Constrained and reconfigurable flight control

Joosten, Diederick

DOI
10.4233/uuid:5d53817c-0956-4ed7-8716-a8e79eb8c86f

Publication date
2017

Document Version
Final published version

Citation (APA)
Joosten, D. (2017). *Constrained and reconfigurable flight control*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:5d53817c-0956-4ed7-8716-a8e79eb8c86f

Constrained and Reconfigurable Flight Control

CONSTRAINED AND RECONFIGURABLE FLIGHT CONTROL

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. Ir. K.Ch.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op

woensdag 10 mei 2017 om 12.30 uur

door

Diederick Alwin JOOSTEN

Master of Science in Systems and Control
geboren te Leiderdorp

Dit proefschrift is goedgekeurd door de promotor:

Prof. dr. ir. M. Verhaegen

en de copromotor:

Dr. ir. A.J.J. van den Boom

Samenstelling promotiecommisie:

Rector Magnificus,	voorzitter
Prof. dr. ir. M. Verhaegen,	Technische Universiteit Delft, promotor
Dr. ir. A.J.J. van den Boom,	Technische Universiteit Delft, copromotor

Onafhankelijke leden:

Prof. dr. ir. J. Hellendoorn,	Technische Universiteit Delft
Prof. dr. ir. J.A. Mulder,	Technische Universiteit Delft
Prof. dr. ir. A.C.P.M. Backx,	Technische Universiteit Eindhoven
Prof. dr. ir. J.M.A. Scherpen,	Rijksuniversiteit Groningen
Dr. Q.P. Chu,	Technische Universiteit Delft

This dissertation has been completed in partial fulfillment of the requirements of the Dutch Institute of Systems and Control DISC for graduate study.

This research has been supported financially by technology foundation STW under project number dmr6515.

Printed in the Netherlands.

Acknowledgments

Looking back, the road has been long and winding. I started my work as a PhD-student in 2005 immediately following my graduation at Delft Center for Systems and Control. I have enjoyed my time in Delft thoroughly and am grateful that I took the opportunity to participate in teaching and supervision of students.

The research itself proved to be a challenge in many ways. In research I have sought scientific elegance, yet in search of such elusive results I have strayed off track. It took a long time before I accepted that there is always another layer, and that that is what scientists call *future work*.

Ton and Michel, thank you for your support in the completion of this thesis. I have appreciated your insights and our scientific discussions greatly. Thank you both and thank you Hans and Mascha for providing the incentive to finish the thesis, so many years since I left Delft.

Former DCSC colleagues, I truly miss our discussions, during work, lunch and outside of office hours. I believe that most but all social and cultural aspects of your (mostly) international backgrounds have been discussed diligently. PhD-students are *isolated* in their research by definition and I have found this to create a form of group cohesion that I still miss today.

Dear Jaap and Ronald, thank you for your support, positively pressuring me to complete the thesis, and for granting me the time to complete this thesis while I was *on the clock*.

Dear family, friends and (former) colleagues, thank you. Partly, I am who I am because of you, our interactions and our mutual adventures.

Dear parents, thank you for your unconditional support. Dear Max and Hanna, every day you continue to amaze me. I wish that Ilja and I will be able to provide for you in the way my parents have always done for me and my siblings, such that you may enjoy the education that you wish and that you have the freedom to make life's choices accordingly. Ilja, thank you for your love and patience. Looking at our kids, I feel that thus far we have done great.

Delft, April 11, 2017
Diederick Joosten

Contents

1 CHAPTER

Introduction

This chapter sets the stage for the remainder of the thesis through demonstrating the relevance of the investigation of fault-tolerant flight control and through providing the required background information. Such basic information includes a brief survey on fault tolerant control research and the basics of model predictive control.

1.1 Motivation of fault-tolerant reconfigurable flight control

Why does one investigate fault-tolerant flight control (FTFC)? Well, for one, *because we can*, for two, *because it is relevant*. Fault tolerant control can contribute significantly towards an overall increase in flight safety and aircraft availability.

Mankind has truly experienced a jump in its technological abilities over the past century. It took only 44 years from the first motorized flight (Wright Flyer 1903, Fig. 1.1a) for Yeager to fly the Bell X-1 past the sound barrier in 1947 (Fig. 1.1b). After that, it took only 22 years until Neil Armstrong got to speak his famous words upon setting foot on the moon for the very first time in history: "*That's one small step for a man, one giant leap for mankind*." It is safe to say that the technological advances of the past century are remarkable and that this holds for the aerospace domain in particular. Advances in aviation have had a significant impact on globalization as a whole.

Technological advances do have their drawbacks. With advance comes an increase in complexity. When one compares the Wright Flyer with a modern jet fighter, then, besides the obvious differences in performance, one major aspect is the enormous difference between the both in the number of components and systems. Current jet fighters and modern airliners are hugely complex pieces of machinery. A well-known example of this explosion in complexity is illustrated by *Moore's law*, first coined in 1965, which states that the number of transistors per area doubles

(a) 1903 Wright Flyer (Daniels 1903)

(b) 1946 Bell X-1 (Hoover 2006)

Figure 1.1: From Wright Flyer to Bell X-1 in less than 50 years time

approximately every two years. Aviation certainly has benefitted from the rapid growth in computing power that this has brought. Fly-by-wire systems, advanced stability augmentation systems and collision avoidance systems have all become reality because of these advances. All of these systems either provide the pilot with more information, or automate the task at hand.

The drawback of this exponential growth in complexity lies in the corresponding growth of the number of systems and subsystems that may fail for one reason or another. Given the systems complexity of aircraft it is no longer easily possible for the crew to establish what exactly has happened when these fail. It is therefore that we need to provide means for the diagnosis of failures and automated recovery.

The aerospace industry is especially conscious of safety and related aspects. The certification of a new aircraft type or subsystem is a lengthy process that is growing ever more complex. Fault-tolerant Flight Control (FTFC) can play a major role in improving the safety, reliability and availability of aircraft. The continuous increase in the number and complexity of onboard systems of aircraft has created demand for a supervisory system that continually monitors the health of onboard systems and reconfigures them when needed. The growth in computing power enables the design of such systems.

In 2015 3% of the accidents in global aviation were contributed to loss of control of the aircraft in-flight (LOC-I), leading to 33% of the fatal accident (See Fig. 1.2). It is postulated here that fault-tolerant flight control could have been of life-saving importance in these cases and its investigation is worth our while.

This chapter motivates the investigation of FTFC and provides the research objectives of the thesis in combination with required background information. The basics of flight control are introduced here, followed by a short introduction to fault detection and diagnosis and an introduction to fault tolerant control in general. The chapter continues with the introduction of Model Predictive Control and its potential use in flight control. The text ends with an overview of the organization of the thesis before continuing to its main body.

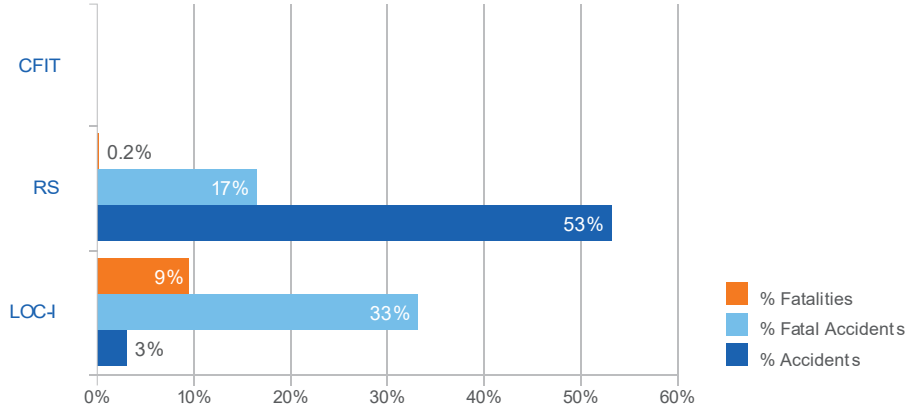

Figure 1.2: Fatalities, fatal accidents and Accidents as recorded by the International Civil Aviation Authority (ICAO 2016). The charts shows accidents, fatal accidents and fatalities for three high risk occurrence categories in 2015. LOC-I means loss of control in-flight, RS means runway safety, and CFIT means controlled flight into terrain. There were no CFIT accidents in 2015.

1.2 Background

This thesis aims to investigate FTFC which lives at the intersection of fault-tolerant control and flight control. The sections below provide background on both subjects.

1.2.1 Introduction to flight control

The construction of a heavier-than-air machine that will fly is but one of the challenges that the pioneers faced in the early days of aviation. Equally important is the ability to control the aircraft in order to have authority over its flightpath. Flight control technology has evolved considerably over the past century.

Aircraft can be modeled as a point mass moving through the air. The wings provide the lift that is needed to sustain the weight of the aircraft in straight and level flight, whereas the engines provide the thrust that is needed to cancel the drag that the aircraft experiences. A pilot needs to rotate around the pitch axis in order to change altitude (climb or descend). If the pilot wants to change direction he will need to use a combination roll, yaw and pitch in order to make the aircraft turn in the desired direction. Several methods can be used to control the attitude of the aircraft.

flight control mechanisms
Otto Lilienthal (1848-1896) made some of the first documented gliding flights using an early version of what one might compare to a modern hang-glider. Much like hang-gliders Lilienthal could control the glider by changing the center of grav-

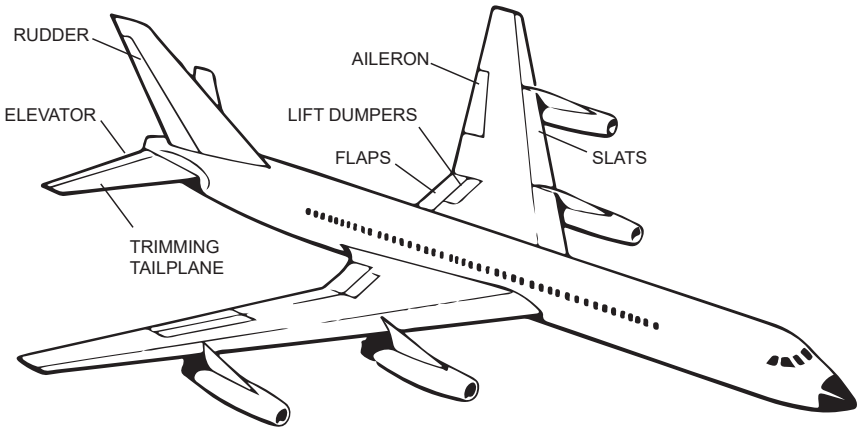

Figure 1.3: Example of flight control surfaces

ity through shifting his body. The glider was difficult to control and eventually Lilienthal fell from 17 meters when the glider lost lift. Lilienthal died saying: "*Kleine Opfer müssen gebracht werden*" (small sacrifices must be made). In subsequent years Wilbur and Orville Wright built their own gliders. They were certain that weight-shift control was not the way forward and came up with a method of *wing-warping* to control the rolling motion of the aircraft. For pitch control they employed a movable canard wing.

Finally, fixed wing aircraft settled on independent moving surfaces to control pitching, yawing and rolling motion independently. The *primary* control surfaces used for these motions are the elevator, the rudder and the ailerons. These surfaces are controlled from a stick, or yoke, and the pedals in the cockpit. Two methods are common in connecting the pilot controls in the cockpit to the control surfaces. These methods are either *cable and pulley* systems, or *push-pull control* rod systems. Next to the primary flight control surfaces, most civilian and military aircraft have *secondary* flight control surfaces. Such secondary flight control surfaces include trailing-edge flaps, leading-edge slats and airbrakes. These flight control surfaces are typically used to control the amount of lift that the aircraft generates, either through changing the wing surface or wing camber (slats and flaps), or through lift 'dumping' (airbrakes). Although most secondary control surfaces are typically used symmetrically, airbrakes or spoilers are also used to aid the ailerons in roll control.

Growth in aircraft size has made it more difficult for pilots to control the aircraft because of the high forces needed to move the control surfaces. This created the necessity to use additional power sources and subsequently *hydraulic boosters* were introduced (at the end of the second World War) to lighten the physical workload for pilots. This system is comparable to power-steering in a car. In a subsequent step of evolution in flight control fully power-operated controls were introduced that are irreversible because the aerodynamic hinge moment has no effect on the

deflection of the surface. The advantage of this system is that it increases the stiffness in the control system which improves the flutter characteristics of the aircraft. At this moment in time the role of mechanical linkage between the cockpit controls and the surface actuators was reduced to one of signaling and no longer for the transmitting of power.

Obviously mechanical linkages are simple and reliable, but electrical signaling allows for implementation of more complex and sophisticated flight controls, possibly enhancing the handling qualities of the aircraft. Traditional benefits of fly-by-wire technology for civil aircraft are (Pratt 2000, p.19):

- the improvement of natural aircraft dynamic behavior, that is: stability, handling qualities, turbulence suppression and passenger comfort;
- the provision of flight envelope protection that allows full pilot commands, if necessary, without danger of either leaving the safe flight envelope or overstressing the aircraft;
- the increase in safety by reduction of pilot workload in routine control tasks, which allow him to concentrate on higher level flight guidance tasks;
- the reduction of airline crew training costs by offering commonality within an aircraft family (cross-crew qualification);
- the more efficient use of crew resources, as one pilot can fly different aircraft types with the same type rating;
- configuration changes can easily be implemented, offering development flexibility and growth potential;
- reduced operational costs, through improved maintainability and a higher dispatch reliability;
- aircraft mass can be reduced, as heavy mechanical parts can be eliminated.

It is fly-by-wire that opens the door to fault-tolerant flight control. There no longer is a mechanical linkage between the cockpit controls and the control surfaces. Hence, it is possible to use the freedom to use each control surface individually, which can be advantageous in a faulty scenario. Some examples include the following:

- propulsion control: given the fact that the engines of a multi-engine aircraft are mounted away from the center of gravity it is possible to create a moment in the yaw, roll and pitch direction, hence making it possible to control the attitude of the aircraft.
- roll control using asymmetric use of flaps or elevator halves: normally extensive mechanisms exist to make sure that flaps are extended symmetrically, but if ailerons become inoperative, if might be important to be able to use flaps or spoilers asymmetrically such that a rolling motion can be achieved using alternative means.
- weight shift control: trim in the roll direction through pumping fuel from one wing tank to another.

Flight critical systems such as the flight control system (FCS), require the highest integrity: system failures which would result in the loss of the aircraft have to be *extremely improbable*, i.e. its probability has to be less than 10^{-9} per flight hour (EASA 2016, AMC 25.1309). This requires redundant and highly reliable components. Furthermore, additional redundant components are installed because airlines need good dispatch rates, i.e. they want to continue with revenue flights safely, even after certain failures have occurred and while being far away from the maintenance base (Pratt 2000). Multiple redundant lanes or channels of computing and actuation are used to achieve this. Additionally extensive integrity monitoring is used to detect faults at the system level. Hardware and software diversity are also important aspects that contribute towards achieving the required safety levels.

Given the knowledge that failures of the present-day flight control system and its individual components are extremely improbable, it may be assumed that it is already particularly unlikely that a fault in an individual actuator or flight control computer will be outright catastrophic. *Why investigate FTFC then?* Serious problems may arise when faults are injected at higher hierarchical level. Examples thereof are situations in which all hydraulic pressure is lost and authority over all primary control surfaces is lost. All the actuator redundancy in the world will not currently solve this problem when hydraulics are the only power source for the flight control surfaces. Another major issue is that structural damage to the aircraft may cause the closed loop of aircraft and autopilot to become unstable. Structural damage may also take the aircraft out of trim, or make it open-loop unstable.

Structural failures
Structural defects lead to a change in the *behavior of the system*. An example of such a failure can be the loss of a vertical fin. Losing the fin causes several problems: the stability in yaw direction is lost, there no longer is a rudder to control yaw, and loss of the rudder has probably caused a leak in the hydraulic system. This is a striking example where the existing flight control law and hardware redundancy are no longer valid.

Sensor failures
Sensor failures are not investigated in this thesis, but actuator failures are.

Actuator failures
Single actuator failures are served relatively well by means of hardware redundancy. It is only when major systems start failing (e.g. total loss of hydraulics, complete loss of elevator authority, etc.) that we are likely to need to control the aircraft using secondary actuators or using secondary effects of the primary actuators.

An alternative reasoning is that through inclusion of fault-tolerant flight control hardware it may be possible to reduce the amount of redundant hardware. Software does not cause a weight penalty, whereas hardware does. Another possible benefit lies in increased dispatch reliability, a major factor in operating economy.

In summary, fault tolerant control methods can either reduce the necessity for hardware redundancy or broaden the scope and number of faults that can be handled. We focus on faults at the system level and assume that single actuator failures do not, or only slightly, affect the behavior of the closed loop of flight control system, airplane structure and actuators, whereas a complete failure of a control surface, loss of an engine or complete hydraulics will.

1.2.2 Failure detection and Fault-tolerant control overview

Fault: *In the general sense, a fault is something that changes the behavior of a system such that the system no longer satisfies its purpose (Blanke 2003).*

Hence, fault-tolerant control has to prevent a fault from causing a failure at the system level.

In larger systems different components typically each have their own purpose. This means that a single fault in a component will in most cases change the performance of the overall system. *Fault tolerant control* is an attempt at finding faults swiftly and at subsequently stopping the propagation of the fault such as to prevent damage to the overall system and human operators. It is the control system that has to deal with this task. We strive to develop a control algorithm that adapts to the faulty plant. Hence, in general the procedure to make a system fault-tolerant consists of two steps (Blanke 2003):

1. Fault diagnosis: the existence of faults has to be detected and the faults have to be identified; and
2. Control re-design: the controller has to be adapted to the faulty situation so that the overall system continues to satisfy its goal.

Classification of Faults

Faults can be classified according to their location in the system, their nature or their time-characteristics. When faults are ordered with respect to their location we distinguish: actuator faults, plant faults and sensor faults (Verhaegen et al. 2010):

Actuator faults represent partial or total loss of control action. Partial loss of control action can be the loss of range in an actuator, or it may be the loss of reaction speed. Partially failed actuators produce only part of the nominal behavior. When full loss of an actuator presents itself this may lead to an actuator that is stuck at a certain position, or an actuator that is floating.

Sensor faults represent incorrect readings from the sensors that the system is fitted with. Faulty sensors may provide the system with amplified versions of the original system, a bias might be included, or the signal may be prone to high noise levels. Either way, the quality of the signal is lost.

Plant faults relate to faults that are associated with changes in the physical parameters of the system. These are the faults that can neither be attributed to actuators or to the sensors. Typical examples would be failures of the system itself, such as damage to the wing of an aircraft.

Faults are typically classified as either being *additive* or *multiplicative*. Sensor and actuator faults can typically be modeled best as multiplicative faults and plant faults are best modeled as additive faults.

Faults are not necessarily observable from the system behavior, but when they are we have to take the effect into account. Fault tolerant control requires that we *detect* that a fault has occurred and that the location of the fault is identified together with *identification* of its severity

Fault tolerant control

Various control methods are suitable for fault tolerant control purposes. At a high level of abstraction one can divide FTC methods into two categories: *active* and *passive* FTC (Jones 2005). Passive methods allow for fault accommodation, whereas active methods use control reconfiguration as a starting point.

Robust control solutions are an example of passive FTC methods. In robust methods, the design of the controller is such that the closed-loop of plant and controller is stable for a whole set of plants. This set can be designed such that certain faulty behavior of the plant fits inside the set. The trade-off in selecting such a robust control law is that one trades performance for robustness. The advantage, however, is that an online FDI system is not strictly required.

In active methods, one does make use of available FDI information and the entire control loop is reconfigured as is necessary. It is possible that the structure of the existing controller remains the same, but that the tuning of its parameters has to be changed in order to accommodate for the fault. Active control reconfiguration can become necessary.

The interested reader is referred to the following references for a generic introduction to fault tolerant control Blanke and Schröder (2006), Patton (2015), Patton (1997). Figure 1.4 provides a classification of fault tolerant control methods, ref Edwards et al. (2010).

1.2.3 Fault tolerant flight control

As stated, FTFC unifies the topic matter of flight control and fault tolerant control. Surveys of, and generic introductions to, FTFC are: Steinberg (2005), Edwards et al. (2010), Zolghadri et al. (2014). Many control methods are suited to the purposes of a fault-tolerant flight controller:

- Multiple model control - the multiple model concept is based on a set of models, each model representing a different operating condition of the plant

(e.g. a fault condition). A controller is designed for each plant model. Central to the control method is an online method to determine a weighted combination of the different controllers that is to be employed. A disadvantage is that the method only considers a finite number of local models (i.e. fault conditions).

- Multiple model and switching (MMST) - in multiple model and switching control a series of models and controllers exist. For each time step the model that is most representative is determined and the corresponding controller selected. Most MMST controllers comprise a tuning mechanism that is applied to the model that corresponds to the active controller. Stability results exist that require a sufficiently dense set of models and a sampling that is fast enough.

- Interacting multiple models (IMM) - IMM attempts to relieve the limitations of the previous two methods in the sense that every fault condition must have been modeled a priori. A key assumption in IMM is that every failure can be modeled as a convex combination of models in a set of models. In theory multiple failures can be handled through combination of single failure models.

- Control Allocation - desired forces and moments about the aircraft center of gravity are inputs to this method. Based on these forces and moments, and estimates of control effector efficiencies and stability derivatives, to compute the inputs necessary to achieve such forces and moments.

- Adaptive feedback linearization - is based on a dynamic inversion controller in an explicit model following architecture. An adaptive neural network is used to adaptively regulate the error between the desired response model and response of the vehicle. Control allocation is applied to generate individual control effector commands (Wise et al. 1999).

- Sliding mode control - sliding mode systems are designed to drive the system states onto a particular surface in the state space, named sliding surface. Once the sliding surface is reached, sliding mode control keeps the states on the close neighborhood of the sliding surface. Hence, sliding mode control is a two part controller design. The first part involves the design of a sliding surface so that the sliding motion satisfies design specifications. The second is concerned with the selection of a control law that will make the switching surface attractive to the system state (Shtessel et al. 2014).

- Eigenstructure assignment - in eigenstructure assignment a linear state feedback controller is obtained through pole placement after which the remaining design freedom is used to align the eigenvectors as accurately as possible.

- Model reference adaptive control - The goal of adaptive model-following is to force the plant output to track a reference model. This can either be done in indirect form through online identification of the plant parameters, or through direct estimation of the controller parameters.

- MPC - applies online optimization to control multivariable systems and is regarded for its ability to incorporate and accommodate various constraints. MPC forms the main content matter of this thesis.
- Knowledge based methods - an artificial neural network (ANN) is a network inspired by biological neural networks. ANN are typified by their ability to accommodate changing behavior, in a certain sense *learning* what has changed. ANN have been applied towards FTFC for purposes of failure detection and identification of a control surface (Napolitano et al. 2000) and cancelation of residual errors in feedback linearization (Kim and Calise 1997).

Figure 1.4 provides a graphic overview of the methods listed above and Table 1.1 from Edwards et al. (2010) provides a comparison of fault tolerant control methods, applicable for reconfigurable flight control, considered in this survey. Filled circles mean that the method has the indicated property while empty circles imply that an author has suggested that the approach could be modified to incorporate the property. The columns are explained as follows:

- Failures: Types of failures that the method can handle
- Robust: The method uses robust control techniques
- Adaptive: The method uses adaptive control techniques
- Fault Model:
  - FDI: An FDI algorithm is incorporated into the method
  - Assumed: The method assumes an algorithm which provides a fault model
- Constraints: The method can handle actuator constraints
- Model Type: The type of internal model used

1.3 Model Predictive Control

Model predictive control (MPC) is central to the theory in this thesis, hence it has to be discussed in this introductory chapter. MPC is a widely used and well accepted controller design method in the process industry (ref. Allgöwer et al. (1999); Biegler (2000); Camacho and Bordons (1995); Clarke et al. (1987a); Garcia et al. (1989); Cutler and Ramaker (1979); Morari and Zafiriou (1989); Richalet et al. (1978)). This motivates the extension of the benefits provided by the MPC framework to high-bandwidth flight systems.

Various methods have been developed since the Seventies for the design of model based control systems for robust multivariable control of industrial processes (Boyd and Barratt 1991; Doyle et al. 1992, 1989; Garcia et al. 1988; Maciejowski 1989; Morari and Zafiriou 1989).

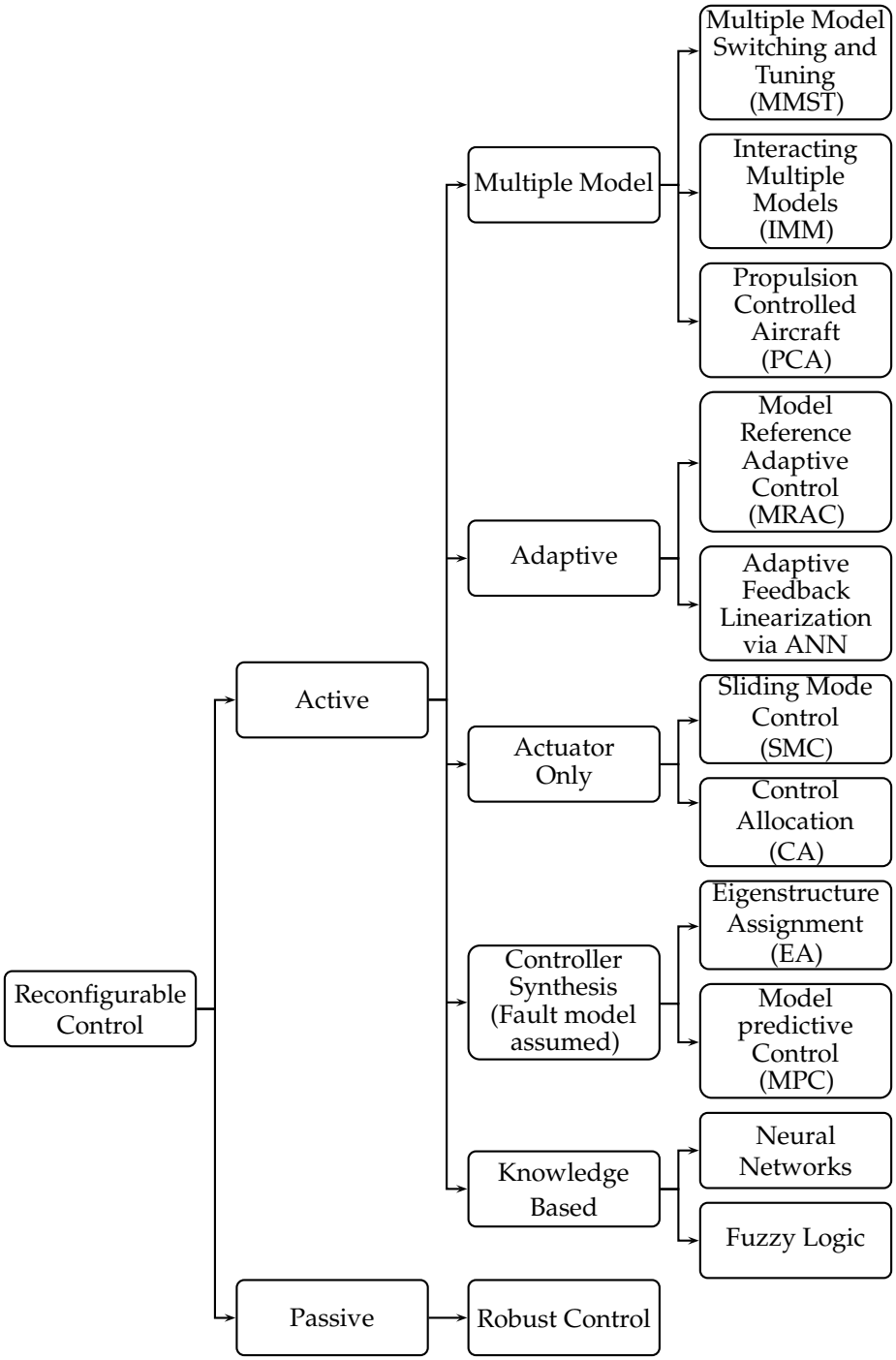

Figure 1.4: Overview and classification of fault tolerant flight control methods.

Table 1.1: FTFC methods, from (Edwards et al. 2010, p.84)

Method	Failures		Robust	Adaptive	Fault Model		Constraints	Model type	
	Actuator	Structural			FDI	Assumed		Linear	Nonlinear
Multiple Model Switching and Tuning (MMST)		●		●	●			●	
Interacting Multiple model (IMM)		●		●	●		○	●	
Propulsion controlled aircraft (PCA)	●		○			●		●	●
Control Allocation (CA)	●					●	○	●	
Feedback Linearization (FL)	●	●		●	●				●
Sliding Mode Control (SMC)	○ [a]	●	● [b]			●			●
Eigenstructure Assignment (EA)		●				●		●	
Pseudo Inverse Method (PIM)		●				●		●	
Model Reference Adaptive Control (MRAC)		●		●	●			●	○
Model Predictive Control (MPC)	●	●	○	○	●	●	●	●	●

[a] Can handle partial loss of effectiveness of actuators, but not complete loss
[b] Assumes robust control can handle all forms of structural failures

Predictive control was pioneered simultaneously by Richalet et al. (1976), Richalet et al. (1978) and Cutler and Ramaker (1980). Model Predictive Control technology has evolved from a basic multivariable process control technology to a technology that enables operation of processes within well-defined operating constraints (Allgöwer et al. 1999; Bequette 1991; Qin and Badgewell 1997). The contributors to the acceptance of MPC technology by the process industry since the 1980's are the following:

- MPC is a model based controller design procedure, which can easily handle processes with large time-delays, non-minimum phase and unstable processes.
- (Industrial) processes typically have limitations in, for instance, valve capacity and other technological requirements and are required to deliver output products against detailed quality specifications. MPC can handle such constraints in a systematic way during design and implementation of the controller.
- Finally, MPC can incorporate structural changes, such as sensor and actuator failures, changes in system parameters and system structure by adapting the control strategy in between measurement samples.

However, the main reasons for its popularity are the constraint-handling capabilities, the straightforward extension to multi-variable processes and, most of all, the possibility to increase process quality and profit margins. From academic side the interest in MPC initially came from the field of self-tuning control. The problem of Minimum Variance control (Åström and Wittenmark (1973)) was studied while minimizing the cost function

$$J(u,k) = \mathrm{E}\left\{ \left(r(k+d) - y(k+d) \right)^2 \right\} \tag{1.1}$$

at time k, where $y(k)$ is the process output signal, $u(k)$ is the control signal, $r(k)$ is the reference signal, $\mathrm{E}(\cdot)$ stands for expectation and d is the process dead-time. To overcome stability problems with non-minimum phase plants, the cost function was modified by adding a penalty on the control signal $u(k)$. Later this $u(k)$ in the cost function was replaced by the increment of the control signal $\Delta u(k) = u(k) - u(k-1)$ to guarantee a zero steady-state error. To handle a wider class of unstable and non-minimum phase systems and systems with poorly known delay the Generalized Predictive Control (GPC) scheme (Clarke and Mohtadi 1989; Clarke et al. 1987a) was introduced with a quadratic cost function.

In GPC mostly polynomial based models are used. For instance, Controlled AutoRegressive Moving Average (CARMA) models or Controlled AutoRegressive Integrated Moving Average (CARIMA) models are popular. These models describe the process using a minimum number of parameters and therefore lead to effective and compact algorithms. Most GPC-literature in this area is based on Single-Input Single-Output (SISO) models. However, the extension to Multiple-Input Multiple-Output (MIMO) systems is straightforward as was shown by De Vries and Verbruggen (1994) using a MIMO polynomial model, and by Kinnaert (1989) using a state-space models.

This text covers state-of-the-art technologies for model predictive process control that are good candidates for future generations of industrial model predictive control systems. Like all other controller design methodologies, MPC also has its drawbacks:

- A detailed plant model is required. Good insight in the physical behavior of the plant is required or system identification techniques have to be applied to obtain a good model.
- The methodology is open, and has given rise to many variations. Such variations include: IDCOM (Richalet et al. 1978), DMC (Cutler and Ramaker 1979), EPSAC (De Keyser and van Cauwenberghe 1982), MAC (Rouhani and Mehra 1982), QDMC (Garcia and Morshedi 1986), GPC (Clarke et al. 1987a) and (Clarke et al. 1987b), PFC (Richalet 1993), UPC (Soeterboek 1992).
- Although, in practice, stability and robustness are easily obtained by accurate tuning, theoretical results on stability and robustness properties are difficult to achieve.

Industry specialists often prefer MPC for supervisory optimizing control of multivariable processes over other controller design methods, such as PID, LQ and H_∞. A PID controller is easily tuned but is basically limited to SISO systems. LQ and H_∞ can be applied to MIMO systems, but cannot incorporate constraints in an adequate way. These techniques also exhibit difficulties in realizing robust performance for varying operating conditions. Key element in model predictive control is the use of a model that can simulate dynamic behavior of the process at in a certain condition. In this respect, model predictive control differs from most of the model based control technologies that have been studied in academia in the Sixties, Seventies and Eighties. Academic research has mostly focused on the use of models for controller design and robustness analysis of control systems for quite some time. With their initial work on internal model based control, Garcia and Morari (1982) made a first step towards bridging academic research in the area of process control and industrial developments in this area. Significant progress has been made in understanding the behavior of model predictive control systems, and a numerous results have been obtained on stability, robustness and performance of MPC (Soeterboek (1992), Camacho and Bordons (1995), Maciejowski (2002a), Rossiter (2003)).

Since the pioneering work at the end of the Seventies and early Eighties, MPC has become the most widely applied supervisory control technique in the process industry. Many papers report successful applications (see Richalet (1993), and Qin and Badgewell (1997)).

In the Eighties and Nineties MPC was mainly applied in the process industry, since the slow dynamics permit the inter-sampling computations for model update and optimal future control sequence determination. The dramatically increasing available computer power now allows the extension of the computational demanding MPC technology to high-bandwidth flight control systems (Keviczky and Balas 2003).

The MPC scheme works as follows (see also Figure 1.5). A model (in our case

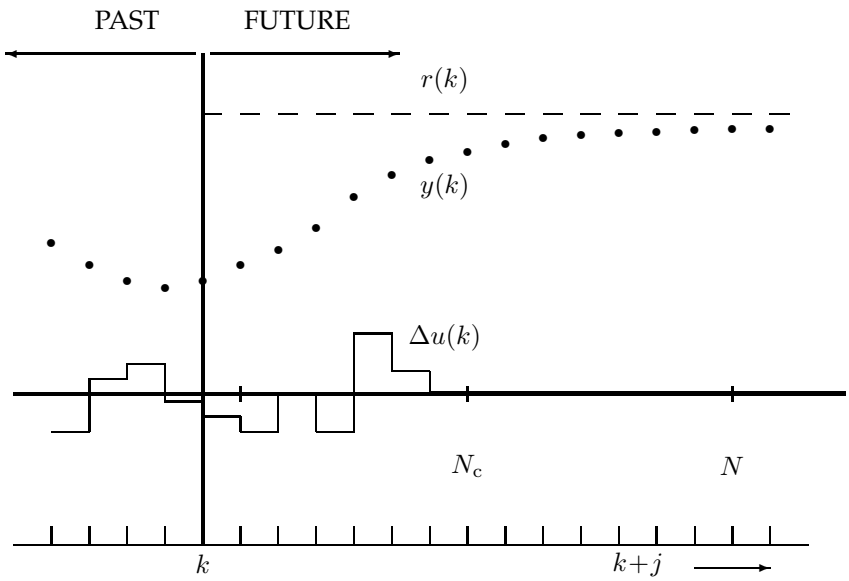

Figure 1.5: The 'Receding horizon' in predictive control (van den Boom and Stoorvogel 2010, Fig 2.5., p.29)

a linear time-invariant discrete-time description of relevant process dynamics) is used to predict the outcome $y(k)$ (the controlled variables) of the process based on an input sequence $u(k)$ (the sequence of control inputs or manipulated variables) supplied to the process and on past measurements of the process. The goal is to achieve that a tracking error signal $z(k)$ (often reflecting the difference between the output signal $y(k)$ and a given reference trajectory $r(k)$) that remains small with reasonable control costs (related to e.g. energy consumption and pollution). In many applications we will use the increment input $\Delta u(k) = u(k) - u(k-1)$ for this will automatically lead to an integrating action in the controller, which is useful for reducing the steady state error. A cost criterion reflects the reference tracking error and the control effort. The prediction horizon N is the number of time steps at which the tracking error signal should be minimized. The optimal input sequence over a given horizon can now be computed by solving an optimization problem (i.e. minimize the cost criterion over the allowed input sequences – and the corresponding $y(k)$ predicted on the basis of the model – given the information of the past behavior of the process). Let us look at the procedure more closely.

Linear MPC uses a linear time-invariant (LTI) state-space representation for the model:

$$x(k+1) = Ax(k) + B\Delta u(k) + H\,e(k) \tag{1.2}$$
$$y(k) = Cx(k) + e(k) \tag{1.3}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $H \in \mathbb{R}^{n\times m}$, and $C \in \mathbb{R}^{m\times n}$. The vector $x \in \mathbb{R}^n$ represents the state, $\Delta u \in \mathbb{R}^p$ the input, $y \in \mathbb{R}^m$ the output, $e \in \mathbb{R}^m$ is zero-mean

white noise, and $k \in \mathbb{Z}$ is the discrete time counter. In this thesis we use a cost function with one term that reflect the tracking error and one term that reflects the control action. The following 2-norm cost function is introduced:

$$J(u,k) = \sum_{j=1}^{N} \hat{z}^T(k+j-1|k)\hat{z}(k+j-1|k) \tag{1.4}$$

where we defined the cost signal

$$z(k) = C_z\, x(k) + D_z\, \Delta u(k) + E_z\, e(k) + F_z\, r(k) \tag{1.5}$$

in which $\hat{z}(k+j-1|k)$ is the prediction of $z(k+j-1)$ at time k, Δ is the difference operator such that $\Delta u(k) = u(k) - \Delta u(k-1)$ and $\hat{z}(k+j-1|k)$ is the prediction of $z(k+j-1)$ given the information up to time instant k.

A key advantage of MPC is that we can immediately accommodate for constraints on the input and outputs of the process. This changes the optimization problem only by incorporating the additional limitations. However, this renders the optimization much more complex and will require more computation time.

In MPC the input is often taken to be constant from a certain point onward: $u(k+j) = u(k+N_c-1)$ (or equivalently $\Delta u(k+j) = 0$) for $j = N_c, \ldots, N-1$ where N_c is the control horizon. The use of a control horizon leads to a reduction of the number of optimization variables. This results in a decrease of the computational burden, a smoother controller signal (because of the emphasis on the average behavior rather than on aggressive noise reduction), and a stabilizing effect (since the output signal is forced to its steady-state value).

MPC uses a receding horizon principle. At time step k the future control sequence $\Delta u(k), \ldots, \Delta u(k+N_c-1)$ is determined such that the cost criterion is minimized subject to the constraints. At time step k the first element of the optimal sequence ($\Delta u(k)$) is applied to the process. At the next time step the horizon is shifted, the model is updated with new information of the measurements, and a new optimization at time step $k+1$ is performed.

By successive substitution of (1.5) in (1.2), estimates of the future values of the output can be computed (Camacho and Bordons 1995). In matrix notation we obtain:

$$\tilde{\mathbf{z}}(k) = \tilde{\mathbf{C}}\, x(k) + \tilde{\mathbf{D}}\, \tilde{\mathbf{u}}(k) + \tilde{\mathbf{E}}\, e(k) + \tilde{\mathbf{F}}\, \tilde{\mathbf{r}}(k)$$

with

$$\tilde{\mathbf{z}}(k) = \begin{bmatrix} \hat{z}(k|k) \\ \hat{z}(k+1|k) \\ \vdots \\ \hat{z}(k+N-1|k) \end{bmatrix},\ \tilde{\mathbf{r}}(k) = \begin{bmatrix} r(k) \\ r(k+1) \\ \vdots \\ r(k+N-1) \end{bmatrix},\ \tilde{\mathbf{u}}(k) = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N-1) \end{bmatrix}, \tag{1.6}$$

$$\tilde{\mathbf{C}} = \begin{bmatrix} C_z \\ C_zA \\ \vdots \\ C_zA^{N-1} \end{bmatrix}, \quad \tilde{\mathbf{D}} = \begin{bmatrix} D_z & 0 & \cdots & 0 & 0 \\ C_zB & D_z & \ddots & \vdots & \vdots \\ C_zAB & C_zB & \ddots & 0 & 0 \\ \vdots & & \ddots & D_z & 0 \\ C_zA^{N-2}B & & \cdots & C_zB & D_z \end{bmatrix}, \tag{1.7}$$

$$\tilde{\mathbf{E}} = \begin{bmatrix} E_z \\ C_zH \\ C_zAH \\ \vdots \\ C_zA^{N-2}H \end{bmatrix}, \quad \tilde{\mathbf{F}} = \mathtt{diag}(F_z, \ldots, F_z). \tag{1.8}$$

where $\mathtt{diag}(A_1, \ldots, A_n)$ is defined as a block diagonal matrix with the blocks A_1 through A_n on its diagonal. The cost function (1.4) can now be written as

$$\begin{aligned} J(u,k) &= \sum_{j=N_\mathrm{m}}^{N} \hat{z}^T(k+j-1|k)\hat{z}(k+j-1|k) \\ &= \tilde{\mathbf{z}}^T(k)\tilde{\mathbf{z}}(k) \\ &= (\tilde{\mathbf{C}}\,x(k) + \tilde{\mathbf{D}}\,\tilde{\mathbf{u}}(k) + \tilde{\mathbf{E}}\,e(k) + \tilde{\mathbf{F}}\,\tilde{\mathbf{r}}(k))^T\,(\tilde{\mathbf{C}}\,x(k) + \tilde{\mathbf{D}}\,\tilde{\mathbf{u}}(k) \\ &\quad +\tilde{\mathbf{E}}\,e(k) + \tilde{\mathbf{F}}\,\tilde{\mathbf{r}}(k)) \\ &= \tilde{\mathbf{u}}^T(k)\,(\tilde{\mathbf{D}}^T\tilde{\mathbf{D}})\tilde{\mathbf{u}}(k) \\ &\quad +2\,(\tilde{\mathbf{C}}\,x(k) + \tilde{\mathbf{E}}\,e(k) + \tilde{\mathbf{F}}\,\tilde{\mathbf{r}}(k))^T\,\tilde{\mathbf{D}}(k)\tilde{\mathbf{u}}(k) \\ &\quad +(\tilde{\mathbf{C}}\,x(k) + \tilde{\mathbf{E}}\,e(k) + \tilde{\mathbf{F}}\,\tilde{\mathbf{r}}(k))^T\,(\tilde{\mathbf{C}}\,x(k) + \tilde{\mathbf{E}}\,e(k) + \tilde{\mathbf{F}}\,\tilde{\mathbf{r}}(k)) \\ &= \tilde{\mathbf{u}}^T(k)H\tilde{\mathbf{u}}(k) + f^T(k)\tilde{\mathbf{u}}(k) + c(k) \end{aligned}$$

This means that the cost function is quadratic in the control variable $\tilde{\mathbf{u}}(k)$ and so by omitting the constant term $c(k)$ we obtain:

$$J(u,k) = \tilde{\mathbf{u}}^T(k)H\tilde{\mathbf{u}}(k) + f^T(k)\tilde{\mathbf{u}}(k) \tag{1.9}$$

In practical applications signals are always constrained. We consider the linear constraint

$$\tilde{\mathbf{C}}_c(k)x(k) + \tilde{\mathbf{D}}_c(k)\tilde{\mathbf{u}}(k) + \tilde{\mathbf{E}}_c(k)e(k) + \tilde{\mathbf{F}}_c(k)\tilde{\mathbf{r}}(k) + \tilde{\mathbf{J}}\mathbf{u}(k-1) \leqslant \tilde{\mathbf{h}}(k) \tag{1.10}$$

with $\tilde{\mathbf{C}}_c(k) \in \mathbb{R}^{l\times n}$, $\tilde{\mathbf{D}}_c(k) \in \mathbb{R}^{l\times pN}$, $\tilde{\mathbf{E}}_c(k) \in \mathbb{R}^{l\times m}$, $\tilde{\mathbf{F}}_c(k) \in \mathbb{R}^{l\times mN}$, $\tilde{\mathbf{J}} \in \mathbb{R}^{l\times p}$, $\tilde{\mathbf{h}}(k) \in \mathbb{R}^l$ for some integer l. Finally, we introduce the control horizon constraint

$$\Delta u(k+j) = 0 \qquad \text{for } j = N_\mathrm{c}, N_\mathrm{c}+1, \ldots, N-1 \tag{1.11}$$

to reduce computational complexity and to smoothen the input signal's behavior.

The MPC problem at time step k for linear time invariant systems is defined as follows:

Find the input sequence $u(k), \ldots, u(k+N_c-1)$ that minimizes the cost function (1.9) subject to the inequality constraints (1.10) and subject to the control horizon constraint (1.11).

Note that the MPC problem boils down to the following quadratic programming problem

$$\begin{aligned} &\min_{\tilde{\mathbf{u}}(k)} \ \tilde{\mathbf{u}}^T(k) H \tilde{\mathbf{u}}(k) + f^T(k)\tilde{\mathbf{u}}(k) \\ &\text{subject to} \ \ \tilde{\mathbf{C}}_c(k)x(k) + \tilde{\mathbf{D}}_c(k)\tilde{\mathbf{u}}(k) + \tilde{\mathbf{E}}_c(k)e(k) + \tilde{\mathbf{F}}_c(k)\tilde{\mathbf{r}}(k) + \tilde{\mathbf{J}}\mathbf{u}(k-1) \leqslant \tilde{\mathbf{h}}(k) \end{aligned}$$

In the absence of constraints (1.10) and (1.11) the solution can easily be found by setting the gradient of the cost function to zero, resulting in

$$\nabla_{\tilde{\mathbf{u}}(k)} \tilde{\mathbf{u}}^T(k) H \tilde{\mathbf{u}}(k) + f^T(k)\tilde{\mathbf{u}}(k) + c(k) = 2H\tilde{\mathbf{u}}(k) + f(k) = 0$$

and so for the unconstrained case we find the optimal solution $\tilde{\mathbf{u}}^*(k) = -H^{-1}f(k)$. Using the receding horizon principle we can compute the optimal control signal at time k as

$$\Delta u(k) = -E_u H^{-1} f(k) \tag{1.12}$$

where

$$E_u = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix} \tag{1.13}$$

The solution can be realized with a feedback law

$$\Delta u(k) = -K\, x(k) + D_e\ \mathrm{e}(k) + D_r\, \tilde{\mathbf{r}}(k) \tag{1.14}$$

with $K = E_u H^{-1}\tilde{\mathbf{C}}$, $D_e = -E_u H^{-1}\tilde{\mathbf{E}}$ and $D_r = -E_u H^{-1}\tilde{\mathbf{F}}$.

Quadratic programming problems can be solved very efficiently. Various algorithms to solve the quadratic programming problem exist: the modified simplex method (algorithms that use a modified version of the simplex method are Wolfe's algorithm (Wolfe 1959) and the pivoting algorithm of Lemke (1968) are most efficient for small and medium-sized problems). The algorithm will find the optimum in a finite number of steps. An alternative for large-sized quadratic programming problems is the interior point method Nesterov and Nemirovskii (1994). A disadvantage of this method is that the optimum can only be approximated. However, bounds for the approximation can be derived.

Predictive control design does not give an a priori guaranteed stabilizing controller. To enforce closed-loop stability we can introduce the following infinite horizon cost function $N = \infty$:

$$J(u,k) = \sum_{j=1}^{\infty} \hat{z}^T(k+j-1|k)\hat{z}(k+j-1|k) \tag{1.15}$$

where r_{ss} is the steady-state reference signal, for which there holds $r(k+j|k) = r_{ss}$ for $j \leq 0$. The steady state x_{ss} can be computed by solving the following equations:

$$\begin{aligned} x_{ss} &= A\, x_{ss} \\ z_{ss} &= C_z\, x_{ss} + F_z\, r_{ss} = 0 \end{aligned}$$

The results are summarized in the following theorem:

Theorem 1.1
Consider the system

$$\begin{aligned} x(k+1) &= A\, x(k) + H\, e(k) + B\, \Delta u(k) \\ y(k) &= C\, x(k) + e(k) \\ z(k) &= C_z\, x(k) + D_z\, \Delta u(k) + E_z\, \hat{e}(k) + F_z\, r(k) \end{aligned}$$

with $r(k+j|k) = r_{ss}$ for $j \leq 0$. Let P be the solution of the algebraic Riccati equation

$$P = A^T P A - (A^T P B + C_z^T D_z)(B^T P B + D_z^T D_z)^{-1}(B^T P A + D_z^T C_z) + C_z^T C_z$$

The unconstrained infinite horizon standard predictive control problem of minimizing performance index

$$J(v,k) = \sum_{j=0}^{\infty} \hat{z}^T(k+j|k)\hat{z}(k+j|k) \tag{1.16}$$

is solved by control law

$$\Delta u(k) \quad = \quad -K\,(x(k|k) - x_{ss}) + D_e\, e(k) \tag{1.17}$$

where

$$\begin{aligned} K &= (B^T P B + D_z^T D_z)^{-1}(B^T P A + D_z^T C_z) \\ D_e &= -\,(B^T P B + D_z^T D_z)^{-1}(B^T P H + D_z^T E_z) \end{aligned}$$

.

Proof: we consider the unconstrained infinite horizon standard predictive control problem. The system to be controlled is described as:

$$\begin{aligned} x(k+1) &= A\, x(k) + H\, e(k) + B\, \Delta u(k) \\ y(k) &= C\, x(k) + e(k) \\ z(k) &= C_z\, x(k) + E_z\, \hat{e}(k) + D_z\, \Delta u(k) + F_z r(k) \end{aligned}$$

with $r(k+j|k) = r_{ss}$ for $j \leq 0$. Prediction:

$$\begin{aligned} \hat{x}(k+j+1|k) &= A\, \hat{x}(k+j|k) + H\, \hat{e}(k+j|k) + B\, v(k+j|k) \\ \hat{z}(k+j|k) &= C_z\, \hat{x}(k+j|k) + E_z\, \hat{e}(k+j|k) + D_z\, v(k+j|k) \end{aligned}$$

Choosing

$$\bar{x}(k+j|k) = \begin{bmatrix} \hat{x}(k+j|k) \\ \hat{e}(k+j|k) \end{bmatrix}$$

we obtain:

$$\begin{aligned} \bar{x}(k+j+1|k) &= \bar{A}\,\bar{x}(k+j|k) + \bar{B}\,v(k+j|k) \\ \hat{z}(k+j|k) &= \bar{C}\,\bar{x}(k+j|k) + \bar{D}\,v(k+j|k) \end{aligned}$$

where

$$\begin{aligned} \bar{A} &= \begin{bmatrix} A & H \\ 0 & 0 \end{bmatrix} \\ \bar{B} &= \begin{bmatrix} B \\ 0 \end{bmatrix} \\ \bar{C} &= \begin{bmatrix} C_z & E_z \end{bmatrix} \\ \bar{D} &= D_z \end{aligned} \tag{1.18}$$

Substitution in the performance index leads to:

$$\begin{aligned} J(v,k) =& \sum_{j=0}^{\infty} \hat{z}^T(k+j|k)\Gamma(j)\hat{z}(k+j|k) \\ =& \sum_{j=0}^{\infty} \big(\bar{x}^T(k+j|k)\bar{C}^T\bar{C}\bar{x}(k+j|k) \\ & +2\bar{x}^T(k+j|k)\bar{C}^T\bar{D}\,v(k+j|k) + v^T(k+j|k)\bar{D}^T\bar{D}\,v(k+j|k)\big) \\ =& \sum_{j=0}^{\infty} \bar{x}^T(k+j|k)\bar{Q}\bar{x}(k+j|k) + \bar{x}^T(k+j|k)\bar{S}v(k+j|k) \\ & + v^T(k+j|k)\bar{R}v(k+j|k) \end{aligned}$$

where

$$\begin{aligned} \bar{Q} &= \bar{C}^T\bar{C} = \begin{bmatrix} C_z & E_z \end{bmatrix}^T \begin{bmatrix} C_z & E_z \end{bmatrix}, \\ \bar{S} &= \bar{C}^T\bar{D} = \begin{bmatrix} C_z & E_z \end{bmatrix}^T D_z, \\ \bar{R} &= \bar{D}^T\bar{D} = D_z^T D_z. \end{aligned} \tag{1.19}$$

Minimization of the performance index is equivalent to the design of an LQR regulator with the solution

$$v(k) = (\bar{B}^T\bar{P}\bar{B} + \bar{R})^{-1}(\bar{B}^T\bar{P}\bar{A} + \bar{S}^T)\bar{x}(k) \tag{1.20}$$

where $P \geq 0$ is the smallest positive semi-definite solution of the discrete time Riccati equation

$$P = \bar{A}^T P\bar{A} - (\bar{A}^T P\bar{B} + \bar{S})(\bar{B}^T P\bar{B} + \bar{R})^{-1}(\bar{B}^T P\bar{A} + S^T) + \bar{Q}$$

which exists due to stabilizability of $(\bar{A}, \bar{B})$ and invertibility of $\bar{R}$. Note that

$$\begin{aligned}
\begin{bmatrix} P_1 & P_2 \\ P_2^T & P_3 \end{bmatrix} &= \begin{bmatrix} A_1^T & 0 \\ A_2^T & 0 \end{bmatrix} \begin{bmatrix} P_1 & P_2 \\ P_2^T & P_3 \end{bmatrix} \begin{bmatrix} A & H \\ 0 & 0 \end{bmatrix} \\
&\quad - \left(\begin{bmatrix} A^T & 0 \\ H^T & 0 \end{bmatrix} \begin{bmatrix} P_1 & P_2 \\ P_2^T & P_3 \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix} + \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \right) \\
&\quad \times \left(\begin{bmatrix} B^T & 0 \end{bmatrix} \begin{bmatrix} P_1 & P_2 \\ P_2^T & P_3 \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \right)^{-1} \\
&\quad \times \left(\begin{bmatrix} B^T & 0 \end{bmatrix} \begin{bmatrix} P_1 & P_2 \\ P_2^T & P_3 \end{bmatrix} \begin{bmatrix} A & H \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} S_1^T & S_2^T \end{bmatrix} \right) + \bar{Q} \\
&= \begin{bmatrix} A^T P_1 A & A^T P_1 H \\ H^T P_1^T A & H^T P_1^T H \end{bmatrix} \\
&\quad - \begin{bmatrix} A^T P_1 B + S_1 \\ H^T P_1 B + S_2 \end{bmatrix} (B^T P_1 B + D_z^T D_z)^{-1} \\
&\quad \times \begin{bmatrix} B^T P_1 A + S_1^T & B^T P_1 H + S_2^T \end{bmatrix} + \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
P_1 &= A^T P_1 A - (A^T P_1 B + S_1)(B^T P_1 B + D_z^T D_z)^{-1}(B^T P_1 A + S_1^T) + Q_1 \\
P_2 &= A^T P_1 H - (A^T P_1 B + S_1)(B^T P_1 B + D_z^T D_z)^{-1}(B^T P_1 H + S_2^T) + Q_2 \\
P_3 &= H^T P_1 H - (H^T P_1 B + S_2)(B^T P_1 B + D_z^T D_z)^{-1}(B^T P_1 H + S_2^T) + Q_3
\end{aligned}$$

so we can write

$$\begin{aligned}
v(k) &= (\bar{B}^T P \bar{B} + \bar{R})^{-1}(\bar{B}^T P \bar{A} + \bar{S}^T)\bar{x}(k) \\
&= \bar{K}\, \bar{x}(k) \\
&= (B^T P_1 B + D_z^T D_z)^{-1} \begin{bmatrix} B^T P_1 A + S_1^T & B^T P_1 H + S_2^T \end{bmatrix} \begin{bmatrix} \hat{x}(k) \\ \hat{e}(k) \end{bmatrix} \\
&= -K\, x(k|k) + D_e\, e(k|k)
\end{aligned}$$

which constitutes the discrete time LQR problem, where

$$\begin{aligned}
K &= (B^T P_1 B + D_z^T D_z)^{-1}(B^T P_1 A + D_z^T C_z) \\
D_e &= -\,(B^T P_1 B + D_z^T D_z)^{-1}(B^T P_1 H + D_z^T E_z)
\end{aligned}$$

.

□

Bitmead et al. (1990) showed that infinite horizon cost function (1.16) is equivalent to the following cost function

$$\begin{aligned}
\min_{\tilde{v}(k)} J(\tilde{v}, k) = \min_{\tilde{v}(k)} \Bigg\{ &\Big(x(k+N|k) - x_{ss} \Big)^T P_0 \Big(x(k+N|k) - x_{ss} \Big) \\
&+ \sum_{j=1}^{N} \hat{z}^T(k+j-1|k)\hat{z}(k+j-1|k) \Bigg\}, \qquad (1.21)
\end{aligned}$$

where P_0 is the solution to the Riccati equation

$$\bar{A}^T P_0 \bar{A} - \bar{A}^T P_0 B(B^T P_0 B + B^T C^T Q C B)^{-1} B^T P_0 \bar{A} + A^T C^T Q (I - CB(B^T C^T Q C B)^{-1} B^T C^T) Q C A - P_0 = 0, \quad (1.22)$$

where $\bar{A} = A - B(B^T C^T Q C B + R)^{-1} B^T C^T Q C A$.

Note that cost function (1.22) is a finite horizon cost function with an additional terminal cost $\Big(x(k+N|k) - x_{ss}\Big)^T P_0 \Big(x(k+N|k) - x_{ss}\Big)$. The predictive control law, minimizing (1.21), results in a stable closed loop (Bitmead et al. 1990).

The main disadvantage of the terminal cost function is that it can only handle the unconstrained case. If we introduce a terminal constraint set we can ensure closed-loop stability for the constrained case in a non-conservative way:

Theorem 1.2 *(Gilbert and Tan (1991), Scokaert and Rawlings (1998), Sznaier and Damborg (1987)) Consider the LTI system (1.2)-(1.3) with cost function (1.21). Let the signal constraints be defined by*

$$\tilde{\mathbf{F}} x(k-1) + \mathbf{G}\tilde{\mathbf{r}}(k) + \mathbf{H}\tilde{\mathbf{u}}(k) \leqslant \tilde{\mathbf{h}}, \quad (1.23)$$

where $\tilde{\mathbf{F}}$, $\mathbf{G}$, $\mathbf{H}$, and $\tilde{\mathbf{h}}$ are constant matrices. Let $r(k) = r_{ss}$ for $k \geq 0$, and consider the linear control law

$$v(k+j|k) = \Big((B^T P_0 B + B^T C^T Q C B)^{-1} B^T P_0 \bar{A} + (B^T C^T Q C B)^{-1} B^T C^T Q C A\Big)\Big(x(k+j|k) - x_{ss}\Big). \quad (1.24)$$

Finally let $\mathcal{W}$ be the set of all states for which 1.23 holds under control law 1.24, and assume

$$\mathcal{D} \subset \mathcal{W}. \quad (1.25)$$

Then the predictive control law, minimizing (1.21), subject to 1.23 and terminal constraint

$$x(k+N|k) \in \mathcal{D}, \quad (1.26)$$

results in a stable closed loop.

1.3.1 MPC in flight control

First advances in the direction of MPC for use in flight control system applications have among others been reported in Heise and Maciejowski (1996), Singh et al. (1995), Maciejowski and Jones (2003).

This thesis investigates the applicability of MPC for reconfigurable flight control because we deem it particularly suitable to the task in view of the advantages (and disadvantages) mentioned in the leading paragraphs of 1.3. These are repeated here for reasons of convenience.

Advantages of MPC:

- one can modify the model that is used in the computation phase of MPC in between the time steps (i.e. the state-space system matrices) because it is optimization based.
- one can change the plant constraints on the input, output and states in between the time steps
- MPC has some inherent robustness against modeling errors and disturbances because it recomputes the optimal input sequence at each time step.

Drawbacks of MPC:

- a stability proof is more difficult to provide
- modeling and control in the discrete-time domain is not always very well accepted
- MPC can be computationally intensive for complex systems, with the risk of calculation not completing before the end of the sampling interval/discrete time step.
- switching between models is not necessarily a smooth phenomenon.

It is also important to notice that MPC can be seen as a control allocation method that takes dynamics into account. Formulated in reverse, control allocation is MPC with a prediction horizon equal to $N = 1$. Both MPC and Control Allocation offer maximum flexibility in the distribution of desired control effort over the available actuators but MPC will generally give a smoother response. Control allocation methods are quite well-known in flight control theory literature.

1.4 Towards MPC based FTFC

This section forms a prelude to the main body of this thesis. The research objectives include the synthesis of MPC type controllers that allow for different levels of performance, or otherwise formulated, controllers that allow for gradual degradation of performance or more strict operational constraints for the system under control. Research constraints and assumptions are also presented here.

1.4.1 Synthesis of the research objectives

Formulating the research objectives requires a clear framework of what it is that should be achieved. This thesis focuses on the application of modern control methods towards reconfigurable flight control. The latter does not mean that methods described in this thesis are not applicable to other systems, but the focus is on aircraft due to the relevance demonstrated in the introductory section of this chapter.

Aircraft are designed with safety aspects in mind. Extensive redundancy is typically built into the flight control system. Most airliners have two, three, or even

four engines and can sustain normal flight with one or several engines inoperative. The same redundancy is available in other aircraft systems. Typical airliners use multiple actuators per control surface, or they have multiple control surfaces altogether. Even flight-control computers and sensors typically have double our triple backups. This form of redundancy is well suited to tackle problems that arise when single failures arise. Where lies the need for more complicated FTFC then? It is useful when a surface becomes inoperative altogether, when multiple systems fail such that some or all primary flight controls are lost.

It is the objective of this thesis to investigate fault tolerant flight control in the event of actuator or plant faults. Table 1.1 suggests that model predictive control (MPC) is a very suitable for use as fault tolerant flight control method due to its ability to incorporate various constraints. In summary, it is the objective of this thesis to investigate the use of MPC as FTFC method.

An MPC problem is sought, changing the objective function when necessary, such that the controller offers three distinct levels of performance:

- level 1: nominal operation, existing autopilot works properly.
- level 2: operation in which the desired closed-loop performance can be realized within the operational constraints (including input constraints).
- level 3: operation in which not all operational constraints can be met, but the plant is still stabilizable.

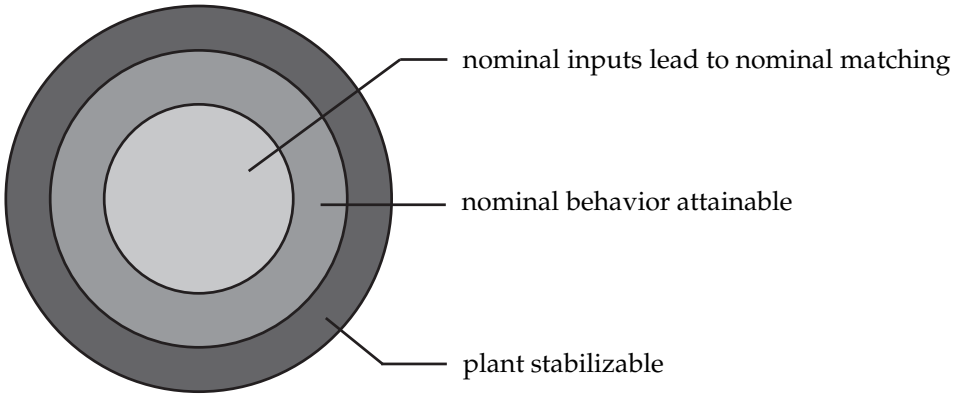

Figure 1.6: Illustration of different performance levels that can be attained with controller matching.

For obvious reasons, level 1 performance will only be attainable in those cases where the plant has no failures. Level 2 performance is achievable in case of actuator failures and only when redundant actuators are available such that the original closed-loop behavior can be matched. The final level, level 3 performance can be

regarded as a back-up mode. Level 3 abandons all desire to match the original behavior, but focuses on stabilization of the plant. Level 1 is subset of level 2, and level 2 is a subset of level 3 behavior. Figure 1.6 illustrates this. Because of reasons that focus on stability of the closed loop it is desirable that these three performance levels are included into one multi-objective cost-function such that switching between different cost-functions can be avoided.

A potential feasibility recovery technique for level 3 performance is based on prioritization of the constraints. The constraints are ordered from lowest to highest priority. In the (nominal) optimization problem becomes infeasible we start by dropping the lowest constraints and see if the resulting reduced optimization problem becomes feasible. As long as the problem is not feasible we continue by dropping more and more constraints until the optimization is feasible again. This means we solve a sequence of quadratic programming problems in the case of infeasibility. The algorithm minimizes the violations of the constraints which cannot be fulfilled. Note that it may take several trials of dropping constraints and trying to find an optimal solution, which is not desirable in any real time application.

Figure 1.7 shows how the MPC framework is applied to achieve conformance with the performance levels introduced in the previous section (Figure 1.6)

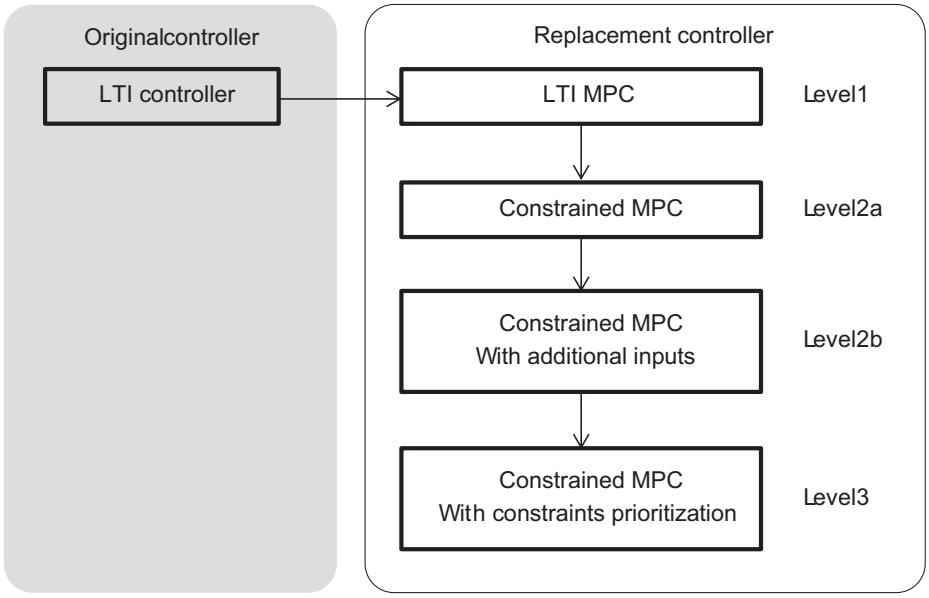

Figure 1.7: Illustration of different performance levels that can be attained with controller matching.

Research constraints and assumptions

In the remainder of this thesis, and in investigating the use of MPC as FTFC method, it is assumed that we are dealing with fixed wing aircraft. The assumption is made that it is possible to control all actuator surfaces independently in the nominal and fault-free case (e.g. left and right wing aileron are independent). Finally, we assume that the aircraft has some redundancy in actuators such that at least one alternate means of control around one of the rotational axes exists. The engines and (possibly asymmetrical use of) secondary control surfaces are assumed to be good candidate alternate means. Furthermore, trimmed flight is assumed to still be possible. Control methods are developed that allow us to make efficient use of the remaining control surfaces.

In this thesis it is assumed that failure detection and identification information becomes available following the introduction of a fault. For instance, use can be made of an online identification algorithm such as described by Lombaerts (2010). This method continuously identifies the aerodynamic aircraft parameters from online measurement data. In that sense, it is not explicitly a failure detection and isolation method, but it rather is an online identification method. Furthermore, it does not identify changes in the physical aircraft parameters like mass and inertia. It is, however, very well possible to extract actuator failure information from the fact that some actuator efficiency is identified to be zero (no effectiveness lost, e.g. the actuator has locked into place).

This thesis poses an exploration of possibilities. There exist many obstacles before practical application of such methods will become feasible in real life situations. Such limitations include the current deterministic methods for clearance of flight control laws that appear to not handle changing controller parameters very well, and acceptance by flight crews due to the inherent loss of situational awareness associated with fault tolerant flight control. These aspects are not investigated in this thesis.

1.5 Organization of the thesis

This thesis is organized into the following chapters:

- Chapter 1: Introduction
- Chapter 2: MPC based controller matching
- Chapter 3: Model Predictive Control and Feedback Linearization
- Chapter 4: Polytope projection
- Chapter 5: Boeing 747 simulation study
- Chapter 6: Conclusions and Recommendations

Chapter 1 introduces the justification and high level goals of MPC based FTFC and Chapter 2 computes an MPC formulation that (in the nominal case) approxi-

mates an existing linear time-invariant controller as Maciejowski and Jones (2003) show that MPC is suitable as fault tolerant control method, but that initial tuning of the MPC controller is not a straightforward problem. The presented method extends existing literature with a method that allows for replication of an original controller that includes direct output feedback in the form of an MPC controller.

Chapter 3 takes into account the fact that an aircraft has nonlinear dynamics and investigates the combination of nonlinear inversion of the aircraft dynamics and model predictive control. The method requires a computationally efficient projection method for the input constraints of the aircraft as these are affected by the nonlinear inversion. Chapter 4 introduces the aforementioned computationally effective projection method.

Chapter 5 applies the methods of Chapters 2, 3 and 4 to a detailed simulation model of a Boeing 747-100 aircraft that allows for the inclusion of a variety of system and actuator faults.

The thesis concludes with conclusions and recommendations in Chapter 6.

2 CHAPTER

MPC based controller matching

This chapter investigates the qualities of a method for finding both a state-observer and the cost function associated with a model predictive controller based on an already existing linear time invariant output feedback controller. The goal of this exercise is to retain the properties of the existing controller, while adding the constraint handling capabilities of MPC. Consistent satisfaction of constraints is deemed an enabling quality for the application of MPC as a fault-tolerant controller for the aircraft benchmark under consideration.

2.1 Introduction

MPC is one of the few control methods that can actively take constraints into account. Such constraints include input, state, and output constraints. It is hypothesized here that the latter makes MPC especially suitable for FTC purposes, whilst actuator faults can be accommodated for through adaptation of the constraints (Maciejowski 2002b). Additionally, the internal model can be changed to incorporate knowledge of faults that affect the dynamics of the system under control. Furthermore, MPC has a certain degree of fault-tolerance to actuator faults, even if the fault is not detected (Maciejowski 1998).

Although MPC is a serious candidate for FTC purposes in theory, it has been argued by Maciejowski and Jones (2003) that proper tuning of MPC is required in order to construct an MPC problem that has acceptable fault-tolerant properties. In general, however, this chapter will look to replace an existing controller with MPC such that constraint handling properties can be incorporated. The existing controller generally has been tuned to exhibit desired transient response, hence construction of MPC through matching of an existing controller offers a good starting point.

The objective of this chapter is to match an existing linear time-invariant (LTI), possibly dynamic, controller with MPC such as to incorporate the desired con-

straint handling capabilities. These constraint handling capabilities can be of vital importance in case of a failure. More precisely: a controller is sought that will retain transient behavior of the nominal closed loop while input and state constraints permit this. If this is no longer possible, the solution should at least be stabilizing. In doing so the aforementioned burdensome tuning of an MPC cost function is avoided. Furthermore, the influence of different failures on the tunable parameters in the MPC problem will be investigated.

2.2 Problem definition and chapter structure

Starting point of this chapter is the desire to match an existing and accurately tuned controller using MPC. Two different theories available in the literature are discussed that allow us to derive the corresponding cost-function for the MPC problem. The first method obtains the tuning parameters in the cost function through direct computation. The second method takes a general dynamic controller which is subsequently manipulated such that an estimator-regulator form of the original controller is obtained. In Section 2.4 a new method for MPC controller matching for plants with direct output feedback is put forward.

2.3 Controller matching using MPC

This section is limited to the investigation of linear time invariant (LTI) plants and controllers. Nominal systems without actuator failures are studied, incorporation of actuator failures is discussed at the end of the chapter.

The state space form is applied because of the multivariable nature of plant and controller. Also the plant and controller are represented in discrete-time form,

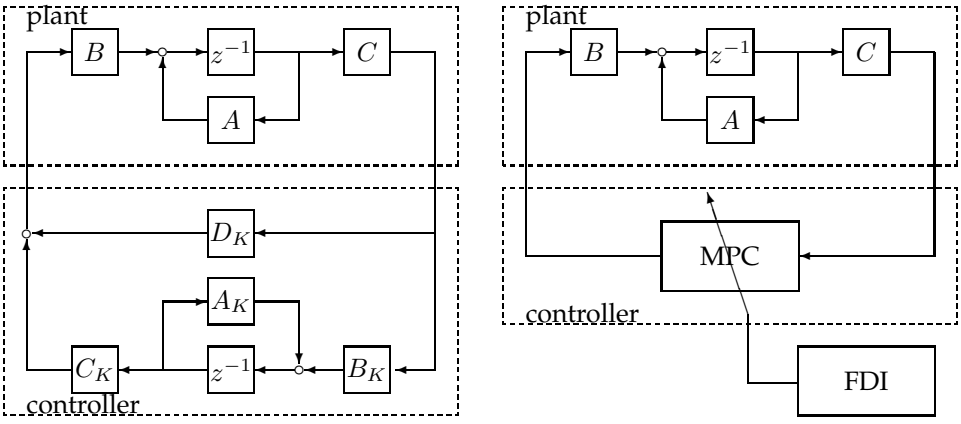

Figure 2.1: Plant and controller; left: linear plant and controller, right: linear plant and model predictive controller

while assuming that this representation has been obtained through discretization of a continuous time plant when necessary. As a starting point it is assumed that the discrete-time representation of the plant is strictly proper and that it has state-space dynamics

$$\begin{bmatrix} x(k+1) \\ y(k) \end{bmatrix} = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \Delta u(k) \end{bmatrix}. \tag{2.1}$$

The original controller can either be static or it may have dynamics. Let the original controller have the realization

$$\begin{bmatrix} x_K(k+1) \\ \Delta u(k) \end{bmatrix} = \begin{bmatrix} A_K & B_K \\ C_K & D_K \end{bmatrix} \begin{bmatrix} x_K(k) \\ y(k) \end{bmatrix} \tag{2.2}$$

or, alternatively, when it has no dynamics, let it be

$$\Delta u(k) = K\, x(k) \tag{2.3}$$

for a state-feedback controller. Figure 2.2 provides a schematic representation of the original plant and controller. Note that in this chapter it is assumed that the reference signal equals zero ($r(k) = 0,\ \forall k$).

As a point of departure for the discussion on controller matching the following papers by Cairano and Bemporad (2010), Maciejowski (2007) and Hartley and Maciejowski (2009) are considered in conjunction with the following cost function

$$J(u,k) = \sum_{j=k}^{k+N-1} \begin{bmatrix} \hat{x}^T(j|k) & \Delta u^T(j|k) \end{bmatrix} \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} \hat{x}(j|k) \\ \Delta u(j|k) \end{bmatrix} \tag{2.4}$$

Note that for a reference signal $r(k) = 0$ the cost signal (1.5), the cost function (1.4) of Chapter 1 can be rewritten as

$$\begin{aligned} J(u,k) &= \sum_{j=k}^{k+N-1} \hat{z}^T(j|k)\hat{z}(j|k) \\ &= \sum_{j=k}^{k+N-1} \Big(C_z\, x(j|k) + D_z\, \Delta u(j|k)\Big)\Big(C_z\, x(j|k) + D_z\, \Delta u(j|k)\Big) \\ &= \sum_{j=k}^{k+N-1} \hat{x}^T(j|k) C_z^T C_z \hat{x}(j|k) + 2\,\hat{x}^T(j|k) C_z^T D_z\, \Delta u(j|k) C_z\, x(j|k) \\ &\quad + \Delta u^T(k+j-1|k) D_z^T D_z \Delta u(j|k) \\ &= \sum_{j=k}^{k+N-1} \hat{x}^T(j|k)\, Q\, \hat{x}(j|k) + 2\,\hat{x}^T(j|k)\, S\, \Delta u(j|k) C_z\, x(j|k) \\ &\quad + \Delta u^T(j|k)\, R\, \Delta u(j|k) \end{aligned}$$

where $Q = C_z^T C_z$, $S = C_z^T D_z$, and $R = D_z^T D_z$.

2.3.1 Direct matching method

Cairano and Bemporad (2010) start their discussion of controller matching, the replication of an existing controller, with the investigation of static controllers. They formulate the MPC matching problem, where the weighting matrices in the cost function must be tuned so that when the constraints are not active, the synthesized MPC feedback law is equivalent to a given linear state-feedback controller. A general solution, based on a bilinear matrix inequality (BMI), is introduced as well as a parameterization of the problem that leads to a linear matrix inequality (LMI) formulation. Two such parameterizations are introduced by Cairano and Bemporad (2010). Finally, the design is extended to dynamic compensators.

Cairano and Bemporad (2010) pose the following problem:

Problem 2.1 *(MPC matching): For a pre-assigned "favorite controller"*

$$u_{fv}(k) = Kx(k),\ K \in \mathbb{R}^{m \times n} \tag{2.5}$$

define the MPC cost function such that the ***unconstrained*** *MPC controller as discussed in Section 1.3 is equal to the favorite controller, that is* $u_{fv} = -E_u H^{-1} \tilde{\mathbf{C}} x(k)$.

Problem 2.1 is immediately solved if

$$-E_u H^{-1} \tilde{\mathbf{C}} x(k) = Kx(k) \tag{2.6}$$

The following solution to problem is posed, first of all E_u is removed from (2.6) setting:

$$H^{-1} \tilde{\mathbf{C}} x(k) = -\begin{bmatrix} \kappa_0 \\ \kappa_1 \\ \vdots \\ \kappa_{N-1} \end{bmatrix} x(k) \tag{2.7}$$

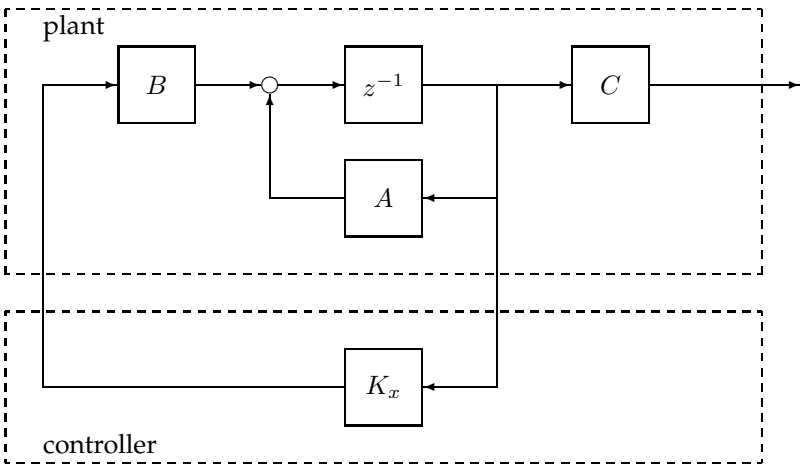

Figure 2.2: Original plant with state-feedback

where $\kappa_0 = K$, and where $\kappa_i \in \mathbb{R}^{m\times m}, i \in \mathbb{Z}_{[1,N-1]}$ are free matrices. Hence (2.7) accounts for the whole optimal input sequence, but a match to the original controller is only enforced for the first control action, which corresponds to the receding horizon mechanism of MPC.

Now define the following matrices for use in the following lemma:

$$\mathcal{T} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad \mathcal{S} = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ A^{N-2}B & & \cdots & B \end{bmatrix}$$

$$\mathcal{Q} = \mathtt{diag}(Q, \cdots, Q), \quad \mathcal{R} = \mathtt{diag}(R, \cdots, R)$$

Lemma 2.1 *(Cairano and Bemporad 2010): Let $(\hat{\mathcal{K}}, \hat{Q}, \hat{R}, \hat{P})$ be any feasible solution of the following problem*

$$\min_{\mathcal{K},Q,R,P} \quad J(\mathcal{K}, Q, R, P) \tag{2.8}$$
$$s.t. \quad Q \geq 0, P \geq 0, R \geq \sigma I \tag{2.9}$$
$$(\mathcal{R} + \mathcal{S}'\mathcal{Q}\mathcal{S})\mathcal{K} + \mathcal{S}'\mathcal{Q}\mathcal{T} = 0 \tag{2.10}$$
$$\kappa_0 = K \tag{2.11}$$

where $\mathcal{K} = [\kappa_0' \dots \kappa_{N-1}']'$, and $J : \mathbb{R}^{Nm\times m} \times \mathbb{R}^{n\times n} \times \mathbb{R}^{m\times m} \times \mathbb{R}^{n\times n} \to \mathbb{R}$ is an arbitrary objective function. Then, the MPC strategy based on the optimal control problem where we set $Q = \hat{Q}$, $P = \hat{P}$, $R = \hat{R}$, solves Problem 2.1.

Proof: see Cairano and Bemporad (2010).

The latter gives rise to a non-convex mathematical problem due to the bilinear constraint (2.10). J remains free to be chosen in this problem, but for obvious reasons, the resulting optimal triplet (Q, R, P) affects the behavior when constraints are active. A possible choice for J is to specify a triplet $(\bar{Q}, \bar{R}, \bar{P})$ of desired weights and set

$$J(\mathcal{K}, Q, R, P) = ||Q - \bar{Q}|| + w_R||R - \bar{R}|| + w_P||P - \bar{P}|| \tag{2.12}$$

where $w_R, w_P \in \mathbb{R}_0+$ and $||.||$ is any matrix norm. The solution that is introduced to construct an LMI problem based on (2.8) is to consider the following convex problem in LMI constraints

$$J^* = \min_{Q,R} ||(\mathcal{R} + \mathcal{S}'\mathcal{Q}\mathcal{S})\bar{K} + \mathcal{S}'\mathcal{Q}\mathcal{T}|| \tag{2.13}$$
$$s.t. \quad P \geq 0, R_i \geq \sigma I, i = 0 \dots N-1 \tag{2.14}$$
$$Q_i \geq 0, i = 1 \dots N-1 \tag{2.15}$$

The second method introduced by Cairano and Bemporad (2010) pertains to match-

ing based on the inverse LQR problem. The latter involves solving

$$\min_{Q,R,P} J(Q,R,P) \tag{2.16}$$

$$s.t. P \geq 0, R \geq \sigma I, Q \geq 0 \tag{2.17}$$

$$P = A'PA + A'PBK + Q \tag{2.18}$$

$$B'PA = -(B'PB + R)K \tag{2.19}$$

where J is convex (e.g. as in (2.12)). Let $\tilde{Q}, \tilde{R}, \tilde{P}$ be any feasible solution (not necessarily the optimal one) of (2.16). Then the MPC strategy based on the optimal control problem (2.5) where we set $Q = \tilde{Q}$,$P = \tilde{P}$, $R = \tilde{R}$, solves Problem 2.1. Cairano and Bemporad (2010) include dynamic controllers by including the dynamics of the controller in the dynamics of the plant and consequently applying the previously introduced theory for static controllers.

2.3.2 Matching observer based realization of controllers

Construction of an MPC problem based on an existing output feedback controller requires two steps in this section. The first step consists of obtaining an observer-based realization of the original controller. The work of Bender and Fowell (1985) and Alazard and Apkarian (1999) provides a method for obtaining a combination of an observer and a state-feedback gain from a given linear time-invariant output feedback controller. This methodology has subsequently been improved upon by Alazard (2002) and Delmond et al. (2006) where an optimal control problem is constructed for which the optimal solution is the existing output feedback controller. A comprehensive overview of the theory of observer based realizations can be found in the recent book by Alazard (2013).

It was proposed by Maciejowski (2007) to use such an observer-based realization as the basis for a second step during which an MPC cost-function is calculated. This methodology has subsequently been investigated further by Hartley and Maciejowski (2009), who explore the trade-offs and design decisions that are involved in this procedure. This section draws heavily upon what has been introduced in the previous two references, while acknowledging that the provided information is tailored with the application example in mind.

Obtaining an observer-based realization of the controller

In order to simplify what is to follow, it is assumed throughout the chapter that both the plant and original controller are represented in discrete time. Whilst both are typically provided in a continuous time representation, it is assumed that both have been discretized before obtaining the observer based realization. As a starting point it is assumed that the discrete-time representation of the plant is strictly proper and that it has state-space dynamics (2.1). Additionally, let the original stabilizing controller have the realization (2.2), and let the order of the plant be n, and let the order of the controller be n_K, respectively. Here, only the situation where

$n_K \leq n$ is considered for that is the situation which will be encountered in the example in this chapter and in Chapter 5, despite the unmodified controller having $n_K > n$. Furthermore, both the plant and controller realization are assumed to be minimal, thus not having any uncontrollable or unobservable modes.

It must be stressed that both the plant and the controller are assumed to be strictly proper. While most physical systems are strictly proper, this does not hold true for controllers which typically include some form of direct feedthrough term $\Delta u(k) = D_K y(k)$. This requirement stems from the fact that it is only possible to construct a controller with direct feedthrough using a combination of a discrete time observer and MPC in a very limited number of cases. It is therefore assumed that the original controller is made proper through direct inclusion of the feedthrough term in the plant model, making the plant model

$$\begin{bmatrix} x(k+1) \\ y(k) \end{bmatrix} = \begin{bmatrix} A + BD_K C & B \\ C & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \Delta u(k) \end{bmatrix}, \tag{2.20}$$

as shown in Figure 2.3 or, alternatively, that the feedthrough term is passed to the plant via a low-pass filter or a unit delay, hence creating extra states in the controller.

Given the controller and plant pair, we search for an observer, in this case having a *predictor* structure, of the form

$$\hat{x}(k+1) = (A - HC)\hat{x}(k) + B\Delta u(k) + Hy(k), \tag{2.21}$$

where H is the observer gain, such that the controller state and the estimated state are related via a transformation T, as in $x_K = T\hat{x}(k)$. The observer and (positive) state-feedback representation of the original controller in this case yield

$$\begin{bmatrix} \hat{x}(k+1) \\ \Delta u(k) \end{bmatrix} = \begin{bmatrix} A - HC + BK & H \\ K & 0 \end{bmatrix} \begin{bmatrix} \hat{x}(k) \\ y(k) \end{bmatrix} \tag{2.22}$$

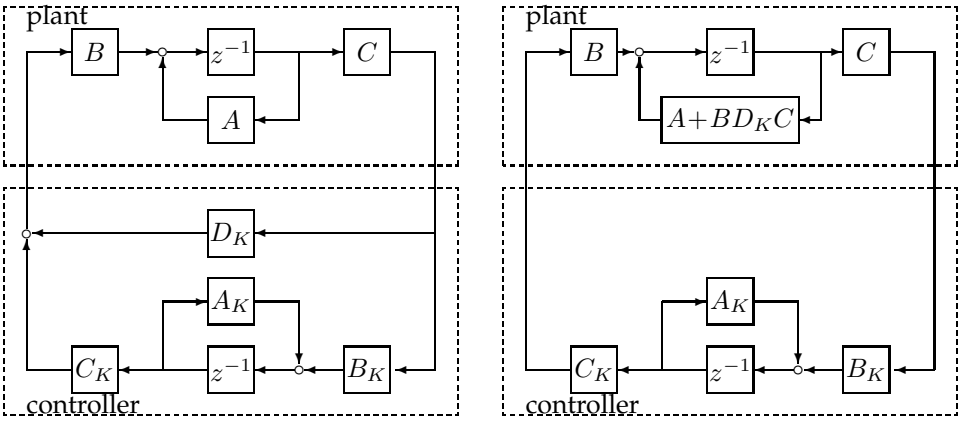

Figure 2.3: Plant and controller; left: linear plant and controller with direct feedthrough, right: linear plant and proper model predictive controller

where K is the state feedback gain. In order to ensure that the original closed loop remains unchanged, both (2.2) and (2.22) must have identical input-output behavior. That, combined with the fact that $x_K = T\hat{x}(k)$ gives rise to the controller and observer gains

$$K = C_K T \tag{2.23}$$
$$H = T^{\dagger} B_K \tag{2.24}$$

where $T^{\dagger}$ is a right inverse of T (see Bender and Fowell (1985) and Alazard and Apkarian (1999) for a complete derivation). At this point it must be remarked that alternative observer formulations exist, e.g. the *filter* estimator, which can be applied in a similar manner (see Hartley and Maciejowski 2009).

Finding T requires solving of the non-symmetric and rectangular Riccati equation

$$\begin{bmatrix} -T & I \end{bmatrix} \underbrace{\begin{bmatrix} A + BD_K C & BC_K \\ B_K C & A_K \end{bmatrix}}_{=A_{cl}} \begin{bmatrix} I \\ T \end{bmatrix} = 0 \tag{2.25}$$

which can be done through application of standard invariant subspace techniques such as the Schur method for solving algebraic Riccati equations, which was introduced by Laub (1979). The Schur decomposition (Golub and van Loan 1996) of a matrix A reads as follows: if $A \in \mathbb{C}^{n+n_K \times n+n_K}$, then there exists a unitary $Q \in \mathbb{C}^{n+n_K \times n+n_K}$ such that

$$Q^H A_{cl} Q = S = D + N \tag{2.26}$$

where $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_{n+n_K})$ and $N \in \mathbb{C}^{n \times n}$ is strictly upper triangular. The Hermitian adjoint Q^H of Q is defined by $Q^H = \bar{Q}^T$, where $\bar{Q}$ is the componentwise conjugate of Q. Furthermore, Q can be chosen such that the eigenvalues λ_i appear in any order along the diagonal of D. Using this result a solution to the Riccati equation can be computed through suitably partitioning of Q as

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \qquad Q_{11} \in \mathbb{R}^{n \times n}, \quad Q_{22} \in \mathbb{R}^{n_K \times n_K} \tag{2.27}$$

which leads to the solution

$$T = Q_{21} Q_{11}^{-1}. \tag{2.28}$$

With respect to this result it must be remarked that, in general, there exist finitely many solutions to the above problem. These solutions correspond to a different choice of eigenvalues from the group of closed loop eigenvalues. The partitioning of the subspaces of A_{cl} determines which poles appear in the state feedback dynamics and which states appear in the observer error dynamics. *Remark*: splitting conjugate pairs of eigenvalues leads to a solution for T that contains complex numbers, and hence to a complex solution for K and H respectively, which is to be avoided.

Under the assumption that indeed $n_K < n$, the observer based controller will be a non-minimal realization of the original controller. Hence, the closed loop of the

plant and observer based controller will contain $n - n_K$ poles that did not appear in the original closed-loop system. The observer modes corresponding to these poles lie in the nullspace of T and are unobservable through K. Furthermore, they are freely assignable through the choice of alternative $T^\dagger$ (see Delmond et al. (2006) or Hartley and Maciejowski (2009) as on how to do so).

Relationship between Cairano and Maciejowski methods

Cairano and Bemporad (2010) try to match a model predictive controller to an existing linear controller and pose their MPC problem as

$$\begin{aligned}
\mathcal{V}(x(k)) &= \min_{U(k)} \sum_{i=0}^{N-1} \begin{bmatrix} x(i|k) \\ u(i|k) \end{bmatrix}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x(i|k) \\ u(i|k) \end{bmatrix} + x^T(N|k) P x(N|k) && (2.29)\\
s.t. \quad & x(i+1|k) = Ax(i|k) + Bu(i|k) && (2.30)\\
& x_{\min} \leq x(i|k) \leq x_{\max} && (2.31)\\
& u_{\min} \leq u(i|k) \leq u_{\max} && (2.32)\\
& x(0|k) = x(k) && (2.33)
\end{aligned}$$

They pose an LMI problem and search for a triplet Q, R, P such that $u(k) = Kx(k)$ for the first time step.

Hartley and Maciejowski (2009) choose another route and pose the MPC problem as follows

$$\begin{aligned}
\mathcal{V}(x(k)) &= \min_{U(k)} \sum_{i=0}^{N-1} \begin{bmatrix} x(i|k) \\ u(i|k) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x(i|k) \\ u(i|k) \end{bmatrix} + x^T(N|k) P x(N|k) && (2.34)\\
s.t. \quad & x(i+1|k) = Ax(i|k) + Bu(i|k) && (2.35)\\
& x_{\min} \leq x(i|k) \leq x_{\max} && (2.36)\\
& u_{\min} \leq u(i|k) \leq u_{\max} && (2.37)\\
& x(0|k) = x(k) && (2.38)
\end{aligned}$$

and set

$$\begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} = \begin{bmatrix} K^T R K & -K^T R \\ -RK & R \end{bmatrix} \tag{2.39}$$

where R can be chosen freely. Through this choice of weighting matrices the optimization problem has an optimum cost of zero and hence P is a zero matrix.

As can be seen, the only difference between both approaches is that Hartley and Maciejowski (2009) allow for off-diagonal entries (S) in the cost function, whilst Cairano and Bemporad (2010) do not. The choice that Hartley and Maciejowski (2009) make immediately leads to $u^* = Kx$ when no constraints are active, whereas Cairano and Bemporad (2010) have to solve a rather involved set of LMIs in order to compute Q, R, P such that the same holds.

Note that the performance index (2.29) in Cairano and Bemporad (2010) is equivalent to performance index (2.34) in Hartley and Maciejowski (2009) for $S = 0$. If S is not zero for the performance index (2.34), it can be transformed in performance index (2.29) by introducing an input-feedback $u = v - R^{-1} S\, x$ and substituting this in (2.34). We obtain:

$$\begin{aligned}
x^T Q\, x + & u^T S x + x^T S^T u + u^T R\, u \\
&= x^T Q\, x + (v^T - x^T S^T R^{-1}) S x \\
&\quad + x^T S^T (v - R^{-1} S\, x) + (v^T - x^T S^T R^{-1})\, R\, (v - R^{-1} S\, x) \\
&= x^T (Q - S^T R^{-1} S - S^T R^{-1} S + S^T R^{-1} S)\, x + v^T S\, x - v^T R\, R^{-1} S\, x \\
&\quad + x^T S^T v - x^T S^T R\, R^{-1} v + v^T R\, v \\
&= x^T \left(Q - S^T R^{-1} S \right) x + v^T R\, v \\
&= x^T \bar{Q}\, x + v^T R\, v
\end{aligned}$$

which result in a new performance index (2.29) where v is the new control variable and $\bar{Q} = Q - S^T R^{-1} S$.

MPC formulation

Now that an estimator form of the original controller has been obtained it is possible to replace the state-feedback with a predictive controller such that its optimal solution is equal to the state feedback law $u(k) = K\hat{x}(k)$. A candidate cost-function that corresponds to this requirement is

$$\begin{aligned}
& \textstyle\sum_{k=0}^{\infty} \quad (u(k) - K\hat{x}(k))^T R (u(k) - K\hat{x}(k)) \\
= \; & \textstyle\sum_{k=0}^{\infty} \quad \underbrace{\begin{bmatrix} x^T(k) & u^T(k) \end{bmatrix} \begin{bmatrix} K^T R K & -K^T R \\ -RK & R \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}}_{=l(k)}
\end{aligned} \tag{2.40}$$

for any input weighting matrix $R = R^T > 0$ (Kreindler and Jameson 1972).

Typically MPC implementations perform an optimization over a finite but receding horizon. A infinite horizon implementation can be obtained by using the candidate cost-function over a finite horizon of length N and using the discrete-time algebraic Riccati equation P as terminal cost. The finite horizon cost function is then

$$\left(\sum_{k=0}^{N-1} l(k) \right) + x^T(N) P x(N) \tag{2.41}$$

although for N is large enough, the terminal cost is negligible. Additionally, since the cost-function represents a quadratic programming problem, linear and ellipsoidal constraints may be added to the controller without having to sacrifice the fact that minimization over these cost-functions leads to a globally optimal solution. These constraints are enforced over the finite prediction horizon N.

The cost-functions (2.40,2.41) presented above, have a very intuitive physical interpretation, i.e. in the absence of constraints the input will match the input that would have resulted from the state-feedback gain K. However, in the presence of constraints it is not at all guaranteed that the properties of the original controller are maintained. Furthermore, it might be more intuitive to opt for the use of other objectives in the cost-function. Especially in failure cases it may be beneficial to minimize the difference between the predicted state trajectory and the state trajectory that would have resulted from the state-feedback gain. The latter opens up the possibility to achieve the same control goals whilst using different and redundant actuators in case of an actuator failure.

2.4 MPC for controllers with direct feedthrough matrix

This section presents a solution to the limitation that the observer based realization of an existing LTI controller does not include direct output feedback. Direct output feedback always remains a separate element in parallel to the observer and state-feedback. It is this fact that makes inclusion of the direct output feedback into the final MPC problem more difficult, as the MPC problem takes the state estimate $\hat{x}(k)$ as its input. Common solutions are to include this term into the plant dynamics prior to MPC design, or to include delay terms between the plant input and the direct feedback term. The first solution makes satisfaction of the input constraints complex and the other changes the dynamics of the closed loop.

In order to arrive at this solution the Youla parameterization of an LTI controller is used as starting point. From Figure 2.4 one can see that this parameterization can include direct output feedback in the form of the term D_e, whereas the remainder constitutes the combination of an observer and state-feedback gain.

From Theorem 1.1 in Section 1.3 we learn that the infinite horizon control law is given by

$$\Delta u(k) = -K\, x_c(k|k) + D_e\, \hat{e}(k|k)$$

where

$$\hat{e}(k|k) = y(k) - C\, x_c(k|k)$$

Substitution in the controller gives the following form:

$$\begin{aligned} x_c(k+1) &= (A - BK - HC - BD_eC)\, x_c(k|k) + (H + BD_e)\, y(k) \\ v(k) &= (-K - D_eC)\, x_c(k|k) + D_e\, y(k) \end{aligned}$$

Let an arbitrary LTI controller be given by

$$\begin{aligned} x'_c(k+1) &= A_K\, x'_c(k|k) + B_K\, y(k) \\ v(k) &= C_K\, x'_c(k|k) + D_K\, y(k) \end{aligned}$$

where x_c' is the state of the LTI controller. Now there must exist a state transformation such that $x_c'(k) = T\,x_c(k)$ and

$$\begin{aligned} TA_K &= (A - BK - HC - BD_eC)T \\ TB_K &= H + BD_e \\ C_K &= (-K - D_eC)T \\ D_K &= D_e \end{aligned}$$

From the second and third equation we derive with $H = TB_K - BD_e$ and $KT = -C_K - D_eCT$ and we find

$$\begin{aligned} TA_K - &(A - BK - HC - BD_eC)T = \\ &= TA_K - AT + B(-C_K - D_eCT) + (TB_K - BD_e)CT + BD_eCT \\ &= TA_K - AT - BC_K - BD_eCT + TB_KCT - BD_eCT + BD_eCT \\ &= TA_K - (A + BD_eC)T - BC_K + TB_KCT \\ &= \begin{bmatrix} T & I \end{bmatrix} \begin{bmatrix} A_K & -B_KC \\ -BC_K & A + BD_eC \end{bmatrix} \begin{bmatrix} I \\ -T \end{bmatrix} \\ &= \begin{bmatrix} T & I \end{bmatrix} A_{cl} \begin{bmatrix} I \\ -T \end{bmatrix} \end{aligned}$$

The transformation matrix T can be found using the method presented in Section 2.3.2, Equation 2.25 onward. With the transformation matrix T we can derive the

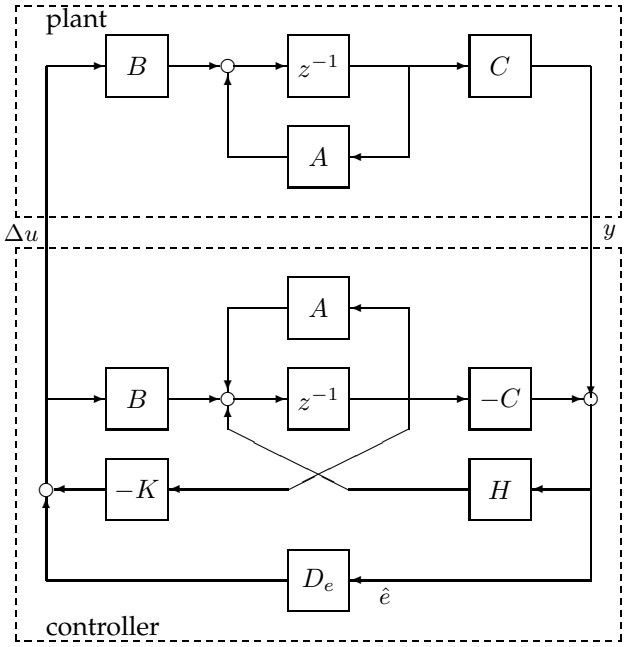

Figure 2.4: Youla parameterization, observer and state feedback

state feedback K and output feedback H as follows:

$$H = TB_K - BD_K$$
$$K = -C_K T^{-1} - D_K C$$

or

$$K = -C_K T^{\dagger} - D_K C$$

where $T^{\dagger}$ is the Moore-Penrose pseudo inverse of T in case the controller order is smaller than that of the palnt $n_K < n$.

controller matching

Now that an estimator form of the original controller has been obtained it is possible to replace the state-feedback with a predictive controller such that its optimal solution is equal to the state feedback law

$$\Delta u(k) = -K\, x_c(k|k) + D_e\, \hat{e}(k|k) \quad .$$

A possible candidate cost-function that corresponds to this requirement is

$$\begin{aligned}
& \textstyle\sum_{k=0}^{\infty} \quad (\Delta u(k) + K\, x(k|k) - D_e\, e(k))^T R (\Delta u(k) + K\, x(k) - D_e\, e(k)) \\
= \; & \textstyle\sum_{k=0}^{\infty} \quad \begin{bmatrix} x^T(k) & e^T(k) & \Delta u^T(k) \end{bmatrix} \begin{bmatrix} K^T R K & -K^T R D_e & K^T R \\ -D_e^T R K & D_e^T R D_e & D_e^T \\ R K & R D_e & R \end{bmatrix} \begin{bmatrix} x(k) \\ e(k) \\ \Delta u(k) \end{bmatrix} \\
= \; & \textstyle\sum_{k=0}^{\infty} \quad z^T(k)\, z(k) \\
= \; & \textstyle\sum_{k=0}^{\infty} \quad \begin{bmatrix} x^T(k) & e^T(k) & \Delta u^T(k) \end{bmatrix} \begin{bmatrix} C_z^T C_z & C_z^T E_z & C_z^T D_z \\ E_z^T C_z & E_z^T E_z & E_z^T D_z \\ D_z^T C_z & D_z^T E_z & D_z^T D_z \end{bmatrix} \begin{bmatrix} x(k) \\ e(k) \\ \Delta u(k) \end{bmatrix}
\end{aligned}$$

This gives us:

$$\begin{bmatrix} C_z & E_z & D_z \end{bmatrix} = R^{1/2} \begin{bmatrix} F & -D_e & I \end{bmatrix} \tag{2.42}$$

for any input weighting matrix $R = R^T > 0$.

Note that the matrix R in equation 2.42 can be chosen freely. It is important to scale the input signals in an adequate way (see Bryson 1975, p.149). Let the required range of the i-th input be given by e given by r_i, so $|\, u_i \,| \leq r_i$, for $i = 1, \ldots, p$. Then a scaling matrix can be given by

$$R = \text{diag}(r_1^{-2}, r_2^{-2}, \ldots, r_p^{-2})$$

The variables $r_i^{-2} > 0$ are a measure of how much the costs should increase if the i-th input $u_i(k)$ increases by 1.

Design considerations

Several choices have to be made in the process that has been outlined in the previous sections. Some of these have been addressed already, such as the choice to discretize both the plant and controller before starting the reverse-engineering process, but others have yet to be mentioned. This section is all but an exhaustive discussion of these design considerations and the reader is referred to Hartley and Maciejowski (2009) and Alazard (2013) for an in-depth discussion of such issues. The properties mentioned here, represent those that are of importance in the discussion of the simulation example in this Chapter.

It was already mentioned in Section 2.3.2 that it is of importance that the controller is strictly proper. The two options that were mentioned for situations where this is not true consisted of adding the feedthrough term D_K to the plant directly, through *loop-shifting*, or alternatively, of adding a filter or unit delay on the output of the controller. The latter results in additional states in the controller, which may have a negative influence on the stability margins of the closed-loop. On the other hand, this does allow to satisfy hard input constraints, whereas loop-shifting does not, due to the fact that the observer error comes into play when making predictions of the output over the horizon.

Additionally, when $n_K \leq n$, the process of designing a state-feedback and observer realization of the original controller leads to two distinct problems involving poles. First of all, one has to distribute the closed loop eigenvalues over the state feedback dynamics $(A + BK)$ and the observer error dynamics $(A - HC)$. Although all ordering of eigenvalues among the two sets gives the same closed-loop behavior (in discrete-time), it appears to be a wise choice to allocate the fast closed-loop poles to the observer poles. The reason for doing so is that the model-predictive controller relies on the estimated state $\hat{x}$ and therefore it is of great importance that the quality of the estimated states is high. The other design choice related to pole locations stems from the fact that when $n_K \leq n$, $n - n_K$ free poles appear in the observer realization. These can be placed freely whilst the corresponding observer error dynamics lie in the null-space of T, but they come into play as soon as one of the constraints becomes active.

2.5 Simulation Example

This example is based on data presented by van Keulen (1991) on the real-time simulation and analysis of the automatic flight control system of the Boeing 747-200. The aforementioned reference is based on data presented by Hanke and Nordwall (1970); Hanke (1970) which links this example to the full nonlinear aircraft model and simulations in Chapter 5. In the thesis van Keulen (1991) presents a (continuous time) linearization of the symmetric equations of motion of the 747 aircraft in combination with descriptions of the 747 autopilot system. The state-space matri-

ces below are for cruise conditions ($V = 423$ ft/sec and $h = 5000$ ft).

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0.286 & -0.572 & 0 & 0 \\ -0.00947 & -0.529 & 0 & 0.967 & -0.0383 \\ 0 & 0 & 0 & 1.0 & 0 \\ 0.00239 & -1.22 & 0 & -0.631 & -1.65 \\ \hline 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 & 0 \end{array}\right]$$

the states of this system are, u (ft), α (deg), θ (deg), q (deg/sec) and the input is the elevator δ_e. A full overview of the inner loop of the autopilot is presented in Figure 5.6, which can be simplified to the following transfer functions

$$\delta_e = -3.32(\theta_{ref} - \theta) - \frac{294s}{(s+20)(s+10)(s+0.5)}q$$

The discrete time state-space form (sample time $T_s = 0.1s$) of the plant is

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{cccc|c} 1.0 & 0.0279 & -0.0572 & -0.00147 & 2.73 \cdot 10^{-5} \\ -9.09 \cdot 10^{-4} & 0.943 & 2.64 \cdot 10^{-5} & 0.0911 & -0.0114 \\ 1.36 \cdot 10^{-5} & -0.00587 & 1.0 & 0.0967 & -0.00809 \\ 2.87 \cdot 10^{-4} & -0.115 & -7.76 \cdot 10^{-6} & 0.933 & -0.16 \\ \hline 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 & 0 \end{array}\right]$$

and the discrete time state-space representation of the controller is

$$\left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right] = \left[\begin{array}{ccc|cc} 0.135 & -0.560 & 0.705 & 0 & 0 \\ 0 & 0.368 & 0.795 & 0 & 0 \\ 0 & 0 & 0.951 & 0 & 1.0 \\ \hline 0.406 & -0.457 & 0.575 & 3.320 & 0 \end{array}\right]$$

Application of the theory in this Chapter and Section 2.4 in particular leads to the following matrices for the infinite MPC problem described in Section 1.3

$$K = \left[\begin{array}{cccc} 134.0 & 130.0 & -154.0 & -9.91 \end{array}\right]$$

$$H = \left[\begin{array}{cc} -9.06 \cdot 10^{-5} & 0.00737 \\ 0.0379 & 0.101 \\ 0.0268 & 0.09 \\ 0.531 & 0.114 \end{array}\right]$$

$$D_e = \left[\begin{array}{cc} 3.32 & 0 \end{array}\right]$$

Simulations were made comparing the closed system for three different controllers:

1. the original controller;
2. an infinite horizon MPC controller;
3. an finite horizon MPC controller (horizon N=5).

The Figures 2.5 through 2.7 show the results of the simulation when the model is initiated in an off-equilibrium condition (pitch angle $\theta_0 = 0.15$ [rad]. What is shown are the elevator input (Fig 2.5), the state-response (for the pitch angle θ only, Fig. 2.6) and the difference (error) between the pitch rate response for the three controllers 2.7). What can be seen from the figures is that the matching procedure leads to a controller with behavior that is identical to the original controller. All controllers show stabilizing behavior as the pitch angle θ returns to zero.

2.6 Conclusions

This chapter has introduced methods for obtaining a state observer in combination with a model-predictive controller. Based on a linear time-invariant representation of both the existing autopilot and the aircraft it is possible to arrive at such a controller structure. When the original controller contains both a direct feedthrough term and integral action, this goal cannot be achieved without the necessary caution. Section 2.4 presents a novel way to do so. In the absence of input constraints this controller shows tracking performance that is on par with the original output feedback controller without requiring extensive tuning of the cost-function weighting matrices and quantities such as the prediction horizon.

In the presence of constraints, however, the performance of the reverse-engineered controller cannot always be guaranteed, especially when one or more constraints become active. Whilst the desire to introduce constrained control to the aircraft benchmark is a driving force in this work, it is deemed very instructive to investigate different cost-function formulations for the MPC problem. Specifically cost-functions that do not weigh the difference between the predicted state-feedback quantities and the input are assumed to be of great value. Methods that weigh the difference between the predicted plant state and the predicted closed loop dynamics are thought to be more than worth the investigation. The latter can offer better ways to make use of the available redundancy in the sense of actuators. Additionally, further future work on the applied aircraft model should also include an exhaustive investigation of the different ways in which the closed-loop poles may be distributed among the observer error-dynamics and the state-feedback dynamics, as well as where to place the free poles in the controller.

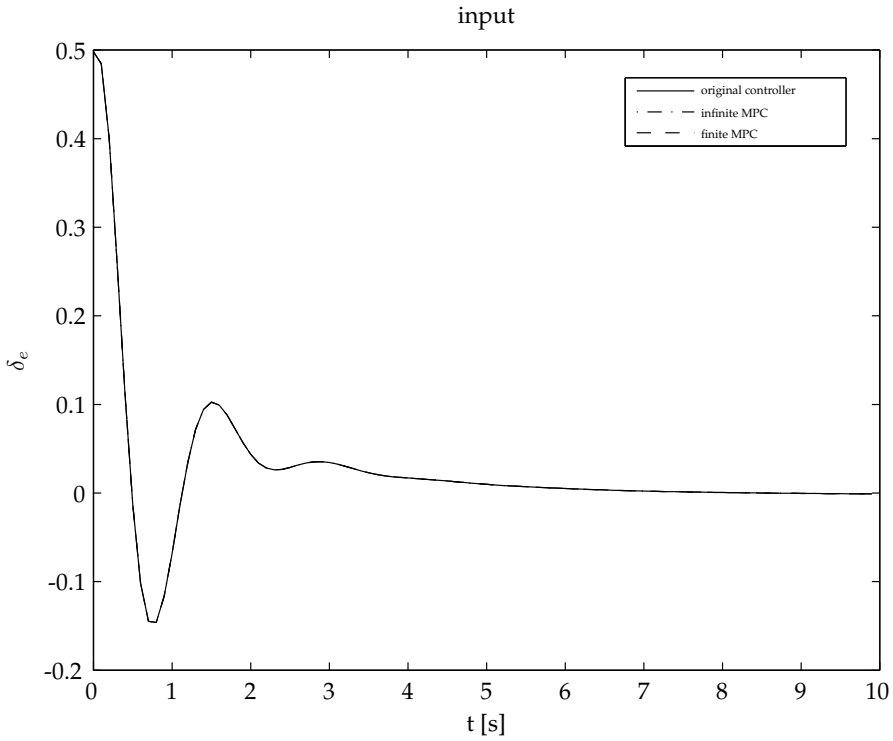

Figure 2.5: Comparison of the elevator input δ_e for a) the combination of the observer based realization of the original controller plus infinite horizon MPC, and b) the same for finite horizon MPC, and c) of the original controller for the example in Section 2.5.

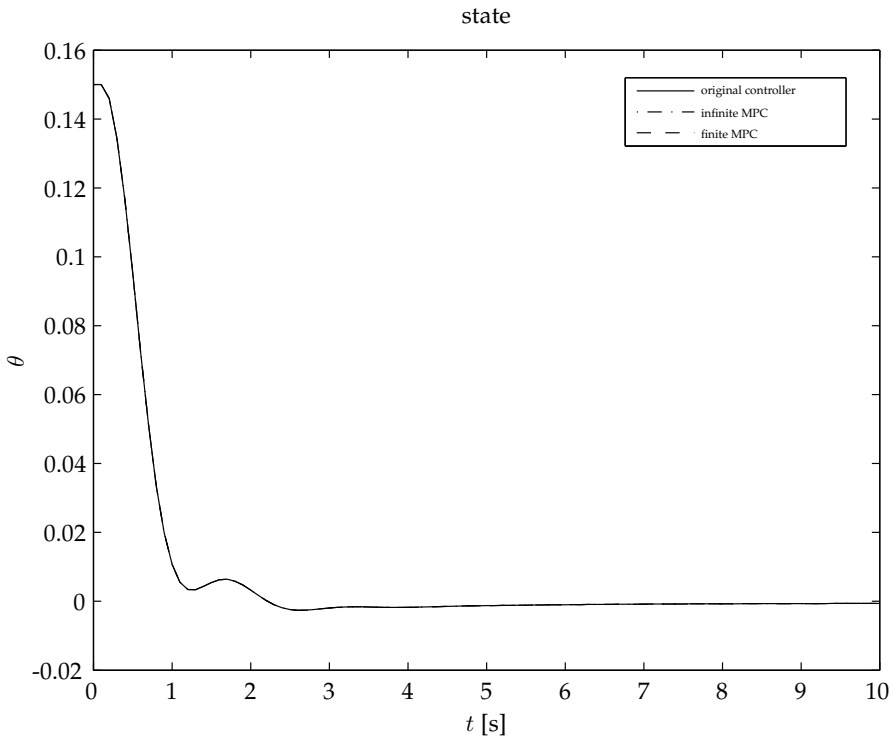

Figure 2.6: Comparison of the state behavior θ (pitch angle) a) the combination of the observer based realization of the original controller plus infinite horizon MPC, and b) the same for finite horizon MPC, and c) of the original controller for the example in Section 2.5.

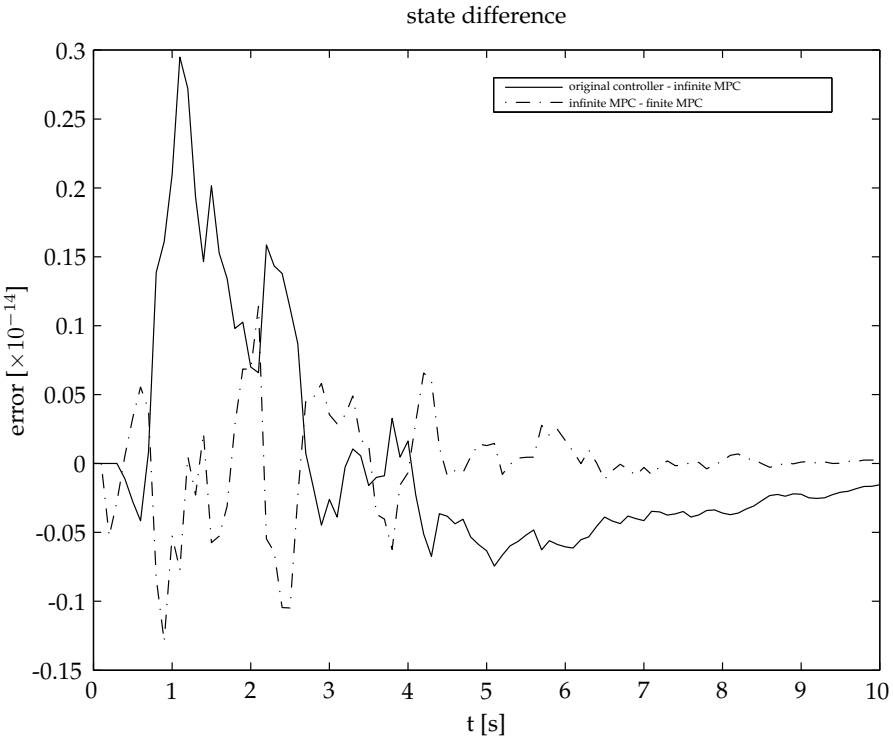

Figure 2.7: Comparison of the error when comparing the state (pitch angle θ) behavior for the example in Section 2.5. Two errors are shown, a) the difference between the original controller and the infinite horizon MPC controller based on reverse-engineering, and b) the difference between the finite horizon and the infinite horizon implementation of the aforementioned MPC controller.

3 CHAPTER

Model Predictive Control and Feedback Linearization

This chapter features the combination of model-based predictive control and the inversion of the dynamics of the system under control into a constrained and globally valid control method for fault-tolerant flight-control purposes. The fact that the approach allows for the incorporation of constraints creates the possibility to incorporate additional constraints in case of a failure. Such failures range from relatively straightforward actuator failures to more complicated structural breakdowns where, through the addition of constraints, the aircraft can be kept within its remaining flight envelope. Furthermore, the method is model-based, which allows modification of the system model in case of a failure. Both of these properties lead to the fault-tolerant qualities of the method presented.

3.1 Introduction

Starting point of this chapter is again the assumption that model predictive control (MPC) is well-suited to the needs of a reconfigurable control method. The latter is also concluded by Jones (2005) where MPC is compared to several other control methods that are deemed suitable. The previous statement is motivated through inspection of the following properties of MPC: as a control strategy MPC is based on online optimization that applies a model of the system under control, which means that the internal model may be changed in between the time steps of the optimization algorithm; furthermore, MPC is a constrained control method which means that actuator failures, like jammed control surfaces can relatively easily be incorporated and hence, accommodated for; and finally, MPC inherently incorporates a control allocation method, which indicates that it is also possible to give preference to the use of certain actuators in order to perform a maneuver. The multi-variable setting is natural to MPC, hence strengthening the motivation

of its suitability as a reconfigurable control method. Therefore, MPC has a definite advantage over many other control methods. This advantage does come at a cost however, for MPC requires an optimization problem to be solved at every time step. Furthermore, a proof of closed-loop stability is more difficult to provide.

In the particular case of reconfigurable flight control, which is what we strive for, we have to take into account that the system under control has nonlinear dynamics. More specifically, aircraft have nonlinear kinematics, as well as nonlinear aerodynamics. We would like to take this knowledge into account whilst this can be of importance in the case of a failure. In modeling dynamic systems, one can use systems structures that vary in complexity. Linear time invariant systems are amongst the most simple implementations of a dynamic model, but these are only valid in a relatively small region around the chosen equilibrium point. A fully nonlinear model, based on first principles modeling, offers a far more precise representation of the physical system, but these models can be very complex. In this chapter we will investigate the use of a nonlinear system model in combination with MPC. We assume that nonlinearities in the dynamics of the aircraft may become of special importance once a failure has been introduced to the system.

The theory of MPC, however, is not very well developed where the control of nonlinear systems is concerned. It may be concluded from different surveys on MPC (Mayne et al. 2000; Bemporad and Morari 1999; Qin and Badgwell 2003; Maciejowski 2002b) that MPC is well-suited to LTI systems. The main reason for this lies in the structure of the optimization problem that forms the basis of MPC. MPC in combination with an LTI system model will lead to a convex programming problem, whereas a nonlinear system model will only lead to a convex programming problem in very specific cases. Hence, no guarantees for the convergence of the optimization problem can be given in cases where a nonlinear system model is used. It is for this reason that it is deemed worthwhile to combine MPC with a nonlinear control method in order to obtain a reconfigurable flight control system. Dynamic inversion is such a method. It allows for the inversion of the nonlinear kinematics of the aircraft such that linear and time-invariant behavior is obtained. This linear behavior can be controlled with one of the commonly available MPC algorithms. Some special measures are needed though, because of the interconnection between both control methods and constraints.

The combination of MPC and NDI into one controller is not new. An example of the combination of MPC and feedback linearization (FBL), which is a more strict variation upon NDI, in order to obtain globally valid and constrained control for the flight of a re-entry vehicle is to be found in van Soest et al. (2006), and the combination of robust MPC and feedback linearization is evaluated in van den Boom (1997). The theory presented in this chapter differs from existing literature in two aspects; the first of which is that the combination of NDI and MPC is not only applied as a form of globally valid and constrained nonlinear control, but also as a reconfigurable method; the second difference lies in the fact that it is assumed here that the system has control effector redundancy in the nominal and fault-free case, i.e. that it is over-actuated. The latter is not the case in the previously mentioned

references. Next to these, Kale and Chipperfield (2005) provide an application of robust MPC to achieve reconfigurable behavior, linear subspace identification and predictive control are synthesized into one in Hallouzi and Verhaegen (2008), NDI and online identification of the aerodynamic derivatives of the aircraft are combined in Lombaerts et al. (2009). An example that considers the use of MPC, without NDI, in a simulation of the Bijlmermeer accident scenario is to be found in Maciejowski and Jones (2003).

In order to tackle the previously mentioned challenges, we introduce the suggested synthesis between model-predictive control (MPC) and a nonlinear dynamic inversion method (NDI) in the following sections. Section 3.2 provides the motivation for this setup, and furthermore, the section provides a clear introduction as to how both methods interact. Section 3.2.2 and 3.2.3 provide a discussion of the theory of MPC and dynamic inversion, whereas Section 3.2.5 introduces control allocation, and Section 3.2.4 is on the mapping of constraints, together providing the theory that is required to make the proposed combination of MPC and dynamic inversion interact correctly. Sections 3.3 and 3.4 conclude the chapter and provide a simulation example and summary and conclusion.

3.2 Overall Control-Setup

Figure 3.1 provides an overview of how MPC and NDI are combined in this chapter. The concept of a combination between NDI and MPC such as to form a reconfigurable, globally valid, nonlinear, and constrained controller seems intuitive, but there are several interconnection issues that require attention. Such issues are caused by the fact that the number of system inputs is in general much larger than the number of states that are to be controlled, which is actually a prerequisite for FTFC. The latter forces us to include control allocation in between the NDI block and the aircraft. This will be elaborated upon in Section 3.2.5. Furthermore, it is not a priori clear how the constraints on the inputs relate to the constraints of the MPC controller.

Subsection 3.2.1 introduces the model structure and section 3.2.2 introduces dynamic inversion. The next subsection provides the details of the MPC strategy that has been applied. Finally, subsection 3.2.5 provides details on how to distribute the desired control effort over the physical inputs.

For reasons of clarity, several assumptions, mainly because of simplicity, are posed here that hold throughout the entire chapter. It is assumed that new system parameters will become available, e.g. through online identification of the aerodynamic parameters based on the work presented by Lombaerts et al. (2009). Other assumptions that are made are that full-state information is to be available, and more importantly, we assume that there are redundant control effectors, such that these can be applied in case a primary actuator fails. Finally, it is such that this method is best suited for failures of actuators/control surfaces and structural failures of the airframe. Sensor failures are not considered here.

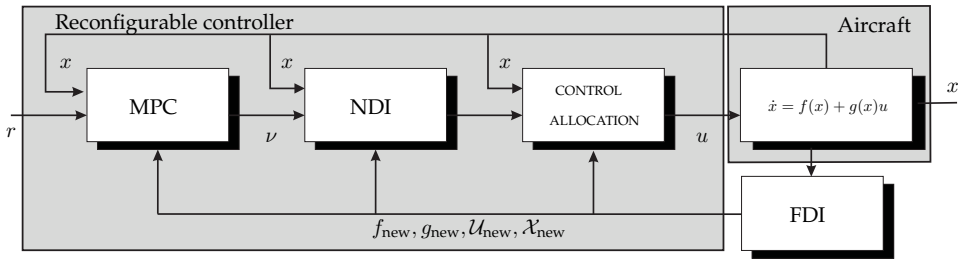

Figure 3.1: Overview of the complete FTFC loop and the individual components. Additionally, the FDI block is shown to stress the importance of a failure detection method that delivers a new system description and a new set of constraints after the introduction of a failure.

3.2.1 Model Structure

This section starts with an introduction of the system-type that is considered and continues to present the aspects that are involved in the combination of feedback linearization and model predictive control. In this chapter we consider nonlinear discrete-time systems, that are modeled to be affine in the input, like

$$x(k+1) = f(x(k)) + g(x(k))\Delta u(k) \tag{3.1}$$
$$y(k) = h(x(k)) \tag{3.2}$$

where $x(k) \in \mathbb{R}^n$ is the state vector, where $u(k) \in \mathbb{R}^m$ is the vector of inputs, and where k indicates that this system is a discrete-time system with sampling-interval T. Furthermore, $f(x) \in \mathbb{R}^{n\times 1}$, $g(x) \in \mathbb{R}^{n\times m}$. Both the input $u \in \mathcal{U}$ and $x \in \mathcal{X}$ belong to a polyhedral set, i.e. they can be written as

$$\mathcal{U} = \{\Delta u \in \mathbb{R}^m | \, A_u \, \Delta u \leq b_u\}, \tag{3.3}$$
$$\mathcal{X} = \{x \in \mathbb{R}^n | \, A_x \, x \leq b_x\} \tag{3.4}$$

for some matrices A_u, A_x and vectors b_u, b_x. Furthermore, it is assumed that the output $y(k) = x(k)$, such that $h(x(k)) = x(k)$.

It must be remarked that it is also possible to apply FBL to the system in continuous time. This, however, leads to issues with respect to the control allocation problem such as depicted in Figure 3.1. The control allocation will consist of a constrained quadratic programming problem and will necessarily be performed in discrete-time. It is therefore more logical to perform all steps in discrete-time, and as such, to discretize the nonlinear system before applying FBL.

3.2.2 Nonlinear Dynamic Inversion

Feedback linearization is a control method that will obtain linear and decoupled input-output behavior through application of a static and nonlinear feedback law.

Aspects like relative degree, partial feedback linearization and uncontrollable internal dynamics are important issues within the standard framework of feedback linearization as presented by Isidori (1995); Slotine and Li (1991). Feedback linearization in its most basic form, (partial) input-state linearization, is what is applied here. Input-state linearization avoids the aforementioned issues to some extent. The presented implementation applies the concept of a virtual input and hence allows for the use of the available control effector redundancy in a further step, whereas FBL in its purest form does not.

It is necessary to include dummy outputs in 3.1 for input-state linearization when $m \geq n$ in order to be able to apply FBL, since u and y, or x in this particular case, are required to be sized equally. Alternatively, it is possible to introduce a virtual input $w(x(k), \Delta u(k)) = g(x(k))\Delta u(k),\ z \in \mathbb{R}^n$ and to split up the problem of input-state, or possibly partial state, linearization and control allocation, such that

$$x(k+1) = f(x(k)) + w(x(k), \Delta u(k)), \tag{3.5}$$

where $z(x(k), u(k))$ is assumed to be a virtual input of the system that can be used for linearization purposes. This relation between $w(x(k), \Delta u(k))$ and $\Delta u(k)$, and how to make use of the freedom therein, is the topic of Section 3.2.5 on control allocation.

It is clear to see that in order to invert the nonlinear dynamics, a choice of

$$w(k) = g(x(k))\Delta u(k) = -f(x(k)) + \nu(k), \tag{3.6}$$

will result in decoupled closed-loop behavior that equals

$$x(k+1) = \nu(k), \tag{3.7}$$

where $\nu(k) \in \mathbb{R}^n$ is a new input to the inverted system. The latter equation shows that the chosen control law decouples the system, such that the closed-loop constitutes a series of integrators in parallel, but optionally, through proper selection of $z(k)$ the linear dynamics can incorporate some desired dynamics such that $x(k+1) = A_{des}x(k) + \nu(k)$. Furthermore, it is clear to see that when the number of inputs m is smaller than the number of states n, it will be impossible to invert the entire dynamics. When $m = n$ there will exist a unique solution to Equation (3.6) and when $m > n$ then there will exist a whole set of solutions $\Delta u(k)$ to this equation. It is necessary to make the remark that it is assumed in this chapter that $m > n$, and hence input redundancy exists. Therefore, the input $u(k)$ will have to be allocated at every discrete-time step. The latter is commonly called nonlinear dynamic inversion (NDI) instead of FBL.

In summary, the input-state linearization that is presented in this section leads to LTI behavior that relates $\nu(k)$ to $x(k)$, and retains freedom in the allocation of $\Delta u(k)$. A restrictive result of the above is that the original input constraints on $\Delta u(k)$ must now be mapped into constraints on ν, since $\nu(k)$ will be controlled using model predictive control (see Figure 3.1). The next section will introduce an MPC algorithm that has been tailored to this situation, such that this issue can be avoided to a large degree.

3.2.3 Model predictive control

Now that a linear discrete-time system (3.7) has been obtained through NDI, it is straightforward to apply model predictive control (MPC). MPC makes use of a system model that provides a description of the dynamic behavior of a system. Linear MPC typically uses linear time-invariant (LTI) systems in state-space form

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) && (3.8)\\ y(k) &= Cx(k) && (3.9) \end{aligned}$$

For reasons of simplicity we do not take the noise signal $e(k)$ into account in this Chapter.

We choose to use the objective function, where the prediction horizon is chosen equal to N, as follows

$$\begin{aligned} J(u,k) = &\sum_{i=N_{\mathrm{m}}}^{N-1} \left(\hat{x}(k+i|k) - x_r(k+i)\right)^T Q \left(\hat{x}(k+i|k) - x_r(k+i)\right)^T + \\ &\sum_{i=1}^{N-1} \Delta u(k+i-1|k)^T R \Delta u(k+i-1|k) \end{aligned} \tag{3.10}$$

where $\hat{x}(k+i|k)$ is the predicted value of $x(k+i)$ at time k. $x_r(k) \in \mathbb{R}^n$ is the reference state and $Q \succeq 0, R \succ 0$ are a state weighting matrix and an input weighting matrix respectively. Note that this objective function corresponds to a performance signal

$$\begin{aligned} z(k) &= \begin{bmatrix} Q^{1/2}(x(k+1) - x_r(k+1)) \\ R^{1/2}\Delta u(k) \end{bmatrix} \\ &= \begin{bmatrix} Q^{1/2}A \\ 0 \end{bmatrix} x(k) + \begin{bmatrix} Q^{1/2}B \\ R^{1/2} \end{bmatrix} \Delta u(k) \\ &\quad + \begin{bmatrix} -Q^{1/2}x_r(k+1) \\ 0 \end{bmatrix} r(k) \end{aligned} \tag{3.11}$$

If we introduce the following variables

$$\tilde{x} = \begin{bmatrix} x(k+1|k) \\ x(k+2|k) \\ \vdots \\ x(k+N|k) \end{bmatrix}, \qquad \tilde{x}_r = \begin{bmatrix} x_r(k+1|k) \\ x_r(k+2|k) \\ \vdots \\ x_r(k+N|k) \end{bmatrix}$$

$$\tilde{u} = \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+N-1|k) \end{bmatrix}, \qquad \tilde{\nu} = \begin{bmatrix} \nu(k|k) \\ \nu(k+1|k)_r \\ \vdots \\ \nu(k+N-1|k)_r \end{bmatrix}$$

and

$$\begin{aligned} \tilde{Q} &= I_N \otimes Q, \\ \tilde{R} &= \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix} \otimes R \end{aligned} \tag{3.12}$$

where I_N is an identity matrix of size N, and where the operator $\otimes$ indicates the Kronecker product of two matrices. The Kronecker product of two matrices A and B is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}$$

where a_{ij} is the i,j-th entry of matrix $A \in \mathbb{R}^{m \times n}$. Now, using relationship (3.7) the above objective function (3.10) can be expanded into

$$\begin{aligned} J(\nu(k)) &= (\tilde{x} - \tilde{x}_r)^T \tilde{Q} (\tilde{x} - \tilde{x}_r) + \tilde{\mathbf{u}}^T R \tilde{\mathbf{u}} \\ &= (\tilde{\nu} - \tilde{x}_r)^T \tilde{Q} (\tilde{\nu} - \tilde{x}_r) + \tilde{\mathbf{u}}^T R \tilde{\mathbf{u}} \\ &= \tilde{\nu}^T \tilde{Q} \tilde{\nu} - 2\tilde{x}_r^T \tilde{Q} \tilde{\nu} - 2\tilde{x}_r^T \tilde{Q} \tilde{x}_r + \tilde{\mathbf{u}}^T R \tilde{\mathbf{u}}. \end{aligned} \tag{3.13}$$

In order to be able to take into account the constraints on the physical input $u(k)$ it is necessary to incorporate Equation (3.6) which denotes the relationship between $\nu(k)$ and $u(k)$ and the constraints on input $u(k)$ as in (3.3). Both of these can be expanded over the horizon as follows

$$\begin{aligned} \underbrace{\begin{bmatrix} g(x(k)) & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & g(x(k+N-1)) \end{bmatrix}}_{=\tilde{C}(x)} \tilde{u}(k) &= \\ \underbrace{\begin{bmatrix} -f(x(k)) \\ -f(x(k+1)) \\ \vdots \\ -f(x(k+N-1)) \end{bmatrix}}_{=\tilde{b}_{eq}(x)} &+ \tilde{\nu}(k) \end{aligned} \tag{3.14}$$

and

$$\underbrace{(I_N \otimes A_u)}_{=\tilde{A}_u} \tilde{u}(k) \leq \underbrace{\begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T \otimes b_u}_{=\tilde{b}_u} \tag{3.15}$$

Hence, it can be concluded that the optimization of cost-function (3.13) subject to (3.14) and (3.15) will produce the optimal vector $\tilde{\nu}^*(k)$. It must be noted, however, that $\tilde{u}(k)$ appears in the equality constraint (3.14) and that the same constraint also

depends nonlinearly on the state $\tilde{x}(k)$. $\tilde{u}(k)$ is an independent variable and therefore it is necessary to append it to the cost-function (3.13) such that the constraints can also be incorporated into the problem as follows

$$\min_{\tilde{\nu},\tilde{u}} \quad \begin{bmatrix}\tilde{u}\\ \tilde{\nu}\end{bmatrix}^T \begin{bmatrix}R & 0\\ 0 & \tilde{Q}\end{bmatrix} \begin{bmatrix}\tilde{u}\\ \tilde{\nu}\end{bmatrix} + \begin{bmatrix}0\\ -2\tilde{x}_r^T\tilde{Q}\end{bmatrix}^T \begin{bmatrix}\tilde{u}\\ \tilde{\nu}\end{bmatrix} \tag{3.16}$$

$$\text{s.t.} \quad \begin{bmatrix}\tilde{C} & | -I_{Nn}\end{bmatrix} \begin{bmatrix}\tilde{u}\\ \tilde{\nu}\end{bmatrix} = \tilde{b}_{eq} \tag{3.17}$$

$$\begin{bmatrix}\tilde{A}_u & 0\end{bmatrix} \begin{bmatrix}\tilde{u}\\ \tilde{\nu}\end{bmatrix} \leq \tilde{b}_u \tag{3.18}$$

The minimization of (3.16), s.t. (3.17) and (3.18) leads to a feasible $\tilde{u}^*$ and an optimal $\tilde{\nu}^*$. Note that the above Equation 3.17 incorporates the relationship between the virtual input w, the physical input Δu, and the variable ν (see remark). The latter may be interpreted as if the dynamic inversion were embedded into the MPC problem. It must be noted, however, that it is not possible to weight the input $\tilde{u}(k)$ during this phase because that impairs the state-tracking capability of the controller. The argument of the optimization $\tilde{u}^*$ is not unique, since $g(x(k))$ is a wide matrix. Hence, it is possible to pose a second optimization problem in the form of a control allocation problem, which will be the subject of one of the following sections.

One issue, that was already mentioned in the previous paragraph, is that the equality constraint (3.17) depends on the state in a nonlinear fashion. This constraint therefore has to be approximated such that it is either constant or linearly dependent of the state at time k. Several possible approximations are:

1. Assume that $x(k)$ is constant over the horizon such that

$$\tilde{C} \approx I_n \otimes g(x(k)), \tilde{b}_{eq} \approx \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}^T f(x(k));$$

2. Apply the input that was computed for the previous time step to predict the evolution of the state over the horizon;
3. Assume that the system state will follow the reference state according to a stable and linear time-invariant (LTI) reference system;
4. Exploit a Jacobian linearization of $f(x(k))$ and $g(x(k))$ to obtain a local LTI model that can be applied to predict the evolution of the state over the horizon.

It is acknowledged here that what is presented in this section is a tailor-made MPC implementation, and refer to Maciejowski (2002b) for an in-depth investigation of MPC and its properties in general. Furthermore, experience has shown that the proposed MPC problem is computationally very intensive. The latter is predominantly caused by the fact that both the physical input u and the input of the inverted system ν are free variables which simultaneously appear in the MPC problem, together with the relation between them and the constraints on the input u.

Now define matrix $F(x) \in \mathbb{R}^{n\times n}$ be such that $f(x) = f(0) + F(x)\,x(k)$. We now obtain

$$\Delta u \approx G^l(x)\Big(\nu(k) - F(x(k))\,x(k) - f(0)\Big)$$

where $G_l(x)$ is a pseudo left-inverse of $G(x)$. For the prediction we can find matrices $\tilde{\mathbf{C}}_F$, $\tilde{\mathbf{D}}_F$ and $\tilde{\mathbf{E}}_f$ such that

$$\tilde{\mathbf{u}}(k) \approx \tilde{\mathbf{C}}_F(x)x(k) + \mathcal{D}_f(x)\nu(k) + \tilde{E}_f(x)\,f(0)$$

If we assume that $x(k)$ is constant over the horizon we obtain the objective function

$$\begin{aligned} J(\tilde{\nu}) \quad &\approx \tilde{\nu}^T(\tilde{Q} + \mathcal{D}_f^T\tilde{R}\mathcal{D}_f)\tilde{\nu}^T + 2\tilde{\nu}^T(\tilde{C}_f x(k) + \tilde{E}_f f(0) - \tilde{Q}\,\hat{x}_r(k)) \\ &+(\tilde{C}_f x(k) + \tilde{E}_f f(0) - \tilde{Q}\,\hat{x}_r(k))^T\tilde{R}(\tilde{C}_f x(k) + \tilde{E}_f f(0) - \tilde{Q}\,\hat{x}_r(k)) \\ &= \tilde{\nu}^T H\tilde{\nu} + \tilde{\nu}^T\mu + c \end{aligned} \tag{3.19}$$

From a computational point of view it is much more desirable to solve a MPC problem of the following form

$$\min_{\tilde{\nu}} \quad \tilde{\nu}^T H\tilde{\nu} + \tilde{\nu}^T\mu + c \tag{3.20}$$

$$\text{s.t.} \quad \tilde{A}_\nu\tilde{\nu} \leq \tilde{b}_\nu \tag{3.21}$$

which is a function of ν only. The latter is possible, but requires a relationship between the constraints on Δu and those on ν which can be computed by means of a constraint mapping algorithm, which will be the topic of the next section.

3.2.4 Constraint mapping through polytope projection

In order to arrive at the computationally less expensive MPC problem in (3.20) we require the mapping of the polytope $\mathcal{U}$ that bounds $u(k)$ to a polytope that bounds $\nu(k)$ via the relationship

$$g(x(k))\Delta u(k) = -f(x(k)) + \nu(k) \tag{3.22}$$

or, when the state is plugged in (according to one of the previously mentioned approximations)

$$\nu(k) = G\Delta u(k) + f$$

where $G = g(x)|_{x=x(k)}$ and $f = f(x)|_{x=x(k)}$. It can very easily be shown that when Δu lies inside a convex set, that ν will also be bound to lie inside a convex set. This holds true since we have assumed that $m > n$ and hence the projection matrix G projects $\mathcal{U}$ onto a lower dimensional subspace. Such a projection preserves convexity.

Performing this mapping must be done every time step and is very closely related to the subject of computational geometry. It is however well-known that projection methods, like vertex enumeration methods and Fourier-Motzkin elimination,

as are described in Preparata and Shamos (1985), are computationally very expensive and therefore not suitable for this application which will typically be applied to systems with fast dynamics. Even the more advanced and much faster methods like the equality set projection algorithm in Jones et al. (2004) was shown to be prohibitive where computational complexity is concerned.

It is, however, possible to very efficiently compute the bound $A_\nu \nu \leq \nu$ using the results of the following proposition:

Define the sets $\mathcal{U}$ and $\mathcal{V}$ as follows:

$$\mathcal{U} = \{\Delta u \in \mathbb{R}^n | -1 \leq \Delta u_i \leq 1, i = 1, \ldots, n\} \tag{3.23}$$

and

$$\mathcal{V} = \{v \in \mathbb{R}^m,\ w \in \mathbb{R}^n \mid v = Gu,\ u \in \mathcal{W}\} \tag{3.24}$$

The following theorem shows how to rewrite the set $\mathcal{V}$ of (4.7) on page 73 into the following form:

$$\mathcal{V} = \{\nu \in \mathbb{R}^m \mid -b_\nu \leq\ A_\nu \nu \leq b_\nu\}. \tag{3.25}$$

Theorem 3.1 (Hypercube projection) *Given the sets $\mathcal{U}$ and $\mathcal{V}$ as defined in (3.24)-(3.25), and a matrix $G \in \mathbb{R}^{m\times n}$, $m < n$. Let $s = \begin{pmatrix} n \\ m-1 \end{pmatrix}$ and define the set $\mathcal{T} = \{T_1, T_2, \ldots, T_s\}$ where the matrices $T_i \in \mathbb{R}^{n\times(m-1)}$, $i = 1, \ldots, s$ are all possible combinations of $(m-1)$ columns of the unit matrix $I^{n\times n}$, and let $T_i^\perp \in \mathbb{R}^{n\times(n-m+1)}$ consist of all remaining $(n-m+1)$ columns of the unit matrix $I^{n\times n}$.*

For all $i = 1, \ldots, s$ we can define the row-vector, if the $\operatorname{rank}(GT_i) = (m-1)$ then choose a_i as

$$a_i = \operatorname{Ker}((G\,T_i)^T)\ \in \mathbb{R}^{m\times 1}, \tag{3.26}$$

and the scalar value

$$b_i = |a_i^T\, G\, T_i^\perp|\, \bar{1}_{n-m+1},$$

where $|\cdot|$ denotes the element-wise absolute value. Now define

$$A_\nu = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_s^T \end{bmatrix}, \quad b_\nu = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_s \end{bmatrix},$$

then the set $\mathcal{V}$ can be written as

$$\mathcal{V} = \{v \in \mathbb{R}^m \mid\ -b_\nu \leq A_\nu\, v \leq b_\nu\}$$

Remark: the method introduced in Proposition 3.1 can be extended to apply to projection of zonotopes. Zonotopes are polytopes that are point-symmetric about the origin. The extensions toward generic convex polytopes is given in Chapter 4

Proposition 3.1 provides a very efficient algorithm to project the constraints that limit the physical system input Δu onto a set of constraints on ν, thus greatly

simplifying MPC problem (3.16) and reducing it into problem (3.20). Next to that it can be remarked that, although Δu was assumed to lie inside a hypercube, the presented method holds equally well for constraints of the form

$$\underline{u}_i \leq \quad \Delta u_i \quad \leq \bar{u}_i \tag{3.27}$$

where $\underline{u}_i$ and $\bar{u}_i$ define componentwise lower and upper bounds on the increment input signal. The latter more general problem reduces to the case where Δu is assumed to lie inside a hypercube when the input is appropriately scaled and translated. This linear transformation must then be reversed after the projective step.

An example that illustrates the projection of a hypercube in $\mathbb{R}^3$ onto $\mathbb{R}^2$ using a random projection matrix G is provided in Figures 3.2 and 3.3. The algorithm has been tested extensively using different dimensions m and n. Especially for higher dimensions (e.q. $u \in \mathbb{R}^{10} - \mathbb{R}^{30}$ this polytope projection has a clear advantage over other geometric projection methods. Since inputs of physical systems generally have upper and lower bounds, the assumption that u is constrained to lie inside a box-like set is deemed only mildly restrictive.

We have now succeeded at separating the NDI part of the controller from the MPC part. The latter reduces the overall controller from one very large optimization problem into two smaller problems (MPC and NDI). MPC has now been covered, what remains is to show how NDI can be incorporated into a efficient control allocation scheme. This will be the topic of the next section.

3.2.5 Computationally Efficient Control Allocation

The previous sections have shown that it is possible to construct a globally valid, but constrained and nonlinear controller by means of a combination of MPC and FBL. Until now, however, we have only computed the FBL input ν_k^* in the previous section, which is related to the desired forces and moments $z(k)$ via the relationship $\nu(k) = f(x(k)) + z(k)$. These desired forces and moments need to be translated, since in general the number of inputs is known to be larger than the number of controlled states. It is desirable to allocate the available inputs such that the desired forces and moments are achieved, and such that for instance, the absolute size of the inputs is minimal, or such that the change of the input with respect to the previous time step is minimized.

Hence, since $m \geq n$ there is freedom in choosing u. One way to solve this problem involves the following quadratic programming problem

$$\begin{aligned} \min_{\Delta u} \quad & J(\Delta u(k)) = u(k)^T Q_u u(k) \\ & + \Delta u(k)^T R_u \Delta u(k) \\ \text{s.t.} \quad & g(x(k))u(k) = -f(x(k)) + \nu^*(k) \\ & Au(k) \leq b \end{aligned} \tag{3.28}$$

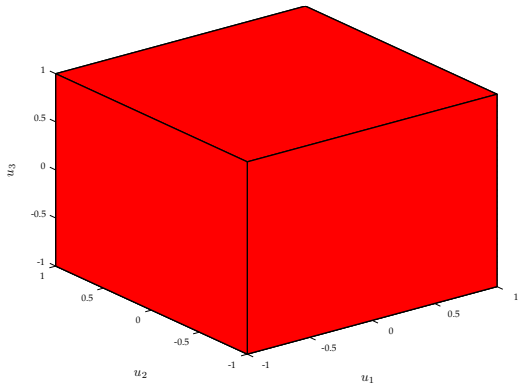

Figure 3.2: Part 1 of the projection example: this cube in $\mathbb{R}^3$ represents the original constraint on the variable u.

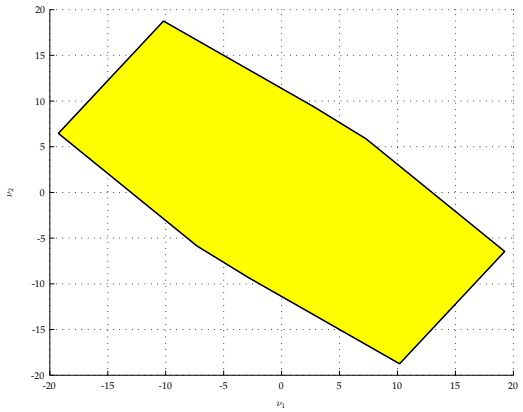

Figure 3.3: Part two of the projection example: the depicted polytope represents the constraints on the variable ν that follow from the relationship $\nu = Gu + f$ and the affine projection of he bounds on u onto bounds on ν using this relationship.

where $\nu^*(k)$ is the optimal input resulting from MPC (3.20), $\Delta u = u(k) - u(k-1)$ and where $Q_u, R_u \succeq 0$ are input weighting matrices.

The above optimization problem may be interpreted as follows: given one feasible input $\nu^*(k)$ that results from the MPC step, this control allocation problem will find a $u(k)$ that satisfies the mixed objective posed above: a weighted minimization of the inputs and minimization of the change of $u(k)$ with respect to the previous time step, while satisfying the control allocation goal by means of the equality constraint $g(x(k))u(k) = -f(x(k)) + \nu(k)$. The goal of this cost-function is bi-fold: on the one hand it is desired to minimize the 2-norm of the input, which can be interpreted as the desire to apply small inputs, whereas on the other hand the minimization of the 2-norm of $\Delta u(k)$ makes the controller much less aggressive. The second item can be compared to the addition of integral action to the controller.

At this point it must be remarked that both MPC and the control allocation method that is presented in this section require computation of a quadratic program with a QP-solver. Although efficient implementations of QP-solvers exist, it must be noted that both MPC and control allocation account for the bulk of the computational effort that is required in the entire flight control setup which is presented in this paper. It goes without saying that the latter is an important aspect in flight control applications.

Furthermore, as a result of the fact that MPC, FBL, control allocation are implemented as individual entities here, there no longer is a link between which input is applied towards which goal, i.e. it is not a priori such that this controller will apply ailerons to achieve a certain roll rate. It might for instance be such that the controller of this setup will apply differential spoilers to achieve the same control action. This can be avoided to a certain degree through an appropriate choice of the weighting matrices Q_u and R_u.

Both issues lead to the conclusion that in the nominal case it is better, i.e. when failures are absent, to apply only a reduced subset of the available inputs. This is especially true for the example which will be presented in Chapter 5, where the total number of individual inputs equals 30. It is therefore suggested to fly with a reduced set of inputs as long as (3.28) is feasible. This reduced set of inputs will in general consist of the primary actuators (e.g. elevator, rudder, ailerons, engine thrust). It is this approach that allows for a significant speed-up of the control allocation algorithm and, furthermore, it forces the controller to apply more 'natural' control inputs in the nominal case.

When (3.28) is not feasible, e.g. in a failure case, this means that the chosen subset of inputs is no longer sufficient to achieve the desired control action. What we present here is an intelligent method to compute the set of most effective inputs such that it can still be avoided to use the full set of inputs. This method is based on the unconstrained solution to the optimization presented in (3.28). Using this unconstrained solution it is possible compute what the effectiveness of each input is, such that they can be ranked accordingly.

It is, however, problematic that the actual control allocation is somewhat hidden in

the complete problem in the form of an equality constraint. We therefore start this search for the set of most effective inputs from the Lagrange dual of the control allocation problem (3.28). Therefore we combine the equality constraint and cost-function, whilst leaving out the input constraints ($Au < b$). For reasons of clarity we will represent the control allocation problem in shorthand notation:

$$\begin{aligned} \min_{w} \quad & w^T H w + f^T w \\ \text{s.t.} \quad & A_w w = b_w \end{aligned} \tag{3.29}$$

where $w = \Delta u(k), H = (Q_u + R_u), f^T = 2u(k-1)Q_u$. The corresponding Lagrangian equals

$$L(w, \lambda) \quad = \quad w^T H w + f^T w + \lambda^T (A_w w - b_w) \tag{3.30}$$

Since $L(w, \lambda)$ is a convex quadratic function of w we can find the minimizing w from the optimality condition

$$\nabla_w L(w, \lambda) = 2Hx + f + A_w^T \lambda = 0 \tag{3.31}$$

which yields

$$w = -\frac{1}{2} H^{-1} (f + A_w^T \lambda) \tag{3.32}$$

and therefore the dual cost-function is

$$\begin{aligned} g(\lambda) \quad &= \quad L(-\frac{1}{2} H^{-1} (f + A_w^T \lambda), \lambda) \\ &= \quad -\frac{1}{4} \lambda^T A_w H^{-1} A_w^T \lambda - (\frac{1}{2} f^T H A_w^T - b) \lambda \end{aligned} \tag{3.33}$$

Now remember from convex optimization theory that $\max g(\lambda) \leq \min J$ always holds. The dual cost-function has its maximum when

$$-\frac{1}{2} A_w T H^{-1} A_w^T \lambda - \frac{1}{2} f^T H A_w^T - b = 0 \tag{3.34}$$

Obviously the minimum λ^* of the latter is not unique because this problem is underdetermined and remember that in the latter the constraints are not taken into account. Solving (3.34) can be reformatted as a least-squares problem where

$$\begin{aligned} \lambda^* \quad &= \quad \arg\min_{\lambda} || - (\frac{1}{2} A_w H^{-1} A_w^T) \lambda \\ &\qquad\qquad -(\frac{1}{2} f^T H A_w^T - b_w)||_2^2 \end{aligned} \tag{3.35}$$

One possible solution, the so-called minimum 2-norm solution can easily be computed using the singular value decomposition of $-(\frac{1}{2} A_w H^{-1} A_w^T)$ such that

$$\begin{aligned} -(\frac{1}{2} A_w T H^{-1} A_w^T) \quad &= \quad \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \\ &= \quad U_1 \Sigma V_1 \end{aligned} \tag{3.36}$$

The minimum 2-norm solution to (3.35) now equals

$$\lambda^* = V_1 \Sigma^{-1} U_1^T (-\frac{1}{2} f^T H A^T + b) \tag{3.37}$$

This result can now be applied to obtain the unconstrained minimizer of the original cost function and control allocation problem. (3.32) relates λ^* to the unconstrained minimum w^*. Using this unconstrained solution we finally arrive at the following

$$\frac{\partial J}{\partial w}|_{w=w^*} = Hw^* + f^T \tag{3.38}$$

which gives an indication of how the individual inputs contribute to the minimization of the cost-function J based on a ranking of the magnitude of the partial derivatives. Based on this knowledge, whilst remembering that $w = \Delta u$, a new and larger set of inputs can be selected from the available inputs when the control allocation problem based on the reduced set of inputs is no longer feasible. This approach can be characterized as being a ranking of the set of available inputs ordered based upon their effectiveness in the current situation. Obviously, this method says nothing regarding how many inputs are required to regain feasibility of the optimization problem, some iterations are hence necessary. The major benefit lies in the computational attractiveness of this methods when compared to optimizing over all available inputs.

It is this control allocation strategy that completes the FTFC setup that has been presented in this section, and the next section will show the merits of this FTFC method by means of an example that involves the nonlinear equations of motion of a fixed-wing aircraft.

3.3 Simulation Example

This example is loosely based on the (simplified) nonlinear longitudinal equations of motion of a fixed wing aircraft where the symmetric stability and control derivatives data of a Citation business jet is applied (Mulder et al. 2006, Table B-1, p.546). We describe the dynamics as follows.

$$\dot{x} = f(x) + g(x)u \tag{3.39}$$

where x is the state vector and $u = [\delta_e T]^T$ the vector of inputs. The state vector comprises the states $x = [V, \gamma, q, \theta]^T$, where V is the airspeed [m/s], γ [rad] the flightpath angle, q [rad/s] and θ [rad] the pitch attitude. The input vector consists of the elevator angle δ_e [rad] and the thrust [N]. The equation elements $f(x)$ and $g(x)$ are as follows:

$$f(x) = \begin{bmatrix} -g\sin(\gamma) - D/m \\ (-g\cos(\gamma)/V) + L/(mV) \\ (1/I_{yy})M \\ q - ((-g\cos(\gamma)/V) + L/(mV)) \end{bmatrix} \tag{3.40}$$

Table 3.1: stability and control derivatives for the Cessna Ce550, based on (Mulder et al. 2006, Table B-1, p.546)

C_{M_0}	=	0.0009	C_{X_0}	=	0	C_{Z_0}	=	-1.1360
			C_{X_u}	=	-0.2199	C_{Z_u}	=	2.2720
C_{M_α}	=	-0.4300	$C_{X\alpha}$	=	0.4653	C_{Z_α}	=	-5.1600
$C_{M_{\delta_e}}$	=	-1.5530	$C_{X_{\delta_e}}$	=	0	$C_{X_{\delta_e}}$	=	-0.6238

and

$$g(x) = \begin{bmatrix} -D_u/m & \cos(\alpha)/m \\ L_u/(mV) & \sin(\alpha)/m \\ (1/I_{yy})M_u & 0 \\ -(L_u/(mV) & -\sin(\alpha)/m \end{bmatrix} \tag{3.41}$$

where g is the gravitational acceleration, I_{yy} the moment of inertia about the pitch axis. The lift (L) and drag (D) forces and the pitching moment M can be expressed as follows:

$$\begin{bmatrix} M \\ L \\ D \end{bmatrix} = \frac{1}{2}\rho V^2 S \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} C_{M_0} + C_{M_\alpha}\alpha \\ C_{X_0} + C_{X_u}(u/V) + C_{X_\alpha}\alpha \\ C_{Z_0} + C_{Z_u}(u/V) + C_{Z_\alpha}\alpha \end{bmatrix} \tag{3.42}$$

$$\begin{bmatrix} M_u \\ L_u \\ D_u \end{bmatrix} = \frac{1}{2}\rho V^2 S \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} C_{M_{\delta_e}} \\ C_{X_{\delta_e}} \\ C_{Z_{\delta_e}} \end{bmatrix} \tag{3.43}$$

The stability and control derivatives are listed in Table 3.1 and are representative for straight and level flight ($\gamma = 0, q = 0$) where the airspeed $V = 59.9$ [m/s] and the pitch attitude $\theta = 0.0163$ [rad]. The inputs required for steady trimmed flight in this condition are $\delta_e = -0.0039$ [rad] and $T = 56.2$ [kN]. The nonlinear state equations are also linearized to obtain the following state space-matrices:

$$\dot{x} = f(x) + g(x)u \approx A(x - x_{trim}) + B(u - u_{trim}) \tag{3.44}$$

where

$$A = \begin{bmatrix} -0.4128 & -9.8100 & 0 & 58.3724 \\ 0.0021 & 0 & 0 & 12.6444 \\ 0 & 0 & 0 & -1.0006 \\ -0.0021 & 0 & 1.0000 & -12.6444 \end{bmatrix} \tag{3.45}$$

$$B = \begin{bmatrix} 7.2936 & 2.1985 \times 10^{-4} \\ -0.0020 & 3.589411 \times 10^{-6} \\ -3.6138 & 0 \\ 0.0020 & -3.589411 \times 10^{-6} \end{bmatrix} \tag{3.46}$$

In constructing the nonlinear dynamic inversion we will set the desired behavior to match the linearized state space system. The effect of this choice is that nonlinearities to be canceled through dynamic inversion are small in the vicinity of this operating point and that hence the effect of modeling mismatches is smaller in such case.

In this particular example there are two inputs and four states. Common sense dictates that the number of states to be controlled are to be reduced. A logical choice for such reduction is to focus on the airspeed and pitch rate. This reduces $f(x)$ and $g(x)$ to where

$$\tilde{f}(x) = \begin{bmatrix} -g\sin(\gamma) - D/m \\ (1/I_{yy})M \end{bmatrix} \tag{3.47}$$

and

$$\tilde{g}(x) = \begin{bmatrix} -D_u/m & \cos(\alpha)/m \\ (1/I_{yy})M_u & 0 \end{bmatrix} \tag{3.48}$$

which together with reduced state space matrices A_r and B_r and $x_r = [Vq]^T$ leads to the dynamic inversion control law

$$u = (\tilde{g}^{-1}(x))(-\tilde{f}(x) + A_r(x_r - x_{r_{trim}}) + B_r(\nu - u_{trim}) \tag{3.49}$$

where ν is the new input to the inverted system.

An MPC controller in the sense of Chapter 2 is applied here although, admittedly, an original controller must be found for that first. To this purpose a discrete time state feedback controller of the form $u_r = -Fx_r(k)$ is chosen that places the poles of the linearized system (A, B) at $p_{des} = [0.9048, 0.9512]^T$ after zero-order-hold discretization of (A, B) with sampling time $T_s = 0.1$ [s]. The resulting state feedback matrix is obtained through pole placement and equals

$$F = \begin{bmatrix} 0 & -0.1384 \\ 2.67 \times 10^3 & 4.5901 \times 10^3 \end{bmatrix} \tag{3.50}$$

An MPC controller is subsequently constructed along the lines of Section 1.3, i.e. matching the original controller through MPC, in this case matching the state feedback controller. While this is not strictly necessary, the latter design choices are representative of the case where one wishes to match the behavior of an original autopilot, hence the matching of the linearized system and controller.

Figures 3.4 and 3.5 show the simulation results for the closed loop of plant, NDI and state-feedback/MPC for an initial condition that has an airspeed 10 m/s above the trimmed condition. The NDI controlled states converge nicely but note that the flight path γ in 3.5 slowly diverges. This is due to the fact that this variable is not controlled in this example. Typically flight control systems consist of an inner loop and an outer loop. The inner loop controls the *fast* variables and the outer loop controls the *slower* variables. Flight path control would typically be implemented in an outer loop. This example foregoes the design of the outer loop.

3.4 Discussion and conclusion

A method that combines the constraint handling capabilities of MPC with NDI has been introduced. The challenge thereof lies in the resulting nonlinear transformation of the original input constraints of the plant. This is both computationally

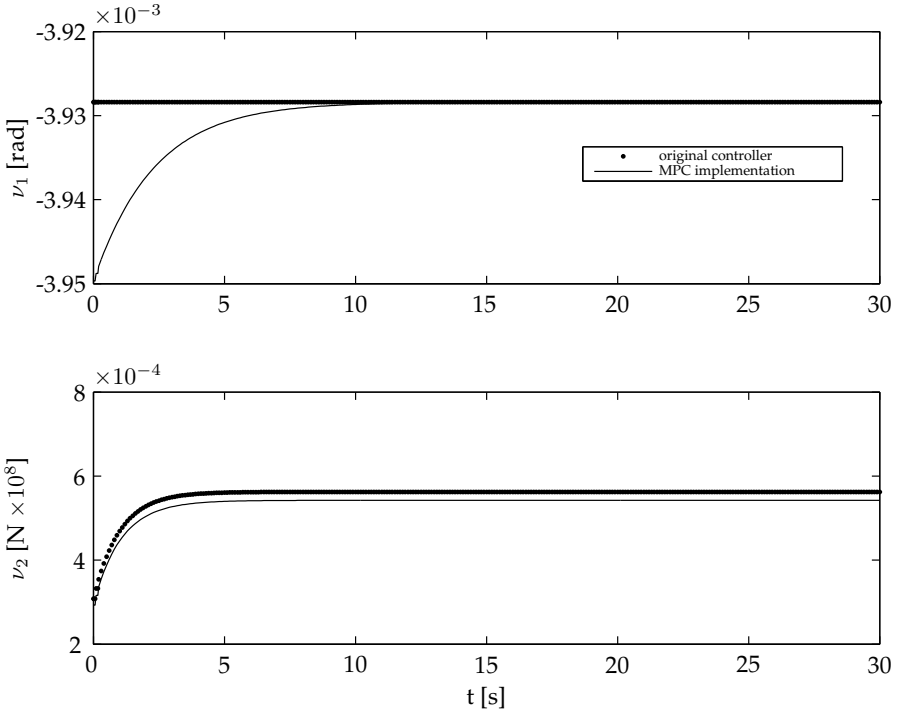

Figure 3.4: Inputs $\nu_{1,2}$ to the dynamic inversion part of the closed loop of plant, dynamic inversion and either state feedback controller or the equivalent MPC controller (prediction horizon $N = 20$).

intensive and the expansion of this transformation over the prediction horizon remains an approximation due to the nonlinear plant.

A drawback of the methods introduced is that NDI or feedback linearization is known to be very sensitive to modeling errors and MPC is a computationally intensive methods. Even the comparatively simple example introduced at the end of this chapter supports the notion that these properties do not combine well with a system that has fast dynamics such as an aircraft.

Having said this, this thesis does present a method to perform the mapping of constraints onto the inputs of the NDI controller in an efficient manner. Chapter 4 investigates the matter of constraint mapping more in-depth. This alleviates one aspect of the aforementioned issues. Also the example shows the feasibility of the method as a whole.

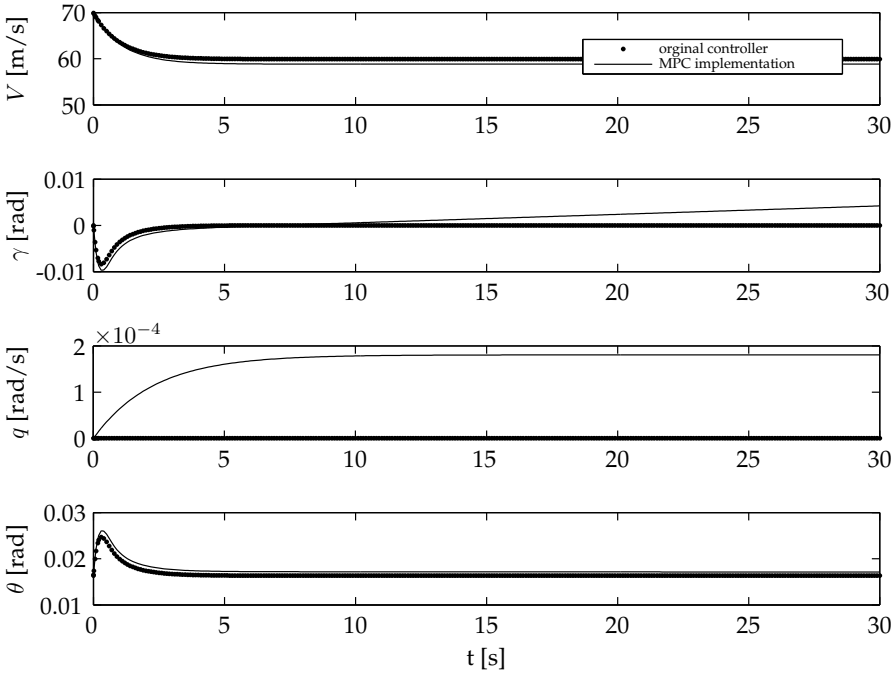

Figure 3.5: States for simulation of the closed loop of plant, dynamic inversion and either state feedback controller or the equivalent MPC controller (prediction horizon $N = 20$).

4 CHAPTER

Polytope projection

This chapter investigates the projection of a polytope onto a lower dimensional subspace. Projection methods that depend on the vertex descriptions of the original polytope and its image under projection can be very computationally expensive. Methods are proposed here that make use of the halfspace description to the maximum extent possible. This particularly relevant to the theory in Chapter 3 of this thesis, but there is also a link with optimization algorithms in general.

4.1 Introduction

In the MPC procedure of Chapter 3 feedback linearization has been applied that linearizes and decouples the system under control. The actuator input signal $u(k)$ is mapped onto a new linearized input signal $v(k)$ at each time step k using

$$v(k) = G\,u(k) + f$$

Note that for a large aircraft we have that the number of control surfaces is much larger than the six degrees of freedom we need to control it. This means that with $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^m$ we have $n \gg m$ (for a Boeing 747 aircraft we have $n = 30$ and $m = 6$).

If we have to solve the MPC problem the constraints will have the form

$$A\tilde{\mathbf{u}}(k) \leq b$$

where

$$\tilde{\mathbf{u}}(k) = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N-1) \end{bmatrix}$$

with $\Delta u(k) = u(k) - u(k-1)$. The following paragraphs provide details as on how to include either constraints on the input increments, the inputs themselves, or the combination of these both in this format.

It is this constraint set in variable $u(k)$ that we need to project onto $\mathbb{R}^m$ such that we arrive at the new constraint set required for the MPC problem

$$A_v \mathbf{v}(k) \leq b_v$$

Increment Input constraints:

Input constraints in MPC are usually given in the form of upper and lower bounds on the individual incremental inputs:

$$\Delta u_{i,\min} \leq \Delta u_i(k) \leq \Delta u_{i,\max} \ , \quad \forall i \in [1, n]$$

where the index i denotes the i-th input. We obtain the constraint:

$$\tilde{\mathbf{u}}_{\min} \leq \tilde{\mathbf{u}}(k) \leq \tilde{\mathbf{u}}_{\max}$$

or

$$\begin{bmatrix} I \\ -I \end{bmatrix} \tilde{\mathbf{u}}(k) \leq \begin{bmatrix} \tilde{\mathbf{u}}_{\max} \\ -\tilde{\mathbf{u}}_{\min} \end{bmatrix} \tag{4.1}$$

where

$$\tilde{\mathbf{u}}_{\min} = \begin{bmatrix} \Delta u_{\min} \\ \Delta u_{\min} \\ \vdots \\ \Delta u_{\min} \end{bmatrix}$$

and the same holds for $\Delta\tilde{\mathbf{u}}_{\max}$.

(Non-increment) Input constraints:

If we deal with an MPC problem with (non-incremental) inputs we consider the constraints:

$$u_{i,\min} \leq u_i(k) \leq u_{i,\max} \ , \quad \forall i \in [1, n]$$

Define

$$\tilde{\mathbf{u}}'(k) = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-1) \end{bmatrix}$$

Then we obtain the constraint:

$$\tilde{\mathbf{u}}'_{\min} \leq \tilde{\mathbf{u}}'(k) \leq \tilde{\mathbf{u}}'_{\max}$$

or

$$\begin{bmatrix} I \\ -I \end{bmatrix} \tilde{\mathbf{u}}'(k) \leq \begin{bmatrix} \tilde{\mathbf{u}}'_{\max} \\ -\tilde{\mathbf{u}}'_{\min} \end{bmatrix}$$

with $\tilde{\mathbf{u}}'_{\max}, \tilde{\mathbf{u}}'_{\min}$ defined similar to $\tilde{\mathbf{u}}_{\max}$ and $\tilde{\mathbf{u}}_{\min}$. Now let

$$M = \begin{bmatrix} I & 0 & 0\dots & 0 \\ I & I & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ I & I & \dots & I \end{bmatrix} \in \mathbb{R}^{Nn \times Nn}, \qquad L = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} \in \mathbb{R}^{Nn \times n},$$

Then with

$$\begin{aligned} \tilde{\mathbf{u}}'(k) &= \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-1) \end{bmatrix} \\ &= \begin{bmatrix} \Delta u(k) + u(k-1) \\ \Delta u(k+1) + \Delta u(k) + u(k-1) \\ \vdots \\ \Delta u(k+N-1) + \Delta u(k+N-2) + \ldots + \Delta u(k) + u(k-1) \end{bmatrix} \\ &= M\,\tilde{\mathbf{u}} + Lu(k-1) \end{aligned}$$

We obtain the constraint

$$\tilde{\mathbf{u}}'_{\min} \leq M\,\tilde{\mathbf{u}} + Lu(k-1) \leq \tilde{\mathbf{u}}'_{\max}$$

or

$$\begin{bmatrix} M \\ -M \end{bmatrix} \tilde{\mathbf{u}}(k) \leq \begin{bmatrix} \tilde{\mathbf{u}}'_{\max} - Lu(k-1) \\ -\tilde{\mathbf{u}}'_{\min} + Lu(k-1) \end{bmatrix} \tag{4.2}$$

Mixed Input constraints:

If we have constraints on both inputs and increment inputs we have to consider the intersection of both constraints and we obtain

$$\begin{bmatrix} I \\ -I \\ M \\ -M \end{bmatrix} \tilde{\mathbf{u}}(k) \leq \begin{bmatrix} \tilde{\mathbf{u}}_{\max} \\ -\tilde{\mathbf{u}}_{\min} \\ \tilde{\mathbf{u}}'_{\max} - Lu(k-1) \\ -\tilde{\mathbf{u}}'_{\min} + Lu(k-1) \end{bmatrix} \tag{4.3}$$

General constraints:

In Chapter 1 we introduced the model predictive control problem with inequality constraint (1.10):

$$\tilde{\mathbf{F}}(k)x(k-1) + \tilde{\mathbf{G}}(k)\tilde{\mathbf{r}}(k) + \tilde{\mathbf{H}}(k)\tilde{\mathbf{u}}(k) + \tilde{\mathbf{J}}(k)u(k-1) \leqslant \tilde{\mathbf{h}}(k)$$

or with $A_u = \tilde{\mathbf{H}}(k)$ and $b_u = \tilde{\mathbf{h}}(k) - \tilde{\mathbf{F}}(k)x(k-1) - \tilde{\mathbf{G}}(k)\tilde{\mathbf{r}}(k) - \tilde{\mathbf{J}}(k)u(k-1)$

$$A_u\tilde{\mathbf{u}}(k) \leqslant b_u \tag{4.4}$$

Note that in the constraints (4.3) the final constraints have the form of (4.4) where b_u is affine in a variable $u(k-1)$, $x(k-1)$ and/or $\tilde{\mathbf{r}}(k)$, which are known at time sample k. Now define the set of all $\tilde{\mathbf{u}}(k)$ that satisfy (4.4) as $\mathcal{U}$, or

$$\mathcal{U} = \{\tilde{\mathbf{u}} \in \mathbb{R}^m \,|A_u\, \tilde{\mathbf{u}} \leq b_u\}. \tag{4.5}$$

then $\mathcal{U}$ is a convex polytopic set.

Now consider the projection

$$\tilde{\mathbf{v}}(k) = \tilde{G}\, \tilde{\mathbf{u}}(k) + \tilde{f}$$

where

$$\tilde{G} = I_N \otimes G$$

with I_N the $N \times N$ identity matrix, and

$$\tilde{f} = [1 \quad 1 \ldots 1]^T \otimes f$$

which constitutes an affine transformation of the original input constraints on $\tilde{u}$. This projection forms the starting point for the remainder of this chapter as this is the problem we would like in a computationally efficient manner.

4.2 Projection

For ease of notation we use u and v instead of $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ in this section. Suppose $u \in \mathcal{U}$ where $\mathcal{U}$ is in a convex polytopic set. The set of all feasible linearized input signals is defined as

$$\mathcal{V} = \{v \in \mathbb{R}^m,\ u \in \mathbb{R}^n \mid v = Gu,\ u \in \mathcal{U}\}$$

which constitutes the projection of actuator space $\mathcal{U}$ on the space $\mathcal{V}$. Note that due to the fact that $\mathcal{U}$ is a polytope we find that $\mathcal{V}$ will also be a polytope, which means that there exist a matrix A and a vector b, such that

$$\mathcal{V} = \{v \in \mathbb{R}^m \,|A\, v \leq b\}$$

As already stated in Chapter 3, it is important to find an efficient and fast algorithm to compute that matrix A and vector b for the MPC algorithm to use at each time step.

Various methods for such projection of constraints, i.e. polytope projection, exist and the interested reader is referred to technical reports such as those by Jones et al. (2004) and Fukuda (2004) for an introduction to the theory of polyhedrons, polytopes and their associated terminology. Jones et al. (2004) provide an orthogonal polytope projection method and list a brief overview of associated literature on the matter. Well-known polytope projection methods include Fourier-Motzkin elimination and methods based on the vertices of the polytope.

Fourier-Motzkin (FM) elimination in linear inequalities, first described by Fourier in 1824, is in fact similar to Gaussian elimination for linear equalities. In FM elimination the original polytope in $\mathbb{R}^n$ is recursively projected one dimension lower ($\mathbb{R}^{n-1}$) until the required dimension is reached. The disadvantage of the method is that it generates many redundant inequalities at each iteration. Removing the redundant inequalities makes this method computationally expensive.

Vertex based projection methods apply the vertices, the extreme points of the polytope that is to be projected, as projection of the individual vertices onto a lower dimension by multiplying each vertex with G. The disadvantage is that reconstruction of the convex hull of the image under projection is again computationally expensive as typically the number of vertices that describe the polytope is much larger than the number of linear inequalities that describe it (the half-space description) and as projection of the vertices leads to many redundant vertices that have to be removed in order to arrive at an irredundant (minimal) description of the projected image. A hypercube in $\mathbb{R}^d$, the higher-dimensional equivalent of a box in $\mathbb{R}^3$ or a square in $\mathbb{R}^2$, for instance requires $2d$ linear inequalities for its half-space description, whereas it has 2^d vertices.

This chapter continues to provide a method that has lower computational complexity than the aforementioned methods but is tailored to the projection of hypercubes, which is applicable to the projection problem in the introduction of this chapter when each individual input is bound to a range between a maximum and a minimum value. The method is computationally efficient as it is based on half-space description of the hypercube only. The chapter concludes with an extension of the hypercube projection methods to general polytopes that uses a mix of the vertices and the half-space description of the original and the projected polytope.

Problem definition

The *Projection problem* is defined as follows:
Given a convex polytope $\mathcal{U}$,

$$\mathcal{U} = \{u \in \mathbb{R}^n, A_u u \leq b_u\} \tag{4.6}$$

a projection matrix $G \in \mathbb{R}^{m \times n}, m < n$ in

$$v = Gu$$

compute matrix A and vector b such that

$$\mathcal{V} = \{v \in \mathbb{R}^m,\ u \in \mathbb{R}^n \mid v = Gu,\ u \in \mathcal{U}\} \tag{4.7}$$

reduces to

$$\mathcal{V} = \{v \in \mathbb{R}^m \,|A\, v \leq b\}\,.$$

Goal is to find a computationally efficient manner of performing a polytopic projection, where possible avoiding the application of linear programming methods

and the use of the vertices of the polytope. The preferred method is one that makes use of the halfspace description only. Such methods are relevant in a whole range of problems.

In Section 4.3 we will consider the projection of a hypercube. In Section 4.4 we will look at the projection of a convex polytope.

Some definitions

Definition 4.1 (hypercube). *A hypercube is the equivalent of a square in n dimensions. The n-hypercube can be described as follows*

$$\mathcal{W} = \{u \in \mathbb{R}^n | -1 \leq u_i \leq 1, i = 1, \ldots, n\} \tag{4.8}$$

Definition 4.2 (n-zonotope). *A convex n-polytope that is point-symmetric about the origin, i.e. the faces of the zonotope are defined as*

$$\begin{bmatrix} A \\ -A \end{bmatrix} x \leq \begin{bmatrix} b \\ -b \end{bmatrix} \quad \textit{for} \quad x \in \mathbb{R}^n$$

Remark: Note that a zonotope is point-symmetric, so if a point v is in the interior of the zonotope also $-v$ will be in the interior of the zonotope.

Definition 4.3 (vector of ones) *The vector $\bar{1}_p = [1\ 1\ \ldots\ 1]^T$ is a vector in $\mathbb{R}^p$.*

Definition 4.4 (matrix kernel) *The kernel of a matrix $A \in \mathbb{R}^{m \times n}$ with $n > m$ and rank$(A) = m$, is defined as*

$$Ker(A) = \{x \in \mathbb{R}^{n \times (n-m)} \mid A\,x = 0\ \}$$

4.3 Hypercube projection

In this section we consider the projection of a hypercube onto a lower dimension. First of all note that the projection with projection matrix G of a hypercube preserves convexity and hence the image of the projection on $\mathbb{R}^m$ will also be a closed and convex polytope. Furthermore, the closed convex polytope will be point-symmetric about the origin, i.e. a zonotope. The convex hull of the image results from the projection of the hull of the original hypercube.

More specifically, the inequalities that define the hull of the image are formed through projection of the $(m-1)$-faces of the hypercube that have exactly $(m-1)$ degrees of freedom.

4.3.1 A Projection algorithm

This section introduces a computationally cheap method for the projection of hypercubes onto a lower dimensional subspace.

Theorem 4.1 (Hypercube projection) *Given $\mathcal{U}$, $G \in \mathbb{R}^{m\times n}$, $m < n$ and $\mathcal{V}$ as defined in (4.6),(4.7). Let $s = \begin{pmatrix} n \\ m-1 \end{pmatrix}$ and define the set $\mathcal{T} = \{T_1, T_2, \ldots, T_s\}$ where the matrices $T_i \in \mathbb{R}^{n\times(m-1)}$, $i = 1, \ldots, s$ are all possible combinations of $(m-1)$ columns of the unit matrix $I^{n\times n}$, and let $T_i^\perp \in \mathbb{R}^{n\times(n-m+1)}$ consist of all remaining $(n-m+1)$ columns of the unit matrix $I^{n\times n}$.*

For all $i = 1, \ldots, s$ we can define the row-vector a_i. If the $\operatorname{rank}(GT_i) = (m-1)$ then choose a_i as

$$a_i = \operatorname{Ker}((G\,T_i)^T) \ \in \mathbb{R}^{m\times 1}, \tag{4.9}$$

and the scalar value

$$b_i = |a_i^T\, G\, T_i^\perp|\, \bar{1}_{n-m+1},$$

where $|\cdot|$ denotes the element-wise absolute value. Now define

$$A = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_s^T \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_s \end{bmatrix},$$

then the set $\mathcal{V}$ can be written as

$$\mathcal{V} = \{v \in \mathbb{R}^m \mid \ -b \le A\, v \le b\}$$

Proof:

Let us now consider $(m-1)$-face i of a hypercube. Note that

$$\begin{bmatrix} T_i & T_i^\perp \end{bmatrix}^T \begin{bmatrix} T_i & T_i^\perp \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

Any u_i on this face can adequately be described as

$$u_i = T_i\, \eta + T_i^\perp\, w$$

where η is a free variable in $\mathbb{R}^{m-1}$, $w_i \in -1, 1 \forall i \in [1, n-m+1]$. Then plugging in $v = G\, u$ gives:

$$v = G\,(T_i\, \eta + T_i^\perp\, w).$$

Now consider a_i as defined in (4.9), then a_i is orthonormal with respect to face i, so

$$a_i^T\, G\, T_i\, \eta = 0$$

This gives us:

$$a_i^T v = a_i^T G\, T_i^\perp\, w \tag{4.10}$$

Note that any hypercube in $\mathbb{R}^n$ has exactly $2n-m+1$ faces with $m-1$ degrees of freedom which are parallel to face i and hence lead to the same a_i after projection. The vector w, however, differs up to a permutation of the sign of its entries. We

now aim to find the boundary faces by finding the maximum value $b_{i,\max}$ and minimum values $b_{i,\min}$ such that

$$b_{i,\min} \leq a_i^T G T_i^{\perp} w \leq b_{i,\max} \ , \ \ \forall \ w \in \mathbb{B}^{n-m+1}$$

One may compute the maximum value as

$$b_{i,\max} = b_i = \max_{w \in \mathbb{B}^{n-m+1}} a_i^T G T_i^{\perp} w = |a_i G T_i^{\perp}| \bar{1}_{n-m+1}$$

Because of the symmetry we find $b_{\min} = -b_i$ and so

$$-b_i \leq a_i^T v \leq b_i$$

□

4.3.2 Examples

Example 4.1 (Example of hypercube projection)

This is an example of the projection of a cube in R^3 onto R^2. The projection matrix is chosen to be

$$G = \begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & 1 \end{bmatrix}$$

with $v = Gu$.

We derive

$$T_i \in \{T_1, T_2, T_3\} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} , \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} , \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

and

$$T_i^{\perp} \in \{T_1^{\perp}, T_2^{\perp}, T_3^{\perp}\} = \left\{ \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} , \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} , \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \right\}$$

Figure 4.1 gives a graphical representation of both the original cube and the image after projection. Indicated in red in the cube are four parallel edges of the cube. They correspond to the case $i = 1$. A description of u lying on these edges can be given as

$$u = T_1 \eta + T_1^{\perp} w = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \eta + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} w$$

where η is a free scalar variable and $w \in \mathbb{B}^2$. The mapping of these edges in $\mathbb{R}^2$ now gives:

$$v = G T_1 \eta + G T_1^{\perp} w = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \eta + \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \pm 1 \\ \pm 1 \end{bmatrix}$$

We compute

$$a_1 = \mathrm{Ker}((G T_1)^T) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

and

$$b_1 = |a_1^T G T_1^{\perp}| \bar{1}_2 = | \begin{bmatrix} 2 & -1 \end{bmatrix} | \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 3$$

Repeating the steps for the other two sets of parallel edges of the cube leads to the polytope description of the image which equals:

$$-\begin{bmatrix} 3 \\ 3 \\ 2 \end{bmatrix} \leq \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} v \leq \begin{bmatrix} 3 \\ 3 \\ 2 \end{bmatrix}$$

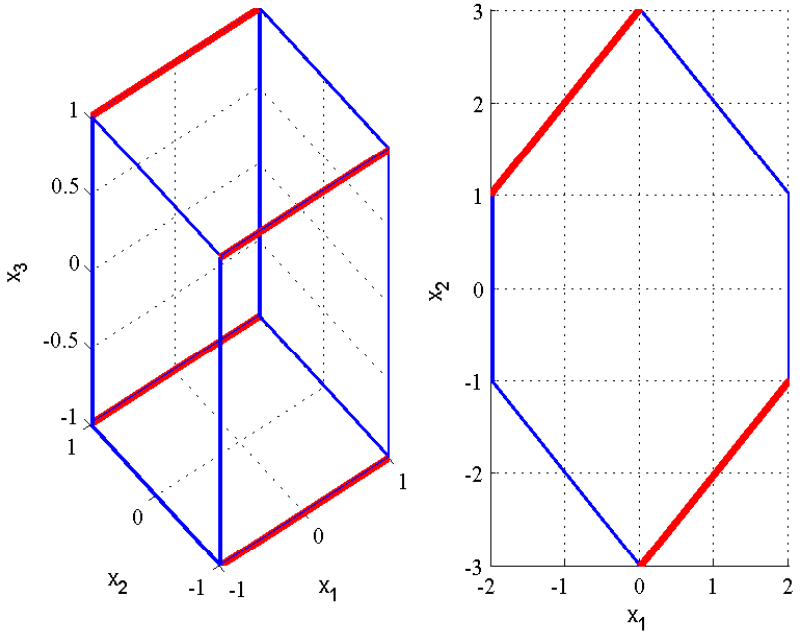

Figure 4.1: Left: original hypercube in $\mathbb{R}^3$, and right the image after projection. Also indicated in the figure in red in the cube are four parallel faces (in this case the edges) that have $n-1$ degrees of freedom, the red lines in the image on the right are the images of the projection of these four edges

Example 4.2 (Example of hypercube projection from $\mathbb{R}^8$ to $\mathbb{R}^2$) *Starting point of this example is the projection matrix $G \in \mathbb{R}^{8\times 2}$:*

$$G = \begin{bmatrix} 23 & -6 & -11 & -6 & 24 & 2 & 22 & -7 \\ -9 & -5 & -10 & 24 & -15 & -37 & 0 & 9 \end{bmatrix}$$

Using the algorithm we derive

$$A = \begin{bmatrix} 0.3644 & 0.9312 \\ -0.6402 & 0.7682 \\ -0.6727 & 0.7399 \\ 0.9701 & 0.2425 \\ 0.5300 & 0.8480 \\ 0.9985 & 0.0540 \\ 0 & 1.0000 \\ 0.7894 & 0.6139 \\ -0.3644 & -0.9312 \\ 0.6402 & -0.7682 \\ 0.6727 & -0.7399 \\ -0.9701 & -0.2425 \\ -0.5300 & -0.8480 \\ -0.9985 & -0.0540 \\ 0 & -1.0000 \\ -0.7894 & -0.6139 \end{bmatrix} \quad b = \begin{bmatrix} 93.1243 \\ 126.6285 \\ 126.3952 \\ 92.8911 \\ 89.3578 \\ 96.5887 \\ 109.0000 \\ 93.4944 \\ 93.1243 \\ 126.6285 \\ 126.3952 \\ 92.8911 \\ 89.3578 \\ 96.5887 \\ 109.0000 \\ 93.4944 \end{bmatrix}$$

In figure 4.2 the resulting 2-dimensional zonotope is given (boundaries are given by the red solid lines). Also the projection of all 1-dimensional faces of the original 8-dimensional hypercube are given (blue dotted lines).

4.3.3 Projection of a hypercube after a linear mapping

Theorem 4.1 can be extended to hold for hypercubes subject to a linear mapping, i.e. a linear translation and/or scaling. A hypercube subject to a linear mapping can be described as follows

$$\mathcal{U} = \{u \in \mathbb{R}^n,\ w \in \mathbb{R}^n \mid u = M\,w + u_0,\ -1 \leq w_i \leq 1, i = 1, \ldots, n\}$$

The projection

$$v = G\,u$$

becomes

$$v = GM\,w + G\,u_0$$

after substitution of the linear mapping. Setting the new variables $v' = v - Gu_0$ and $G' = GM$ leads to the projection of the hypercube in w

$$\mathcal{U}' = \{w \in \mathbb{R}^n \mid\ -1 \leq w_i \leq 1, i = 1, \ldots, n\}$$

onto v' using

$$v' = G'\,w$$

which is identical to the projection problem of Theorem 4.1 and after substitution of the original variables leads to the following description of the image after projection:

$$\mathcal{V} = \{v \in \mathbb{R}^m \mid b_{\min} \leq A\,v \leq b_{\max}\}$$

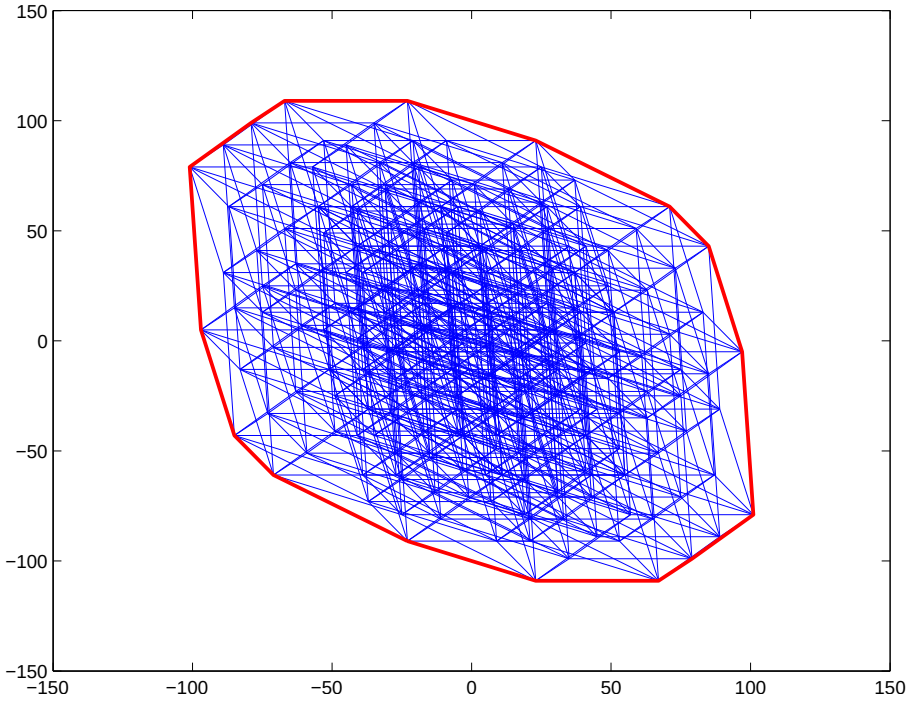

Figure 4.2: Projection of hypercube in $\mathbb{R}^8$ onto the 2-dimensional plane. The projection of all 1-dimensional faces are given as dotted lines. The faces of the 2 dimensional zonotope are given in red.

with

$$A = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_s^T \end{bmatrix}, \quad b_{\min} = \begin{bmatrix} b_{\min,1} \\ b_{\min,2} \\ \vdots \\ b_{\min,s} \end{bmatrix}, \quad b_{\max} = \begin{bmatrix} b_{\max,1} \\ b_{\max,2} \\ \vdots \\ b_{\max,s} \end{bmatrix},$$

where

$$a_i = \mathrm{Ker}((G\,M\,T_i)^T) \in \mathbb{R}^{m\times 1}$$
$$b_{\min,i} = -|a_i^T\,G\,T_i^{\perp}| + ai^T\,G\,u_0$$
$$b_{\max,i} = |a_i^T\,G\,T_i^{\perp}| + ai^T\,G\,u_0$$

4.4 Projection of a convex polytope

In this section we aim for projection of polytopes (constraint sets) as described by Equations (4.3). This can be interpreted as the projection of a convex polytope that is the result of intersection of two hypercubes, both after linear mapping. The

presented theory, however, generalizes to convex polytopes as a whole. Let a convex polytope $\mathcal{W}_\theta$ be given by

$$\mathcal{W}_\theta = \{w \in \mathbb{R}^n \mid A_w w \leq b_w\}. \tag{4.11}$$

where $A_w \in \mathbb{R}^{p\times n}$ and $b_w = \alpha + \beta\,\theta \in \mathbb{R}^{p\times 1}$, where $p > n$ and θ is a free parameter.

Remark 4.1 *Note that this description covers the feasibility regions described in (4.1)-(4.4) where $w = \tilde{\mathbf{u}}$. The vector θ is known at time sample k and contains values of $u(k-1)$, $x(k-1)$ and/or $r(k)$.*

Let $\mathcal{S} = \{S_1, \ldots, S_L\}$ be the set of all $n \times p$ submatrices of the $p \times p$ identity matrix such that the matrix $S_i\,A_w$ is invertible. Let $\xi_i = \{\nu_{i,1}, \nu_{i,2}, \ldots, \nu_{i,n}\}$ be the set of indices of the corresponding selected rows of the identity matrix. Now

$$v_i = (S_i\,A_w)^{-1} S_i \beta\,\theta + (S_i\,A_w)^{-1} S_i \alpha = \tau_i\,\theta + \sigma_i$$

is a vertex of the region $\mathcal{W}_\theta$ if $A_w\,v_i \leq b_w$, or in terms of θ if

$$(A_w (S_i\,A_w)^{-1} S_i - I)\beta\,\theta \leq (I - A_w (S_i\,A_w)^{-1} S_i)\alpha$$

Alternatively, with $R_i = (A_w (S_i\,A_w)^{-1} S_i - I)\beta$ and $Q_i = (I - A_w (S_i\,A_w)^{-1} S_i)\alpha$ we obtain that $z_i = \tau_i\,\theta + \sigma_i$ is an active vertex of $\mathcal{W}_\theta$ if $R_i\,\theta \leq Q_i$. The term *active vertex* in this context indicates that a vertex non-redundant in the description of the polytope $\mathcal{W}_\theta$. All possible vertices z_i are given by the set

$$\mathcal{Z} = \{z_1, \ldots, z_L\}$$

Define the set $\mathcal{L}_\theta = \{\ell_1, \ell_2, \ldots, \ell_{n_\theta}\}$ with all the indices for vertices $z_i \in \mathcal{Z}$ that are active for a given θ. The set of active vertices for a given θ is equal to

$$\mathcal{Z}_\theta = \{z_{\ell_1}, z_{\ell_2}, \ldots, z_{\ell_{n_\theta}}\}$$

Then polytope $\mathcal{W}_\theta$ can now be written as:

$$\mathcal{W}_\theta = \mathrm{Co}(z_{\ell_1}, z_{\ell_2}, \ldots, z_{\ell_{n_\theta}})$$

Now consider the projection of $\mathcal{W}_\theta$ that leads to the set $\mathcal{V} = \{v \in \mathbb{R}^m,\ w \in \mathbb{R}^n \mid v = Gw,\ w \in \mathcal{W}_\theta, m \leq n\}$. This set can now be written as

$$\mathcal{V} = \mathrm{Co}(G\,z_{\ell_1}, G\,z_{\ell_2}, \ldots, G\,z_{\ell_{n_\theta}})$$

Removal of the redundant vertices in $\mathcal{V}$ can for instance be achieved through solving ℓ_n linear programming problems. After removing all redundant vertices, we obtain

$$\mathcal{V}_\theta = \mathrm{Co}(G\,z_{m_1}, G\,z_{m_2}, \ldots, G\,z_{m_{n_v}})$$

where the indices to the vertices in $\mathcal{Z}_\theta$ that lead to the vertices of $\mathcal{V}_\theta$ after projection are given by

$$\mu_{\theta,G} = \{m_1, m_2, \ldots, m_{n_v}\}$$

To find the half-space description of this projected polytope $\mathcal{V}_\theta$ we start to consider all possible $(m-1)$ faces of the polytope $\mathcal{W}_\theta$, as the projection thereof will form the faces of $\mathcal{V}_\theta$.

To find these faces we consider the set $\mathcal{T} = \{T_1, \ldots, T_K\}$ of all $(n-m+1) \times p$ submatrices of the $p \times p$ identity matrix such that for some index q the matrix $T_q\, A$ has full row-rank, which means that $(m-1)$ face q is spanned by at least m vertices from the set $\mathcal{Z}$.

Let $\eta_q = \{\pi_{q,1}, \pi_{q,2}, \ldots, \pi_{q,n-m+1}\}$ be the set of indices of the corresponding selected rows of the identity matrix. Note that if $\eta_q \subset \xi_i$, then z_i is a vertex on the q-th $(m-1)$ face. The indices of the active vertices in $\mathcal{Z}$ (for given θ) that are on the q-th $(m-1)$ face are denoted by

$$\psi_{q,\theta} = \{p_{q,\theta,1}, \ldots, p_{q,\theta,n_q}\}$$

For a given pair θ, G the active vertices that are on the q-th $(m-1)$ face are given by the set

$$\phi_{q,\theta,G} = \mu_{\theta,G} \cap \psi_{q,\theta} = \{\lambda_{q,\theta,G,1}, \ldots, \lambda_{q,\theta,G,\bar{n}}\}$$

If the number of vertices in $\phi_{q,\theta,G}$ is at least equal to m (or $\#(\phi_{q,\theta,G}) \geq m$), then the mapping of the q-th $(m-1)$ face will be a face of the mapped polytope. First we compute a point, strictly inside the final polytope (if the final polytope is not degenerated):

$$z_0 = (G\, z_{m_1} + G\, z_{m_2} + \ldots + G\, z_{m_{n_v}})/n_v\ .$$

Let the set of active $(m-1)$ faces be given by $Q_{\theta,G} = \{\ q \mid \#(\phi_{q,\theta,G}) \geq m\ \} = \{t_1, \ldots, t_M\}$. Compute for all $i = 1, \ldots, M$:

$$a_i = \mathrm{Ker}((\,[\, G\, z_{p_{t_i,\theta,1}}\ \ldots\ G\, z_{p_{t_i,\theta,n_q}}\,])^T)\ \in \mathbb{R}^{m\times 1}, \tag{4.12}$$

and the scalar value

$$b_i = a_i^T\, G\, z_{p_{t_i,\theta,1}},$$

and

$$s_i = \mathrm{sign}(b_i - a_i^T\, z_0)$$

Now define

$$A = \begin{bmatrix} s_1\, a_1^T \\ s_2\, a_2^T \\ \vdots \\ s_M\, a_M^T \end{bmatrix}, \quad b = \begin{bmatrix} s_1\, b_1 \\ s_2\, b_2 \\ \vdots \\ s_M\, b_M \end{bmatrix},$$

then the set $\mathcal{V}$ can be written as

$$\mathcal{V} = \{v \in \mathbb{R}^m \mid A\, v \leq b\}$$

Example 4.3 (Projection of a simplex in $\mathbb{R}^9$ onto $\mathbb{R}^2$) *This example illustrates the theory introduced in Section 4.4 through projecting a (scaled) simplex in $\mathbb{R}^9$ that has the form*

$$\mathcal{W} = \{w \in \mathbb{R}^n \mid A_w w \leq b_w\}$$

with

$$A_w = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \quad b_w = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \end{bmatrix}$$

onto $\mathbb{R}^2$ *with* $v = Gw$ *and the projection matrix*

$$G = \begin{bmatrix} 4.30 & 1.00 & 4.00 & 3.65 & 2.15 & 1.05 & 4.40 & 3.75 & 0.60 \\ 4.55 & 4.30 & 1.70 & 2.70 & 2.00 & 0.65 & 4.05 & 4.90 & 3.15 \end{bmatrix}.$$

The resulting image under projection is

$$\mathcal{V} = \{v \in \mathbb{R}^m \mid A\,v \leq b\}$$

with

$$A = \begin{bmatrix} -0.9823 & 0.1871 \\ 0.3911 & -0.9203 \\ 0.9858 & -0.1678 \\ -0.9445 & 0.3285 \\ -0.2132 & 0.9770 \\ 0.5369 & 0.8437 \\ 0.9806 & 0.1961 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ 7.3161 \\ 0.9363 \\ 7.9760 \\ 12.2944 \\ 10.2177 \end{bmatrix}$$

Figure 4.3 shows the resulting image under projection. It can clearly be seen that the method in Section 4.4 produces the faces of the image (in red lines) and that the projection of all other $(m-1)$*-faces of the original polytope onto* $\mathbb{R}^2$ *produces half spaces that are not a face of the image (dash-dotted in blue).*

4.5 Discussion and conclusion

A computationally efficient method for the projection of convex polytopes to lower dimensional spaces has been introduced that is suitable for the projection of hypercubes. The efficiency advantage stems from the fact that the algorithm foregoes the projection of the vertices, but applies the polytope more efficient half-space description instead. Extension of the method towards generic convex polytopes has been shown to be feasible but is computationally more intensive in the sense that both the vertex description and the half-space description are involved. The complexity increase is bounded, however, due to the fact that removal of redundancies is necessary in one step only. Determination of the exact computational complexity of the method remains future work.

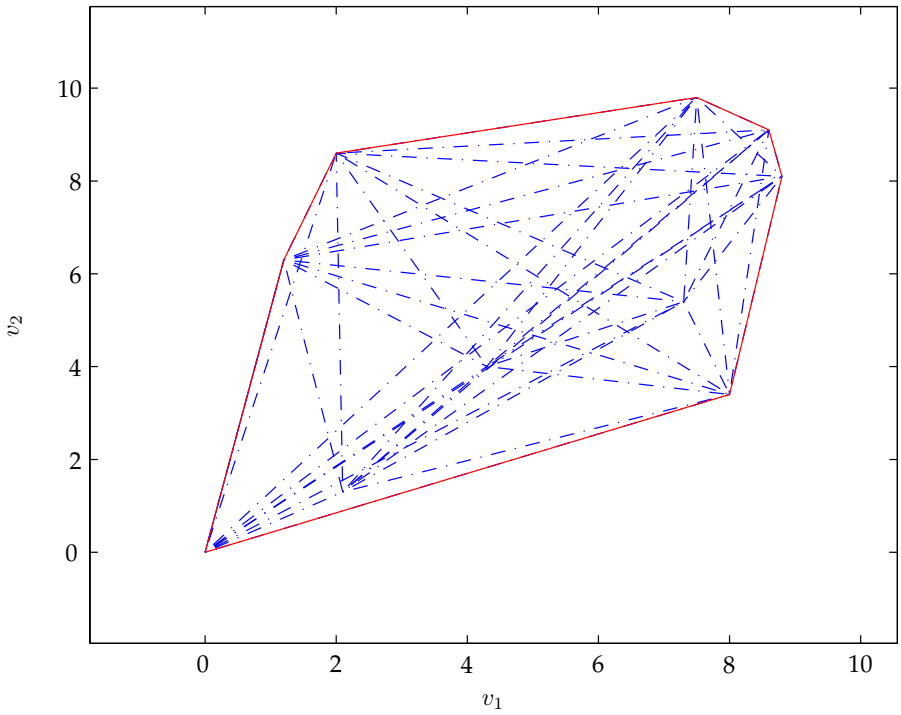

Figure 4.3: Example of the projection of the image under projection onto $\mathbb{R}^9$ of a simplex in $\mathbb{R}^9$. The red lines show the faces of the image, whereas the blue dash-dotted lines show that the projection of all other $(m-1)$-faces of the original polytope are not a face of the image.

5 CHAPTER

Boeing 747 simulation study

This section applies MPC towards fault tolerant flight control. The methods proposed in Chapters 2 and 3 are tested in conjunction with a detailed simulation model of a Boeing 747 aircraft. The particular aircraft type is of interest as it was involved in one of the most well-known aircraft disasters of the Netherlands, i.e. the Bijlmerramp, that occurred in Amsterdam, the Netherlands in 1992.

5.1 Introduction to the Boeing 747 model

The benchmark simulation model applied in this thesis is a precursor to the REconfigurable COntrol for Vehicle Emergency Return (RECOVER) Matlab/Simulink model described in (Edwards et al. 2010, Section 6.3 onward). The simulation model includes a generic 6 degree of freedom (DOF) nonlinear aircraft model and aircraft specific modules including aerodynamics, flight control system and engines. The baseline flight control system model reflects the hydro-mechanical system architecture of the Boeing 747-100/200. Also included are sensor models, an autopilot and a mechanism to introduce different (failure scenarios to the plant). The aircraft specific modeling data and parameters are based on the data presented by Hanke (1970, 1971). The simulation model includes an classical autopilot, sensors models and has been modified to resemble a *fly by wire* aircraft in the sense that the original 11 control inputs have been separated such that each of the 26 aerodynamic surfaces and the four engines can be controlled separately. A schematic of the simulation model is depicted in Figure 5.1.

5.2 Modeling the benchmark

As MPC will be employed in discrete time a nonlinear set of equations for a fixed wing aircraft is introduced here in discretized form. This is not the standard pre-

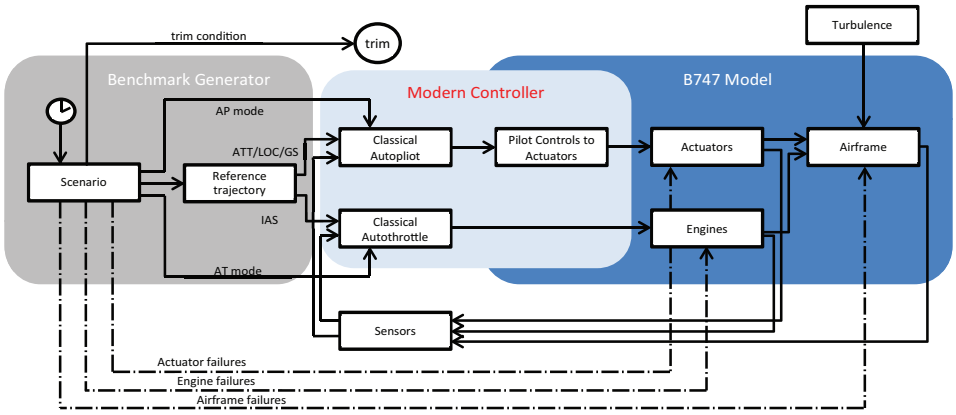

Figure 5.1: Detailed schematic of the GARTEUR benchmark showing model component relationships including test maneuver and failure scenario generation and fault injection (adapted from Edwards et al. (2010, p.197))

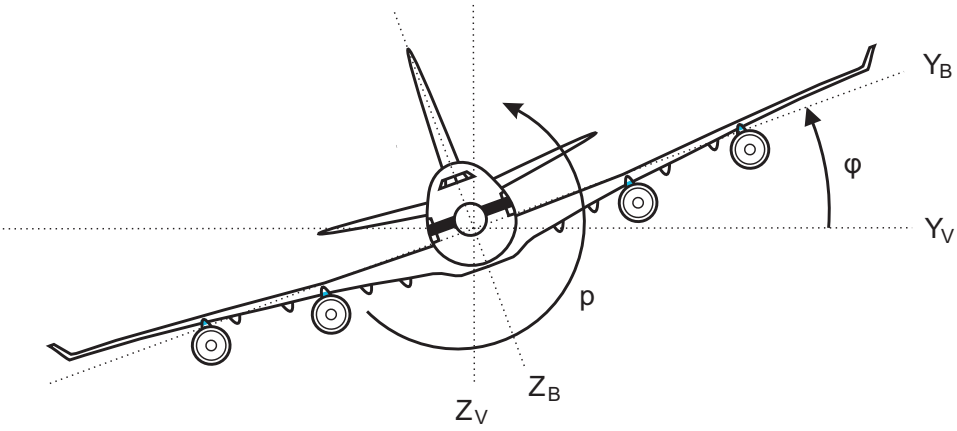

Figure 5.2: Front view of the Boeing 747. (Hallouzi 2008)

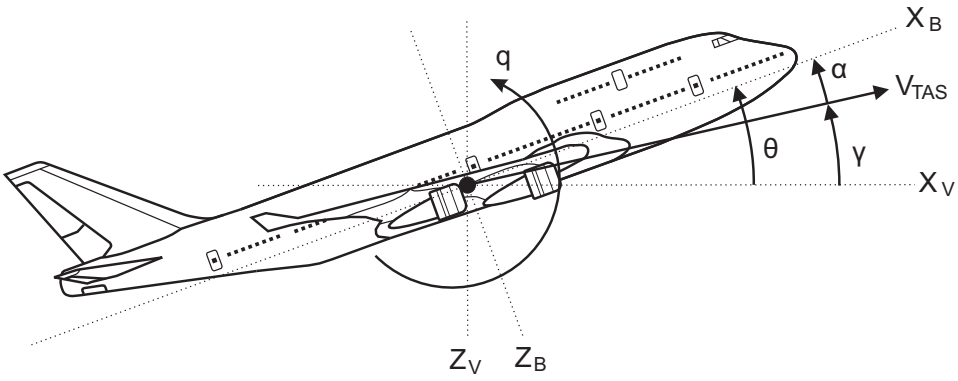

Figure 5.3: Side view of the Boeing 747. (Hallouzi 2008)

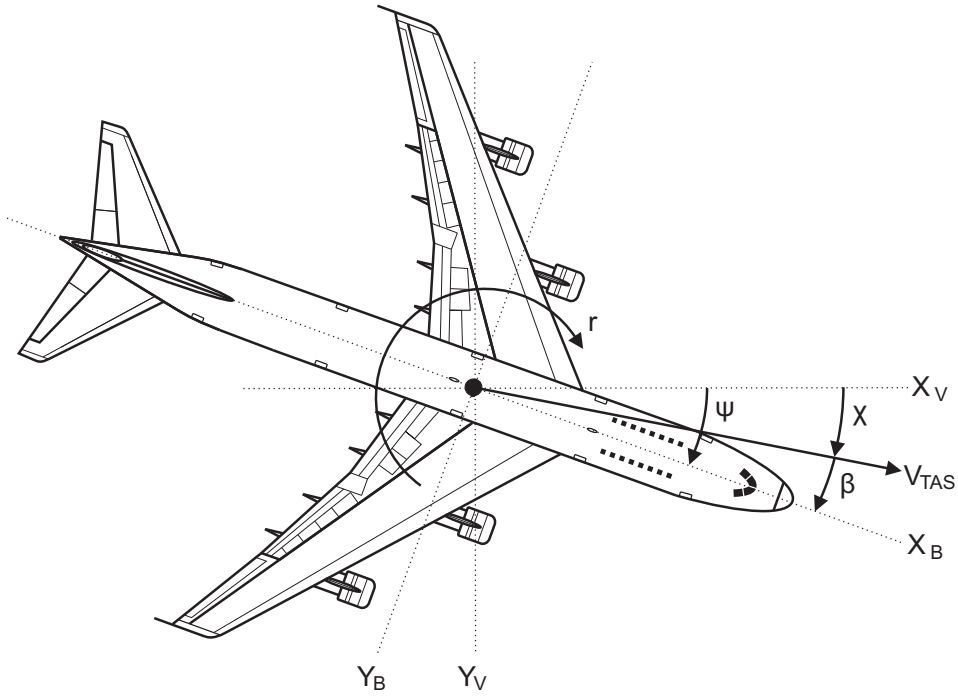

Figure 5.4: Top view of the Boeing 747. (Hallouzi 2008)

Table 5.1: aircraft states

p	roll rate
q	pitch rate
r	yaw rate
V_{TAS}	true airspeed
α	angle of attack
β	sideslip angle
γ	flight path angle
θ	pitch angle
ϕ	roll angle
ψ	yaw angle
χ	tracking angle

Table 5.2: inputs of the Boeing 747 model

Control no.	Description	Mechanical limits	Operation rate
1.	Right inner aileron	-20/20 deg	40/45 deg/s (↓/↑)
2.	Left inner aileron	-20/20 deg	40/45 deg/s (↓/↑)
3.	Right outer aileron	-25/25 deg	45/55 deg/s (↓/↑)
4.	Left outer aileron	-25/25 deg	45/55 deg/s (↓/↑)
5.	Spoiler panel # 1	0/45 deg	75 deg/s
6.	Spoiler panel # 2	0/45 deg	75 deg/s
7.	Spoiler panel # 3	0/45 deg	75 deg/s
8.	Spoiler panel # 4	0/45 deg	75 deg/s
9.	Spoiler panel # 5	0/20 deg	75 deg/s
10.	Spoiler panel # 6	0/20 deg	25 deg/s
11.	Spoiler panel # 7	0/20 deg	25 deg/s
12.	Spoiler panel # 8	0/20 deg	75 deg/s
13.	Spoiler panel # 9	0/45 deg	75 deg/s
14.	Spoiler panel # 10	0/45 deg	75 deg/s
15.	Spoiler panel # 11	0/45 deg	75 deg/s
16.	Spoiler panel # 12	0/45 deg	75 deg/s
17.	Right inner elevator	-23/17 deg	37 deg/s
18.	Left inner elevator	-23/17 deg	37 deg/s
19.	Right outer elevator	-23/17 deg	37 deg/s
20.	Left outer elevator	-23/17 deg	37 deg/s
21.	Stabilizer	-12/3 deg	0.2 to 0.5 deg/s
22.	Upper rudder surface	-25/25 deg	50 deg/s
23.	Lower rudder surface	-25/25 deg	50 deg/s
24.	Outer flaps	0/25 deg	1.8 deg/s
25.	Inner flaps	0/25 deg	1.8 deg/s
26.	Thrust engine # 1	0.7/1.7 EPR[1]	-
27.	Thrust engine # 2	0.7/1.7 EPR	-
28.	Thrust engine # 3	0.7/1.7 EPR	-
29.	Thrust engine # 4	0.7/1.7 EPR	-

EPR = Engine Pressure Ratio

sentation of aircraft flight dynamics in that sense. The reader is directed to Cook (2003) and Mulder et al. (2006) as discretization of nonlinear dynamic systems is not at all trivial. In this chapter the nonlinear system is sampled with sampling interval T and first order Euler integration is applied. The difference equation (3.1) is obtained from the original nonlinear system equation as follows

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \approx \frac{x(k+1) - x(k)}{T} \qquad (5.1) \\ &\Leftrightarrow \\ x(k+1) &\approx Tf(x(k)) + x(k) + Tg(x(k))u. \qquad (5.2) \end{aligned}$$

It is acknowledged here that the Euler method, which is a first-order method, is typically associated with an integration error that is proportional tot the sampling interval T. This makes the Euler method less accurate than higher order methods such as the Runge-Kutta type methods. There are two specific reasons why Euler's method is applied here. For one, use of higher order methods would complicate the dynamic inversion of the nonlinear aircraft model in Section 5.6.1 unnecessarily. Next to that, and more importantly, the simulation settings for the benchmark model are such that the Euler method is applied in the simulation. Hence, the Euler method is chosen over higher-order methods for discretization.

First, we model the discretized but nonlinear equation for the airspeed V of the benchmark aircraft and linearize this. Subsequently, we perform the same actions for the equations that belong to the three attitude states. Additionally, in the first instance we will assume that the forces (X, Y, Z) and moments (L, M, N), that enter the system equations, are inputs to the system.

The nonlinear and discretized state equation for the airspeed is given as follows:

$$V(k+1) = V(k) + \frac{T}{m_{ac}} \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix}^T \begin{bmatrix} F_X(k) \\ F_Y(k) \\ F_Z(k) \end{bmatrix} \qquad (5.3)$$

where α and β are the angle of attack and sideslip angle, respectively, and m_{ac} is the mean aerodynamic chord. The variable T is again the sampling interval.

$$\begin{bmatrix} \phi(k+1) \\ \theta(k+1) \\ \psi(k+1) \end{bmatrix} = T \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p(k) \\ q(k) \\ r(k) \end{bmatrix} + \begin{bmatrix} \phi(k) \\ \theta(k) \\ \psi(k) \end{bmatrix}, \qquad (5.4)$$

where ϕ, θ, ψ are the roll-, pitch and yaw angle and p, q, r are the roll-, pitch- and yaw rate. The following equations govern the states p, q, r,

$$\begin{aligned} \begin{bmatrix} p(k+1) \\ q(k+1) \\ r(k+1) \end{bmatrix} &= \left(-TJ_{ac}^{-1} \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} J_{ac} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} p(k) \\ q(k) \\ r(k) \end{bmatrix} \\ &\quad + TJ_{ac}^{-1} \begin{bmatrix} M_L(k) \\ M_M(k) \\ M_N(k) \end{bmatrix}, \qquad (5.5) \end{aligned}$$

where

$$J_{ac} = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{bmatrix} \tag{5.6}$$

and where I_{**} indicates the inertia.

What remains now is to introduce expressions for the forces $F = [F_X, F_Y, F_Z]^T$ and moments $M = [M_L, M_M, M_N]^T$. The forces are the sum of the external forces and the contribution of the aerodynamics, and the moments are dependent of the aerodynamics only, which leads to the expressions:

$$F = F_{grav} + F_{wind} + F_{aero} + T, \tag{5.7}$$

$$M = M_{aero}, \tag{5.8}$$

where the subscripts indicate the contribution of gravity, the wind and the aerodynamic model, respectively. We model the aerodynamics as follows

$$\begin{aligned} F_{aero} &= \frac{1}{2}\rho V^2 S(C_{F_x}[1 \quad \alpha \quad \alpha^2 \quad \alpha^3 \quad \beta \quad \beta^2 \; \beta^3 \quad \frac{pb}{2V} \quad \frac{qc}{2V} \quad \frac{rb}{2V}]^T \\ &\quad + C_{F_u} u), \end{aligned} \tag{5.9}$$

$$\begin{aligned} M_{aero} &= \frac{1}{2}\rho V^2 S \begin{bmatrix} b & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & b \end{bmatrix} (C_{M_u} u + C_{M_x} \times \\ &\quad \left[1 \; \alpha \; \alpha^2 \; \alpha^3 \; \beta \; \beta^2 \; \beta^3 \; \tfrac{pb}{2V} \; \tfrac{qc}{2V} \; \tfrac{rb}{2V}\right]^T \Big), \end{aligned} \tag{5.10}$$

where ρ is the air density, S, b, c are the wing area, wing span and wing chord, respectively. The input variable u is a vector composed of the control surfaces and engines of the aircraft. In this chapter we make use of a subset of these control effectors. In this particular case we apply our controller to the four elevator surfaces, the four ailerons, the two rudder halves and the four engines, hence $u \in \mathbb{R}^{14}$.

The aerodynamic parameters $C_{F_x}, C_{M_x} \in \mathbb{R}^{3\times 10}$ and $C_{F_u}, C_{M_u} \in \mathbb{R}^{3\times 14}$ are determined online through a recursive identification method, using the approach presented by Lombaerts et al. (2009). Although not strictly required in the nominal and failure-free case, the identification method is applied in both the nominal and the failure case. Because of the fact we apply data from recursive identification, we do not have to model failures explicitly. As an example one might consider a rudder that has become stuck. Such a failure will result in a change in the basic aerodynamic parameters to account for the static aerodynamic moment that this creates. Furthermore the effectiveness of the rudder itself will be reduced to zero. Additionally, although not applied here, it is possible to include direct knowledge of actuator failures in the controller. The uncertainty caused by failures of the aircraft structure or actuators is considered to be small because of the relatively fast response of the identification algorithm. Gravity is incorporated as follows

$$F_{grav} = \begin{bmatrix} -mg\sin\theta \\ mg\sin\phi\cos\theta \\ mg\sin\cos\cos\theta \end{bmatrix} \tag{5.11}$$

5.3 Autopilot model

In contrast to the plant model, a linear representation of the autopilot (for the given flight condition) was manually extracted from the simulation model. A minimal realization of this controller has 15 states, 9 inputs and 14 outputs. The closed loop of the controller and linear aircraft is asymptotically stable and both the linear controller and aircraft were evaluated in closed loop with, or in comparison with, the original nonlinear aircraft model. The latter has led to good confidence in both the model and the controller. Both the controller and the aircraft model were discretized using zero order hold sampling (ZOH) with a sampling interval $h = .05$ [s].

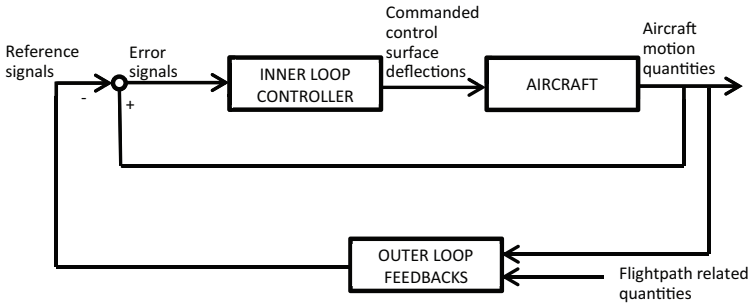

Figure 5.5: General autopilot structure for the Boeing 747-200, adapted from van Keulen (1991)

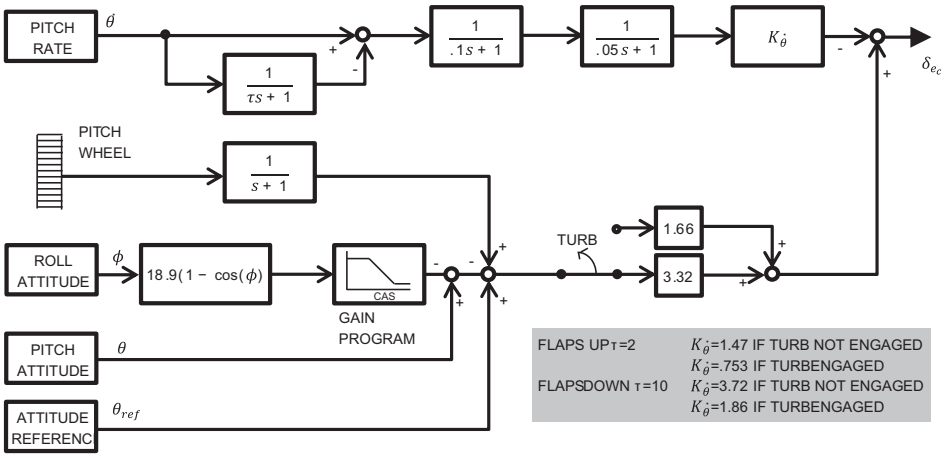

Figure 5.6: Boeing 747-200 AFCS, inner loop, pitch attitude hold, adapted from van Keulen (1991)

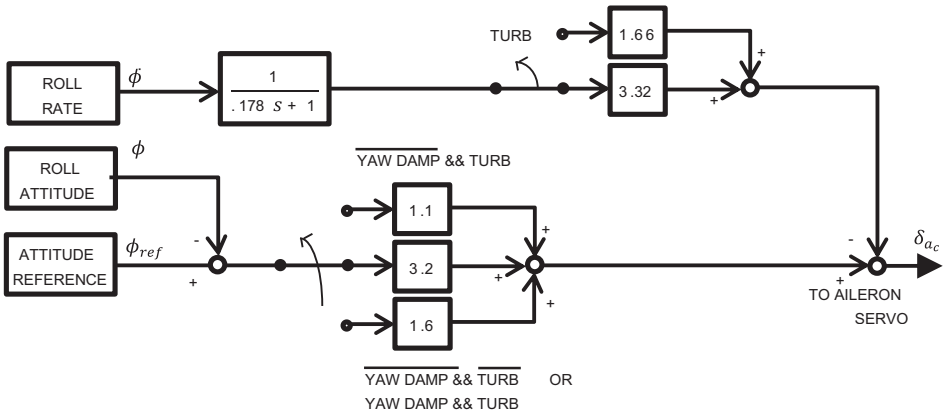

Figure 5.7: Boeing 747-200 AFCS, inner loop, roll attitude hold, adapted from van Keulen (1991)

aileron control

$$\begin{bmatrix} \delta_{air} \\ \delta_{ail} \\ \delta_{aor} \\ \delta_{aol} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} [-T_1 p - K_1 \phi + T_2 \psi] \tag{5.12}$$

elevator control

$$\begin{bmatrix} \delta_{eir} \\ \delta_{eil} \\ \delta_{eor} \\ \delta_{eol} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} [-T_3 q + T_5 T_4 K_2 h - T_5 T_4 K_3 \dot{h} - T_5 \theta] \tag{5.13}$$

rudder control (yaw damper)

$$\begin{bmatrix} \delta_{ru} \\ \delta_{rl} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} [-T_9 (T_7 + T_8) \phi + T_9 T_6 \dot{\psi}] \tag{5.14}$$

throttle (auto-throttle)

$$\begin{bmatrix} \delta_{Tn_1} \\ \delta_{Tn_2} \\ \delta_{Tn_3} \\ \delta_{Tn_4} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} [T_{10} V] \tag{5.15}$$

$$T_1 = \frac{-825.1}{0.178s+1} \tag{5.16}$$

$$T_2 = \frac{793.5s+634.8}{2.5s+1} \tag{5.17}$$

$$T_3 = \frac{-149.2s}{0.01s^3+0.305s^2+2.15s+1} \tag{5.18}$$

$$T_4 = \frac{0.0175s+0.0011}{0.65s^2+s} \tag{5.19}$$

$$T_5 = \frac{-167.4s-33.48}{s} \tag{5.20}$$

$$T_6 = \frac{579.3}{s^2+4.048s+1.354} \tag{5.21}$$

$$T_7 = \frac{21.72}{s^2+4.048s+1.354} \tag{5.22}$$

$$T_8 = \frac{3953}{s^2+20s+100} \tag{5.23}$$

$$T_9 = \frac{31.2s+0.3744}{s^2+32.1s+3.2} \tag{5.24}$$

$$T_{10} = \frac{-0.2s-0.01}{s} \tag{5.25}$$

5.4 Taking MPC towards fault tolerance

We assume that FDI information is available with respect to the status of the inputs and possible faults thereof is available (possibly with a delay). FDI, without a doubt, is a very difficult aspect of FTC, but with ever more complex condition monitoring systems becoming available it is possible to assume that such information is available. In this section we focus specifically at incorporation of actuator failures. To be able to provide alternatives when primary controls fail we need sufficient control effector redundancy. Therefore we first concentrate on that issue.

5.4.1 Control effector redundancy

An aircraft can rotate freely about three axes. In normal flight the rotation around the lateral axis, called pitch, is controlled by the elevators, the rotation around the longitudinal axis passes, called roll is mainly controlled by the ailerons (rudder also has a secondary effect on the roll angle), and finally the rotation around the vertical axis, called yaw, is mainly controlled by the rudder (the ailerons also have a secondary effect on the yaw angle). In the case of an aircraft with disabled flight controls, we can use the redundancy in the control surfaces to control the aircraft. For example, in normal flight, the pitch angle is controlled by the elevators. If there is a failure in the elevators, one can control the pitch angle using thrust if

basic angle	primary control surface	back up control surface
pitch angle	elevators	thrust, ailerons
yaw angle	rudder	thrust
roll angle	ailerons	rudder, elevators

Table 5.3: primary and back up (or secondary) control surfaces for the rotations about three axis of the aircraft.

the engines are mounted under the center of gravity. Increasing the thrust will increase the pitch angle, decreasing the thrust will decrease the pitch angle. If the ailerons are decoupled, they can also be used to control the pitch angle. Furthermore, if the engines are used in a asymmetrical way, the pilot can, in case of a rudder failure, use differential thrust to control the yaw angle of the aircraft. In the same way, if we use the elevators and ailerons in an asymmetric way, we can use the ailerons to control the pitch angle and the elevators to control the roll angle. In Table 5.3 the primary and back up (or secondary) control surfaces are given for the rotations about three axis of the aircraft.

The primary control inputs of an aircraft are thrust ($\Delta u_{\rm thr}$), ailerons ($\Delta u_{\rm ail}$), elevators ($\Delta u_{\rm ele}$) and rudder ($\Delta u_{\rm rud}$). Let us first consider the decoupling of the ailerons. In nominal operation, the values of the left and right aileron are asymmetric, so

$$\Delta u_{\rm ail,l} = \Delta u_{\rm ail} \qquad\qquad \Delta u_{\rm ail,r} = -\Delta u_{\rm ail}$$

where $\Delta u_{\rm ail}$ is the nominal control variable. We add a control variable $u_{a,2}$ to be able to decouple the ailerons:

$$\Delta u_{\rm ail,l} = \Delta u_{\rm ail} + u_{a,2} \qquad\qquad \Delta u_{\rm ail,r} = -\Delta u_{\rm ail} + \Delta u_{a,2}$$

For nominal flight we do not need this additional variable so then $\Delta u_{a,2} = 0$.

We follow a similar procedure for the elevators. In nominal operation, the values of the left and right aileron are symmetric, so

$$\Delta u_{\rm ele,l} = \Delta u_{\rm ele} \qquad\qquad \Delta u_{\rm ele,r} = \Delta u_{\rm ele}$$

where $\Delta u_{\rm ele}$ is the nominal control variable. We add a control variable $\Delta u_{e,2}$ to be able to decouple the elevators:

$$\Delta u_{\rm ele,l} = \Delta u_{\rm ele} + \Delta u_{e,2} \qquad\qquad \Delta u_{\rm ele,r} = \Delta u_{\rm ele} - \Delta u_{e,2}$$

In nominal operation all four engines are controlled by the same thrust value $\Delta u_{\rm thr}$. In the perturbed case we add three new variables $\Delta u_{t,2}$, $\Delta u_{t,3}$, and $\Delta u_{t,4}$ and decouple the engines:

$$\begin{aligned}
\Delta u_{\rm thr,1} &= \Delta u_{\rm thr} + \Delta u_{t,2} + \Delta u_{t,4} \\
\Delta u_{\rm thr,2} &= \Delta u_{\rm thr} + \Delta u_{t,3} - \Delta u_{t,4} \\
\Delta u_{\rm thr,3} &= \Delta u_{\rm thr} - \Delta u_{t,3} - \Delta u_{t,4} \\
\Delta u_{\rm thr,4} &= \Delta u_{\rm thr} - \Delta u_{t,2} + \Delta u_{t,4}
\end{aligned}$$

where Δu_{thr} is the nominal control variable. For nominal flight we do not need this additional variable so then $\Delta u_{t,2} = \Delta u_{t,3} = \Delta u_{t,4} = 0$.

The new variables have to be included in the MPC problem. The number of variables will increase. Let $\Delta u_o = \begin{bmatrix} \Delta u_{\mathrm{ail}} & \Delta u_{\mathrm{ele}} & \Delta u_{\mathrm{thr}} & \Delta u_{\mathrm{rud}} \end{bmatrix}^T$ be the vector with the original four control variables (elevator, aileron, thrust, and rudder) and let the 4×4 matrix R be the weighting matrix in cost function (2.4) on page 31. When extending the vector with the new additional variables

$$\Delta u_{add} = \begin{bmatrix} \Delta u_{a,2} & \Delta u_{e,2} & \Delta u_{t,2} & \Delta u_{t,3} & \Delta u_{t,3} \end{bmatrix}^T \tag{5.26}$$

we have to apply a new larger vector D_z to include the new variables into the cost signal (1.5) on page 16. By choosing $D_{z,add}$ with sufficiently large entries we guarantee that even with the new variables included, the nominal behavior will remain the same in unperturbed flight. The new cost signal is now given by

$$z(k) = C_z\, x(k) + D_z\, \Delta u(k) + D_{z,add}\, \Delta u_{add}(k) + E_z\, e(k) + F_z\, r(k) \tag{5.27}$$

Using this updated MPC formulation we obtain a new MPC controller that will give the same original behavior, but now with the possibility to add constraints that may reflect failures into the problem with sufficient control effector redundancy to obtain the same performance in the perturbed case.

5.4.2 How to include failures into the MPC problem

We will focus on actuator faults in this chapter. Several types of actuator failures exist: stuck, reduced rates, reduced range and floating inputs cover the spectrum. We will mainly be looking at failures that can be accounted for through adaptation of the input constraints.

As to model changes due to structural failures of the system, we will not actively discuss these in this chapter because it is logical that the existing controller is not a priori stabilizing for the plant subject to structural damage. We will, however, rely on the inherent passive robustness of the MPC controller.

Sensor failures are not investigated, whilst we assume these to be available in redundant quantities in our flight control examples.

In normal operating conditions, the optimization algorithm provides an optimal solution within acceptable ranges and limits of the constraints. A drawback of MPC is that we are using (hard) constraints, which may lead to infeasibility: There is no possible control action without violation of the constraints. When there is no feasibility guarantee, stability cannot be proven. If a solution does not exist within the predefined ranges and limits of the constraints, the optimizer should have the means to recover from the infeasibility. Three well-known algorithms to handle infeasibility are the soft-constraint approach (Zheng and Morari 1995), the minimal recovery time approach (Rawlings and Muske 1993), and the constraint prioritization approach (Vada et al. 1999). The feasibility recovery technique we

will discuss is based on the priorities of the constraints. The constraints are ordered from lowest to highest priority. In terms of flight control this means that constraints related to passenger comfort and energy consumption will have a low priority, where control actuator limitations and structure integrity of the aircraft have a high priority. If the (nominal) optimization problem becomes infeasible we start by dropping the lowest constraints and see if the resulting reduced optimization problem becomes feasible. As long as the problem is not feasible we continue by dropping more and more constraints until the optimization is feasible again. This means we solve a sequence of quadratic programming problems in the case of infeasibility. The algorithm minimizes the violations of the constraints which cannot be fulfilled. Note that it may take several trials of dropping constraints and trying to find an optimal solution, which is not desirable in any real time application.

An example of such a disaster involved the El Al Boeing 747 freighter aircraft that crashed in the Bijlmermeer-area, near Amsterdam, The Netherlands, in 1992. In this particular case separation of the two right wing engines caused significant loss of controllability and, next to that, structural changes, that eventually led to the crash. A simulation study has shown that this failure was likely to be survivable, given the correct control inputs and a wisely chosen trajectory (Smaili 1997). This example, and many others, have clearly indicated that it is desirable to develop automatic fault-tolerant flight control mechanisms that can assist the pilots in extraordinary situations.

The introduction of fly-by-wire systems has created the possibility to redistribute control effort over the actuators in an automated fashion. It has been the increase of computational power of such flight control systems that now enables the investigation of fault-tolerant (FTFC) techniques. An overview of control methods that can be applied for FTFC purposes has been compiled by Jones (2005). The latter reference makes a clear distinction between passive and active methods. Passive methods are designed to accommodate failures through control design that is robust with respect to a set of system failures which has been defined a priori. Active methods, on the other hand, assume that a fault detection and identification (FDI) method is available that provides online failure information such that the FTFC controller can be adapted online.

This chapter investigates the use of MPC for FTFC purposes with application to a simulation model that represents the aircraft that was involved in the previously mentioned disaster which took place over Amsterdam in 1992. The successful use of MPC for this aircraft in the given accident scenario has been shown by Maciejowski and Jones (2003), yet at the same time it was mentioned as one of their main conclusions, that considerable tuning of the MPC cost-function had been required in order to achieve the good results that were obtained. The latter observation has led to the notion that an existing controller can be used to obtain a state-observer and initial tuning of an MPC cost-function (Maciejowski (2007)) through a process of reverse-engineering. The application of such a process of reverse engineering as to obtain a valid MPC cost function is the main interest of Section 5.5. The readily available autopilot will be used as a basis for this MPC design. Once obtained, this cost-function will be applied in an example in order

to investigate its usefulness in simulations with constraints present.

The chapter continues with the introduction of simulation results on the theory of NDI and MPC in Section 5.6. Section 5.6.1 introduces the relevant equations of motion of the benchmark aircraft and applies the NDI theory to these. The chapter wraps up with a discussion and conclusion in Section 5.7.

5.5 MPC reverse engineering: a simulation example

Two important design considerations come into play when reverse-engineering the existing controller into observer based form. First of all, the controller is not strictly proper. In this case this problem has been tackled through addition of a unit delay in three channels, hence leading to a total augmented controller having 18 states. Next to that, the controller has built-in integral action, hence requiring augmentation of the plant model with a number of states equal to the number of inputs, thus leading to a plant model with 28 states and disturbance states. Despite starting out with a controller order greater than the plant order ($n_K > n$), we end up with a plant controller pair for which $n_K \leq n$. Both the augmented controller and plant have been used to obtain a state observer and MPC cost-function as per section 2.3.2. Initially the prediction horizon was selected to be $N = 10$.

An experiment was performed in order to verify the quality of obtained MPC controller in comparison to the linear time invariant version of the autopilot. The example consists of a reference tracking task. The aircraft is initialized at an altitude of $h = 980$ [m] whilst flying straight and level with an airspeed of approximately $V = 92$ [m.s^{-1}]. Subsequently, at $t = 10$ [s] a right hand turn of 180 [deg] is initiated and after $t = 150$ [s] a descent toward $h = 50$ [m] is initiated. Figure 5.8 shows a subset of 6 of the aircraft's states when flown in closed loop with either the nominal controller or the MPC controller. Three different trajectories can be distinguished from the figure: the nominal response of the aircraft in closed loop with the original LTI controller, the same situation when using the MPC controller and, finally, a situation in which one outboard aileron is blocked in its neutral position. All three trajectories show that it is possible to fly the same trajectory with the reverse-engineered controller and that the state responses match relatively closely. It must be remarked, however, that in all situations the input constraints had to be softened after one step of the prediction horizon such that the controller and plant continued to show stable behavior. An investigation of this fact suggests that either the quality of the estimated states is not good enough, or that the chosen MPC cost function and the desire to include constraints while tracking a reference do not coincide well.

5.6 MPC and NDI: Simulation Results

5.6.1 Dynamic inversion of the benchmark model

Starting with the airspeed equation 5.3 we choose, using the notational convention of Section 3.2.2, the virtual input z_1 as

$$z_1(k) = \frac{T}{m_{ac}} \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix}^T \begin{bmatrix} F_X(k) \\ F_Y(k) \\ F_Z(k) \end{bmatrix}, \tag{5.28}$$

such that when

$$z_1(k) = (a_{des} - 1)V(k) + \nu_1(k), \tag{5.29}$$

the state equation 5.3 becomes linear and is represented as

$$V(k+1) = a_{des}V(k) + \nu_1(k). \tag{5.30}$$

Performing NDI for the attitude states requires some additional steps, whilst they do not depend on the external forces and moments directly. Starting are the equations for the attitude states as in 5.4. In order to apply NDI we shift these equations one step in time in order to arrive at

$$\begin{bmatrix} \phi(k+2) \\ \theta(k+2) \\ \psi(k+2) \end{bmatrix} = T \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} (k+1) \times \begin{bmatrix} p(k+1) \\ q(k+1) \\ r(k+1) \end{bmatrix} + \begin{bmatrix} \phi(k+1) \\ \theta(k+1) \\ \psi(k+1) \end{bmatrix},$$

substituting 5.6 in order to arrive at

$$\begin{aligned} \begin{bmatrix} \phi(k+2) \\ \theta(k+2) \\ \psi(k+2) \end{bmatrix} = & \begin{bmatrix} \phi(k+1) \\ \theta(k+1) \\ \psi(k+1) \end{bmatrix} + T \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} (k+1) \\ & - \left(TJ_{ac}^{-1} \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} J_{ac} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} p(k) \\ q(k) \\ r(k) \end{bmatrix} \\ & + TJ_{ac}^{-1} \begin{bmatrix} M_L(k) \\ M_M(k) \\ M_N(k) \end{bmatrix}. \end{aligned} \tag{5.31}$$

Using the same method that was applied for the airspeed, we choose the virtual input

$$z_2(k) = TJ_{ac}^{-1} \begin{bmatrix} M_L(k) \\ M_M(k) \\ M_N(k) \end{bmatrix}. \tag{5.32}$$

Choosing this virtual input to equal

$$z_2(k) = (A_{des} - I) \begin{bmatrix} \phi(k+1) \\ \theta(k+1) \\ \psi(k+1) \end{bmatrix} - T \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} (k+1) - \left(TJ_{ac}^{-1} \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} J_{ac} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} p(k) \\ q(k) \\ r(k) \end{bmatrix}, \quad (5.33)$$

leads to the linear and time-invariant closed-loop behavior

$$\begin{bmatrix} \phi(k+2) \\ \theta(k+2) \\ \psi(k+2) \end{bmatrix} = (A_{des} - I) \begin{bmatrix} p(k+1) \\ q(k+1) \\ r(k+1) \end{bmatrix} + \nu_2(k), \quad (5.34)$$

where $A_{des} \in \mathbb{R}^{3\times3}$ is the desired linear time invariant behavior and where ν_2 is the input to the linearized system. At this stage we may conclude that when z_1 and z_2 satisfy equation (5.29) and (5.33) that the linear state behavior equals

$$\begin{bmatrix} V(k+1) \\ \phi(k+2) \\ \theta(k+2) \\ \psi(k+2) \end{bmatrix} = \begin{bmatrix} a_{des} & 0 \\ 0 & A_{des} \end{bmatrix} \begin{bmatrix} V(k) \\ \phi(k+1) \\ \theta(k+1) \\ \psi(k+1) \end{bmatrix} + \begin{bmatrix} \nu_1(k) \\ \nu_2(k) \end{bmatrix}. \quad (5.35)$$

In summary, we may apply MPC to the linear system of equation (5.35), provided that the input u from (5.9)-(5.10) is allocated such that the virtual inputs z_1, z_2 in (5.28) and (5.32) satisfy equations (5.29) and (5.33). Additionally, the physical constraints are entered into the problem to arrive at the MPC problem (3.16,3.17,3.18) and the control allocation and weighting problem (3.28) from Section 3.2.

In this section we evaluate the performance of the combination of MPC and NDI as a reconfigurable control method. We do so in two individual examples. The first example involves a so-called stabilizer runaway of the benchmark aircraft. The second example shows the simulation results when one of the maneuvers from the benchmark assessment criteria is flown.

5.6.2 Reference tracking: stabilizer runaway

Here, it will be shown that the control strategy proposed in this chapter allows retention of a trim condition and tracking of a reference with the benchmark aircraft in the event of a failure.

In this particular example, it is shown that a combination of the reconfigurable controller and the online identification algorithm can retain stability after the introduction of the stabilizer runaway failure at time $t = 10$ [s]. At this time the stabilizer moves to its extreme trim angle of 2°. Next to that, it is shown that, despite the stabilizer being inoperative and stuck at an extreme position, it is still

possible to track a doublet-like reference signal with the pitch rate q [rad/s] using another combination of the control surfaces.

The states that are controlled are the roll attitude ϕ, the pitch attitude θ and the yaw attitude ψ, respectively. The inputs that are used in this example are the four different aileron surfaces, the four elevator surfaces, the two rudder surfaces, and the stabilizer trim angle. The other inputs, including the engines, remain at their trim value for the initial condition.

Figure 5.9 depicts the results that were obtained in simulation. Several important notions can be derived from this figure. First of all, it can be seen from the figure that, although the online identification is initialized with data that was obtained off-line, it takes approximately 3 [s] for the closed loop to stabilize the system for the reference state $p, q, r = 0$. Furthermore, it is clear to see, that although a failure is introduced at $t = 10$ [s] relatively little effect is noticeable in the state-response. The latter indicates that the controller successfully succeeds at redistributing the desired control effort over the remaining control surfaces and that the FDI algorithm identifies the new situation quickly. And finally, it is easily seen from the figure that in spite of the failure of the stabilizer, it is still possible to track a reference on the pitch rate. It is assumed that extensive tuning of parameters like the state- and input weighting matrices Q, Q_u, R_u, the selected sampling interval T, and the prediction horizon N will lead to greatly improved tracking behavior.

What remains to be said about this example is that the computational complexity of the control method is quite high. It is expected that this can be greatly improved upon through a more efficient implementation of the controller. Furthermore, although not visible in the provided results, the online identification algorithm suffers from lack of excitation when the system is controlled to be in steady-state for extended periods of time. Both of these issues are not addressed in here, but will be the topic of future research.

5.6.3 Right turn and localizer intercept

What may be concluded from the previous example is that the method is very much dependent of the quality of the model that is identified online. This holds particularly true for control based on NDI in this setting. Because of the fact that the aircraft is simulated in closed loop with the controller, it is also very important that the quality of the initial estimate of the aircraft parameters is high. Furthermore, the aerodynamic model of the benchmark may basically be regarded to be a black-box system, hence it is not possible to use exact knowledge of this model for testing purposes. This, combined with the fact that the control method is particularly sensitive to tuning of the weighting matrices in both MPC and the control allocation method, makes it difficult to achieve proper results for flying full maneuvers from the list of assessment criteria. In order to show the applicability of the method, provided that the uncertainty of the aerodynamic model is not too high and that the tuning of the controller is appropriately chosen, we show an example maneuver that was obtained through simulation of the benchmark where

the aerodynamics have been replaced by a static but still nonlinear model.

Figures 5.10, 5.11 and 5.12, which are included at the end of the paper, show the results when the aircraft is made to fly a turn to the right followed by a localizer intercept. Figure 5.10 shows a subset of the aircraft states and the angle between the aircraft heading and the localizer beam λ during this particular simulation example. Also indicated in the figure, in red and yellow bands, are the assessment specifications. Figure 5.11 and 5.12 show the accelerations of the aircraft and the horizontal trajectory of the aircraft. The results presented here consider a flight in a fault-free scenario, but given the simplified aerodynamic model, different failure scenarios, with stuck control surfaces perform equally well. What may be concluded from this simulation is that the combination of MPC and the inversion of the nonlinear aircraft kinematics through NDI is valid for FTFC purposes, provided correct knowledge of the aerodynamics of the aircraft is available.

5.7 Conclusion

This chapter has presented, in the form of simulation examples, the theory presented in Chapters 2 and 3, i.e. tuning of an MPC controller based on an existing controller and the combination of model predictive control and nonlinear dynamic inversion into a constrained and globally valid control method. Using the proposed control methods, it is possible to implement a reconfigurable flight controller. The reconfigurable properties are a result of efficient distribution of the desired control effort over the remaining and redundant control inputs. Furthermore, the method can take into account various input, state and output constraints. The latter is particularly useful when actuators get stuck or become unavailable. Similar results have been achieved when building a controller through reverse engineering of the original autopilot.

Practical issues that will be the topic of future research are related to the construction of a more computationally efficient adaptation of this controller. Additionally, it will have to be taken into account that in one example a recursive identification scheme is applied in a closed-loop setting whilst this is not explicitly accounted for at the moment. Increased robustness of the fault tolerant flight control method will be of great importance in applications where there is latency in the failure detection and identification system. Robustness with respect to modeling uncertainty is required to guarantee stability until new and accurate failure information becomes available after a failure has occurred.

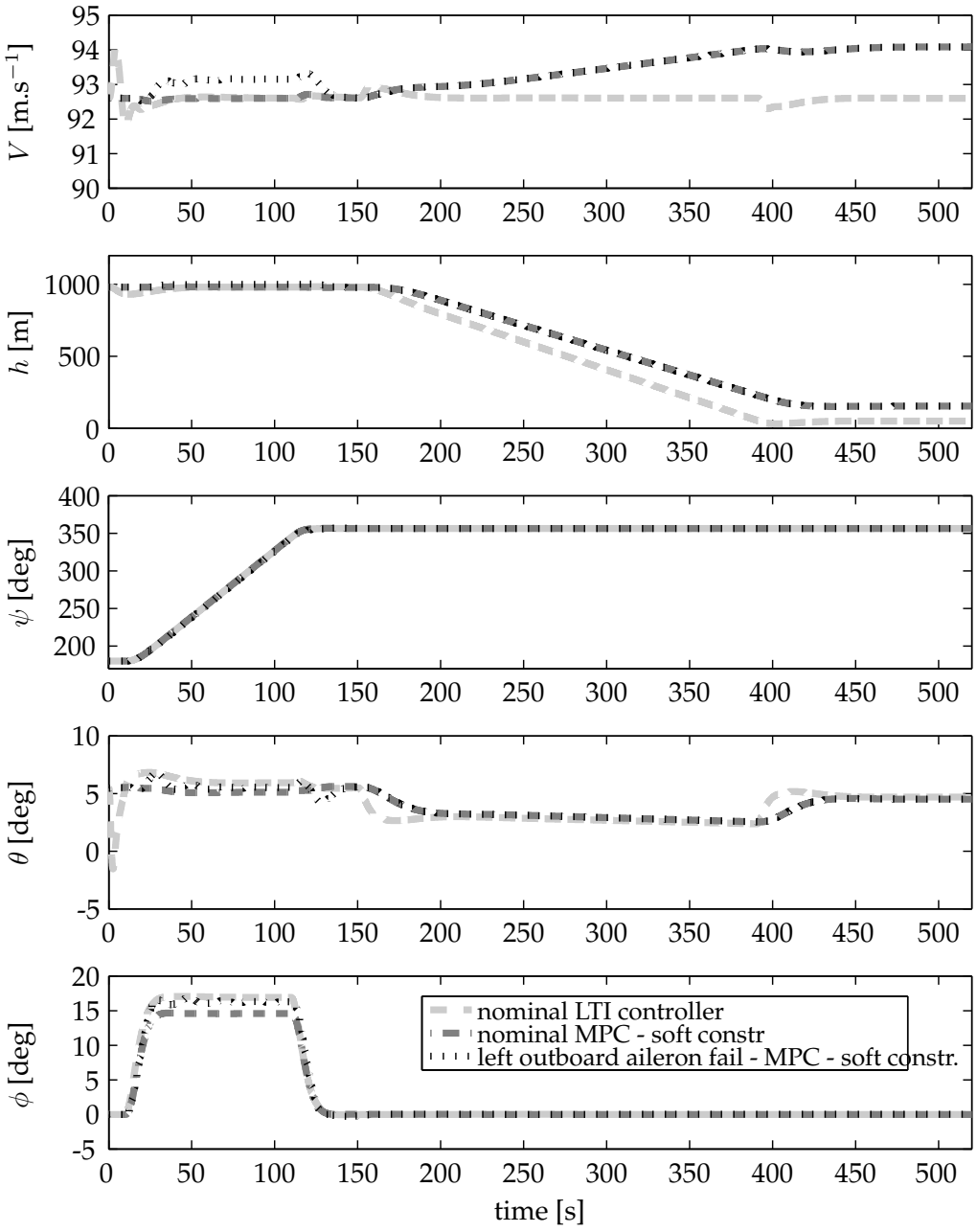

Figure 5.8: Overview of 5 of the aircraft states and the ability to track a specified reference trajectory. Three situations are depicted: the response with the original LTI controller in the nominal case, the response with the MPC controller in the nominal case, and the response with the MPC controller in the case of a jammed aileron (left outboard, jammed at $\delta_{ail} = 0$, starting at $t = 0$)

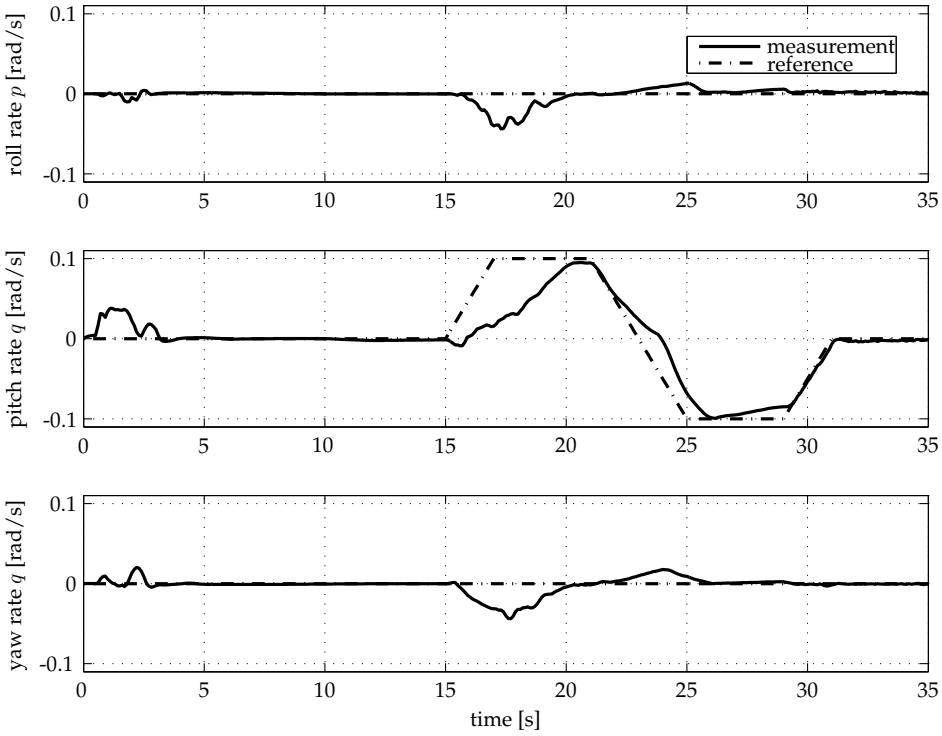

Figure 5.9: Simulation result for the body rates p, q, r with respect to a reference after introduction of a stabilizer runaway fault at $t = 10$ [s]

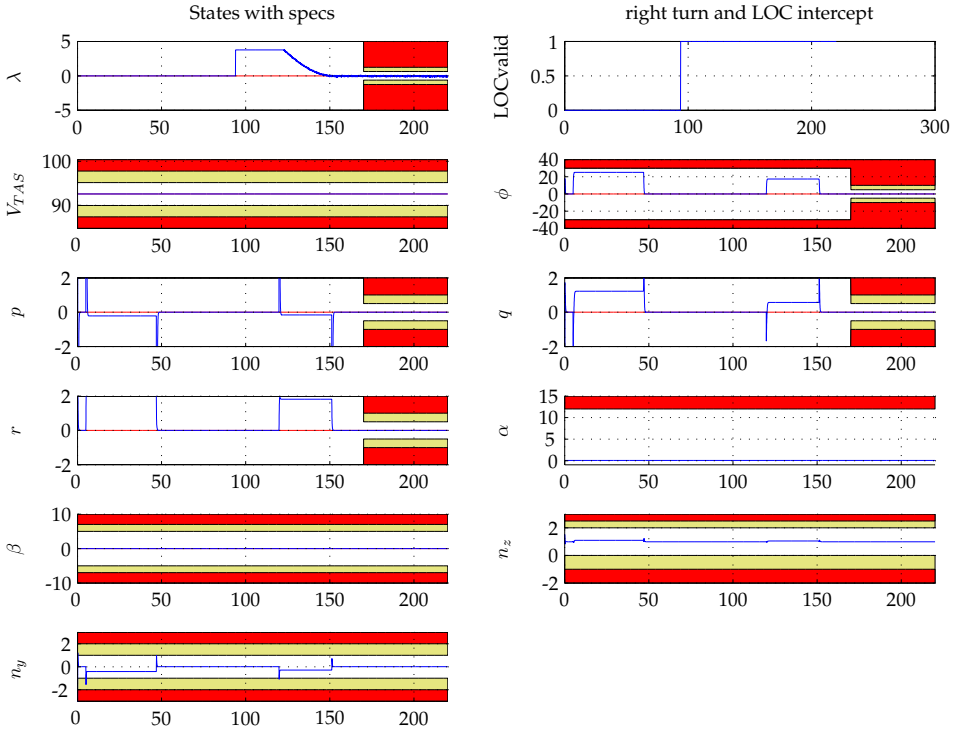

Figure 5.10: Overview of several aircraft states during a right-hand turn and subsequent localizer intercept. The top left and top right graph in the figure depict the angle λ with respect to the localizer beam and the signal that indicates whether the localizer signal is valid.

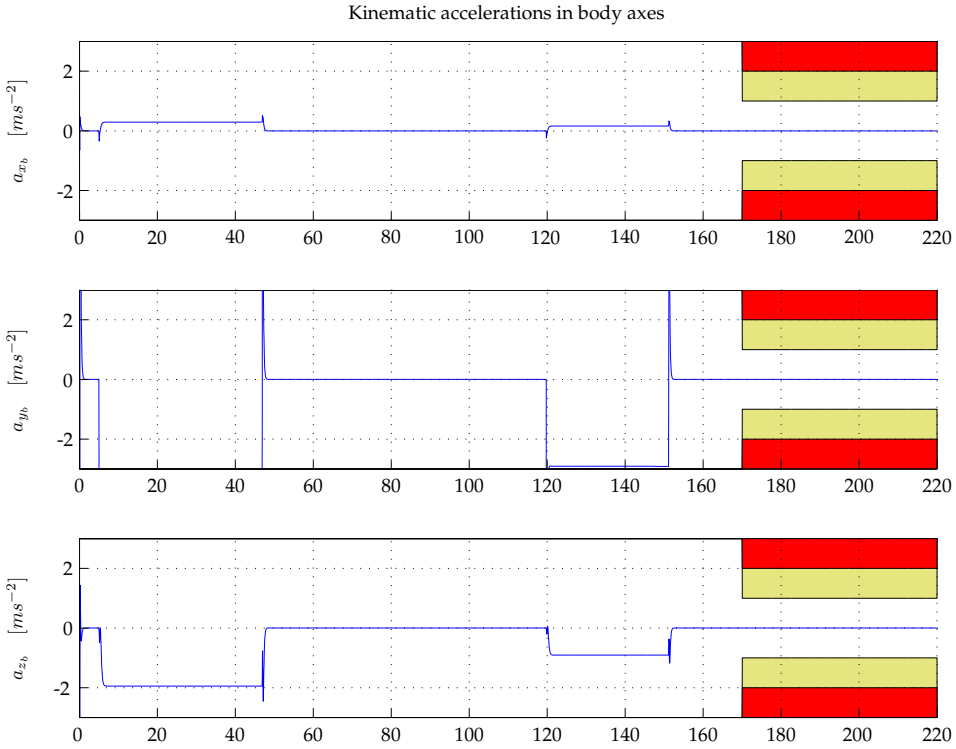

Figure 5.11: Overview of the accelerations of the aircraft body during the right turn and localizer intercept.

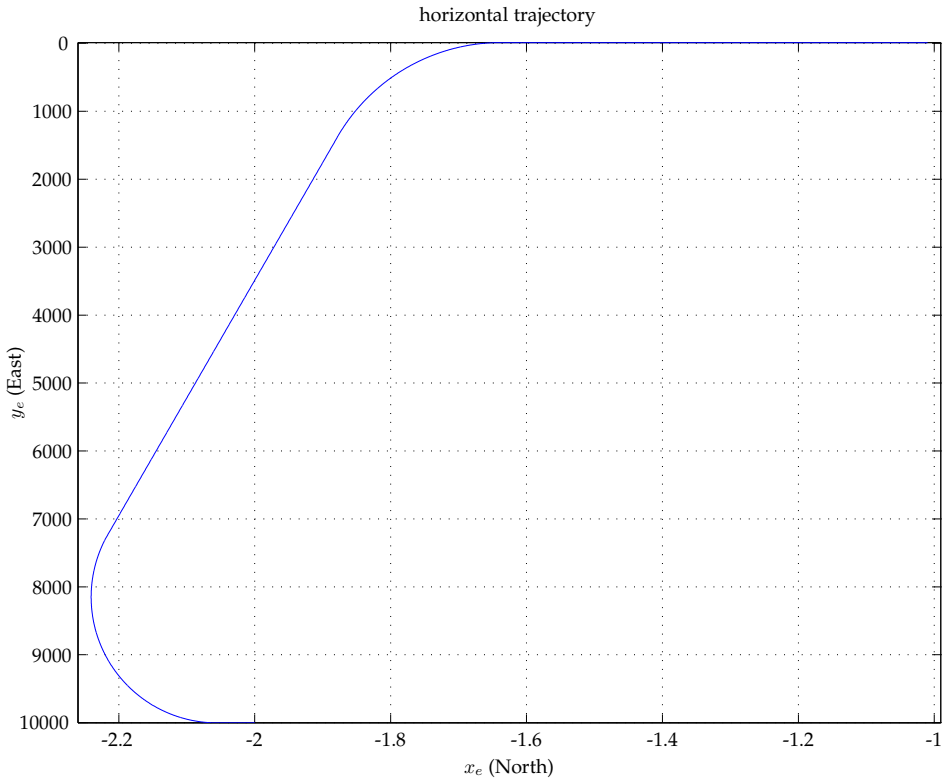

Figure 5.12: Representation of the horizontal trajectory that was flown by the aircraft during the right hand turn and localizer intercept maneuver.

6 CHAPTER

Conclusions and Recommendations

6.1 Discussion & Conclusions

This thesis has investigated the applicability of MPC for fault tolerant flight control purposes. Its inherent constraint handling capabilities and use of an internal plant model were the main cause of interest for the application of MPC. From literature it is known that tuning of the cost function, i.e. the underlying weighting matrices, may not be very straightforward, especially in flight control examples, and that it is therefore of interest to base controller design on that of an existing controller. Also the dynamic equations of motion of an aircraft are nonlinear, it is therefore worthwhile to establish control methods that can accommodate these nonlinearities, while still allowing for the inclusion of constraints on states or inputs. Both aforementioned methods have been investigated in the thesis. The results of the investigation are described in more detail below. Also, the thesis has introduced the notion of levels of performance in case of operating in fault conditions, whereby MPC controllers can be applied such that plant performance degrades gradually with increasing fault complexity, but stability is maintained as long as possible.

A method for obtaining a state observer in combination with a model-predictive controller has been introduced. Based upon a linear time-invariant representation of both the existing autopilot and the aircraft it is possible to arrive at such a controller structure. When the original controller contains both a direct feedthrough term and integral action, this goal cannot be achieved without the necessary caution. Section 2.4 presented a novel way to do so. In the absence of input constraints this controller shows tracking performance that is on par with the original output feedback controller without requiring extensive tuning of the cost-function weighting matrices and parameters such as the prediction horizon. The latter opens the door to reconfigurable and fault tolerant flight control as constraints can now be taken into account in the MPC criterion. Such constraints can be used to include faults (e.g. stuck control surfaces) and maintain stable behavior although it has to be remarked that the behavior of the closed loop will start to deviate from

the original closed loop behavior as the system moves out of its unconstrained space.

Chapter 3 has shown a method that combines the constraint handling capabilities of MPC with NDI. The challenge thereof lies in the resulting nonlinear transformation of the original input constraints of the plant. This is both computationally intensive and the expansion of this transformation over the prediction horizon remains an approximation due to the nonlinear plant dynamics. A drawback of the methods introduced is that NDI or feedback linearization is known to be very sensitive to modeling errors and MPC is a computationally intensive method. Even comparatively simple examples support the notion that these properties do not combine well with a system that has fast dynamics such as an aircraft. It must be noted that, despite the fact that a nonlinear control method is combined with MPC, the constraint mapping presents a limitation on the closed loop as these are currently kept constant over the prediction horizon whilst the system itself can move through state-space in a nonlinear fashion.

Having said this, this thesis does present a method to perform the mapping of constraints onto the inputs of the NDI controller in an efficient manner. Chapter 4 investigates the matter of constraint mapping more in-depth . This alleviates one aspect of the aforementioned issues. Chapter 4 introduced a computationally very efficient method for the projection of convex polytopes to lower dimensional spaces that is suitable for the projection of hypercubes. The efficiency advantage stems from the fact that the algorithm foregoes the projection of the vertices, but applies the more efficient half-space description instead.

Chapter 5 has presented the combination of model predictive control and nonlinear dynamic inversion into a constrained and globally valid control method. Using the proposed control method, it is possible to implement a reconfigurable flight control-law that is valid throughout the flight envelope. The reconfigurable properties are a result of efficient distribution of the desired control effort over the remaining and redundant control inputs. Furthermore, the method can take into account various input, state and output constraints. The latter is particularly useful when actuators get stuck or become unavailable. Similar results have been achieved when building a controller through reverse engineering of the original autopilot.

6.2 Recommendations

This thesis poses an exploration of possibilities. There exist many obstacles before practical application of such methods will become feasible in real life situations. Such limitations include the current deterministic methods for clearance of flight control laws that appear to not handle changing control parameters very well, and acceptance by flight crews due to the inherent loss of situational awareness associated with fault tolerant flight control. These aspects have not been investigated in this thesis.

Recommendations for future work on MPC based on the reverse engineering of an existing controller (Chapter 2) include the following: In the presence of constraints the performance of the reverse-engineered controller cannot always be guaranteed, especially when one or more constraints become active. Whilst the desire to introduce constrained control to the aircraft benchmark is a driving force in this work, it is deemed instructive to investigate different cost-function formulations for the MPC problem. Specifically cost-functions that do not weigh the difference between the predicted state-feedback quantities and the input are assumed to be of great value. Methods that weigh the difference between the predicted plant state and the predicted closed loop dynamics are thought to be more than worth the investigation. The latter can offer better ways to make use of the available redundancy in the sense of actuators. Additionally, further future work on the applied aircraft model should also include an exhaustive investigation of the different ways in which the closed-loop poles may be distributed among the observer error-dynamics and the state-feedback dynamics, as well as where to place the free poles in the controller.

As to the projection of polytopes presented in Chapter 4 there appear to exist extensions towards generic non-symmetric polytopes appear feasible, but lead to redundant half spaces. This theory and an efficient method for the removal of the redundant half spaces are recommended for future research.

Further practical issues that will be the topic of future research are related to the construction of a more computationally efficient implementation of this controller. Additionally, it will have to be taken into account that the online identification schemes applied in a closed-loop may cause estimation bias. Increased robustness of the fault tolerant flight control method will be of great importance in applications where there is latency in the failure detection and identification system. Robustness with respect to modeling uncertainty is required to guarantee stability until new and accurate failure information becomes available after a failure has occurred.

Bibliography

Alazard, D. (2002). Cross standard form for generalized inverse problem: application to lateral flight control of a highly flexible aircraft. In *Proc. Int. Conf. Nonlinear Problems in Aviation and Aerospace*, Melbourne (Florida) (May).

Alazard, Daniel (2013). *Reverse engineering in control design*. John Wiley & Sons.

Alazard, D. and P. Apkarian (1999). Exact observer-based structures for arbitrary compensators. *International Journal of Robust Nonlinear Control* **9**, 101–118.

Allgöwer, F., T.A. Badgwell, J.S. Qin and J.B. Rawlings (1999). Nonlinear predictive control and moving horizon estimation – an introductory overview. In *Advances in Control, Highlights of ECC'99, Edited by F. Frank*, Springer-Verlag, London, UK.

Åström, K.J. and B. Wittenmark (1973). On self-tuning regulators. *Automatica* **9**, 185–199.

Bemporad, A. and Manfred Morari (1999). Robust model predictive control: A survey. *Robustness in identification and control, Lecture Notes in Control and Information Sciences* **245**, 207–226. Springer Verlag.

Bender, D.J. and R.A. Fowell (1985). Computing the estimator-controller form of a compensator. *International Journal of Control* **41**(6), 1565–1575.

Bequette, B.W. (1991). Nonlinear control of chemical processes: A review. *Ind. Eng. Chem. Res.* **30**(7), 1391–1413.

Biegler, L. (2000). *Efficient solution of dynamic optimization and NMPC problems*. volume 26 of *Progress in Systems and Control Theory*. Basel, Switzerland: Birkhäuser Verlag: In: F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control*.

Bitmead, R.R., M. Gevers and V. Wertz (1990). *Adaptive Optimal Control. The Thinking Man's GPC*. Upper Saddle River, New Jersey: Prentice Hall.

Blanke, M. (2003). *Diagnosis and Fault-Tolerant Control*. Springer.

Blanke, Mogens and Jochen Schröder (2006). *Diagnosis and fault-tolerant control*, Volume 691. Springer.

Boyd, S.P. and C.H. Barratt (1991). *Linear Controller Design, Limits of performance*. Englewood Cliffs, New Jersey: Prentice Hall, Information and System Sciences Series.

Bryson, Arthur Earl (1975). *Applied optimal control: optimization, estimation and control*. CRC Press.

Cairano, S. Di and A. Bemporad (2010). Model predictive control tuning by controller matching. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL* **55**(1), 185 – 190.

Camacho, E.F. and C. Bordons (1995). *Model Predictive Control in the Process Industry, Advances in Industrial Control*. London: Springer.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987a). Generalized predictive control - part 1. the basic algorithm. *Automatica* **23**(2), 137–148.

Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987b). Generalized predictive control - part 2. extensions and interpretations. *Automatica* **23**(2), 149–160.

Clarke, David W and C Mohtadi (1989). Properties of generalized predictive control. *Automatica* **25**(6), 859–875.

Cook, M.V. (2003). *Flight Dynamics Principles*. Burlington, MA 01803: Elsevier Butterworth-Heinemann.

Cutler, C.R. and B.L. Ramaker (1979). Dynamic matrix control - a computer control algorithm. In *AIChE Nat. Mtg*.

Cutler, C.R. and B.L. Ramaker (1980). Dynamic matrix control - a computer control algorithm. In *Proceeding Joint American Control Conference, San Francisco, CA, USA*.

Daniels, J.T. (1903). First flight, 120 feet in 12 seconds, 10:35 a.m.; kitty hawk, north carolina. web. `http://www.loc.gov/pictures/item/00652085/`, retrieved 2016-10-12, picture in public domain.

De Keyser, R.M.C. and A.R. van Cauwenberghe (1982). Typical application possibilities for self-tuning predictive control. In *IFAC Symp.Ident.& Syst.Par.Est.*

De Vries, R.A.J. and H.B Verbruggen (1994). Multivariable process and prediction models in predictive control – a unified approach. *Int. J. Adapt. Contr.& Sign. Proc.* **8**, 261–278.

Delmond, F., D. Alazard and Cumer. C (2006). Cross standard form: a solution to improve a given controller with h_2 or h_∞ specifications. *International Journal of Control* **79**(4), 279–287.

Doyle, J.C., B.A. Francis and A.R. Tannenbaum (1992). *Feedback control systems*. New York, USA: MacMillan Publishing Company.

Doyle, J.C., K. Glover, P.P. Khargonekar and B.A. Francis (1989). State-space solutions to standard H_2 and H_∞ control problems. *IEEE AC* **34**, pp831–847.

EASA, Certification Specification (2016). Certification specifications and acceptable means of compliance for large aeroplanes cs-25, amdt. 18. Technical report, European Aviation Safety Agency.

Edwards, Christopher, Thomas Lombaerts and Hafid Smaili (2010). *Fault tolerant flight control*, Volume 399. Springer.

Fukuda, K. (2004). Frequently asked questions in polyhedral computation. Technical report, Institute for Operations Research, ETH Zürich. Available from Internet: `ftp.math.ethz.ch/users/fukudak/reports/polyfaq040618.pdf`.

Garcia, C.E. and M. Morari (1982). Internal model control. 1. a unifying review and some new results. *Ind.Eng.Chem.Proc.Des.Dev.* **21**(2), 308–323.

Garcia, C.E. and A.M. Morshedi (1986). Quadratic programming solution of dynamic matrix control (QDMC). *Chem.Eng.Comm.* **46**(11), 73–87.

Garcia, C.E., D.M. Prett and M. Morari (1989). Model predictive control: Theory and practice - a survey. *Automatica* **25**(3), 335–348.

Garcia, C.E., D.M. Prett and B.L. Ramaker (1988). *Fundamental Process Control*. Stoneham, MA: Butterworths.

Gilbert, E.G. and K.T. Tan (1991). Linear systems with state and control constraints: the theory and application of maximal output admissible sets. *IEEE AC* **36**, 1008–1020.

Golub, Gene H. and Charles F. van Loan (1996). *Matrix Computations*. The Johns Hopkins University Press.

Hallouzi, R. (2008). *Multiple-Model Based Diagnosis for Adaptive Fault-Tolerant Control*. Ph. D. thesis, Delft University of Technology.

Hallouzi, R and M Verhaegen (2008). Fault-tolerant subspace predictive control applied to a boeing 747 model. *Journal of Guidance, Control, and Dynamics* **31**(4), 873 –883. AIAA paper 0731-5090.

Hanke, C.R. (1970). The simulation of a large transport aircraft. modeling data, vol. ii. Technical Report CR-114494, NASA.

Hanke, C.R. (1971). The simulation of a large transport aircraft. mathematical model, vol. i. Technical Report CR-1756, NASA.

Hanke, C.R. and D.R. Nordwall (1970). The simulation of a jumbo jet transport aircraft. Technical Report D6-30643, National Aeronautics and Space Administration. Volume II: Modeling Data.

Hartley, E. N. and J. M. Maciejowski (2009). Initial tuning of predictive controllers by reverse engineering. In *Proceedings of the European Control Conference 2009*, Budapest, Hungary (August).

Heise, SA and JM Maciejowski (1996). Model predictive control of a supermaneuverable aircraft. In *AIAA Meeting Papers on Disc, Jul., Paper No. AIAA*, pp. 96–3768.

Hoover, Lt. Robert A. (2006). File:bell x-1 color.jpg. web. `https://en.wikipedia.org/wiki/File:Bell_X-1_color.jpg`, retrieved 2016-10-12, picture in public domain.

ICAO (2016). Icao safety report. Technical report, International Civil Aviation Organization, 999 Boulevard Robert-Bourassa, Montréal, QC, Canada, H3C 5H7.

Isidori, A. (1995). *Nonlinear control systems*. New York: Springer Verlag. ISBN ISBN: 978-3-540-19916-8. pp. 219 – 277.

Jones, C.N. (2005). Reconfigurable flight control. Technical report TR2005304, Control Group, Engineering Dept, University of Cambridge, Cambridge, UK.

Jones, C.N., E.C. Kerigan and J.M. Maciejowski (2004). Equality set projection: A new algorithm for the projection of polytopes in halfspace representation. Technical Report CUED/F-INFENG/TR.463, Department of Engineering, University of Cambridge. Available from Internet: `http://www-control.eng.cam.ac.uk`.

Kale, M.M. and A.J. Chipperfield (2005). Stabilized mpc formulations for robust reconfigurable flight control. *Control Engineering Practice* **13**(6), 771 – 788.

Keviczky, Tamas and Gary J Balas (2003). Software enabled flight control using receding horizon techniques. In *AIAA Guidance, Navigation, and Control Conference*.

Kim, Byoung S and Anthony J Calise (1997). Nonlinear flight control using neural networks. *Journal of Guidance, Control, and Dynamics* **20**(1), 26–33.

Kinnaert, M. (1989). Adaptive generalized predictive controller for mimo systems. *Int.J. Contr.* **50**(1), 161–172.

Kreindler, E. and A. Jameson (1972). Optimality of linear control systems. *IEEE Trans. Automat. Control* **AC-17**, 349–351.

Laub, Alan J. (1979). A schur method for solving algebraic riccati equations. *IEEE Trans. Automat. Control* **AC-24**, 913–921.

Lemke, C.E. (1968). *On complementary pivot theory*, pp. 95–114. Mathematics of the decision sciences, Part 1. Providence: American Mathematical Society.

Lombaerts, T.J.J. (2010). *Fault Tolerant Flight Control - A Physical Model Approach*. Delft, NL, ISBN: 978-90-815443-1-3: PhD-thesis, Delft University of Technology.

Lombaerts, T., H. Huisman, Q. Chu, J. Mulder and D. Joosten (2009). Nonlinear reconfiguring flight control based on online physical model identification. *Journal of Guidance, Control, and Dynamics* **32**(3), 727 – 748. AIAA paper 0731-5090.

Maciejowski, J.M. (1989). *Multivariable Feedback Control Design*. Wokingham, UK: Addison-Wesley Publishers.

Maciejowski, J.M. (1998). The implicit daisy-chaining property of constrained predictive control. *Applied Mathematics and Computer Science* **8**(4), 101–117.

Maciejowski, J.M. (2002a). *Predictive control with constraints*. Pearson Education Limited, Harlow, UK: Prentice Hall.

Maciejowski, J.M. (2002b). *Predictive control: with constraints* (1st ed.). New York: Harlow: Pearson Education. ISBN ISBN 0-201-39823-0. Ch. 2–3.

Maciejowski, J.M. (2007). Reverse engineering existing controller for mpc design. In *IFAC symposium on System Structure and Control*, October 17-19).

Maciejowski, J.M. and C.N. Jones (2003). Mpc fault-tolerant flight control case study: flight 1862. In *Proceedings of the 5th International Federation of Automatic Control on Safeprocess Sympoisum*, Number M1-C1, Washington, DC, pp. 119–124.

Mayne, D.Q., J.B. Rawlings, C.V. Rao and P.O.M. Scokaert (2000). Constrained model predictive control: stability and optimality. *Automatica* **36**(6), 789–814.

Moore, G.E. (1965). Cramming more components onto integrated circuits. Volume 38.

Morari, M. and E. Zafiriou (1989). *Robust Process Control*. Englewood Cliffs, New Jersey: Prentice Hall.

Mulder, J.A., W.H.J.J. van Staveren, J.C. van der Vaart and E. de Weerdt (2006). Flight dynamics ae3-302. lecture notes, Delft University of Technology.

Napolitano, Marcello R, Younghwan An and Brad A Seanor (2000). A fault tolerant flight control system for sensor and actuator failures using neural networks. *Aircraft Design* **3**(2), 103–128.

Nesterov, Y. and A. Nemirovskii (1994). *Interior-Point Polynomial Algorithms in Convex Programming*. Philadelphia: SIAM.

Patton, R.J. (1997). Fault-tolerant control systems: The 1997 situation. In *Proc. IFAC Symp. Fault Detection, Supervision Safety Technical Processes*, Volume 3, pp. 1033–1054.

Patton, Ron J (2015). Fault-tolerant control. *Encyclopedia of Systems and Control*, 422–428.

Pratt, Roger W. (Ed.) (2000). *Flight Control Systems; practical issues in design and implementation*. The Institution of Electrical Engineers and The American Institute of Aeronautics and Astronautics.

Preparata, F.P. and M.I. Shamos (1985). *Computational geometry: an introduction*. New York: Springer. ISBN ISBN: 0-387-96131-3. pp. 95 – 148.

Qin, S.J. and T.A. Badgewell (1997). An overview of industrial model predictive control technology. In *Chemical Process Control - V, AIChe Symposium Series - American Institute of Chemical Engineers*, Volume 93, pp. 232–256.

Qin, S.J. and T.A. Badgwell (2003). A survey of industrial model predictive control technology. *Control Engineering Practice* **11**(7), 733–764.

Rawlings, J.B. and K.R. Muske (1993). The stability of constrained receding horizon control. *IEEE AC* **38**, 1512–1516.

Richalet, J. (1993). Industrial applications of model based predictive control. *Automatica* **29**(5), 1251–1274.

Richalet, J., A. Rault, J.L. Testud and J. Papon (1976). Algorithmic control of industrial processes. In *Proceedings of the 4th IFAC Symposium on Identification and System Parameter Estimation, Tbilisi*.

Richalet, J., A. Rault, J.L. Testud and J. Papon (1978). Model predictive heuristic control: Applications to industrial processes. *Automatica* **14**(1), 413–428.

Rossiter, J.A. (2003). *Model-Based Predictive Control: A Practical Approach*. CRC Press, Inc.

Rouhani, R. and R.K. Mehra (1982). Model algorithmic control (mac): Basic theoretical properties. *Automatica* **18**(4), 401–414.

Scokaert, P.O.M. and J.B. Rawlings (1998). Constrained linear quadratic regulation. Volume 43, pp. 1163–1169.

Shtessel, Yuri, Christopher Edwards, Leonid Fridman and Arie Levant (2014). *Sliding mode control and observation*. Springer.

Singh, Sahjendra N, Marc Steinberg and Robert D DiGirolamo (1995). Nonlinear predictive control of feedback linearizable systems and flight control system design. *Journal of Guidance, Control, and Dynamics* **18**(5), 1023–1028.

Slotine, J.J.E. and W. Li (1991). *Applied nonlinear control*. Prentice Hall Englewood Cliffs, NJ. ISBN ISBN: 0-13-040890-5. pp. 207 – 275.

Smaili, M.H. (1997). Flight data reconstruction and simulation of El Al flight 1862. Master's thesis, Delft University of Technology, Delft, The Netherlands.

Soeterboek, A.R.M. (1992). *Predictive control - A unified approach*. Englewood Cliffs, NJ: Prentice Hall.

Steinberg, Marc (2005). Historical overview of research in reconfigurable flight control. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* **219**(4), 263–275.

Sznaier, M. and M.J. Damborg (1987). Suboptimal control of linear systems with state and control inequality constraints. In *Proceedings of the 26th IEEE conference on decision and control*, Los Angeles, pp. 761–762.

Vada, J., O. Slupphaug and B.A. Foss (1999). Infeasibility handling in linear mpc subject to prioritized constraints. In *IFAC'99 14th World Congress, Beijing, China, July 1999*.

van den Boom, T.J.J. (1997). Robust nonlinear predictive control using feedback linearization and linear matrix inequalities. In *Proceedings of the American Control Conference*, Volume 5, June 4–6), pp. 3068–3072.

van den Boom, T.J.J. and A.A. Stoorvogel (2010). Model predictive control. lecture notes, Dutch Institute for Systems and Control.

van Keulen, R. (1991). Real-time simulation and analysis of the automatic flight control system of the boeing 747-200. Master's thesis, Delft University of Technology.

van Soest, W.R., Q.P. Chu and J.A. Mulder (2006). Combined feedback linearisation and constrained model predictive control for entry flight. *Journal of Guidance, Control and Dynamics* **29**(2), 427–434. AIAA paper 0731-5090.

Verhaegen, Michel, Stoyan Kanev, Redouane Hallouzi, Colin Jones, Jan Maciejowski and H. Hafid Smaili (2010). *Fault Tolerant Flight Control - A Survey*, Chapter Fault Tolerant Flight Control, pp. 47 – 89. Springer.

Wise, Kevin A, Joseph S Brinker, Anthony J Calise, Dale F Enns, Michael R Elgersma and Petros Voulgaris (1999). Direct adaptive reconfigurable flight control for a tailless advanced fighter aircraft. *International Journal of Robust and Nonlinear Control* **9**(14), 999–1012.

Wolfe, P. (1959). The simplex method for quadratic programming. *Econometrica* **27**, 382–398.

Zheng, A. and M. Morari (1995). Stability of model predictive control with mixed constraints. *IEEE AC* **40**, 1818.

Zolghadri, Ali, David Henry, Jérôme Cieslak, D Efimov and Philippe Goupil (2014). *Fault diagnosis and fault-tolerant control and guidance for aerospace vehicles*. Springer.

List of Abbreviations

AFCS	Automatic flight control system
AMC	Acceptable means of compliance
ANN	Artificial neural network
BMI	Bilinear matrix inequality
CA	Control allocation
CARMA	Controlled autoregressive moving average
CARIMA	Controlled autoregressive integrated moving average
DISC	Dutch institute of systems and control
DMC	Dynamic matrix control
DCSC	Delft Center for Systems and Control
DOF	Degree of Freedom
CFIT	Controlled flight into terrain
EASA	European aviation safety agency
EPR	Engine pressure ratio
EPSAC	Extended prediction self adaptive control
FBL	Feedback Linearization
FCS	Flight control system
FDI	Failure detection and identification
FTC	Fault tolerant control
FTFC	Fault tolerant flight control
GARTEUR	Group for Aeronautical Research and Technology in Europe
GPC	Generalized predictive control
ICAO	International Civil Aviation Organization
IDCOM	Identification and command
IMM	Interacting multiple models
LOC-I	Loss of control in-flight
LQ(R)	Linear quadratic (regulator)
LMI	Linear matrix inequality
LTI	Linear time invariant
MAC	model algorithmic control
MIMO	Multiple input multiple output
MRAC	Model reference adaptive control
MPC	Model predictive control
MMST	Multiple model and switching and tuning
NDI	Nonlinear dynamic inversion

PCA	Propulsion controlled aircraft
PFC	Predictive functional control
PID	Proportional integral derivative control
QDMC	Quadratic dynamic matrix control
RECOVER	REconfigurable COntrol for Vehicle Emergency Return
RS	Runway safety
SMC	Sliding mode control
SPCP	Standard predictive control problem
STW	Stichting technische wetenschappen
UPC	Unified predictive controller
ZOH	zero order hold

List of Figures

List of Publications

Book Chapter

Joosten, D., van den Boom T., Verhaegen M (2010). Fault-Tolerant Control through a Synthesis of Model-Predictive Control and Nonlinear Inversion. In *Fault Tolerant Flight Control A Benchmark Challenge*,(p. 319-336), Christopher Edwards, Thomas Lombaerts, Hafid Smaili (Eds.), Springer Verlag, 2010.

Conference proceedings

D.A. Joosten and T.J.J. van den Boom (2006). Towards fault tolerant flight control using model predictive control. In *25th Benelux meeting on systems and control*, page 137.

D.A. Joosten, T.J.J. van den Boom, and T.J.J. Lombaerts (2007). Effective control allocation in fault-tolerant flight control using feedback linearisation and model predictive control. In *Benelux meeting on systems and control*.

D.A. Joosten, T.J.J. van den Boom, and T.J.J. Lombaerts (2007). Effective control allocation in fault-tolerant flight control with MPC and feedback linearisation. In *Proceedings of the European Conference on Systems and control*, pages 3552–3559, Kos, Greece.

D.A. Joosten, T.J.J. van den Boom, and T.J.J. Lombaerts (2008). Fault-tolerant control using dynamic inversion and model-predictive control applied to an aerospace benchmark. In *the Proceedings of the 17th IFAC world congress*, volume 17, Seoul, South Korea.

D.A. Joosten, T.J.J. van den Boom, and T.J.J. Lombaerts (2008). Computationally Efficient Use of MPC and Dynamic Inversion for Reconfigurable Flight Control. In *the Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii.

D.A. Joosten, J.M.M. Maciejowski (2009). MPC Design for Fault-Tolerant Flight Control Purposes Based Upon an Existing Output Feedback Controller. In *Proceedings of the IFAC Safeprocess Conference*, Barcelona, Spain, June 30- July 3.

Publications, second author

T. Lombaerts, Q. Chu, J. Mulder, and D. Joosten. Real time damaged aircraft model identification for reconfiguring flight control. In *proceedings of the AIAA Atmospheric Flight Mechanics Conference and Exhibit*, number AIAA-2007-6717, 2007.

T.J.J. Lombaerts, H.O. Huisman, Q.P. Chu, J.A. Mulder, and D.A. Joosten. Computer based real time damaged aircraft model identification with two step method and modified stepwise regression. In *Proceedings of the 18th SFTE symposium on flight test challenges in a changing world environment*, 2007.

T. Lombaerts, H. Huisman, Q.P. Chu, J. Mulder, D. Joosten. Nonlinear Reconfiguring Flight Control Based on Online Physical Model Identification In *Journal of Guidance, Control, and Dynamics*, vol.32(3), p. 727–748, 2009

H. Smaili, J. Breeman, T. Lombaerts, and D. Joosten. A simulation benchmark for integrated fault tolerant flight control evaluation. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*, number AIAA-2006-6471, Keystone, CO, August 2006.

H. Smaili, J. Breeman, T. Lombaerts, and D. Joosten. Surviving the improbable: Towards self-repairing flight control systems. In *Air & Space Europe*, Vienna, Austria, 2006.

Summary

Current jet fighters and modern airliners are hugely complex pieces of machinery. The drawback of this complexity lies in the number of systems and subsystems that may fail for one reason or another. Given the systems complexity of aircraft it is no longer easily possible for the crew to establish what exactly has happened when these fail. This motivates the need to provide means for the diagnosis of failures and automated recovery, i.e. fault tolerant flight control.

In general the procedure to make a system fault-tolerant consists of two steps: *1*) Fault diagnosis: the existence of faults has to be detected and the faults have to be identified, *2*) Control re-design: the controller has to be adapted to the faulty situation so that the overall system continues to satisfy its goal. This thesis focuses on the latter through the application of modern control methods towards reconfigurable flight control.

This thesis investigates fault tolerant flight control in the event of actuator or plant faults. A literature survey suggests that model predictive control (MPC) is very suitable for use as fault tolerant flight control method due to its ability to incorporate various constraints. Application of MPC in this setting is the central topic in this thesis.

MPC is applied in two different ways in the text. First, a method is presented for finding both a state-observer and the cost function associated with a model predictive controller, based on an already existing output feedback controller. The goal of this exercise is to retain the properties of the existing controller, while adding the constraint handling capabilities of MPC.

The second way features the combination of model-based predictive control and the inversion of the dynamics of the system under control into a constrained and globally valid control method for fault-tolerant flight-control purposes. The fact that the approach allows the incorporation of constraints creates the possibility to incorporate additional constraints in case of a failure. Such failures range from relatively straightforward actuator failures to more complicated structural breakdowns where, through the addition of constraints, the aircraft can be kept within its remaining flight envelope. Furthermore, the method is model-based, which allows adaptation of the system model in case of a failure. Both of these properties lead to the fault-tolerant qualities of the method presented. Projection of a polytope onto a lower dimensional polytope is an important element in the com-

bination of MPC and dynamic inversion. A method is presented that avoids the computation of the polytope's vertices and the application of linear programming methods.

The theory presented in this thesis is applied to a benchmark model which constitutes a detailed simulation model of a Boeing 747-200 aircraft, like the aircraft that crashed in the Amsterdam Bijlmer area in 1992.

Samenvatting

Moderne straaljagers en verkeersvliegtuigen zijn buitengewoon complexe machines. Het nadeel van complexiteit ligt in het aantal systemen en subsystemen dat om welke reden dan ook foutief gedrag kan vertonen, of anderszins kan falen. Daarnaast is het zo dat de systeemcomplexiteit van vliegtuigen het in zekere gevallen moeilijk maakt voor de vliegers om vast te stellen wat er is gebeurd als deze systemen falen. Het voorgaande motiveert en justifieert onderzoek naar methoden voor foutdiagnose en foutherstel, oftewel fout-tolerante vliegtuigbesturing.

In het algemeen vergt het maken van een fouttolerant systeem twee stappen: *1* foutdiagnose: het optreden van een fout moet onderkend worden en geïdentificeerd, *2* foutherstel: de regelaar moet worden aangepast aan de foutsituatie opdat het systeem als geheel de bedoelde functie blijft vervullen. Dit proefschrift onderzoekt het laatstgenoemde onderwerp door toepassing van moderne regeltechnieken ten behoeve van fout-tolerante vliegtuigbesturing.

Dit proefschrift onderzoekt fout-tolerante vliegtuigbesturing in het geval van fouten in de actuatoren of het systeem (vliegtuig) zelf. Uit literatuuronderzoek blijkt dat modelgebaseerd voorspellend regelen (*model predictive control, MPC*) bijzonder geschikt is voor dit doel vanwege de mogelijkheid om diverse systeembeperkingen op te leggen en mee te nemen in de berekening van het stuursignaal. De toepassing van MPC vormt het centrale onderwerp van dit proefschrift.

MPC wordt op twee verschillende manieren toegepast. Allereerst wordt een methode gepresenteerd die een toestandschatter en kostenfunctie van een MPC regelaar bepaald aan de hand van een bestaande lineair tijd-invariante regelaar. Het doel van deze methode ligt in het behoud van de eigenschappen van de bestaande regelaar, terwijl tegelijkertijd de mogelijkheden om systeembeperkingen mee te nemen worden toegevoegd om daarmee fouten op te vangen.

De tweede methode bestaat uit de combinatie van een MPC regelaar en het inverteren van de niet-lineaire dynamica van het te besturen systeem. Deze combinatie leidt tot een regelaar die in het gehele werkgebied van het te besturen systeem geldig is en fouten mee kan nemen door de introductie van systeembeperkingen. Fouten beslaan het bereik van relatief eenvoudig falen van actuatoren tot complexe beschadigingen aan de structuur van het vliegtuig of systeem. Door het toevoegen van beperkingen aan de regelaar kan het vliegtuig in die toestand(en) worden gehouden waarin een verdere stabiele en bestuurbare vlucht mogelijk is.

Deze elementen leiden tot de fout-tolerante eigenschappen van de gepresenteerde regelmethode. Het projecteren van polytopen naar een lager-dimensionele ruimte is een belangrijk element in de eerder genoemde combinatie van regeltechnieken. In dit proefschrift wordt daartoe een (rekentechnisch) efficiënte methode gepresenteerd.

De in dit proefschrift gepresenteerde theorie wordt toegepast op een aangepast simulatiemodel van een Boeing 747-200 vliegtuig, het type vliegtuig dat in 1992 in Amsterdam Bijlmermeer is neergestort.

About The Author

Diederick Joosten was born on August 4, 1979 in Leiderdorp, the Netherlands. He received his secondary education at *Stedelijk Gymnasium Leeuwarden*. From 1997 to 2005 he studied at Delft University of Technology and he received the Bachelor of Science degree in Electrical Engineering in 2004 and the Master of Science degree (with honors) in Systems and Control in 2005. The title of the MSc thesis was *LFT subspace identification; case study: application to the model of a mini-helicopter*. Hereafter he stayed on at the Delft Center for Systems and Control as a PhD student in the STW project *Reconfigurable Handling & Flying Qualities for Degraded Flight Systems using Model Predictive Control*. During his PhD project he took graduate courses at the Dutch Institute for Systems and Control (DISC) and received the DISC certificate. In 2010 he joined the Netherlands Aircraft Company, becoming director marketing in 2012. Since October 2015 he is investment manager at Panta Holdings, a privately held investment company.