Dipl-Ing. Martin Kirchengast, BSc

Control Allocation Techniques for Redundantly Actuated Systems

Dissertation
to achieve the university degree of
Doctor of Engineering Sciences (Dr.techn.)
Doctoral Programme: Doctoral School of Electrical Engineering

submitted to

Graz University of Technology

Supervisor
Univ.-Prof. Dipl-Ing. Dr.techn. Martin Horn

Co-supervisor
Ass.Prof. Dipl.-Ing. Dr.techn. Martin Steinberger

Institute of Automation and Control
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Martin Horn

Graz, May 2018

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present dissertation.

Date	Signature

Abstract

Systems equipped with a redundant set of actuators can be found in many domains which are characterized by high demands on reliability, fault tolerance, and performance, as for instance in flight control. Due to the input redundancy in general infinitely many combinations of actuator actions result in the same effect on the system in the absence of faults. The implementation of a hierarchical control system with control allocation (CA) is one strategy to deal with this problem class. In this case the first control layer operates based on a mathematical system model which neglects the redundancy by replacing the real actuators with a smaller number of artificial ones. Subsequently, the CA algorithm distributes the desired virtual control effort from the overlying controller among the real actuators resting upon optimization- and/or rule-based approaches. The principal difficulty of CA originates from the fact that the actuators always exhibit a limited operation range leading to constrained optimization problems. The main purpose of this work is the development of computationally efficient methods for constrained CA.

Three methods with increasing effectivity and complexity are proposed. The first one is a new approach to compute generalized inverses with regard to the constraints where the major computational burden can be handled offline, i.e. prior to the execution of the control algorithm. It is based on geometric considerations and it requires considerably less computation time than the conventional method. High performance application scenarios typically involve the utilization of actuators close to their limits. Therefore, an extension of an established iterative CA algorithm dedicated to these very situations is presented. It enables the targeted influence on the error between desired and actually achieved virtual control effort if an exact solution is impossible due to the constraints. A popular CA problem formulation yields a quadratic program with linear constraints which has to be solved during the operation of the control loop. For this purpose an efficient solver based on the penalty function and gradient projection approaches is introduced. Although, originally designed for CA it is shown that it can also be successfully applied in different areas such as Model Predictive Control (MPC). Apart from testing the developed methods in numerical simulations they are also implemented on a laboratory setup and compared with two alternative methods to handle input redundancy: Linear Quadratic Regulation and MPC. The results demonstrate the key benefits of CA, namely good performance and relatively low computational effort.

Kurzfassung

Um höchsten Anforderungen hinsichtlich Zuverlässigkeit, Fehlertoleranz und Leistungsfähigkeit entsprechen zu können, werden technische Systeme in zahlreichen Anwendungsgebieten (z.B. Luftfahrt) mit einem Überschuss an Aktuatoren ausgestattet. Im fehlerfreien Betrieb ist es aufgrund der redundanten Aktuatorik möglich das Systemverhalten mit unterschiedlichen Aktuatoreingriffen auf ein und dieselbe Art und Weise zu beeinflussen. Der Einsatz von hierarchisch Regelungskonzepten mit sogenannter Control Allocation (CA) stellt eine Methode dar, um dieser Problemstellung zu begegnen. Der Regler im engeren Sinn operiert dabei auf Basis eines mathematischen Systemmodells, welches durch die Einführung einer geringeren Anzahl an virtuellen Aktuatoren die Redundanz umgeht. Die daraus resultierenden virtuellen Stellgrößensignale werden in einem zweiten Schritt vom CA-Algorithmus auf die tatsächlich vorhandenen Aktuatoren aufgeteilt, wobei die Lösung von Optimierungsproblemen und/oder regelbasierte Ansätze die Entscheidungsgrundlage bilden. Eine der größten Herausforderungen für CA stellen hierbei die in allen praktischen Anwendungen vorhandenen Stellgrößenbegrenzungen dar, welche zu beschränkten Optimierungsproblemen führen. Der Schwerpunkt der vorliegenden Arbeit liegt in der Entwicklung von effizienten Algorithmen zur Lösung von CA Problemstellungen unter Berücksichtigung von Aktuatorbeschränkungen (Constrained CA).

Drei Methoden mit steigender Effektivität und Komplexität werden vorgestellt. Zunächst wird eine neue Vorgehensweise zur Bestimmung von Pseudoinversen entwickelt, die es erlaubt die Aktuatorbeschränkungen miteinzubeziehen und größtenteils vorab, d.h. nicht erst zur Laufzeit des Reglers, berechnet werden kann. Sie basiert auf geometrischen Überlegungen und benötigt weniger Rechenzeit als die konventionelle Methode. In vielen Fällen ist es erforderlich die Aktuatoren nahe ihren Beschränkungen zu betreiben, um die bestmögliche Leistung des Gesamtsystems zu erzielen. Für diese Fälle wird eine Erweiterung eines iterativen CA-Algorithmus präsentiert, die es erlaubt den Fehler zwischen gewünschten und erzielten virtuellen Stellgrößen zu beeinflussen, wenn eine exakte Lösung aufgrund der Beschränkungen nicht möglich ist. CA-Problemstellungen werden häufig als Quadratische Programme (QP) mit linearen Beschränkungen formuliert, welche zur Laufzeit des Reglers online gelöst werden müssen. Basierend auf der Methode der Straffunktionen und der Gradientenprojektion wird ein effizienter Lösungsalgorithmus für QPs entwickelt. Obwohl ursprünglich für CA konzipiert, kann die Methode auch in anderen Bereichen wie beispielsweise modellprädiktiven Regelungen (MPR) erfolgreich eingesetzt werden. Die entworfenen Verfahren werden nicht nur in numerischen Simulationen ausführlich getestet, sondern auch an einem Labormodell implementiert und mit zwei alternativen Ansätzen verglichen: Linear Quadratische Regelung und MPR. Im Zuge der Experimente zeigt sich, dass mithilfe von CA nicht nur hinsichtlich Rechenaufwand, sondern auch Leistungsfähigkeit sehr gute Ergebnisse erzielt werden können.

Acknowledgments

First, I would like to express my gratitude to Prof. Martin Horn for drawing my attention to control engineering, for giving me the opportunity to work on several interesting projects at the Institute of Automation and Control, and thus making this thesis possible. My special thanks go to Dr. Martin Steinberger for the numerous valuable discussions, ideas, and advices, which contributed significantly to the successful completion of this thesis. I also give thanks to the entire staff at the Institute of Automation and Control for creating such a pleasant working environment. Parts of this work originate from a collaboration with Magna Powertrain and representative for all former colleagues from Plant Albersdorf I want to thank Dr. Daniel Lindvai-Soos and his predecessor Dr. Martin Ringdorfer for their support. Finally, I give thanks to all my family and friends for their encouragement as well as for giving me the necessary distraction from work.

Martin Kirchengast

Notation

Scalars are represented as small letters, vectors as small letters in boldface, and matrices as capital letters in boldface. The i-th row of a matrix $\boldsymbol{A}$ is written as $\boldsymbol{A_{i.}}$ and the i-th element of a vector $\boldsymbol{x}$ reads as x_i. Relational operators which are applied on vectors have to be fulfilled element-wise, for example $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{x} \leq \boldsymbol{y} \Leftrightarrow \forall i = 1, \ldots, n : x_i \leq y_i$. $\boldsymbol{A} \succ 0$ and $\boldsymbol{A} \succeq 0$ denote positive definite and positive semi-definite matrices, respectively. The right nullspace of matrix $\boldsymbol{A}$ is $\mathcal{N}_{\mathrm{r}}(\boldsymbol{A})$, the left nullspace is $\mathcal{N}_{\mathrm{l}}(\boldsymbol{A})$, and its range is given by $\mathcal{R}(\boldsymbol{A})$. Gradient and Hessian of a function $f(\boldsymbol{x})$ are denoted as $\boldsymbol{\nabla}_f = \frac{\partial f}{\partial \boldsymbol{x}}$ and $\boldsymbol{\mathcal{H}}_f = \frac{\partial^2 f}{\partial \boldsymbol{x}^2}$, respectively. Given an angle α the short notations of sine, cosine, tangent, and secant are si_α, co_α, ta_α, and se_α. The unit step function reads as

$$\sigma(T_{step}, t) = \begin{cases} 0 & \text{if } t < T_{step} \\ 1 & \text{else} \end{cases}$$

Assignments in algorithms are indicated with the $\leftarrow$ operator. For example, $i \leftarrow i + 1$ means that the value of i is increased by one. The number of elements of a finite set S is denoted by $|S|$. The convex hull of a set of points P is labeled as $\mathrm{conv}(P)$.

Contents

Part I.

Motivation and Background

1. Introduction

1.1. Over-actuation, input redundancy, and control allocation

The terms over-actuation and input redundancy are closely related although not the same. The exact definition of over-actuation for a general system is ambiguous. However, primarily this notion is used in context of motion control where a system whose number of actuators exceeds those of its degrees of freedom (DOFs) is called over-actuated. This always implies some kind of redundancy in the actuation of the system, i.e. the plant comes with more actuators than strictly needed to achieve the control objectives. There are several reasons to equip complex systems with redundant sets of actuators. Probably the most important one is the need to guarantee fault-tolerance in safety-critical applications. Redundant actuators might take over the tasks of defective ones if necessary. In fault-free cases they can be used to support the overall actuation of the plant. High performance requirements might also induce the usage of some additional actuators to increase the total control power. Another reason for input redundancy is the desire to reduce the energy-consumption of systems by means of actuators which have different efficiency characteristics over the system's operating range (e.g. hybrid electric cars). Finally, actuator sharing among multiple control systems is a source of input redundancy. For example an electric car's motor can not only be used for propulsion but also as additional actuator of the braking control system [1].

Input redundancy on the other hand is a property of the mathematical description of a system. In principle a distinction is made between *strong* and *weak* input redundancy. In case of strong input redundancy infinitely many combinations of actuator actions achieve the same effect on the dynamics of the system. The mathematical formulation of this property given in [2] is based on the linear plant model

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B_u}\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D_u}\boldsymbol{u} \end{aligned} \tag{1.1}$$

with state vector $\boldsymbol{x} \in \mathbb{R}^n$, input $\boldsymbol{u} \in \mathbb{R}^m$, output $\boldsymbol{y} \in \mathbb{R}^p$, and matrices $\boldsymbol{A}$, $\boldsymbol{B_u}$, $\boldsymbol{C}$, and $\boldsymbol{D_u}$ being of appropriate dimensions. System (1.1) is called strongly input redundant if

$$\mathcal{N}_\mathrm{r}\Bigg(\underbrace{\begin{bmatrix}\boldsymbol{B_u}\\ \boldsymbol{D_u}\end{bmatrix}}_{\overline{\boldsymbol{B_u}}}\Bigg) \neq \boldsymbol{0} \tag{1.2}$$

where $\mathcal{N}_\mathrm{r}(.)$ denotes the right nullspace of a linear mapping. It is a subspace of control space $\mathbb{R}^m$ which contains those elements which get mapped to zero by multiplication with $\overline{\boldsymbol{B_u}}$. Thus $\forall \boldsymbol{u_0} \in \mathcal{N}_\mathrm{r}(\overline{\boldsymbol{B_u}}), \boldsymbol{u} \notin \mathcal{N}_\mathrm{r}(\overline{\boldsymbol{B_u}}) : \overline{\boldsymbol{B_u}}\boldsymbol{u} = \overline{\boldsymbol{B_u}}\,(\boldsymbol{u} + \boldsymbol{u_0})$. Note that the strong input redundancy condition (1.2) remains meaningful in case of affine-input nonlinear systems where matrices $\boldsymbol{B_u}$ and $\boldsymbol{D_u}$ can be state-dependent.

Weakly input redundant systems are characterized by allowing infinitely many input vectors to result in the same steady-state output. If

$$\boldsymbol{P}^* = \lim_{s \to 0} \Big[\boldsymbol{C}\,(s\boldsymbol{I_n} - \boldsymbol{A})^{-1}\,\boldsymbol{B_u} + \boldsymbol{D_u}\Big] \tag{1.3}$$

is a finite matrix and

$$\mathcal{N}_{\mathrm{r}}\left(\boldsymbol{P}^{*}\right) \neq \mathbf{0} \tag{1.4}$$

then (1.1) is called weakly input redundant from $\boldsymbol{u}$ to $\boldsymbol{y}$ [2]. Choosing the system's output $\boldsymbol{y}$ as the DOFs over-actuation implies weak input redundancy, but not vice versa.

One possibility to deal with input redundancy is control allocation (CA). Therein, the problem of selecting one specific combination of actuator actions in order to achieve a desired effect is considered. The complexity of a CA problem significantly increases in the case of actuator constraints (e.g. magnitude and rate of each actuating signal). Typically, CA introduces an additional layer into the control system architecture. Figure 1.1 depicts the typical control scheme including CA:

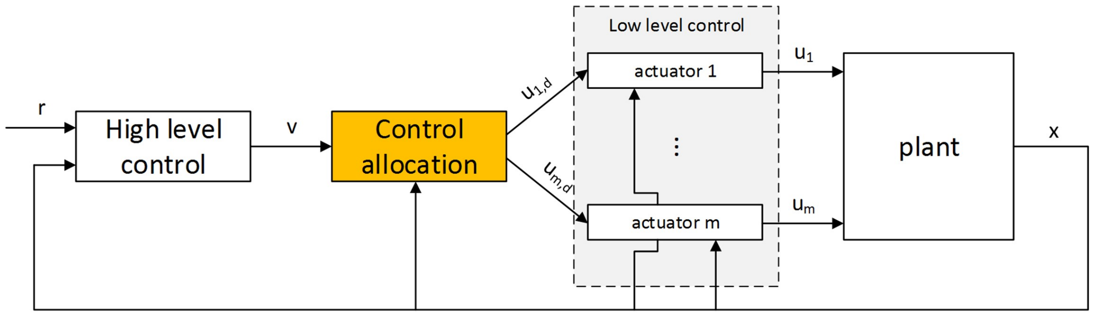

Figure 1.1.: System architecture incorporating control allocation.

A so-called high-level controller determines a virtual control signal $\boldsymbol{v}$, which serves as the input of the CA unit. This control signal is computed while relying on a redundancy-free plant model, i.e. the redundancy of the original system is not considered at this point. In context of motion control systems $\boldsymbol{v}$ is usually chosen as a number of forces or torques equal to the number of DOFs which should be controlled [1]. The CA algorithm is implemented on an intermediate layer. It distributes the desired virtual control effort among the real actuators, which constitute the low-level control layer. Each of the actuators can have its own controller resulting in a hierarchical control system. The benefits of the mentioned approach include:

- A modular control system that is straightforward to understand and to maintain.
- The high-level controller can be designed without detailed knowledge about the actuation system.
- Actuator constraints are explicitly considered in the CA algorithm.
- Actuator faults may easily be considered in the intermediate layer. They change the relation between virtual and real controls. Up to a certain extend high-level control is not affected, provided the faults are recognized.
- Over-actuation does not have to be considered in the high-level control layer. This broadens up the available techniques of controller design methods of this layer.

Most common CA techniques are based on linear plant models or on models gained by linearization. The fact that there may be infinitely many solutions to the CA task suggests its formulation as a constrained optimization problem. This supports the incorporation of secondary objectives such as minimizing the energy consumption into the CA problem. Popular objective function formulations lead to linear and quadratic programs, which generally require the application of numerical algorithms such as Simplex or Active-Set for their solution. Practically, the problem has to be solved within a discrete time environment with very limited computational resources. This work focuses on algorithm development for the mentioned class of problems. In particular, the capability of the resulting algorithms to operate under real-time conditions and high sampling rates is studied. Consequently, the computational complexity should be kept as small as possible and an efficient implementation is crucial for a successful application.

Originally CA methods were developed for flight control because modern aircraft come with numerous actuators (elevators, ailerons, rudders, canards, ...) which may influence multiple body axes at the same time. The virtual control vector typically consists of the moments about the roll, pitch, and yaw axes. All the available control effectors must be actuated according to the desired angular acceleration coming from the (auto-)pilot ([3], [4], [5], [6], [7], [8], [9]). Another application for CA can be found in marine crafts. Control systems for these vessels have to control surge, sway, and yaw during various positioning operations such as station keeping, low speed maneuvering, or auto-piloting. Actuators include different types of (turnable) propellers, thrusters and jets as well as rudders and fins ([10], [11], [12]). Moreover automotive control provides a lot of opportunities for the use of CA techniques. Powertrain electrification gains more and more importance and introduces additional actuators accompanied by input redundancy ([13], [14], [15]). Vehicle dynamics control is another field of application for CA as it can be seen in [16] and [17].

Of course CA is not the only technique to handle input redundant control problems. Linear Quadratic Regulators (LQR) [18] and Model Predictive Control (MPC) [19] provide alternatives, even though they are not restricted to input redundant plant models. Both methods overcome the ambiguity in the choice of actuator commands by solving optimization problems. While in the LQR case this problem is solved during controller design and constraints cannot be incorporated explicitly, MPC performs the optimization during its operation and facilitates the consideration of constraints. In contrast to CA there is no separation between control law and allocation in those methods.

1.2. A motivating example

In order to get a better understanding of the concepts of over-actuation and input redundancy a descriptive example from the automotive area is presented. Hybrid electric vehicles (HEVs) have the potential to reduce fuel consumption and exhaust emissions by combining electric machine (EM) and internal combustion engines (ICEs). An essential point for achieving better fuel economy than ordinary cars is an effective energy management strategy which determines the energy distribution between conventional and electric parts of the powertrain. The main goal is to meet the driver's power demand at any time by combining the available power sources. Additional objectives such as minimizing fuel consumption and emissions have to be considered while satisfying the constraints on battery state of charge and engine speeds ([20], [21]). The vehicle architecture considered in this example is a parallel HEV which means that power from ICE and EM can be used simultaneously to drive the car [21]. A schematic of the vehicle's powertrain is shown in Figure 1.2. Both ICE and EM propel the driving shaft and are connected by means of an extra clutch which allows disconnecting and shutting down of the ICE during EM-only operation. This is useful for reducing the drag torque and energy consumption. Especially at low speeds most energy management strategies suggest that the

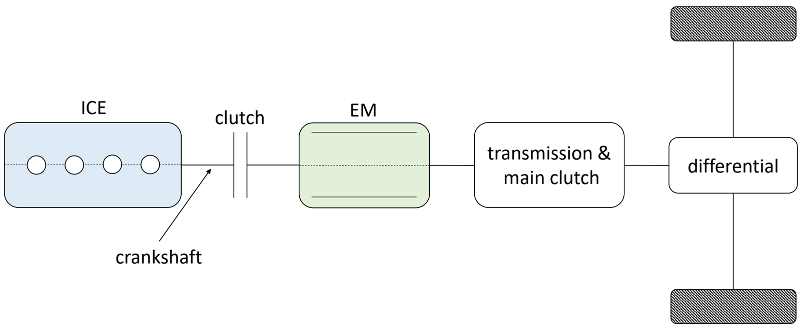

Figure 1.2.: A parallel HEV architecture with a crankshaft-integrated electric machine.

vehicle is driven exclusively by the EM with the ICE being switched off. Hence it is essential to provide a method to smoothly restart the engine while driving. ICE start-up is done in two phases. First the engine is accelerated up to the current powertrain speed by the EM and after that combustion is initiated. During the speed matching phase EM and clutch must be controlled such that the driver's torque request is still met despite the additional load of accelerating the ICE.

For the rest of this section only the driving shaft dynamics of the HEV powertrain are considered. The number of degrees of freedom of the two sides (ICE and EM) of the driving shaft depends on the clutch state. Assuming infinitely stiff shafts there is only one DOF if the clutch is completely closed while there are two DOFs otherwise. Figure 1.3 shows the relevant quantities for the system description. Inertia, internal friction, driving torque, and angular speed of the ICE are denoted as J_1, T_{fr1}, T_1, and ω_1 respectively. Subscript 2 labels the same quantities for the EM. Driving resistance is represented as torque T_{load} acting on the EM shaft. T_c stands for the torque transmitted by the clutch. A simple model for the shaft dynamics of

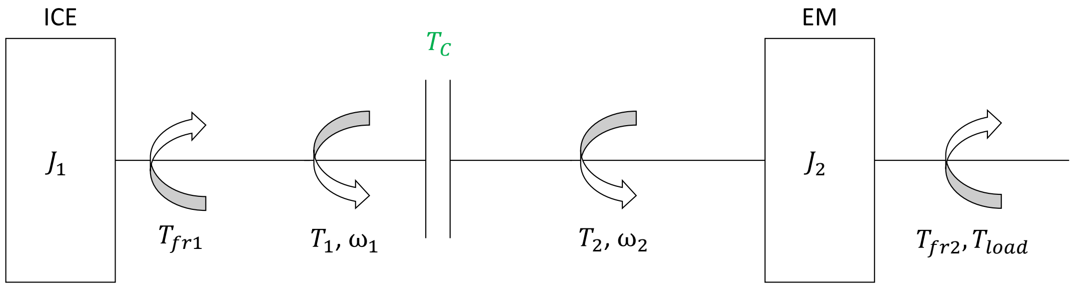

Figure 1.3.: Simplified model for the HEV driving shaft dynamics: two motors can be coupled via clutch.

ICE and EM reads as

$$\dot{\omega}_1 = \frac{1}{J_1} \left[T_1 - T_{fr1}(\omega_1) - T_c \right] \tag{1.5a}$$

$$\dot{\omega}_2 = \frac{1}{J_2} \left[T_2 - T_{fr2}(\omega_2) + T_c - T_{load} \right] . \tag{1.5b}$$

Typical friction clutches consist of at least two discs, each of them being connected to one input shaft. An actuator (e.g. electromechanic, electrohydraulic, or electromagnetic) creates an axial force which compresses the rotating discs. Torque is transmitted between the shafts due to the resulting friction. In order to compute T_c one has to distinguish whether the clutch is *slipping* or *locked* [22]. The effective torque entering the transmission and controlled via the accelerator pedal is

$$T_{eff} = T_2 + T_c. \tag{1.6}$$

1.2.1. Slipping clutch

The clutch is in slipping mode if the angular velocities of the shafts are not equal, i.e. one side rotates faster than the other one or in the opposite direction. In this case torque depends on the friction coefficient μ_{slip}, the active radius r_a, the number of friction surfaces n_s, and the force F_{ax} generated by the actuator. It can be written as [22]

$$T_c = \mu_{slip} r_a n_s F_{ax} \operatorname{sign}(\omega_1 - \omega_2). \tag{1.7}$$

1.2.2. Locked clutch

During locked mode both sides of the clutch spin with equal angular velocity. The system has now only one DOF instead of two. The clutch remains locked as long as the magnitude of the transmitted torque does not exceed the maximum $T_{c,max}$ which is governed by the stiction coefficient μ_{stick} and the actuator's axial force [22]

$$T_{c,max} = \mu_{stick} r_a n_s F_{ax}. \tag{1.8}$$

While the clutch is locked T_c cannot be manipulated directly, instead it follows from $\omega_1 \equiv \omega_2$. Equating (1.5a) and (1.5b) yields

$$\overline{T}_c = \frac{T_1 J_2 - T_{fr1}(\omega_1) J_2 - T_2 J_1 + T_{fr2}(\omega_2) J_1 + T_{load} J_1}{J_1 + J_2}. \tag{1.9}$$

Expression (1.9) ensures that both angular speeds stay identical. The torque which is actually transmitted during locked mode is obtained by combining (1.8) and (1.9)

$$T_c = \begin{cases} \overline{T}_c & \text{if } |\overline{T}_c| \le T_{c,max} \\ \operatorname{sign}(\overline{T}_c) T_{c,max} & \text{else} \end{cases}. \tag{1.10}$$

In fact the behavior of (1.5) in this state could also be described by a single differential equation as long as $\omega_1 = \omega_2$. Substituting (1.9) into (1.5) results in

$$\dot{\omega} = \frac{1}{J_1 + J_2} \left[T_1 + T_2 - T_{fr1}(\omega_1) - T_{fr2}(\omega_2) - T_{load} \right]. \tag{1.11}$$

1.2.3. Actuator dynamics

Equations (1.5) presume infinitely fast actuators which is not possible in reality. Unmodeled dynamics (e.g. combustion and EM dynamics, finite stiffness of shafts, ...) slow down torque generation. Thus the actuators are modeled as PT_1-elements

$$\begin{aligned} \dot{x} &= \frac{1}{\tau_i}(-x + u) \\ y &= x \end{aligned} \tag{1.12}$$

with time constants $\tau_i > 0$ and $i \in \{1, 2, ax\}$. Model (1.12) is an approximation of the closed-loop behavior of an actuator with a dedicated low-level controller.

1.2.4. Control-oriented plant models

Depending on the chosen strategy to perform the ICE start-up and the desired level of detail, different modeling approaches can be used as a basis of controller development. For all following models speed-proportional friction torques are assumed and the unknown input T_{load} is neglected. One possibility to realize the ICE-startup is using speed control for both shaft sides. Introducing the short notation

$$\mu(\boldsymbol{\omega}) = \mu_{slip} r_a n_s \operatorname{sign}(\omega_1 - \omega_2) \tag{1.13}$$

the plant model is obtained from the combination of (1.5) and (1.7) which yields

$$\begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} -\frac{k_1}{J_1} & 0 \\ 0 & -\frac{k_2}{J_2} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{1}{J_1} & 0 & -\frac{\mu(\boldsymbol{\omega})}{J_1} \\ 0 & \frac{1}{J_2} & \frac{\mu(\boldsymbol{\omega})}{J_2} \end{bmatrix}}_{\boldsymbol{B_{u1}}(\boldsymbol{\omega})} \begin{bmatrix} T_1 \\ T_2 \\ F_{ax} \end{bmatrix}. \tag{1.14}$$

The actuator dynamics (1.12) are neglected in (1.14). System (1.14) is strongly input redundant because $\operatorname{rank}(\boldsymbol{B_{u1}(\omega)}) = 2$ or equivalently $\mathcal{N}_r(\boldsymbol{B_{u1}(\omega)}) \neq \mathbf{0}$.

A different model taking the actuator dynamics (1.12) into account as well is given by

$$\begin{bmatrix} \dot{T}_1 \\ \dot{T}_2 \\ \dot{F}_{ax} \\ \dot{\omega}_1 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_1} & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{\tau_2} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\tau_{ax}} & 0 & 0 \\ \frac{1}{J_1} & 0 & -\frac{\mu(\boldsymbol{\omega})}{J_1} & -\frac{k_1}{J_1} & 0 \\ 0 & \frac{1}{J_2} & \frac{\mu(\boldsymbol{\omega})}{J_2} & 0 & -\frac{k_2}{J_2} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ F_{ax} \\ \omega_1 \\ \omega_2 \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{1}{\tau_1} & 0 & 0 \\ 0 & \frac{1}{\tau_2} & 0 \\ 0 & 0 & \frac{1}{\tau_{ax}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{B_{u2}}(\boldsymbol{\omega})} \begin{bmatrix} T_{1,d} \\ T_{2,d} \\ F_{ax,d} \end{bmatrix} \tag{1.15}$$

with $T_{1,d}$, $T_{2,d}$, and $F_{ax,d}$ being the desired torques and axial force. Owing to $\mathcal{N}_r(\boldsymbol{B_{u2}(\omega)}) = \mathbf{0}$ model (1.15) is not strongly input redundant. However, considering $\boldsymbol{\omega}$ as the system's output, i.e. $C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$, the dc-gain matrix

$$\boldsymbol{P}^* = \begin{bmatrix} \frac{1}{k_1} & 0 & -\frac{\mu(\boldsymbol{\omega})}{k_1} \\ 0 & \frac{1}{k_2} & \frac{\mu(\boldsymbol{\omega})}{k_2} \end{bmatrix} \tag{1.16}$$

is finite and weak input redundancy follows from $\mathcal{N}_r(\boldsymbol{P}^*) \neq \mathbf{0}$.

One recognizes that the chosen mathematical representation of a system determines whether it is strongly or weakly input redundant or neither. Instead over-actuation is a property of the real-world plant and does not depend on the model which is chosen for controller development. The driving shaft dynamics are over-actuated. If the clutch is in slipping mode the system has two DOFs and three inputs and if the clutch is locked one DOF is controlled by two inputs.

1.3. Problem statement

1.3.1. General nonlinear model

The majority of CA algorithms require a system description which is linear with respect to the control inputs. Consider a general nonlinear system of the form

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}) \end{aligned} \tag{1.17}$$

with $\boldsymbol{x} \in \mathbb{R}^n$ being the state vector, $\boldsymbol{u} \in \mathbb{R}^m$ the input vector, and $\boldsymbol{y} \in \mathbb{R}^p$ the system's output. In order to facilitate the separation between regulation and CA (1.17) has to be transformed into a special form. For that purpose one has to introduce the virtual control vector $\hat{\boldsymbol{v}} \in \mathbb{R}^k$ with $k < m$ and find a mapping

$$\hat{\boldsymbol{v}} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \quad \text{such that} \quad \forall \boldsymbol{x}, \boldsymbol{u} : \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \hat{\boldsymbol{f}}(\boldsymbol{x}, \underbrace{\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})}_{\hat{\boldsymbol{v}}}). \tag{1.18}$$

Since $\dot{\boldsymbol{x}}$ remains unaffected by an appropriate mapping (1.18) its existence can be interpreted as condition for the strong input redundancy of nonlinear systems. Next, high-level controller design is carried out for the modified plant dynamics

$$\begin{aligned} \dot{\boldsymbol{x}} &= \hat{\boldsymbol{f}}(\boldsymbol{x}, \hat{\boldsymbol{v}}) \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}). \end{aligned} \tag{1.19}$$

Typically, actuators are subject to constraints which define the feasible subset of m-dimensional (m-D) control space

$$\Omega \subset \mathbb{R}^m. \tag{1.20}$$

The general CA problem is to find a real control vector $\boldsymbol{u} \in \Omega$ which fulfills (1.18) for given virtual controls $\hat{\boldsymbol{v}}$. However, this goal can be unachievable if the desired $\hat{\boldsymbol{v}}$ exceeds the capabilities of the actuators because of their constraints. In such situations the objective of the CA algorithm is to minimize the allocation error in some sense. Hence, the CA problem can be stated as the optimization task

$$\min_{\boldsymbol{u} \in \Omega} \left\| \hat{\boldsymbol{v}} - \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \right\|. \tag{1.21}$$

Due to the fact that (1.18) is an underdetermined nonlinear system of equations its solution (provided that it exists) is not unique in general. This offers the opportunity to introduce secondary objectives by means of a cost function $J(\boldsymbol{x}, \boldsymbol{u})$ and reformulate the CA task as combined optimization problem

$$\min_{\boldsymbol{u} \in \Omega} \left\| \hat{\boldsymbol{v}} - \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \right\| + \gamma J(\boldsymbol{x}, \boldsymbol{u}) \tag{1.22}$$

where $\gamma > 0$ is a sufficiently small weighting parameter. The solution of (1.22) can be computationally expensive because the general formulation allows non-convex cost functions and constraints. In order to enable a real-time solution one common method is the consecutive linearization of (1.18) at each time step [1]. It can be locally approximated around the last commanded control vector $\boldsymbol{u_0}$ by the affine mapping ([7] and [23])

$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \approx \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u_0}) + \underbrace{\frac{\partial \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u_0})}{\partial \boldsymbol{u}}}_{\boldsymbol{B}(\boldsymbol{x})} (\boldsymbol{u} - \boldsymbol{u_0}). \tag{1.23}$$

Now a new CA equation which has a linear relationship between real and virtual controls follows

$$\boldsymbol{v} = \boldsymbol{B}(\boldsymbol{x}) \boldsymbol{u} \tag{1.24}$$

where $\boldsymbol{B}(\boldsymbol{x}) \in \mathbb{R}^{k \times m}$ is called *control effectivity matrix* and $\boldsymbol{v} \in \mathbb{R}^k$ is the new virtual control vector. It is computed from the high-level control's output (1.18) and (1.23) by means of

$$\boldsymbol{v} = \hat{\boldsymbol{v}} - \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u_0}) + \boldsymbol{B}(\boldsymbol{x}) \boldsymbol{u_0}. \tag{1.25}$$

Assuming that the current system state $\boldsymbol{x}$ is known (1.24) is reduced to $\boldsymbol{v} = \boldsymbol{B}\boldsymbol{u}$. This linear underdetermined system of equations is solved at each time step by the CA algorithm.

1.3.2. Input-affine model

Another commonly occurring class of (non)linear systems[1] is input-affine and reads as

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{a}(\boldsymbol{x}) + \boldsymbol{B_u}(\boldsymbol{x})\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{c}(\boldsymbol{x})\end{aligned} \tag{1.26}$$

with input matrix $\boldsymbol{B_u}(\boldsymbol{x}) \in \mathbb{R}^{n \times m}$. Under the assumption that

$$\forall \boldsymbol{x} : k = \operatorname{rank}\left(\boldsymbol{B_u}(\boldsymbol{x})\right) < m \tag{1.27}$$

it follows from the rank-nullity theorem [24] that $\dim\left[\mathcal{N}_{\mathrm{r}}\left(\boldsymbol{B_u}(\boldsymbol{x})\right)\right] = m - k$ which implies that (1.26) is strongly input redundant $\forall \boldsymbol{x}$. In other words $\boldsymbol{B_u}(\boldsymbol{x})$ does not have full column rank and so perturbations of $\boldsymbol{u}$ in nullspace directions do not affect the system dynamics. Condition (1.27) enables an input matrix factorization of $\boldsymbol{B_u}(\boldsymbol{x})$ into a virtual input matrix $\boldsymbol{B_v}(\boldsymbol{x}) \in \mathbb{R}^{n \times k}$ and a control effectivity matrix $\boldsymbol{B}(\boldsymbol{x}) \in \mathbb{R}^{k \times m}$, i.e.

$$\boldsymbol{B_u}(\boldsymbol{x}) = \boldsymbol{B_v}(\boldsymbol{x})\boldsymbol{B}(\boldsymbol{x}) \tag{1.28}$$

with both of their ranks being

$$\operatorname{rank}(\boldsymbol{B_v}(\boldsymbol{x})) = \operatorname{rank}(\boldsymbol{B}(\boldsymbol{x})) = k. \tag{1.29}$$

The factorization of $\boldsymbol{B_u}(\boldsymbol{x})$ according to (1.28) induces the altered plant model

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{a}(\boldsymbol{x}) + \boldsymbol{B_v}(\boldsymbol{x})\boldsymbol{v} \\ \boldsymbol{y} &= \boldsymbol{c}(\boldsymbol{x})\end{aligned} \tag{1.30}$$

which provides the basis for high-level controller development in further consequence. CA algorithms are applied on (1.24) at each time step, just as in the general case.

1.3.3. Constraints

Actuator constraints delimit the admissible subset of control space $\mathbb{R}^m$. The considered class of constraints forms a m-D hyperrectangle or box which is given by

$$\Omega = \left\{\boldsymbol{u} \in \mathbb{R}^m \,\middle|\, \boldsymbol{u_{min}} \leq \boldsymbol{u} \leq \boldsymbol{u_{max}}\right\}. \tag{1.31}$$

Definition (1.31) not only includes constraints regarding the magnitude but also the rate of actuator actions and even state-dependent constraints. All these can be treated as time-varying bounds $\boldsymbol{u_{min}}(t)$ and $\boldsymbol{u_{max}}(t)$ in principle. But due to the fact that CA is carried out in each time step the bounds are still considered as constant during one execution cycle. For example actuator magnitude and rate constraints

$$\underline{\boldsymbol{u}}_{\boldsymbol{min}} \leq \boldsymbol{u} \leq \overline{\boldsymbol{u}}_{\boldsymbol{max}}, \tag{1.32a}$$

$$|\dot{\boldsymbol{u}}| \leq \dot{\boldsymbol{u}}_{\boldsymbol{max}} \tag{1.32b}$$

can be combined to (1.31) with (see [8])

$$\boldsymbol{u_{max}} = \min\left(\overline{\boldsymbol{u}}_{\boldsymbol{max}}, \boldsymbol{u} + T_s\dot{\boldsymbol{u}}_{\boldsymbol{max}}\right), \tag{1.33a}$$

$$\boldsymbol{u_{min}} = \max\left(\underline{\boldsymbol{u}}_{\boldsymbol{min}}, \boldsymbol{u} - T_s\dot{\boldsymbol{u}}_{\boldsymbol{max}}\right). \tag{1.33b}$$

[1]Note that the linear system (1.1) is just a special form of (1.26).

where T_s is the sample time. Accordingly one can define a subset Φ of virtual control space $\mathbb{R}^k$ containing all $\boldsymbol{v}$ which can be achieved by feasible $\boldsymbol{u} \in \Omega$. It is called *attainable moment set* (AMS) [3] and is basically the projection of the m-D box Ω into lower dimensional virtual control space

$$\Phi = \boldsymbol{B}\Omega := \left\{ \boldsymbol{v} \in \mathbb{R}^k \middle| \exists \boldsymbol{u} \in \Omega : \boldsymbol{v} = \boldsymbol{B}\boldsymbol{u} \right\} . \tag{1.34}$$

Whereas the mapping from Ω to Φ is unique this is not the case in the opposite direction due to Ω's higher dimension, meaning that there are infinitely many $\boldsymbol{u} \in \mathbb{R}^m$ (not only in Ω) which fulfill (1.24) for a given $\boldsymbol{v} \in \Phi$. The challenge for CA algorithms is to avoid computing solutions $\boldsymbol{u} \notin \Omega$ to (1.24) although $\boldsymbol{v} \in \Phi$ [25].

The saturation function $\mathrm{sat}_\Omega(\boldsymbol{u})$ of a vector $\boldsymbol{u}$ regarding box-like constraints (1.31) is often employed in the rest of this thesis. It is defined by means of the component-wise saturation function $\mathrm{sat}_{\Omega,i}(\boldsymbol{u})$ with index $i = 1, \ldots, m$ as

$$\mathrm{sat}_\Omega(\boldsymbol{u}) = [\mathrm{sat}_{\Omega,1}(\boldsymbol{u}) \; \ldots \; \mathrm{sat}_{\Omega,m}(\boldsymbol{u})]^T \tag{1.35a}$$

$$\mathrm{sat}_{\Omega,i}(\boldsymbol{u}) = \begin{cases} u_{max,i} & \text{if } u_i > u_{max,i} \\ u_{min,i} & \text{if } u_i < u_{min,i} \\ u_i & \text{else} \end{cases} . \tag{1.35b}$$

1.3.4. Summary

The main objective considered in this thesis is the development of CA-methods for strongly input redundant systems. Mathematically this can be stated as finding computationally efficient solutions to

$$\min_{\boldsymbol{u} \in \Omega} \|\boldsymbol{v} - \boldsymbol{B}\boldsymbol{u}\| . \tag{1.36}$$

As (1.36) may have infinitely many solutions additional goals can be specified as cost functions augmenting the optimization problem.

1.4. Thesis outline

Chapter 2 presents an overview on the standard methods for controller design for input redundant systems. The main focus lies on the description of various CA algorithms. However, the concepts of two very common alternatives are also briefly outlined: LQR and MPC.

The next part of this thesis is dedicated to some theoretical results on linear CA problems. First, the influence of input matrix factorization on standard CA techniques is investigated. Based on those results a new method to generate generalized inverses for CA is presented in Chapter 4. It incorporates the actuator constraints into its computation leading to feasible allocation results for a greater amount of the AMS as it typically is the case for the standard approach. Another outcome of the factorization research is an extension of an established iterative CA algorithm which is developed in Chapter 5. In cases where the desired virtual controls cannot be reached due to the actuator constraints it enables the prioritization of certain virtual control vector elements by the developer. Subsequently, an efficient algorithm for solving quadratic programs with linear constraints is introduced in Chapter 6. It is based on the penalty function and gradient projection methods. Extensive testing and comparisons against free and commercial standard solvers demonstrate the capabilities of the approach. It may not only be used in the context of CA but also, for instance, as solver for MPC.

The final part deals with the application of CA in practical examples. At the beginning solutions of the ICE-startup problem described in Section 1.2 are presented. Two possible schemes of high-level control and CA are derived and evaluated in simulations. In the last chapter CA methods are implemented for the control of a laboratory model: a quadrotor test bed. A comprehensive nonlinear model is developed by means of Lagrangian dynamics and parameter identification is carried out. In order to meet the control objective of attitude tracking various strategies are realized. The performance of CA-based schemes using a *Sliding Mode* approach for high-level control are compared to the alternatives LQR and MPC.

2. Control Design in Consideration of Input Redundancy

In principle, there are two strategies for dealing with input redundancy in controller development. First, there are methods which directly incorporate the distribution of control effort among the actuators in their control laws. Due to the fact that in general there are infinitely many possibilities of blending actuator effects resulting in the same virtue on the plant, these techniques are related to some kind of optimization in order to find a unique solution. Sections 2.2 and 2.3 explain the principles of two such methods, namely LQR and MPC.

Second, one can separate the allocation from the control task by introducing a dedicated algorithm for this purpose as already outlined in the introduction (Chapter 1). In most CA approaches the input redundancy is neglected during high-level control development by introducing the virtual control vector and relying on an altered system description like (1.19) or (1.30). Since high-level controller design is more or less independent from CA it is not treated in this chapter. Actually, it is neither possible nor meaningful to deal with this topic as it is basically possible to use any kind of controller for this task, depending only on the particular application. The following Section 2.1 provides the reader with the principles of common CA techniques.

2.1. Control allocation

Fundamentally, CA algorithms can be divided into two groups. On one hand there is *static allocation* which means that no dynamics are involved in the computations. These CA methods provide a mapping $\boldsymbol{u_{cmd}} = \boldsymbol{f_{st,CA}}(\boldsymbol{v})$ from virtual to real controls which is independent of time. *Dynamic allocation* on the other hand comprises a gradual progress of the allocation result towards the target value, i.e.

$$\dot{\boldsymbol{u}}_{\boldsymbol{cmd}} = \boldsymbol{f_{dy,CA}}(\boldsymbol{v}). \tag{2.1}$$

Typically, the final value is the result of an equivalent optimization problem (see [2] and [26]). It should be emphasized that this distinction does not necessarily have something to do with the fact whether actuator dynamics are considered in the allocation process or not. Expression (2.1) just means that the *commanded* real controls evolve over time in a certain way. Actuator dynamics can be taken into account in both static and dynamic CA. The majority of CA methods relies on linear actuation models

$$\boldsymbol{v} = \boldsymbol{B}\boldsymbol{u} \tag{2.2}$$

with the control effectivity matrix $\boldsymbol{B} \in \mathbb{R}^{k \times m}$ characterizing the relationship between virtual and real controls. However, for most applications matrix $\boldsymbol{B}$ changes with the system state and/or input and other time-varying parameters. Thus, (2.2) does not restrict CA to linear time-invariant (LTI) systems because $\boldsymbol{B}$ gets updated in every sampling instant.

2.1.1. Explicit ganging

In some cases, especially if the number of virtual and real controls is rather small, it is known a priori how to combine the actuator actions to achieve a certain $\boldsymbol{v}$. Therefore, one can use a method called *Explicit Ganging* where the allocation is explicitly predefined by the control designer. Actuators are grouped to so-called *pseudo controls* $\boldsymbol{u_{pseudo}} \in \mathbb{R}^k$ which have the same dimension as the virtual controls $\boldsymbol{v}$. The relationship between pseudo controls and real controls is given by [8]

$$\boldsymbol{u} = \boldsymbol{G}\boldsymbol{u_{pseudo}} \tag{2.3}$$

with $\boldsymbol{G} \in \mathbb{R}^{m \times k}$ having full column rank. Inserting (2.3) into (2.2) yields

$$\boldsymbol{v} = \boldsymbol{B}\boldsymbol{G}\boldsymbol{u_{pseudo}} \quad \Rightarrow \quad \boldsymbol{u_{pseudo}} = (\boldsymbol{B}\boldsymbol{G})^{-1}\,\boldsymbol{v}. \tag{2.4}$$

The real controls $\boldsymbol{u}$ are now obtained from (2.3) and (2.4). Effectively, this method reduces the control space dimension in order to get a unique relationship between virtual and pseudo controls. After that, the predefined mapping $\boldsymbol{G}$ provides the distribution on the real controls [8]. The actuator constraints (1.31) are not considered, excesses are just cut off. The desired virtual controls will in general not be reached in such situations.

2.1.2. Pseudoinverse

Computing a generalized inverse or pseudoinverse is a common way to solve linear systems of equations with non-square parameter matrices. For every matrix $\boldsymbol{B} \in \mathbb{R}^{k \times m}$ there exist matrices $\boldsymbol{P} \in \mathbb{R}^{m \times k}$ satisfying at least one of the four so-called Penrose-equations [25] [27]

$$\boldsymbol{BPB} = \boldsymbol{B} \tag{2.5a}$$
$$\boldsymbol{PBP} = \boldsymbol{P} \tag{2.5b}$$
$$(\boldsymbol{BP})^* = \boldsymbol{BP} \tag{2.5c}$$
$$(\boldsymbol{PB})^* = \boldsymbol{PB}, \tag{2.5d}$$

where $\boldsymbol{A}^*$ denotes the conjugate transpose of a matrix $\boldsymbol{A}$. Since $\boldsymbol{B}$ is assumed to have full row rank a matrix $\boldsymbol{P}$ which satisfies (2.5a) is a right-inverse of $\boldsymbol{B}$, i.e. $\boldsymbol{BP} = \boldsymbol{I_k}$ with $\boldsymbol{I_k} \in \mathbb{R}^{k \times k}$ being the identity matrix [27]. The solution to the CA problem can now be written as [25]

$$\boldsymbol{u} = \boldsymbol{P}\boldsymbol{v}, \tag{2.6}$$

whereby those elements of $\boldsymbol{u}$ which exceed Ω are set to their extremal values ([3] and [4]).

2.1.2.1. Weighted pseudoinverse

The unconstrained control allocation problem (2.2) has infinitely many solutions and a reasonable choice is to pick that with the lowest energy consumption (least-norm solution [28], [29]). Using a constant positive definite weighting matrix $\boldsymbol{W} \succ 0$ and a constant offset vector $\boldsymbol{c} \in \mathbb{R}^m$, an optimization problem can be formulated [1] [8] [30]

$$\min_{\boldsymbol{u}} \; (\boldsymbol{u} + \boldsymbol{c})^T \, \boldsymbol{W} \, (\boldsymbol{u} + \boldsymbol{c}), \tag{2.7}$$

which is subject to equality constraint (2.2). The purpose of $\boldsymbol{c}$ is to specify preferred control positions or to account for faulty actuators [8]. For example assume that the favored actuator positions minimizing the air drag of a plane for its current state are given by $\boldsymbol{u_p}$. Then solving (2.7) with $\boldsymbol{c} = -\boldsymbol{u_p}$ minimizes the deviation from $\boldsymbol{u_p}$. The closed-form solution of (2.7) is derived with the method of Lagrange multipliers and reads as

$$\boldsymbol{u} = -\boldsymbol{c} + \boldsymbol{B}^{\#} (\boldsymbol{v} + \boldsymbol{Bc}) \tag{2.8a}$$

$$\boldsymbol{B}^{\#} = \boldsymbol{W}^{-1}\boldsymbol{B}^{T} \left(\boldsymbol{B}\boldsymbol{W}^{-1}\boldsymbol{B}^{T}\right)^{-1}. \tag{2.8b}$$

$\boldsymbol{B}^{\#} \in \mathbb{R}^{m \times k}$ is called weighted pseudoinverse and if $\boldsymbol{W} = \boldsymbol{I_m}$ then $\boldsymbol{B}^{\#} = \boldsymbol{B}^{\ddagger}$ is the Moore-Penrose pseudoinverse (MPP) [28] [30]. It satisfies all Penrose-equations (2.5) and is unique for a given matrix $\boldsymbol{B}$.

If $\operatorname{rank}(\boldsymbol{B}) < k$, e.g. due to actuator faults, the closed form solution (2.8b) is not applicable any more. In such cases a rank-deficient pseudoinverse can still be calculated using either regularization techniques like

$$\boldsymbol{B}^{\#} = \boldsymbol{W}^{-1}\boldsymbol{B}^{T} \left(\boldsymbol{B}\boldsymbol{W}^{-1}\boldsymbol{B}^{T} + \epsilon \boldsymbol{I_k}\right)^{-1} \tag{2.9}$$

with small $\epsilon > 0$ or the Singular Value Decomposition (SVD, see Chapter 5 for more details). However, in general the desired $\boldsymbol{v}$ cannot be reached even without considering the actuator constraints [1].

2.1.3. Daisy chaining

This approach is used when a fixed hierarchy of actuator utilizations can be specified. In other words, a certain subset of the actuators should be preferably used to accomplish the CA task. If the desired virtual controls cannot be reached by them exclusively the remaining virtual control demand is passed to the next group and so on. For this reason, the real control vector is divided into a fixed number of groups [8]

$$\boldsymbol{u} = \left[\boldsymbol{u_1}^T \dots \boldsymbol{u_N}^T\right]^T \tag{2.10}$$

where a low group number stands for high priority. The control effectivity matrix is partitioned in the same fashion as

$$\boldsymbol{B} = [\boldsymbol{B_1} \dots \boldsymbol{B_N}] \tag{2.11}$$

and for each group $j = 1, \dots, N$ a feasible subset $\Omega_j \subset \Omega$ exists. The algorithm starts with initializing the control vector $\boldsymbol{u} \leftarrow \boldsymbol{0}$, the current group number $j \leftarrow 1$ and the virtual control deviation $\boldsymbol{v_j} = \boldsymbol{v}$. Thereafter, the following steps are taken [4]:

1. Compute the allocation result for the current group by means of the MPP of $\boldsymbol{B_j}$

$$\tilde{\boldsymbol{u}}_{\boldsymbol{j}} = \boldsymbol{B}_{\boldsymbol{j}}^{\ddagger} \boldsymbol{v_j}. \tag{2.12}$$

2. Truncate the previous result

$$\boldsymbol{u_j} = \operatorname{sat}_{\Omega_j}(\tilde{\boldsymbol{u}}_{\boldsymbol{j}}). \tag{2.13}$$

3. If there was a constraint violation in (2.12), i.e. $\tilde{\boldsymbol{u}}_{\boldsymbol{j}} \notin \Omega_j$ then compute the next virtual control deviation from (2.13)

$$\boldsymbol{v_{j+1}} = \boldsymbol{v_j} - \boldsymbol{B_j}\boldsymbol{u_j}, \tag{2.14}$$

set $j \leftarrow j + 1$, and go back to step 1.

4. Otherwise: STOP

Note that depending on rank($\boldsymbol{B_j}$) and the number of group members the MPP in (2.12) can be an ordinary (square) inverse, a right-inverse, a left-inverse, or a rank-deficient inverse [4]. Although Daisy Chaining takes the constraints (1.31) into account during its iterations it is not guaranteed that a feasible $\boldsymbol{u}$ is found $\forall \boldsymbol{v} \in \Phi$. Due to the fact that all the controls in one group are frozen as soon as one of them violates the constraints it can easily happen that a solution is not found, though many actuators are barely utilized [1].

2.1.4. Redistributed pseudoinverse

Redistributed Pseudoinverse (RPINV) is an iterative process that takes the actuator constraints into account. During the first step the unconstrained solution according to the conventional weighted pseudoinverse (2.8b) is calculated. If no controls exceed their limits, no further steps are required. Otherwise, the actuators that violate the constraints are fixed at their extremal values (saturated) and a reduced pseudoinverse is computed for the remaining free controls. This procedure is repeated until no new constraint violations occur or all actuators are at their limits ([1], [8]). Let $\boldsymbol{B_0}$ be the control effectivity matrix, $\boldsymbol{u}^N$ the result of the N-th iteration, and $\boldsymbol{u}^0 = \boldsymbol{0}$. Constraint violations are recorded in an offset vector [30]

$$\boldsymbol{c}^N = [c_1^N \; ... \; c_m^N]^T \tag{2.15a}$$

$$c_i^N = \begin{cases} -u_{max,i} & \text{if } u_i^{N-1} \geq u_{max,i} \\ -u_{min,i} & \text{if } u_i^{N-1} \leq u_{min,i} \\ 0 & \text{else} \end{cases} \quad \text{with } i = 1, ..., m. \tag{2.15b}$$

The set of free actuator indices sorted in ascending order reads as

$$J^N = \{l \in \mathbb{N}^+ | c_l^N = 0\} = \{l_1, ..., l_j\} \tag{2.16}$$

and $j = |J^N|$ denotes its number. Using the i-th unit vector $\boldsymbol{e_i} \in \mathbb{R}^m$ one can define the column-selection matrix [30]

$$\boldsymbol{R} = \left[\boldsymbol{e_{l_1}} \; ... \; \boldsymbol{e_{l_j}}\right]. \tag{2.17}$$

By means of (2.17) the modified control effectivity matrix with columns related to saturated controls set to zero is given by

$$\boldsymbol{B}^N = \boldsymbol{B_0}\boldsymbol{R}\boldsymbol{R}^T. \tag{2.18}$$

Applying (2.8b) the result of the N-th iteration of RPINV reads as

$$\boldsymbol{u}^N = -\boldsymbol{c}^N + \boldsymbol{B}^{\#}\left(\boldsymbol{v} + \boldsymbol{B_0}\boldsymbol{c}^N\right). \tag{2.19}$$

Remark 2.1.1. The resulting virtual control after the application of RPINV and neglecting potentially exceeded constraints is denoted as $\boldsymbol{v_{act}}$. Considering $\boldsymbol{v} = \boldsymbol{B_0}\boldsymbol{u}^N$ and (2.19) one obtains [30]

$$\boldsymbol{v_{act}} = -\boldsymbol{B_0}\boldsymbol{c}^N + \boldsymbol{B_0}\boldsymbol{B}^{\#}\boldsymbol{v_{des}} + \boldsymbol{B_0}\boldsymbol{B}^{\#}\boldsymbol{B_0}\boldsymbol{c}^N \tag{2.20}$$

where $\boldsymbol{v_{des}}$ is the desired value coming from the controller. One realizes from (2.20) that $\boldsymbol{v_{des}}$ can be reached in principle if $\boldsymbol{B_0}\boldsymbol{B}^{\#} = \boldsymbol{I_k}$. Inserting (2.18) into (2.8b) yields $\boldsymbol{B}^{\#} = \boldsymbol{W}^{-1}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{B_0}^T\left(\boldsymbol{B_0}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{W}^{-1}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{B_0}^T\right)^{-1}$. In order to achieve $\boldsymbol{B}\boldsymbol{B}^{\#} = \boldsymbol{B_0}\boldsymbol{B}^{\#} = \boldsymbol{I_k}$ the computation is altered into [30]

$$\boldsymbol{B}^{\#} = \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{W}^{-1}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{B_0}^T\left(\boldsymbol{B_0}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{W}^{-1}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{B_0}^T\right)^{-1}. \tag{2.21}$$

RPINV is a simple and effective method of regarding actuator constraints but just as Daisy Chaining it does not use the entire AMS, though it tends to be more effective [1]. An intuitive explanation is that it is advantageous to retain only those controls which violate the constraints, because this increases the overall actuator utilization.

2.1.5. Direct allocation

Direct allocation (DA) uses geometric principles to compute $\boldsymbol{u}$ from a given virtual control vector $\boldsymbol{v}$. Its main advantage is its ability to determine a $\boldsymbol{u} \in \Omega$ for every $\boldsymbol{v} \in \Phi$, i.e. it facilitates the usage of the entire AMS. Figures 2.1 and 2.2 give a visual interpretation of the DA problem in case of $m = 3$ and $k = 2$. The following requirements have to be met [4] [30]:

- A half-line starting in the origin of Φ intersects its boundary $\partial(\Phi)$ in a single point, see Figure 2.2.
- Every point on $\partial(\Phi)$ uniquely maps to one point on $\partial(\Omega)$ (the boundary of Ω) in each case, i.e.

$$\boldsymbol{v}^* \in \partial(\Phi) \Rightarrow \exists! \boldsymbol{u}^* \in \partial(\Omega) \,:\, \boldsymbol{v}^* = B\boldsymbol{u}^*. \tag{2.22}$$

At first DA computes the intersection of vector $\boldsymbol{v}$ with the boundary $\partial(\Phi)$. This solution $\boldsymbol{v}^*$ is always located on a (k-1)-D object and is the image of one point $\boldsymbol{u}^*$ on $\partial(\Omega)$. Now $\boldsymbol{v}^*$ is scaled by a factor $0 < a \in \mathbb{R}$ to match $\boldsymbol{v}$. In order to obtain $\boldsymbol{u}$ the same scaling can be applied to $\boldsymbol{u}^*$ because of the linearity of the problem [30]

$$\boldsymbol{v} = a\boldsymbol{v}^* = aB\boldsymbol{u}^* = B(a\boldsymbol{u}^*). \tag{2.23}$$

If $a > 1$, the desired virtual control lies outside of Φ. In this case the boundary-intersecting solution $\boldsymbol{u}^*$ is chosen, which preserves the desired virtual control direction ([3], [4])

$$\boldsymbol{u} = \alpha\boldsymbol{u}^* \quad \text{with} \quad \alpha = \begin{cases} a & \text{if } 0 < a \leq 1 \\ 1 & \text{else} \end{cases}. \tag{2.24}$$

Thus, the direction of the desired $\boldsymbol{v}$ is maintained even if $\boldsymbol{v} \notin \Phi$. This preservation of virtual control direction is an important characteristic in flight control systems. One disadvantage of DA is the lack of any possibility for prioritization or influencing the allocation result [31]. A crucial point in DA is the determination of the intersection of $\boldsymbol{v}$ with the boundary of the AMS $\partial(\Phi)$. One possibility is to project all vertices of Ω into $\mathbb{R}^k$, compute the convex hull $\partial(\Phi)$, and iterate through all of its (k-1)-D objects. In order to avoid the expensive convex hull computation [4] presents a method which is based on the relation between the geometry of $\partial(\Phi)$ and $\partial(\Omega)$ to speed up the procedure. The approach exploits the fact that only a small portion of all (k-1)-D object on $\partial(\Omega)$ also lie on $\partial(\Phi)$. It uses an efficient way to determine this objects resulting in a significantly reduced number of computations during the search for the intersection $\boldsymbol{v}^*$ [4]. Alternatively, DA can also be formulated as optimization problem [31]

$$\max_{\rho>0,\ \boldsymbol{u}^*\in\Omega} \rho \qquad \text{subject to} \quad \boldsymbol{Bu}^* = \rho\boldsymbol{v} \tag{2.25a}$$

$$\boldsymbol{u} = \begin{cases} \frac{\boldsymbol{u}^*}{\rho} & \text{if } \rho > 1 \\ \boldsymbol{u}^* & \text{else.} \end{cases} \tag{2.25b}$$

and transformed into a standard linear program (LP). By relying on a dedicated solver for linear programs the advantage of (2.25) is the low development effort.

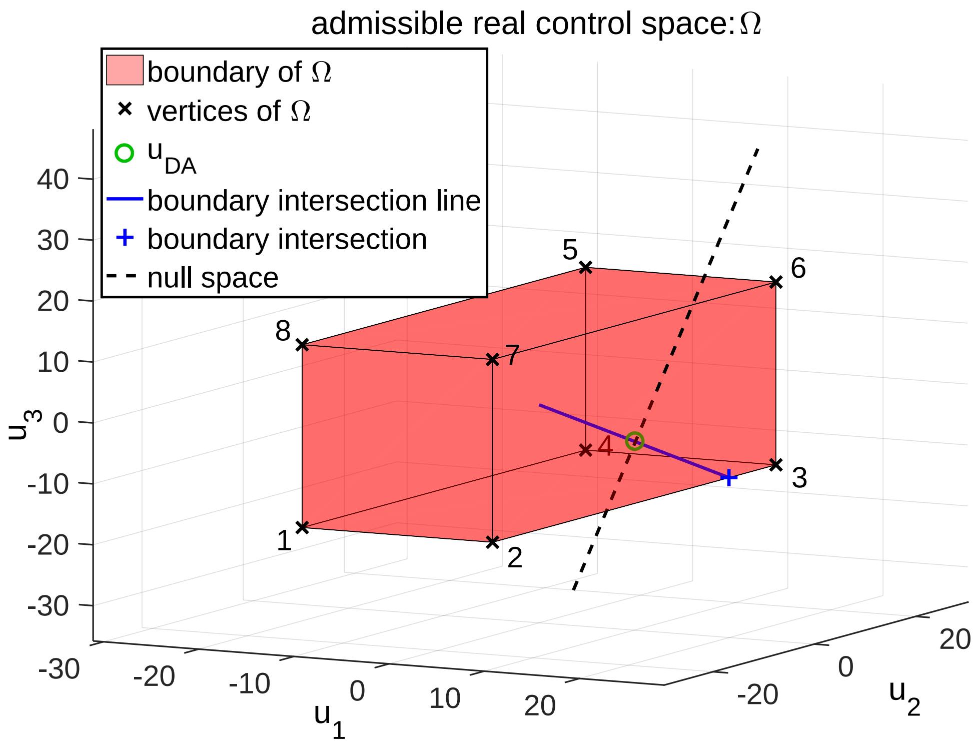

Figure 2.1.: DA computes the solution $\boldsymbol{u_{DA}}$ (green 'o', inside the box) for the desired virtual control $\boldsymbol{v_{des}}$ (see Figure 2.2) by scaling down the boundary intersection (blue '+'). Moving along the (m-k)-D nullspace direction (black dashed line) does not change the resulting virtual control. (© 2018 IEEE, [30]).

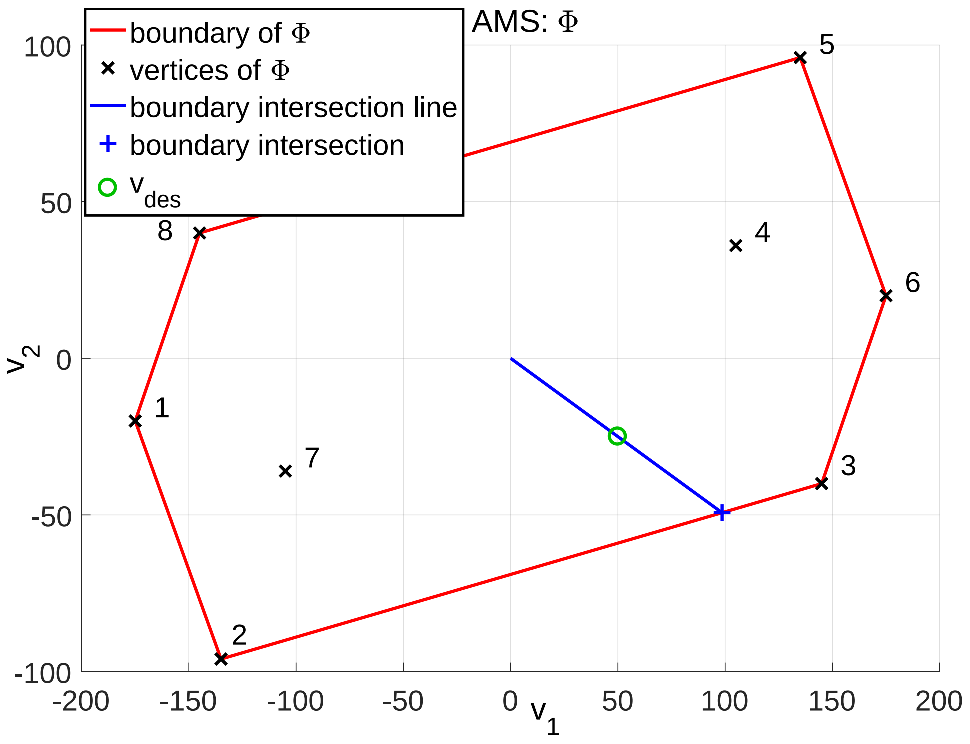

Figure 2.2.: 2-D virtual control space related to Figure 2.1: The numbered black 'x' represent the vertices of Ω. Some of them are not part of $\partial(\Phi)$. For $\boldsymbol{v_{des}} = [50\ \text{-}25]^T$ DA computes the intersection with $\partial(\Phi)$ and only checks those (k-1)-D objects of $\partial(\Omega)$ for intersection which also belong to $\partial(\Phi)$. (© 2018 IEEE, [30]).

2.1.6. CA via numerical optimization

Numerical optimization is a powerful approach to solve the CA problem. Especially, its ability to explicitly consider the actuator constraints (1.31) while still enabling actuator or virtual control weighting distinguishes this method from the previous ones. However, increased computational complexity compared to the pseudoinverse-based algorithms is the price one has to pay in exchange.

The basic problem formulation with respect to the linear actuation model (2.2) is given in the introduction by (1.36) and can be extended by a weighting matrix $\boldsymbol{W_v} \in \mathbb{R}^{k\times k}$ leading to

$$\min_{\boldsymbol{u}\in\Omega} \|\boldsymbol{W_v}(\boldsymbol{Bu}-\boldsymbol{v})\|. \tag{2.26}$$

Problem (2.26) is called *error minimization.* As already discussed the solution of (2.26) is in general not unique for feasible virtual controls, i.e. if $\boldsymbol{v} \in \Phi$ there are infinitely many $\boldsymbol{u} \in \Omega$ making the cost function of (2.26) zero [1] [6] [31]. In such situations a second optimization problem

$$\min_{\boldsymbol{u}\in\Omega} \|\boldsymbol{W_u}(\boldsymbol{u}-\boldsymbol{u_p})\| \qquad \text{subject to} \quad \boldsymbol{Bu}=\boldsymbol{v} \tag{2.27}$$

with weighting matrix $\boldsymbol{W_u} \in \mathbb{R}^{m\times m}$ and preferred control positions $\boldsymbol{u_p} \in \mathbb{R}^m$ can be solved in order to get a unique solution. This two-step procedure is known as *error and control minimization* [8] [31].

A drawback of coping with (2.26) and (2.27) sequentially is the fact that two optimization problems have to be solved (for feasible $\boldsymbol{v}$) which requires considerable effort. In order to avoid this, both objectives can be combined into a single cost function as indicated in (1.22) leading to the *mixed optimization* problem [8] [31]

$$\min_{\boldsymbol{u}\in\Omega} \big(\|\boldsymbol{W_u}(\boldsymbol{u}-\boldsymbol{u_p})\| + \gamma \|\boldsymbol{W_v}(\boldsymbol{Bu}-\boldsymbol{v})\| \big) \tag{2.28}$$

with a large weighting factor $\gamma > 0$ emphasizing the primary CA objective. So far, the 1-norm, the 2-norm, and the ∞-norm have been successfully applied to (2.26) - (2.28). The decision which norm is chosen not only influences the applicable algorithms for solving the optimization problems but obviously also the outcome. As shown in [7] the main differences of the allocation results are:

- The 1-norm solution utilizes as few controls as possible to reach $\boldsymbol{v}$ whereas in case of the 2-norm the control effort is distributed among all elements of $\boldsymbol{u}$.
- For a nonsingular $\boldsymbol{W_u}$ the 2-norm solution is unique. Depending on $\boldsymbol{B}$ and $\boldsymbol{W_u}$ this is not true for the result regarding the 1-norm [32].
- Continuous changes of parameters of $\boldsymbol{B}$ also yield continuous variations in the 2-norm solution while the 1-norm solution varies discontinuously.

Using the ∞-norm which is defined as

$$\|\boldsymbol{u}-\boldsymbol{u_p}\|_\infty = \max_i |u_i - u_{p,i}| \tag{2.29}$$

for (2.27) minimizes the maximum deflection of actuators from their preferred positions. This leads to a more balanced actuator utilization compared to the 1-norm solution. Both 1-norm and ∞-norm problems (2.26) - (2.28) can be transformed into standard LPs [33]. Common algorithms for solving LPs are based on the *simplex method*, the *active-set method*, and *interior-point methods* with the first one being by far the most widely used for CA [1] [31].

The choice of the 2-norm enables the transformation of (2.26) - (2.28) to standard quadratic programs (QP). In context of CA such problems are solved with the *fixed-point method* [34], active-set methods [6], and interior-point methods [35]. Algorithms based on the active-set strategy perform very well in case of a moderate number of actuators and offer the advantage of using the previous allocation result as an initial guess. Interior-point methods on the other hand are superior for large numbers of actuators [35].

2.1.7. Dynamic allocation for linear plant models

All CA algorithms discussed so far belong to the static methods, i.e. the computations directly provide $\boldsymbol{u}$. The *dynamic allocation* approach in contrast uses integral action in the CA law. Starting from a linear plant model (1.1) assume that there is already a linear controller and that the closed-loop system is internally stable. The dynamic allocator (DYA) is introduced in [2] as

$$\dot{\boldsymbol{w}} = -\boldsymbol{K}\boldsymbol{N}^T\boldsymbol{W}\boldsymbol{u} \tag{2.30a}$$

$$\boldsymbol{u} = \boldsymbol{y_c} + \boldsymbol{N}\boldsymbol{w} \tag{2.30b}$$

with $\boldsymbol{w}$ representing the allocator states, $\boldsymbol{K} \succ 0$ and $\boldsymbol{W}$ are gain and weighting matrices of appropriate dimensions, $\mathrm{span}(\boldsymbol{N}) = \mathcal{N}_\mathrm{r}(\overline{\boldsymbol{B_u}})$ spans the nullspace according to definition (1.2), and $\boldsymbol{u}$ is the plant input consisting of the controller output $\boldsymbol{y_c}$ and the allocator output. The DYA is inserted between plant and controller as illustrated in Figure 2.3 [2]. In case of strong

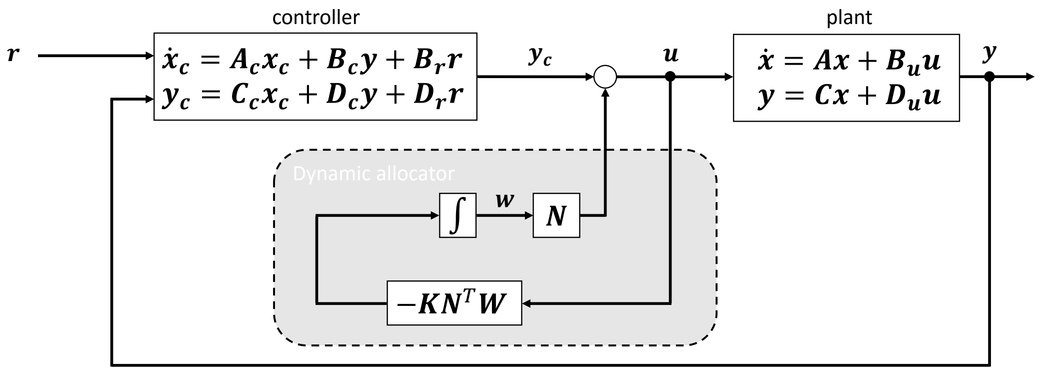

Figure 2.3.: Dynamic allocation scheme with the dynamic allocator being inserted between controller and plant.

input redundancy of (1.1) the allocator is completely invisible at the resulting dynamics and the interconnection as shown in Figure 2.3 is internally stable iff the control system without DYA is internally stable. This is not surprising as the plant dynamics read as [2]

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B_u}\boldsymbol{y_c} + \underbrace{\boldsymbol{B_u}\boldsymbol{N}}_{\boldsymbol{0}}\boldsymbol{w} \tag{2.31}$$

which allows the simplification of the stability analysis by considering a cascade connection between the original control loop and the DYA as depicted in Figure 2.4 instead. At the first glance it might seem pointless to add a zero-term in (2.31). But actually it is the essence of CA for strongly input redundant systems to choose $\boldsymbol{u}$ from a set of inputs which all have the same effect on the system dynamics. The steady-state output of the DYA (for a constant controller output $\boldsymbol{y}_c^*$) is [2]

$$\boldsymbol{u}^* = \left[\boldsymbol{I} - \boldsymbol{N}\left(\boldsymbol{N}^T\boldsymbol{W}\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\boldsymbol{W}\right]\boldsymbol{y}_c^* \tag{2.32}$$

which is the solution of the optimization problem

$$\min_{\boldsymbol{w}} \left(\boldsymbol{y}_c^* + \boldsymbol{N}\boldsymbol{w}\right)^T \boldsymbol{W}\left(\boldsymbol{y}_c^* + \boldsymbol{N}\boldsymbol{w}\right). \tag{2.33}$$

By means of the gain matrix $\boldsymbol{K}$ the allocator's speed of convergence to (2.32) can be adjusted and in case of strong input redundancy there is no restriction in this respect. Static allocation via pseudoinverse yields comparable results [2].

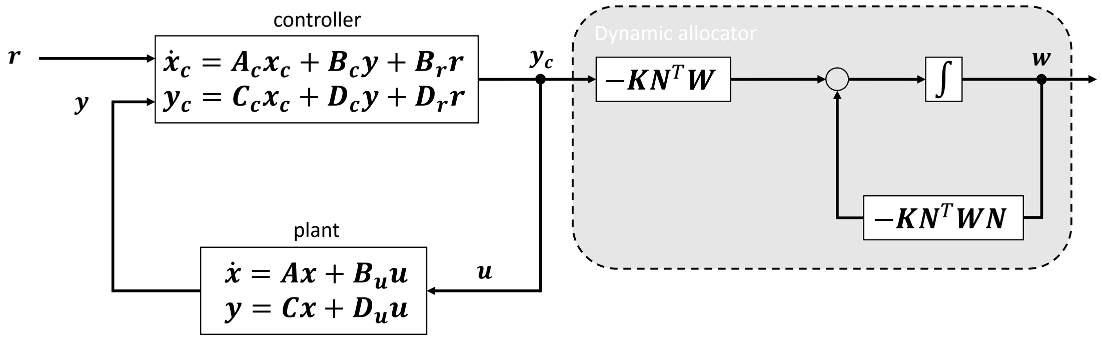

Figure 2.4.: The independence of the plant dynamics from the allocator output in case of strong input redundancy enables the stability analysis for a substitute system.

If (1.1) is weakly input redundant the structure of the allocator (2.30) remains the same. Only the nullspace-spanning matrix changes to $\text{span}(\boldsymbol{N}) = \mathcal{N}_\text{r}(\boldsymbol{P}^*)$ with $\boldsymbol{P}^*$ according to (1.3) and the gain matrix is $\boldsymbol{K} = \rho\overline{\boldsymbol{K}}$ with $\rho > 0$ and $\overline{\boldsymbol{K}} \succ 0$. Provided that the closed loop system without the allocator is internally stable it is shown in [2] that there is a small enough ρ such that the system with DYA (Figure 2.3) is internally stable. Furthermore, the steady-state plant outputs with and without the DYA converge to the same values. In order to include actuator constraints in the dynamic allocation process $\boldsymbol{W}$ in (2.30) can be replaced by a vector function $\boldsymbol{W}(\boldsymbol{u})$ which penalizes the usage of actuators that operate close to their limits [2].

2.2. Linear Quadratic Regulator

LQR is a special form of optimal control that is restricted to linear plant models and yields linear state controllers. As opposed to conventional state controller design where the closed-loop eigenvalues are specified by the control designer, in case of LQR they are defined by the solution of an optimization problem. The origin of this method lies in the following general optimal control problem: Consider a nonlinear plant model

$$\dot{\boldsymbol{x}} = \boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t)] \tag{2.34}$$

with state $\boldsymbol{x}(t) \in \mathbb{R}^n$, input $\boldsymbol{u}(t) \in \mathbb{R}^m$, and known initial state $\boldsymbol{x_0} = \boldsymbol{x}(0)$. Now, compute an input signal $\boldsymbol{u}(t)$ that drives the state to the origin by solving the optimization problem

$$\min_{\boldsymbol{u}} \int_0^T L\left[\boldsymbol{x}(t), \boldsymbol{u}(t), t\right] dt + S\left[\boldsymbol{x}(T)\right] \quad \text{satisfying (2.34).} \tag{2.35}$$

The cost function in (2.35) rates the system's state trajectory as well as the control signal in the range $[0, T]$ via the integral part L and the final state by means of S. Next, the Hamiltonian is defined as

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{2.36}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ is the vector of Lagrange multipliers or costate vector. Assuming that $\boldsymbol{u}$ is not explicitly bounded[1] the following necessary conditions for the optimal state $\boldsymbol{x}^*$ and control $\boldsymbol{u}^*$ vectors can be derived from the Euler-Lagrange equations (p. 101 in [36]):

$$\dot{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{\lambda}} \quad \text{with } \boldsymbol{x}(0) = \boldsymbol{x_0} \tag{2.37a}$$

$$-\dot{\boldsymbol{\lambda}} = \frac{\partial H}{\partial \boldsymbol{x}} \tag{2.37b}$$

$$\boldsymbol{0} = \frac{\partial H}{\partial \boldsymbol{u}} \tag{2.37c}$$

$$\boldsymbol{\lambda}(T) = \left.\frac{\partial S}{\partial \boldsymbol{x}}\right|_{t=T} \tag{2.37d}$$

LQR can now be deduced by focusing on linear plant models $\dot{\boldsymbol{x}} = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B_u}(t)\boldsymbol{u}(t)$ instead of the general form (2.34). For the purpose of obtaining a linear controller a quadratic cost function [18]

$$J = \frac{1}{2}\int_0^T \left[\boldsymbol{x}(t)^T \boldsymbol{Q}(t)\boldsymbol{x}(t) + \boldsymbol{u}(t)^T \boldsymbol{R}(t)\boldsymbol{u}(t)\right] dt + \frac{1}{2}\boldsymbol{x}(T)^T \boldsymbol{S}\boldsymbol{x}(T) \tag{2.38}$$

with $\boldsymbol{Q}(t) \succeq 0$, $\boldsymbol{R}(t) \succ 0$, and $\boldsymbol{S} \succeq 0$ is formulated. Evaluating (2.36) yields $H = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u} + \boldsymbol{\lambda}^T\left(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B_u}\boldsymbol{u}\right)$ which is used together with (2.37a) - (2.37c) to get

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(t)\boldsymbol{x}(t) - \boldsymbol{B_u}(t)\boldsymbol{R}(t)^{-1}\boldsymbol{B_u}(t)^T\boldsymbol{\lambda}(t) \tag{2.39a}$$

$$\dot{\boldsymbol{\lambda}} = -\boldsymbol{Q}(t)\boldsymbol{x}(t) - \boldsymbol{A}(t)^T\boldsymbol{\lambda}(t). \tag{2.39b}$$

From (2.39) it can be shown that there is a linear relationship between the state and costate vectors [18]

$$\boldsymbol{\lambda}(t) = \boldsymbol{P}(t)\boldsymbol{x}(t) \tag{2.40}$$

which leads to the linear control law

$$\boldsymbol{u}(t) = -\boldsymbol{R}(t)^{-1}\boldsymbol{B_u}(t)^T\boldsymbol{P}(t)\boldsymbol{x}(t). \tag{2.41}$$

Taking the time-derivative of (2.40) and considering (2.39) results in the matrix Riccati differential equation [18]

$$\dot{\boldsymbol{P}} = -\boldsymbol{A}(t)^T\boldsymbol{P}(t) - \boldsymbol{P}(t)\boldsymbol{A}(t) - \boldsymbol{Q}(t) + \boldsymbol{P}(t)\boldsymbol{B_u}(t)\boldsymbol{R}(t)^{-1}\boldsymbol{B_u}(t)^T\boldsymbol{P}(t). \tag{2.42}$$

Using (2.40) and (2.37d) leads to $\boldsymbol{\lambda}(T) = \boldsymbol{S}\boldsymbol{x}(T)$ and finally

$$\boldsymbol{P}(T) = \boldsymbol{S}. \tag{2.43}$$

[1]Implicitly, it can be bounded by the choice of the cost function.

2.2.1. Time-invariant results

In many cases the plant is adequately described by a linear time-invariant model $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B_u}\boldsymbol{u}(t)$. If additionally $T \to \infty$ and the time-dependence of the weighting matrices in (2.38) is omitted then a constant control law [18]

$$\boldsymbol{u}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B_u}^T\bar{\boldsymbol{P}}\boldsymbol{x}(t). \tag{2.44}$$

originates from the solution of the optimization problem. Furthermore, $\boldsymbol{S} = \boldsymbol{0}$ applies in this case. Matrix $\bar{\boldsymbol{P}}$ satisfies the algebraic Riccati equation [18]

$$\boldsymbol{0} = -\boldsymbol{A}^T\bar{\boldsymbol{P}} - \bar{\boldsymbol{P}}\boldsymbol{A} - \boldsymbol{Q} + \bar{\boldsymbol{P}}\boldsymbol{B_u}\boldsymbol{R}^{-1}\boldsymbol{B_u}^T\bar{\boldsymbol{P}}. \tag{2.45}$$

Remark 2.2.1. By means of extending $T \to \infty$ it follows that the infinite-time optimization problem

$$J = \frac{1}{2}\int_0^\infty \left[\boldsymbol{x}(t)^T\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{u}(t)^T\boldsymbol{R}\boldsymbol{u}(t)\right] dt \tag{2.46}$$

results in a finite J only if $\boldsymbol{x}(t) \to \boldsymbol{0}$ which requires $[\boldsymbol{A}, \boldsymbol{B_u}]$ to be *stabilizable*. On the other hand (2.46) were also finite if it would not include the unstable elements of $\boldsymbol{x}$. However, the resulting controller were impractical because it would not stabilize the plant. Hence, another reasonable requirement is to choose the state weighting matrix as

$$\boldsymbol{Q} = \boldsymbol{E}^T\boldsymbol{E} \tag{2.47}$$

with $\boldsymbol{E} \in \mathbb{R}^{n\times n}$ and $[\boldsymbol{A}, \boldsymbol{E}]$ being detectable. Note that since $\boldsymbol{Q}$ is positive semidefinite and symmetric a factorization (2.47) is always possible [24].

2.2.2. Extension of the time-invariant controller for tracking

So far, all the LQR controller can do is forcing the system state to the origin. Though, in many practical applications the output values of a plant shall follow a reference signal $\boldsymbol{r}$. This can be realized by a simple extension of the controller involving a prefilter. Consider a linear time-invariant plant model

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B_u}\boldsymbol{u} \tag{2.48a}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} \tag{2.48b}$$

with $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^m$, and $\boldsymbol{y}, \boldsymbol{r} \in \mathbb{R}^p$. The control law is chosen as

$$\boldsymbol{u}(t) = -\underbrace{\boldsymbol{R}^{-1}\boldsymbol{B_u}^T\bar{\boldsymbol{P}}}_{\boldsymbol{K}}\boldsymbol{x}(t) + \boldsymbol{V}\boldsymbol{r}(t) \tag{2.49}$$

with the first part corresponding with the LQR result (2.44) for stabilization of the origin and the prefilter $\boldsymbol{V} \in \mathbb{R}^{m\times p}$ multiplied by the reference signal. The prefilter is designed such that a vanishing steady-state error for constant references $\boldsymbol{r}_\infty$ is achieved, i.e.

$$\lim_{t\to\infty} \boldsymbol{y}(t) \overset{!}{=} \boldsymbol{r}_\infty. \tag{2.50}$$

It follows from (2.50) that $\dot{\boldsymbol{x}} = \boldsymbol{0}$ when $t \to \infty$ which leads to

$$\boldsymbol{0} = (\boldsymbol{A} - \boldsymbol{B_u}\boldsymbol{K})\boldsymbol{x}_\infty + \boldsymbol{B_u}\boldsymbol{V}\boldsymbol{r}_\infty \quad \Rightarrow \quad \boldsymbol{y}_\infty = \underbrace{-\boldsymbol{C}(\boldsymbol{A} - \boldsymbol{B_u}\boldsymbol{K})^{-1}\boldsymbol{B_u}}_{\boldsymbol{L}\in\mathbb{R}^{p\times m}}\boldsymbol{V}\boldsymbol{r}_\infty \overset{!}{=} \boldsymbol{r}_\infty.$$

Assuming that $p \leq m$ and $\operatorname{rank}(\boldsymbol{L}) = p$ the prefilter is computed from $\boldsymbol{L}\boldsymbol{V} = \boldsymbol{I_p}$ as

$$\boldsymbol{V} = \boldsymbol{L}^T\left(\boldsymbol{L}\boldsymbol{L}^T\right)^{-1}. \tag{2.51}$$

2.3. Model Predictive Control

MPC is an optimization-based control technique which is frequently used in academics as well as in various industrial environments. Especially within the chemical-, oil- and process-industry it is recognized as an established control method. This is motivated due to the following characteristics [37]:

1. MPC has the inherent ability to handle multiple-input and multiple-output control problems.
2. It takes constraints regarding actuators, states, and outputs into consideration.
3. The basic underlying principle is easy to understand even for people not familiar with control engineering.
4. The dynamics of plants in process industry is usually rather slow compared to motion control or electrical control problems. Thus, more time can be spent on online-optimization.

However, the ongoing advances in computer hardware also facilitate the application of MPC in more dynamic areas like automotive control (e.g. [38]). The *receding horizon* idea is the basis of MPC and Figure 2.5 demonstrates its principles for a single-input single-output (SISO) system. At the current time index k MPC uses a mathematical model of the plant for predicting its output over the prediction horizon n_p as a function of the chosen input sequence and, depending on the model representation, the internal states of the plant or the past input values. The input sequence $\{u_k, u_{k+1}, \ldots, u_{k+n_c-1}\}$ with n_c denoting the control horizon resulting in the best predicted behavior with respect to an objective function and satisfying an optional set of constraints is computed by solving an optimization problem. The objective function involves at least the error between reference and predicted plant output but can contain several other influencing factors like for example energy consumption. In case of $n_c < n_p$ the input is assumed constant after the end of the control horizon. Although computing n_c future input values, MPC only applies the first element as the plant's actual input signal. In the subsequent sampling instant $k+1$ the whole procedure is repeated to get the next input [37]. In principle this idea is applicable to any kind of plant model (2.34) with general objective functions (2.35) and constraints. Unfortunately this general formulation leads to nonlinear and non-convex optimization problems. But since the solution must be found online the majority of MPC deals with special classes of plant models, objective functions and constraints which facilitate an efficient implementation.

2.3.1. Plant model

Most MPC formulations are based on linear, discrete-time, state-space representations of the plant

$$\boldsymbol{x_{k+1}} = \boldsymbol{A}\boldsymbol{x_k} + \boldsymbol{B}\boldsymbol{u_k} + \boldsymbol{B_d}\boldsymbol{d_k} \tag{2.52a}$$

$$\boldsymbol{y_k} = \boldsymbol{C}\boldsymbol{x_k} + \boldsymbol{D}\boldsymbol{u_k} + \boldsymbol{D_d}\boldsymbol{d_k} \tag{2.52b}$$

with state vector $\boldsymbol{x_k}$, input vector $\boldsymbol{u_k}$, measured disturbances $\boldsymbol{d_k}$, and controlled outputs $\boldsymbol{y_k}$. Measured disturbances can be interpreted as known but uncontrollable inputs. Their consideration enables a feedforward-like behavior since the controller does not have to wait for changes in $\boldsymbol{y_k}$ before it can react [37].

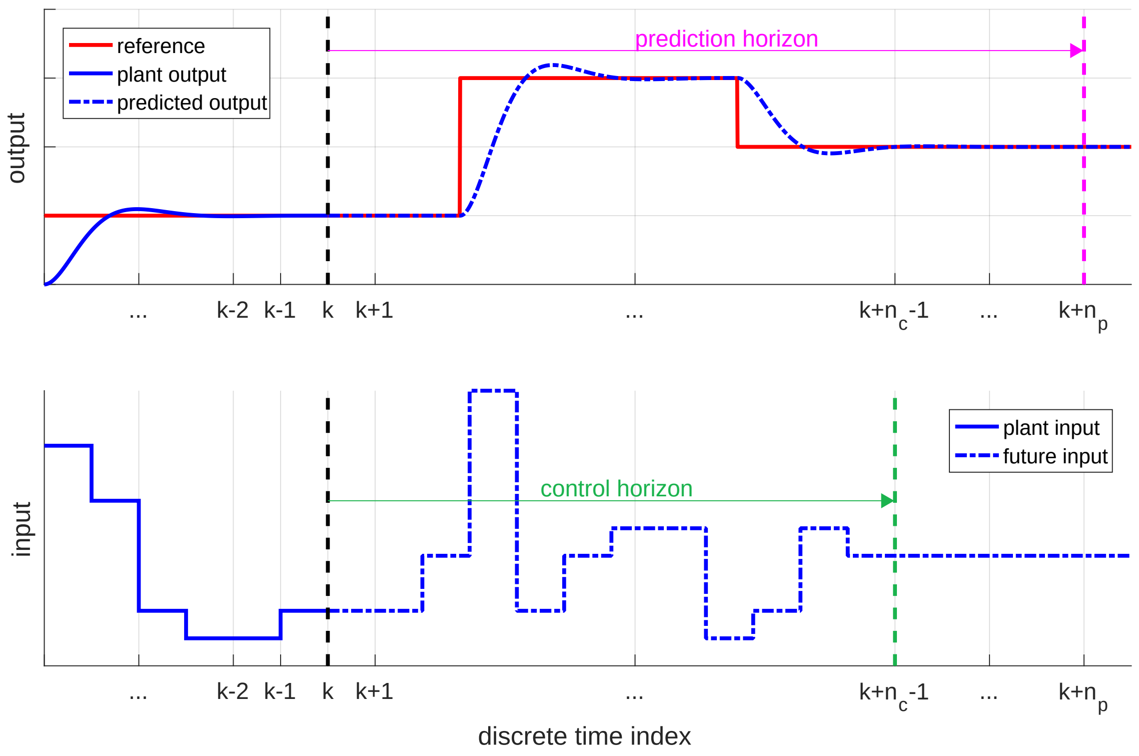

Figure 2.5.: The receding horizon principle for the SISO case: At discrete-time instant k MPC computes an input sequence which is optimal in terms of some cost function that includes the error between reference and predicted output. Only the first element u_k is used, then the process is repeated.

2.3.2. Objective function

One of the most frequently used objective function types for MPC is a sum of quadratic terms [37]. It is very common in MPC literature to choose the input changes $\boldsymbol{u_{\Delta,k}} = \boldsymbol{u_k} - \boldsymbol{u_{k-1}}$ as optimization variables instead of the total input $\boldsymbol{u_k}$. A possible formulation penalizing deviations of the predicted output from the reference and high rate of change of the input signals reads as

$$J(k) = \sum_{i=1}^{n_p} \left(\boldsymbol{y_{k+i}} - \boldsymbol{r_{k+i}}\right)^T \boldsymbol{Q_i} \left(\boldsymbol{y_{k+i}} - \boldsymbol{r_{k+i}}\right) + \sum_{i=1}^{n_c} \boldsymbol{u}^T_{\boldsymbol{\Delta,k+i-1}} \boldsymbol{R_{\Delta,i}} \boldsymbol{u_{\Delta,k+i-1}} \tag{2.53}$$

with $\boldsymbol{Q_i} \succ 0$ and $\boldsymbol{R_{\Delta,i}} \succeq 0$ being the weighting matrices. In most cases only their main diagonals are chosen different from zero.

2.3.3. Constraints

The favorable representations of constraints on inputs, outputs, and states of (2.52) are linear inequalities. They all have to be translated into expressions which only depend on the optimization variables, i.e. the plant inputs, before solving.

2.3.4. Overall optimization problem

If all the assumptions of the previous Sections 2.3.1 - 2.3.3 are satisfied the resulting optimization problem is a QP.

Part II.

Theory and Methodology

3. Input Matrix Factorization

In order to apply control allocation techniques one must transform the plant model into a form that allows the separation of the regulation from the control distribution task ([1], [25], [39]). For a strongly input-redundant plant model that is linear with respect to its inputs, an input matrix factorization has to be conducted, which is not unique for each plant [30]. The main results of this chapter have been published by the author in [25] and [30]. The starting point of considerations is the linear plant model

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B_u}\boldsymbol{u}, \tag{3.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the system matrix, $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, $\boldsymbol{B_u} \in \mathbb{R}^{n\times m}$ is the input matrix, and $\boldsymbol{u} \in \mathbb{R}^m$ is the input vector. Let the rank of the input matrix be [25]

$$k = \operatorname{rank}(\boldsymbol{B_u}) < m, \tag{3.2}$$

which means that $\boldsymbol{B_u}$ does not have full column rank or in other words the plant is equipped with a redundant set of actuators. $\mathcal{N}_\mathrm{r}(\boldsymbol{B_u})$ is the right nullspace of $\boldsymbol{B_u}$ and $\dim\left[\mathcal{N}_\mathrm{r}(\boldsymbol{B_u})\right] = m-k$ follows from the rank-nullity theorem [24]. Hence certain directions in control space $\mathbb{R}^m$ are mapped to zero which means that infinitely many input vectors have the same effect on the plant. Condition (3.2) implies that the input matrix can be factorized into a virtual input matrix $\boldsymbol{B_v} \in \mathbb{R}^{n\times k}$ and a control effectivity matrix $\boldsymbol{B} \in \mathbb{R}^{k\times m}$, i.e. [25]

$$\boldsymbol{B_u} = \boldsymbol{B_v}\boldsymbol{B} \tag{3.3}$$

with their ranks satisfying

$$\operatorname{rank}(\boldsymbol{B_v}) = \operatorname{rank}(\boldsymbol{B}) = k. \tag{3.4}$$

The factorization (3.3) of $\boldsymbol{B_u}$ leads to an alternative plant model

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B_v}\boldsymbol{v} \tag{3.5a}$$

$$\boldsymbol{v} = \boldsymbol{B}\boldsymbol{u} \tag{3.5b}$$

including the virtual control vector $\boldsymbol{v} \in \mathbb{R}^k$ which has, according to (3.2), less elements than $\boldsymbol{u}$. A controller for redundancy-free (3.5a) provides the output $\boldsymbol{v}$, which is distributed among the real actuators $\boldsymbol{u}$ by CA [39]. Usually, the actuators $\boldsymbol{u}^T = [u_1 \dots u_m]$ are subject to constraints, which delimit the admissible subset of the m-dimensional (m-D) control space Ω according to (1.31). In the same way a subset of k-D virtual control space Φ is defined as (1.34) and called AMS [25].

The factorization according to (3.3) is not unique, i.e. the number of possibilities to form $\boldsymbol{B_v}$ and $\boldsymbol{B}$ is infinite. In the case of $n = k$ a natural choice is $\boldsymbol{B_v} = \boldsymbol{I_k}$ and $\boldsymbol{B} = \boldsymbol{B_u}$ with $\boldsymbol{I_k} \in \mathbb{R}^{k\times k}$ being the identity matrix [25]. In case of $n > k$ it is not always so obvious how to accomplish (3.3). Singular value decomposition (SVD) and QR factorization, to name but a few, are possible options [30]. For instance the SVD of $\boldsymbol{B_u}$ yields

$$\boldsymbol{B_u} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \tag{3.6}$$

with orthogonal $\boldsymbol{U} \in \mathbb{R}^{n\times n}$, $\boldsymbol{S} \in \mathbb{R}^{n\times m}$, and orthogonal $\boldsymbol{V} \in \mathbb{R}^{m\times m}$ [28]. Matrix $\boldsymbol{S}$ is all zero except for the first k entries of its main diagonal, which are the nonzero singular values of $\boldsymbol{B_u}$. One possible factorization following from (3.6) is

$$\boldsymbol{B_u} = \underbrace{\boldsymbol{U_k S_k}}_{\boldsymbol{B_v}} \underbrace{\boldsymbol{V_k^T}}_{\boldsymbol{B}} \tag{3.7}$$

where $\boldsymbol{U_k} \in \mathbb{R}^{n\times k}$ consists of the first k columns of $\boldsymbol{U}$, $\boldsymbol{S_k} \in \mathbb{R}^{k\times k}$ is the diagonal matrix of all nonzero singular values, and $\boldsymbol{V_k} \in \mathbb{R}^{m\times k}$ is composed of the first k columns of $\boldsymbol{V}$.

3.1. Relationship between factorizations

Consider two arbitrary input matrix factorizations

$$\boldsymbol{B_u} = \boldsymbol{B_{v1} B_1} = \boldsymbol{B_{v2} B_2} \tag{3.8}$$

whereby according to (3.4) the rank of all matrices is k.

Theorem 3.1.1. *For any pair of input matrix factorizations (3.8) an invertible transformation matrix*

$$\boldsymbol{T} = \left(\boldsymbol{B_{v1}^T B_{v1}}\right)^{-1} \boldsymbol{B_{v1}^T B_{v2}} \tag{3.9}$$

with $\boldsymbol{T} \in \mathbb{R}^{k\times k}$ exists, such that

$$\boldsymbol{B_{v1} T} = \boldsymbol{B_{v2}} \tag{3.10}$$

$$\boldsymbol{T^{-1} B_1} = \boldsymbol{B_2} \tag{3.11}$$

hold [25].

Proof. The first step is to show that $\boldsymbol{T}$ always exists and that it is invertible. In accordance with [24] and [40] $\operatorname{rank}(\boldsymbol{B_{v1}^T B_{v1}}) = \operatorname{rank}(\boldsymbol{B_{v1}}) = k$ and so (3.9) can always be calculated. In order for $\boldsymbol{T}$ being invertible $\operatorname{rank}(\boldsymbol{B_{v1}^T B_{v2}})$ must be k. For the purpose of determining $\operatorname{rank}(\boldsymbol{B_{v1}^T B_{v2}})$ one can write

$$\begin{aligned}\mathcal{N}_\mathrm{r}(\boldsymbol{B_{v1}^T B_{v2}}) = \{\boldsymbol{z} \in \mathbb{R}^k | [\boldsymbol{z} \in \mathcal{N}_\mathrm{r}(\boldsymbol{B_{v2}})] \vee \\ [\boldsymbol{B_{v2} z} \in \mathcal{N}_\mathrm{r}(\boldsymbol{B_{v1}^T})]\}.\end{aligned} \tag{3.12}$$

The rank-nullity theorem [24] yields

$$k = \dim[\mathcal{N}_\mathrm{r}(\boldsymbol{B_{vi}})] + \underbrace{\dim[\operatorname{im}(\boldsymbol{B_{vi}})]}_{=\operatorname{rank}(\boldsymbol{B_{vi}})=k} \quad \forall i \in \{1,2\}, \tag{3.13}$$

which means that

$$\mathcal{N}_\mathrm{r}(\boldsymbol{B_{vi}}) = \boldsymbol{0} \in \mathbb{R}^k \quad \forall i \in \{1,2\}. \tag{3.14}$$

In order to get $\mathcal{N}_\mathrm{r}(\boldsymbol{B_{v1}^T})$ evaluating the rank-nullity theorem once again one obtains

$$k = \dim[\mathcal{N}_\mathrm{r}(\boldsymbol{B_i^T})] + \underbrace{\operatorname{rank}(\boldsymbol{B_i^T})}_{=k} \quad \forall i \in \{1,2\}, \tag{3.15}$$

and this means that $\mathcal{N}_\mathrm{r}(\boldsymbol{B_i}^T) = \mathbf{0} \in \mathbb{R}^k$. Given that the nullspace of the transposed input matrix simplifies to

$$\mathcal{N}_\mathrm{r}(\boldsymbol{B_u}^T) = \mathcal{N}_\mathrm{r}(\boldsymbol{B_i}^T \boldsymbol{B_{vi}}^T) = \mathcal{N}_\mathrm{r}(\boldsymbol{B_{vi}}^T) \quad \forall i \in \{1,2\}, \tag{3.16}$$

which implies that

$$\mathcal{N}_\mathrm{r}(\boldsymbol{B_{v1}}^T) = \mathcal{N}_\mathrm{r}(\boldsymbol{B_{v2}}^T). \tag{3.17}$$

Combining (3.12) and (3.17) results in $\mathcal{N}_\mathrm{r}(\boldsymbol{B_{v1}}^T \boldsymbol{B_{v2}}) = \mathcal{N}_\mathrm{r}(\boldsymbol{B_{v2}}^T \boldsymbol{B_{v2}})$. According to [24] and [40] $\forall \boldsymbol{A} \in \mathbb{R}^{m\times n} : \mathcal{N}_\mathrm{r}(\boldsymbol{A}^T\boldsymbol{A}) = \mathcal{N}_\mathrm{r}(\boldsymbol{A})$. Considering that and (3.14) yields $\mathcal{N}_\mathrm{r}(\boldsymbol{B_{v2}}^T \boldsymbol{B_{v2}}) = \mathcal{N}_\mathrm{r}(\boldsymbol{B_{v2}}) = \mathbf{0}$ which means that $\mathrm{rank}(\boldsymbol{B_{v1}}^T \boldsymbol{B_{v2}}) = k$. In order to prove (3.11) one can rewrite (3.9) to get $\left(\boldsymbol{B_{v1}}^T \boldsymbol{B_{v1}}\right)\boldsymbol{T} = \boldsymbol{B_{v1}}^T \boldsymbol{B_{v2}}$. Multiplying with $\boldsymbol{B_2}$ from the right-hand side and considering (3.8) results in

$$\boldsymbol{B_{v1}}^T \boldsymbol{B_{v1}} \boldsymbol{T} \boldsymbol{B_2} = \boldsymbol{B_{v1}}^T \underbrace{\boldsymbol{B_{v2}} \boldsymbol{B_2}}_{\boldsymbol{B_{v1}}\boldsymbol{B_1}}. \tag{3.18}$$

The square matrix $\boldsymbol{B_{v1}}^T \boldsymbol{B_{v1}}$ has full rank and so (3.11) follows. Inserting (3.11) into (3.8) yields $\left(\boldsymbol{B_{v1}} - \boldsymbol{B_{v2}}\boldsymbol{T}^{-1}\right)\boldsymbol{B_1} = \mathbf{0}$. Taking into account (3.4) and the dimensions of $\boldsymbol{B_1}$ its left nullspace reads as $\mathcal{N}_\mathrm{l}(\boldsymbol{B_1}) = \mathbf{0} \in \mathbb{R}^k$ which implies (3.10) [25]. ■

Remark 3.1.1. In case of $n = k$ the transformation matrix computation simplifies to [25]

$$\boldsymbol{T} = \boldsymbol{B_{v1}}^{-1} \boldsymbol{B_{v2}} \tag{3.19}$$

and the proof of Theorem 3.1.1 becomes trivial.

3.2. Influence on virtual control

The main goal of the controller is to make the state of system (3.1) track a desired trajectory regardless of the chosen factorization method. The state vector $\boldsymbol{x_i}(t)$ for factorization $\boldsymbol{B_u} = \boldsymbol{B_{vi}}\boldsymbol{B_i}$ and initial state $\boldsymbol{x}_0 := \boldsymbol{x}(t=0)$ reads as [25]

$$\boldsymbol{x_i}(t) = \boldsymbol{\Phi_A}(t)\boldsymbol{x}_0 + \int_0^t \left[\boldsymbol{\Phi_A}(t-\tau)\boldsymbol{B_{vi}}\boldsymbol{v_i}(\tau)\right] d\tau \tag{3.20}$$

with $\boldsymbol{\Phi_A}(t)$ being the state-transition matrix. Using (3.10) and (3.20) it follows that the state vectors for different input matrix factorizations are identical, i.e. $\boldsymbol{x_1}(t) \equiv \boldsymbol{x_2}(t)$ if [25]

$$\boldsymbol{v_2}(t) = \boldsymbol{T}^{-1}\boldsymbol{v_1}(t) \quad \forall t. \tag{3.21}$$

This relationship is utilized in Sections 3.4 - 3.6 to analyze the factorization's impact on generalized inverses.

3.3. Influence on admissible subsets

Each input matrix factorization has its own attainable moment set

$$\Phi_i = \left\{\boldsymbol{v_i} \in \mathbb{R}^k \middle| \exists \boldsymbol{u} \in \Omega : \boldsymbol{v_i} = \boldsymbol{B_i}\boldsymbol{u}\right\} \; \forall i \in \{1,2\}. \tag{3.22}$$

In contrast the admissible control space Ω stays the same [25].

3.4. Influence on direct allocation

Theorem 3.4.1. *For any pair of input matrix factorizations (3.8) direct allocation yields the same input vectors* $\boldsymbol{u_1} \equiv \boldsymbol{u_2}$ *if the virtual control vectors* $\boldsymbol{v_1}$ *and* $\boldsymbol{v_2}$ *fulfill (3.21) [30].*

Proof. Each factorization has its own boundary of the related attainable moment set

$$\partial(\Phi_i) = \{\boldsymbol{v_i} \in \Phi_i | (b\boldsymbol{v_i} \notin \Phi_i) \,\forall\, (b > 1)\} \;\; \forall i \in \{1, 2\} \tag{3.23}$$

with $b \in \mathbb{R}$. For one factorization the boundary-intersections $\boldsymbol{v_1^*}$ and $\boldsymbol{u_1^*}$ fulfill

$$\boldsymbol{v_1} = a_1\boldsymbol{v_1^*} = a_1\boldsymbol{B_1}\boldsymbol{u_1^*}. \tag{3.24}$$

Using (3.21) the virtual control vector $\boldsymbol{v_2}$ can be expressed as

$$\boldsymbol{v_2} = \boldsymbol{T}^{-1}\boldsymbol{v_1} = a_1\boldsymbol{T}^{-1}\boldsymbol{v_1^*} = a_1 \underbrace{\boldsymbol{T}^{-1}\boldsymbol{B_1}}_{\boldsymbol{B_2}} \boldsymbol{u_1^*} \tag{3.25}$$

in case of another factorization of the input matrix. Due to the dimension reduction which takes place by the mapping from Ω to Φ_2 not all elements on $\partial(\Omega)$ lie on $\partial(\Phi_2)$ but rather in the interior of Φ_2. However, only if $\boldsymbol{B_2}\boldsymbol{u_1^*} = \boldsymbol{T}^{-1}\boldsymbol{v_1^*} \in \partial(\Phi_2)$ DA will compute the same real control vector for both factorizations. Let $\boldsymbol{v_1^*}$ be located on $\partial(\Phi_1)$, i.e. [30]

$$\boldsymbol{v_1^*} \in \partial(\Phi_1) \quad \Rightarrow \quad \forall\, b > 1 \,:\, b\boldsymbol{v_1^*} \notin \Phi_1. \tag{3.26}$$

Assuming that $\boldsymbol{T}^{-1}\boldsymbol{v_1^*}$ is in the interior of Φ_2 means that $\boldsymbol{T}^{-1}\boldsymbol{v_1^*} \in \Phi_2$ and $\boldsymbol{T}^{-1}\boldsymbol{v_1^*} \notin \partial(\Phi_2)$, which is equivalent to

$$\exists\, b > 1 \,:\, b\boldsymbol{T}^{-1}\boldsymbol{v_1^*} \in \Phi_2. \tag{3.27}$$

Utilizing (3.22) to rewrite (3.27) yields

$$\exists\, b > 1, \exists(b\boldsymbol{u}) \in \Omega \,:\, b\boldsymbol{T}^{-1}\boldsymbol{v_1^*} = b\boldsymbol{B_2}\boldsymbol{u}. \tag{3.28}$$

Now one can see from (3.11) that (3.28) is equivalent to $\exists\, b > 1, \exists(b\boldsymbol{u}) \in \Omega \,:\, b\boldsymbol{v_1^*} = b\boldsymbol{B_1}\boldsymbol{u}$, which means that $\boldsymbol{v_1^*}$ lies in the interior of Φ_1, i.e. $\exists\, b > 1 \,:\, b\boldsymbol{v_1^*} \in \Phi_1$ and this is contradictory to (3.26). Therefore $\boldsymbol{T}^{-1}\boldsymbol{v_1^*}$ is indeed on $\partial(\Phi_2)$ and so it follows from (2.22) that DA will yield the same $\boldsymbol{u}$ for all factorizations [30]. ■

3.5. Influence on generalized inverses

Theorem 3.5.1. *Given two arbitrary input matrix factorizations (3.8), consider a generalized inverse* $\boldsymbol{P_1}$ *for factorization 1 and the transformation matrix* $\boldsymbol{T}$ *according to (3.9). Then a generalized inverse for factorization 2 is given by [25]*

$$\boldsymbol{P_2} = \boldsymbol{P_1}\boldsymbol{T}. \tag{3.29}$$

Proof. Equation (3.29) can be proved by inserting it together with (3.11) into the Penrose-equations (2.5) [25]. ■

Corollary 3.5.1. *Consider two arbitrary input matrix factorizations (3.8) with related generalized inverses* $\boldsymbol{P_1}$ *and* $\boldsymbol{P_2}$ *satisfying (3.29) and virtual control vectors* $\boldsymbol{v_1}$ *and* $\boldsymbol{v_2}$ *fulfilling (3.21). Then (2.6) yields identical input vectors* $\boldsymbol{u_1} \equiv \boldsymbol{u_2}$ *for both factorizations [25].*

Proof. Using (2.6), (3.21), and (3.29) yields $\boldsymbol{u_1} = \boldsymbol{P_1}\boldsymbol{v_1} = \boldsymbol{P_1}\boldsymbol{T}\boldsymbol{T}^{-1}\boldsymbol{v_1} = \boldsymbol{P_2}\boldsymbol{v_2} = \boldsymbol{u_2}$. ■

Hence, CA via generalized inverses remains unaffected by input matrix factorization [25].

3.5.1. Geometrical considerations

With $i \in 1, 2$ representing the factorization number the columns of a generalized inverse $\boldsymbol{P_i}$ span a k-D subspace $\mathcal{R}(\boldsymbol{P_i}) = \left\{\boldsymbol{u} \in \mathbb{R}^m \,|\, (\boldsymbol{u} = \boldsymbol{P_i v}) \,\forall\, (\boldsymbol{v} \in \mathbb{R}^k)\right\} \subset \mathbb{R}^m$ which is called column space or range (see [40] and [27]). Two generalized inverses fulfilling (3.29) have identical ranges, i.e. [25]

$$\mathcal{R}(\boldsymbol{P_2}) = \mathcal{R}(\boldsymbol{P_1 T}) = \mathcal{R}(\boldsymbol{P_1}) \tag{3.30}$$

because $\boldsymbol{T}$ is guaranteed to have full row rank [40]. The intersection of this subspace with the convex set Ω is also convex ([4]) and reads as [25]

$$\Theta(\boldsymbol{P_i}) = \mathcal{R}(\boldsymbol{P_i}) \cap \Omega. \tag{3.31}$$

It forms the feasible set of points that can be reached by means of the pseudoinverse and is invariant to input matrix factorizations because of (3.30). $\Theta(\boldsymbol{P_i})$ can be projected into virtual control space by means of $\boldsymbol{B_i}$, where it denotes the subset Π_i of virtual controls, that gets mapped to feasible real controls for the i-th factorization [25]

$$\Pi_i = \boldsymbol{B_i}\Theta(\boldsymbol{P_i}) \subseteq \Phi_i. \tag{3.32}$$

Fig. 3.1 shows $\Theta(\boldsymbol{P_i})$ and Ω in a 3-D example and Fig. 3.2 depicts its virtual control space for an arbitrary factorization. In order to map more virtual control vectors into Ω the sets Π_i and

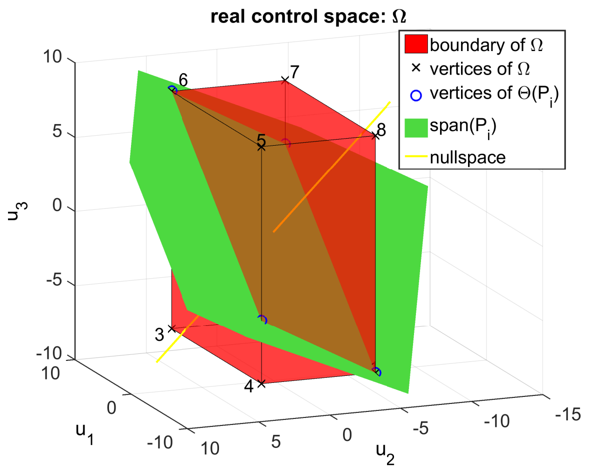

Figure 3.1.: Visualization of a pseudoinverse for $m = 3$ and $k = 2$: The red box depicts Ω. The green plane is $\mathcal{R}(\boldsymbol{P_i})$. The intersection with the red box is the subset $\Theta(\boldsymbol{P_i})$. The yellow line shows the nullspace. Moving along its direction does not change the resulting virtual control. (© 2016 IEEE, [25]).

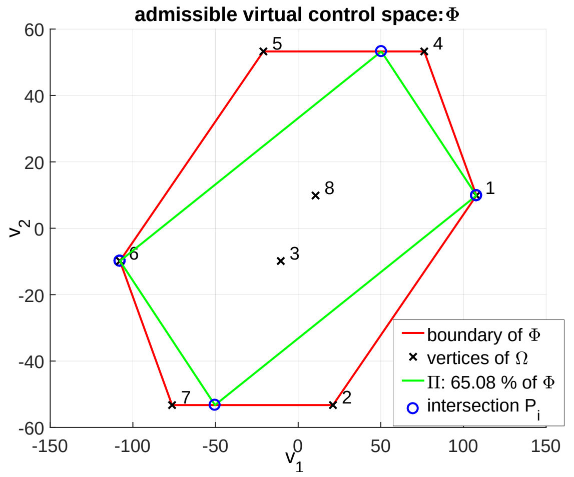

Figure 3.2.: 2-D virtual control space related to the example in Figure 3.1: The red polygon is the projection of Ω into 2-D space and thus it spans the admissible virtual control space Φ_i for a certain factorization. The green polygon is Π_i, which is the projection of $\Theta(\boldsymbol{P_i})$. (© 2016 IEEE, [25]).

Φ_i should coincide as much as possible, i.e. their volumes should be the same. One possibility to determine the volume of a k-D polytope $\boldsymbol{P}$ involves the triangulation of its boundary $\partial(\boldsymbol{P})$ into a set Δ_P of (k-1)-D simplices (special case of polytopes, see [41]). Each (k-1)-simplex forms together with the origin a k-simplex. The volume of $\boldsymbol{P}$ is calculated as the sum of volumes of all k-simplices [25]

$$V(\boldsymbol{P}) = \frac{1}{k!} \sum_{\sigma \in \Delta_P} \left| \det \left[\boldsymbol{p_1}(\sigma) \quad \dots \quad \boldsymbol{p_k}(\sigma) \right] \right| \tag{3.33}$$

with $\boldsymbol{p_j}(\sigma)$ being the j-th vertex of simplex σ [42]. Using (1.34) and (3.11) the vertices of Φ_2 can be written as

$$\boldsymbol{p}(\Phi_2) = \boldsymbol{T}^{-1}\boldsymbol{p}(\Phi_1). \tag{3.34}$$

Consequently, one can utilize (3.32) and (3.11) to express the vertices of Π_2 as

$$\boldsymbol{p}(\Pi_2) = \boldsymbol{T}^{-1}\boldsymbol{p}(\Pi_1). \tag{3.35}$$

Combining (3.33) and (3.34) the volume of Φ_2 reads as

$$\begin{aligned} V(\Phi_2) &= \frac{\left|\det(\boldsymbol{T}^{-1})\right|}{k!} \sum_{\sigma \in \Delta_{\Phi_1}} \det \left| \left[\boldsymbol{p_1}(\sigma) \quad \dots \quad \boldsymbol{p_k}(\sigma) \right] \right| \\ &= \left|\det(\boldsymbol{T}^{-1})\right| V(\Phi_1) \end{aligned} \tag{3.36}$$

and together with (3.35) one obtains

$$\begin{aligned} V(\Pi_2) &= \frac{\left|\det(\boldsymbol{T}^{-1})\right|}{k!} \sum_{\sigma \in \Delta_{\Pi_1}} \det \left| \left[\boldsymbol{p_1}(\sigma) \quad \dots \quad \boldsymbol{p_k}(\sigma) \right] \right| \\ &= \left|\det(\boldsymbol{T}^{-1})\right| V(\Pi_1). \end{aligned} \tag{3.37}$$

It follows from (3.36) and (3.37) that the volume ratio is not affected by factorization, i.e. [25]

$$\frac{V(\Pi_1)}{V(\Phi_1)} = \frac{V(\Pi_2)}{V(\Phi_2)}. \tag{3.38}$$

Figure 3.3 shows the virtual control space of another input matrix factorization related to the example in Figure 3.1. One can observe that the volume ratio is the same as in case of the factorization leading to Figure 3.2.

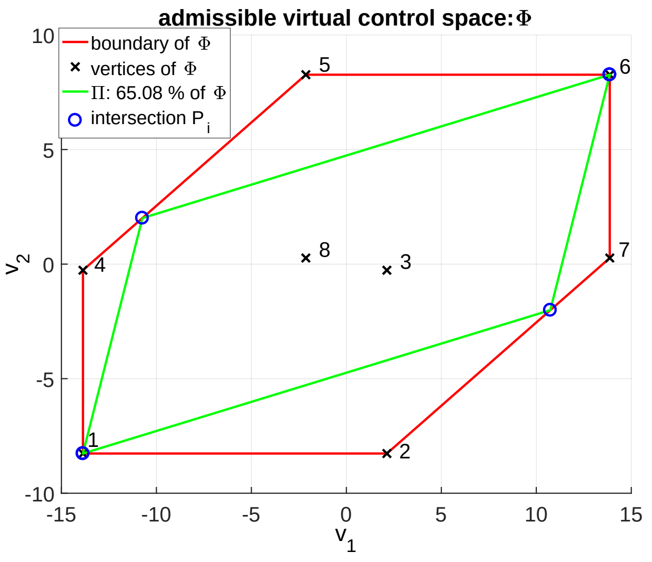

Figure 3.3: 2-D virtual control space related to the example in Figure 3.1 for a different factorization. The red polygon is Φ and the green polygon is Π (the projection of $\Theta(\boldsymbol{P_1}) = \Theta(\boldsymbol{P_2})$).

3.6. Influence on Redistributed Pseudoinverse

Whether the factorization affects RPINV is related with a possible decrease of the input matrix's rank [30]. Let j denote the number of free actuators during an RPINV iteration.

Assumption 3.6.1. Every $k \times k$ submatrix of $\boldsymbol{B_0}$ is full rank.

Assumption 3.6.2. The number of free controls satisfies $j \geq k$.

Theorem 3.6.1. *Under Assumptions 3.6.1 and 3.6.2 RPINV yields the same input vectors $\boldsymbol{u_1} \equiv \boldsymbol{u_2}$ for any pair of input matrix factorizations (3.8) if the virtual control vectors $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ fulfill (3.21) [30].*

Proof. During the first iteration (unconstrained solution) $\boldsymbol{c}^1$ is zero for both factorizations. Considering (3.11), (2.18), and (2.21) the weighted pseudoinverse for the second factorization reads as [30]

$$\begin{aligned}\boldsymbol{B_2^\#} &= \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{W}^{-1}\boldsymbol{B_1^{N,T}}\boldsymbol{T}^{-T}\left[\boldsymbol{T}^{-1}\boldsymbol{B_1^N}\boldsymbol{W}^{-1}\boldsymbol{B_1^{N,T}}\boldsymbol{T}^{-T}\right]^{-1} \\ &= \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{W}^{-1}\boldsymbol{B_1^{N,T}}\left(\boldsymbol{B_1^N}\boldsymbol{W}^{-1}\boldsymbol{B_1^{N,T}}\right)^{-1}\boldsymbol{T} = \boldsymbol{B_1^\#}\boldsymbol{T}\end{aligned} \tag{3.39}$$

and so $\boldsymbol{u_1^1} \equiv \boldsymbol{u_2^1}$ follows from (3.21). Assume that $j \geq k$ actuators remain free in iteration $N \geq 1$. Because of $\boldsymbol{u_1^{N-1}} = \boldsymbol{u_2^{N-1}}$ this leads to the same changes in $\boldsymbol{c}^N$ and $\boldsymbol{B_i^N}$ for both factorizations $i \in \{1,2\}$ and $k = \operatorname{rank}(\boldsymbol{B_i^N}) = \min(k,j)$ still holds. Hence, $\boldsymbol{B_i^N}\boldsymbol{W}^{-1}\boldsymbol{B_i^{N,T}}$ is still invertible and (3.39) can be applied again to show that $\boldsymbol{u_1^N} = \boldsymbol{u_2^N}$ [30]. ■

Remark 3.6.1. Assumption 3.6.1 guarantees that $\operatorname{rank}(\boldsymbol{B}^N) = k$ as long as it contains k nonzero columns. If this assumption is not fulfilled, zeroing columns can lead to a rank-reduction of $\boldsymbol{B}^N$ although Assumption 3.6.2 holds. Whether Theorem 3.6.1 holds depends in this case not only on the number of saturated actuators but also on which of them saturate [30].

In order to analyze the case of $j < k$ assume that $\boldsymbol{W} = \boldsymbol{I_m}$ which causes $\boldsymbol{B}^\#$ to become the MPP[1], i.e. $\boldsymbol{B}^\# = \boldsymbol{B}^\ddagger = \boldsymbol{B}^T(\boldsymbol{B}\boldsymbol{B}^T)^{-1}$. Due to numerical reasons this calculation is not carried out directly, but instead the SVD is used [28]. Consider matrix $\boldsymbol{B}$ as [30]

$$\boldsymbol{B} = \begin{bmatrix} b_{11} & \cdots & b_{1j} & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ b_{k1} & \cdots & b_{kj} & 0 & \cdots & 0 \end{bmatrix} \tag{3.40}$$

with j being the number of nonzero columns[2] and singular values. The SVD yields $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ with the orthonormal columns of $\boldsymbol{U} \in \mathbb{R}^{k\times k}$ being the eigenvectors of $\boldsymbol{B}\boldsymbol{B}^T$ [30],

$$\boldsymbol{S} = \begin{bmatrix} \sigma_1 & & & 0 & \cdots & 0 \\ & \ddots & & \vdots & & \vdots \\ & & \sigma_k & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{k\times m} \tag{3.41}$$

containing the singular values $\sigma_1, \ldots, \sigma_k$ of $\boldsymbol{B}$ (note that only the first j of them are nonzero), and the orthogonal matrix $\boldsymbol{V} \in \mathbb{R}^{m\times m}$. An important observation is that in the first j columns

[1] A generalization for arbitrary $\boldsymbol{W} \succ 0$ is presented later on. In order to keep notations concise superscript 'N' in $\boldsymbol{B}^N$ is omitted from now on.

[2] W.l.o.g. one can assume the nonzero columns to be the first ones, because columns of $\boldsymbol{B}$ and related elements in $\boldsymbol{u}$ may be appropriately interchanged.

of $\boldsymbol{V}$ the last $m-j$ rows are zero, because these columns are the eigenvectors of the nonzero eigenvalues of $\boldsymbol{B}^T\boldsymbol{B}$, whose $m-j$ last rows are zero. The MPP of $\boldsymbol{B}$ reads as [30]

$$\boldsymbol{B}^\ddagger = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T \tag{3.42}$$

with

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^{-1} & & & & \\ & \ddots & & & \\ & & \sigma_j^{-1} & & \\ & & & \ddots & \\ 0 & & \cdots & & 0 \\ \vdots & & & & \vdots \\ 0 & & \cdots & & 0 \end{bmatrix} \in \mathbb{R}^{m\times k}. \tag{3.43}$$

One can see from (3.43) that $\boldsymbol{B}^\ddagger$ is no longer a right-inverse because $\text{rank}(\boldsymbol{\Sigma}) < k$. Matrix $\boldsymbol{\Sigma}$ has $m-j$ zero rows which means together with (3.42) that the last $m-j$ columns of $\boldsymbol{V}$ do not contribute to $\boldsymbol{B}^\ddagger$. Evaluating (3.42) and considering these insights yields [30]

$$\boldsymbol{B}^\ddagger = \begin{bmatrix} \sigma_1^{-1}v_{11} & \cdots & \sigma_j^{-1}v_{1j} & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ \sigma_1^{-1}v_{j1} & \cdots & \sigma_j^{-1}v_{jj} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} \cdot \boldsymbol{U}^T \tag{3.44}$$

with v_{pq} being the element from $\boldsymbol{V}$'s p-th row and q-th column. Equation (3.44) can be rewritten to

$$\boldsymbol{B}^\ddagger = \boldsymbol{R} \begin{bmatrix} v_{11} & \cdots & v_{1j} \\ \vdots & & \vdots \\ v_{j1} & \cdots & v_{jj} \end{bmatrix} \begin{bmatrix} \sigma_1^{-1} & & & 0 & \cdots & 0 \\ & \ddots & & \vdots & & \vdots \\ & & \sigma_j^{-1} & 0 & \cdots & 0 \end{bmatrix} \boldsymbol{U}^T \tag{3.45}$$

with $\boldsymbol{R} \in \mathbb{R}^{m\times j}$ being consistent with (2.17). Expression (3.45) can also be evaluated if $\text{rank}(\boldsymbol{B}) < k$, i.e. less than k actuators are free[3]. Assuming $j < k$ and taking all nonzero columns from (3.40), one obtains [30]

$$\widetilde{\boldsymbol{B}} = \boldsymbol{B}\boldsymbol{R} = \begin{bmatrix} b_{11} & \cdots & b_{1j} \\ \vdots & & \vdots \\ b_{k1} & \cdots & b_{kj} \end{bmatrix} \in \mathbb{R}^{k\times j} \tag{3.46}$$

with $\text{rank}(\widetilde{\boldsymbol{B}}) = j$. Applying the SVD on $\widetilde{\boldsymbol{B}}$ results in $\widetilde{\boldsymbol{B}} = \widetilde{\boldsymbol{U}}\widetilde{\boldsymbol{S}}\widetilde{\boldsymbol{V}}^T$ with $\widetilde{\boldsymbol{U}} \in \mathbb{R}^{k\times k}$, $\widetilde{\boldsymbol{S}} \in \mathbb{R}^{k\times j}$, and $\widetilde{\boldsymbol{V}} \in \mathbb{R}^{j\times j}$. Note that the zero columns of $\boldsymbol{B}$ do not affect $\boldsymbol{U}$, because $\boldsymbol{B}\boldsymbol{B}^T = \widetilde{\boldsymbol{B}}\widetilde{\boldsymbol{B}}^T$. This implies that $\boldsymbol{U} = \widetilde{\boldsymbol{U}}$ and the nonzero singular values of $\boldsymbol{B}$ and $\widetilde{\boldsymbol{B}}$ are the same. Furthermore, it can be shown that the elements of $\widetilde{\boldsymbol{V}}$ coincide with those from the top-left $j \times j$-submatrix of $\boldsymbol{V}$. The MPP of $\widetilde{\boldsymbol{B}}$ is now a left inverse and reads as $\widetilde{\boldsymbol{B}}^\dagger = (\widetilde{\boldsymbol{B}}^T\widetilde{\boldsymbol{B}})^{-1}\widetilde{\boldsymbol{B}}^T = \widetilde{\boldsymbol{V}}\widetilde{\boldsymbol{\Sigma}}\widetilde{\boldsymbol{U}}^T$ [28] or more specifically [30]

$$\widetilde{\boldsymbol{B}}^\dagger = \begin{bmatrix} v_{11} & \cdots & v_{1j} \\ \vdots & & \vdots \\ v_{j1} & \cdots & v_{jj} \end{bmatrix} \begin{bmatrix} \sigma_1^{-1} & & & 0 & \cdots & 0 \\ & \ddots & & \vdots & & \vdots \\ & & \sigma_j^{-1} & 0 & \cdots & 0 \end{bmatrix} \boldsymbol{U}^T. \tag{3.47}$$

[3]In (2.8b) the expression $(\boldsymbol{B}\boldsymbol{B}^T)^{-1}$ does not exist any more.

Comparing (3.45) and (3.47) one realizes that if $j < k$ then

$$\boldsymbol{B}^{\ddagger} = \boldsymbol{R}\widetilde{\boldsymbol{B}}^{\dagger}, \tag{3.48}$$

i.e. the pseudoinverse of $\boldsymbol{B}^N$ is the left inverse of its nonzero columns augmented with rows of zeros [30].

3.6.1. RPINV with non-identity weighting matrices

Now an arbitrary weighting matrix $\boldsymbol{W} \succ 0$ is considered. By means of Cholesky decomposition $\boldsymbol{W}$ can be factorized into $\boldsymbol{W} = \widehat{\boldsymbol{W}}^T\widehat{\boldsymbol{W}}$ and $\widehat{\boldsymbol{W}} \in \mathbb{R}^{m\times m}$ is an upper triangular matrix with positive diagonal entries [24]. Assuming $j \geq k$ an auxiliary matrix is introduced as [30]

$$\boldsymbol{B}_{\boldsymbol{W}} = \boldsymbol{B}_{\boldsymbol{0}}\boldsymbol{R}\boldsymbol{R}^T\widehat{\boldsymbol{W}}^{-1} \tag{3.49}$$

with $\operatorname{rank}(\boldsymbol{B}_{\boldsymbol{W}}) = k$ following from (A.3) in Appendix A. Its MPP is

$$\boldsymbol{B}_{\boldsymbol{W}}^{\ddagger} = \widehat{\boldsymbol{W}}^{-T}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{B}_{\boldsymbol{0}}^T\left(\boldsymbol{B}_{\boldsymbol{0}}\boldsymbol{R}\boldsymbol{R}^T\underbrace{\widehat{\boldsymbol{W}}^{-1}\widehat{\boldsymbol{W}}^{-T}}_{\left(\widehat{\boldsymbol{W}}^T\widehat{\boldsymbol{W}}\right)^{-1}=\boldsymbol{W}^{-1}}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{B}_{\boldsymbol{0}}^T\right)^{-1}. \tag{3.50}$$

Comparing (2.21) and (3.50) it follows that the weighted pseudoinverse of $\boldsymbol{B}$ can be computed by [30]

$$\boldsymbol{B}^{\#} = \boldsymbol{R}\boldsymbol{R}^T\widehat{\boldsymbol{W}}^{-1}\boldsymbol{B}_{\boldsymbol{W}}^{\ddagger}. \tag{3.51}$$

Lemma 3.6.1. *In case of $j < k$ the weighting matrix has no influence on the resulting pseudoinverse [30].*

Proof. Combining (3.46) and (3.49) yields another auxiliary matrix

$$\widetilde{\boldsymbol{B}}_{\boldsymbol{W}} = \boldsymbol{B}_{\boldsymbol{W}}\boldsymbol{R} \tag{3.52}$$

with $\widetilde{\boldsymbol{B}}_{\boldsymbol{W}} \in \mathbb{R}^{k\times j}$. Its MPP reads as

$$\widetilde{\boldsymbol{B}}_{\boldsymbol{W}}^{\dagger} = \left(\boldsymbol{R}^T\widehat{\boldsymbol{W}}^{-T}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{B}_{\boldsymbol{0}}^T\boldsymbol{B}_{\boldsymbol{0}}\boldsymbol{R}\boldsymbol{R}^T\widehat{\boldsymbol{W}}^{-1}\boldsymbol{R}\right)^{-1}\boldsymbol{R}^T\widehat{\boldsymbol{W}}^{-T}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{B}_{\boldsymbol{0}}^T. \tag{3.53}$$

Right-multiplying a matrix with $\boldsymbol{R}$ selects according to (2.16) columns with indices $\{l_1, ..., l_j\}$ and left-multiplying with $\boldsymbol{R}^T$ selects rows with identical indices. Therefore, $\boldsymbol{R}^T\widehat{\boldsymbol{W}}^{-T}\boldsymbol{R}$ and $\boldsymbol{R}^T\widehat{\boldsymbol{W}}^{-1}\boldsymbol{R}$ are triangular matrices with positive diagonal entries which guarantees invertibility. It follows that [30]

$$\widetilde{\boldsymbol{B}}_{\boldsymbol{W}}^{\dagger} = \left(\boldsymbol{R}^T\widehat{\boldsymbol{W}}^{-1}\boldsymbol{R}\right)^{-1}\left(\widetilde{\boldsymbol{B}}^T\widetilde{\boldsymbol{B}}\right)^{-1}\widetilde{\boldsymbol{B}}^T \tag{3.54}$$

and together with (3.48) and (3.51) one obtains [30]

$$\begin{aligned}\boldsymbol{B}^{\#} &= \boldsymbol{R}\left(\boldsymbol{R}^T\widehat{\boldsymbol{W}}^{-1}\boldsymbol{R}\right)\left(\boldsymbol{R}^T\widehat{\boldsymbol{W}}^{-1}\boldsymbol{R}\right)^{-1}\left(\widetilde{\boldsymbol{B}}^T\widetilde{\boldsymbol{B}}\right)^{-1}\widetilde{\boldsymbol{B}}^T \\ &= \boldsymbol{R}\left(\widetilde{\boldsymbol{B}}^T\widetilde{\boldsymbol{B}}\right)^{-1}\widetilde{\boldsymbol{B}}^T = \boldsymbol{R}\widetilde{\boldsymbol{B}}^{\dagger} = \boldsymbol{B}^{\ddagger}\end{aligned}. \tag{3.55}$$

■

Assumption 3.6.3. Two input matrix factorizations (3.8) are related by means of a transformation matrix $\boldsymbol{T}$ with *orthogonal columns*, i.e.

$$\boldsymbol{T}\boldsymbol{T}^T = \boldsymbol{I_k} d \tag{3.56}$$

with $d \in \mathbb{R} \setminus \{0\}$.

Theorem 3.6.2. *Under Assumption 3.6.3 RPINV yields identical input vectors $\boldsymbol{u_1} \equiv \boldsymbol{u_2}$ for any $j \geq 0$ if $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ fulfill (3.21) [30].*

Proof. If $j \geq k$ Theorem 3.6.1 holds. In case of $j < k$ Lemma 3.6.1 enables the restriction of considerations on $\boldsymbol{W} = \boldsymbol{I_m}$. Using (3.11) leads to $\widetilde{\boldsymbol{B}}_{\boldsymbol{2}} = \boldsymbol{T}^{-1}\widetilde{\boldsymbol{B}}_{\boldsymbol{1}}$ and according to (3.55) the rank deficient pseudoinverse for the second factorization is [30]

$$\boldsymbol{B}_{\boldsymbol{2}}^{\#} = \boldsymbol{R}\left[\widetilde{\boldsymbol{B}}_{\boldsymbol{1}}^T\boldsymbol{T}^{-T}\boldsymbol{T}^{-1}\widetilde{\boldsymbol{B}}_{\boldsymbol{1}}\right]^{-1}\widetilde{\boldsymbol{B}}_{\boldsymbol{1}}^T\boldsymbol{T}^{-T}. \tag{3.57}$$

Considering (3.56) yields $\boldsymbol{T}^{-T}\boldsymbol{T}^{-1} = \boldsymbol{I_k}\frac{1}{d}$ and taking that out of the bracket in (3.57) one obtains $\boldsymbol{I_k} d\boldsymbol{T}^{-T} = \boldsymbol{T}$. Consequently, $\boldsymbol{B}_{\boldsymbol{2}}^{\#} = \boldsymbol{B}_{\boldsymbol{1}}^{\#}\boldsymbol{T}$ and together with (3.21) $\boldsymbol{u_1} \equiv \boldsymbol{u_2}$ follows [30]. ∎

Factorizations which do not comply with Assumption 3.6.3 lead to deviating allocation results if the number of free actuators decreases below k. This is due to the fact that $\boldsymbol{B}_{\boldsymbol{2}}^{\#} \neq \boldsymbol{B}_{\boldsymbol{1}}^{\#}\boldsymbol{T}$ and so $\boldsymbol{T}^{-1}$ is not canceled from (3.21) in the corresponding RPINV iteration.

3.7. Conclusion

At the beginning of this chapter transformation matrices are introduced which provide a systematic way of describing the relationship between the major CA-related quantities for different input matrix factorizations. Additionally, the factorization influence on three popular CA algorithms is investigated. Whereas WPINV and DA remain invariant under factorizations RPINV can be influenced depending on the number of actuator saturations. Only those factorizations which are connected by means of transformation matrices with orthogonal columns yield identical RPINV results under all circumstances. In other cases it is crucial when zeroing columns of the control effectivity matrix leads to its rank-reduction, because then the resulting virtual control vector starts to deviate from the desired one [30].

If actuator constraints are considered the AMS Φ and the subset $\Pi \subset \mathbb{R}^k$ which leads to feasible $\boldsymbol{u}$ for a given generalized inverse are convex polytopes. Transformation matrices enable the conversion of their volumes from one factorization to another.

4. Normalized Generalized Inverse

As mentioned in Section 3.5.1 it can be advantageous to use a generalized inverse that maximizes the volume ratio

$$p := \frac{V(\Pi)}{V(\Phi)} \tag{4.1}$$

because it may improve the overall actuator utilization. One way to compute Π-maximizing generalized inverses is known as *Tailored Generalized Inverse* (TINV). The control effectivity matrix $\boldsymbol{B}$ is split into an invertible part $\boldsymbol{B_a} \in \mathbb{R}^{k\times k}$ and $\boldsymbol{B_b} \in \mathbb{R}^{k\times(m-k)}$. As described in [3] and [4] the TINV can now be determined by [25]

$$\boldsymbol{P_{tail}} = \begin{bmatrix} \boldsymbol{B_a}^{-1}(\boldsymbol{I_k} - \boldsymbol{B_b P_b}) \\ \boldsymbol{P_b} \end{bmatrix} \tag{4.2}$$

with $\boldsymbol{P_b} \in \mathbb{R}^{(m-k)\times k}$ containing the free parameters corresponding to the number of DOF one has in the choice of a k-D subspace in $\mathbb{R}^m$ (see [3], [4], and [41]). These are tuned by means of a search algorithm and that changes the orientation of the k-D hyperplane which is spanned by the generalized inverse. Therefore the algorithm not only has to compute the intersections $\Theta(\boldsymbol{P_{tail}})$ with the m-D object Ω during each iteration step, but also the triangulation of $\partial(\Pi)$ before the volume can be determined. This makes the overall computation very expensive. Another major drawback of this optimization procedure is its strong dependability on the initial value of the free parameters $\boldsymbol{P_b}$. In general they are chosen to match with the corresponding entries in (2.8b), but this does not guarantee convergence to the optimum at all [25].

4.1. Algorithm principles

In order to avoid the drawbacks of TINV an alternative problem formulation is presented that tackles the problem from the opposite direction. Major parts of this chapter have been published by the author in [25]. The main ideas behind this method are:

- First, construct a matrix $\boldsymbol{P_n}$ spanning a k-D subspace in Ω using geometric principles.
- After that compute a factorization (3.3) for which $\boldsymbol{P_n}$ is a generalized inverse. We refer to $\boldsymbol{P_n}$ as *Normalized Generalized Inverse* (NINV).

Without loss of generality the method requires that the input constraints are symmetric, i.e. $\boldsymbol{u_{min}} = -\boldsymbol{u_{max}}$ with $\boldsymbol{u_{max}} > 0$. It is shown in [4] that in case of asymmetric constraints an input transformation $\boldsymbol{u'} = \boldsymbol{u} + \boldsymbol{u_\Delta}$ can be applied to overcome this restriction. For a certain class of problems a closed-form solution is stated, but in most cases a search algorithm is used which outperforms the existing approach for typical problem sizes [25].

4.2. Factorization of the input matrix

Theorem 4.2.1. *Let $\boldsymbol{B_u} \in \mathbb{R}^{n\times m}$ fulfilling (3.2) be the input matrix and $\boldsymbol{P_n} \in \mathbb{R}^{m\times k}$ satisfying* $\mathrm{rank}(\boldsymbol{B_u P_n}) = k$*. Then an input matrix factorization can be computed with*

$$\boldsymbol{B_u} = \boldsymbol{B_{v,n} B_n} \tag{4.3a}$$

$$\boldsymbol{B_{v,n}} = \boldsymbol{B_u P_n} \tag{4.3b}$$

$$\boldsymbol{B_n} = \left(\boldsymbol{B}^T_{\boldsymbol{v,n}} \boldsymbol{B_{v,n}}\right)^{-1} \boldsymbol{B}^T_{\boldsymbol{v,n}} \boldsymbol{B_u} \tag{4.3c}$$

and $\boldsymbol{P_n}$ is a right-inverse of $\boldsymbol{B_n}$ [25].

Proof. Considering $\mathrm{rank}(\boldsymbol{B}^T_{\boldsymbol{v,n}}\boldsymbol{B_{v,n}}) = \mathrm{rank}(\boldsymbol{B_{v,n}})$ (see [24] and [40]) and using (4.3b) yields

$$\begin{aligned} k = \mathrm{rank}(\boldsymbol{B}^T_{\boldsymbol{v,n}}\boldsymbol{B_{v,n}}) &= \mathrm{rank}\left[\left(\boldsymbol{B}^T_{\boldsymbol{v,n}}\boldsymbol{B_u}\right)\boldsymbol{P_n}\right] \\ &\leq \min\left[\mathrm{rank}(\boldsymbol{B}^T_{\boldsymbol{v,n}}\boldsymbol{B_u}), \mathrm{rank}(\boldsymbol{P_n})\right]. \end{aligned} \tag{4.4}$$

It follows from (4.4) that $\mathrm{rank}(\boldsymbol{B}^T_{\boldsymbol{v,n}}\boldsymbol{B_u}) = k$ and therefore $\mathrm{rank}(\boldsymbol{B_n}) = k$. Next it is shown that $\boldsymbol{P_n}$ is a right-inverse of $\boldsymbol{B_n}$ by inserting (4.3b) into (4.3c) and multiplying with $\boldsymbol{P_n}$ from the right-hand side [25]

$$\boldsymbol{B_n P_n} = \left(\boldsymbol{P}^T_{\boldsymbol{n}}\boldsymbol{B}^T_{\boldsymbol{u}}\boldsymbol{B_u P_n}\right)^{-1} \boldsymbol{P}^T_{\boldsymbol{n}}\boldsymbol{B}^T_{\boldsymbol{u}}\boldsymbol{B_u P_n} = \boldsymbol{I_k}. \tag{4.5}$$

Now (4.3a) is derived. Because of $\mathrm{rank}(\boldsymbol{P_n}) = \mathrm{rank}(\boldsymbol{B_{v,n}}) = k$ it follows that

$$\mathrm{rank}\underbrace{\left[\boldsymbol{P_n}\left(\boldsymbol{B}^T_{\boldsymbol{v,n}}\boldsymbol{B_{v,n}}\right)^{-1}\boldsymbol{B}^T_{\boldsymbol{v,n}}\right]}_{=:\boldsymbol{G}} = k. \tag{4.6}$$

Inserting (4.3b) into $\boldsymbol{G}$ reveals that $\boldsymbol{G B_u G} = \boldsymbol{G}$, which is one of the Penrose-equations. Due to $\mathrm{rank}(\boldsymbol{G}) = \mathrm{rank}(\boldsymbol{B_u})$ one can conclude that $\boldsymbol{B_u G B_u} = \boldsymbol{B_u}$ (see [27]). Finally, inserting (4.3b) and (4.3c) into that expression yields [25]

$$\underbrace{\boldsymbol{B_u}\left[\boldsymbol{P_n}\left(\boldsymbol{P}^T_{\boldsymbol{n}}\boldsymbol{B}^T_{\boldsymbol{u}}\boldsymbol{B_u P_n}\right)^{-1}\boldsymbol{P}^T_{\boldsymbol{n}}\boldsymbol{B}^T_{\boldsymbol{u}}\right]\boldsymbol{B_u}}_{=\boldsymbol{B_{v,n}B_n}} = \boldsymbol{B_u}. \tag{4.7}$$

■

4.3. Construction of $\boldsymbol{P_n}$ for $k = 1$

In the case of 1-D virtual control space a closed form solution can be derived that ensures 100% usage of Φ [25]. The concept behind NINV is best explained with the aid of the simplest constrained CA problem with $m = 2$ actuators and $k = 1$ virtual control. The actuator constraints can be visualized as red rectangle in Figure 4.1 and the nullspace of $\boldsymbol{B} \in \mathbb{R}^{1\times 2}$ is spanned by $\boldsymbol{N} \in \mathbb{R}^{2\times 1}$ depicted as yellow line. Generalized inverses $\boldsymbol{P} \in \mathbb{R}^{2\times 1}$ can be represented as lines through the origin, i.e. the CA results $\boldsymbol{u} = \boldsymbol{P}v$ are located on those lines. Figure 4.2 shows the corresponding Φ which is delimited by two projected vertices of Ω. The best achievable volume ratio is "1" meaning that there is no $v \in \Phi$ which gets mapped to $\boldsymbol{u} \notin \Omega$ by multiplication with $\boldsymbol{P}$. Drawing the attention back to real control space (Figure 4.1) this is equivalent with finding a line through the origin which does feature any points outside the red rectangle Ω from where Ω's boundary could be reached by moving along the nullspace direction. In the given example the optimal solution is clearly the green line intersecting vertices 2 and 3. This idea is now generalized for an arbitrary number of actuators.

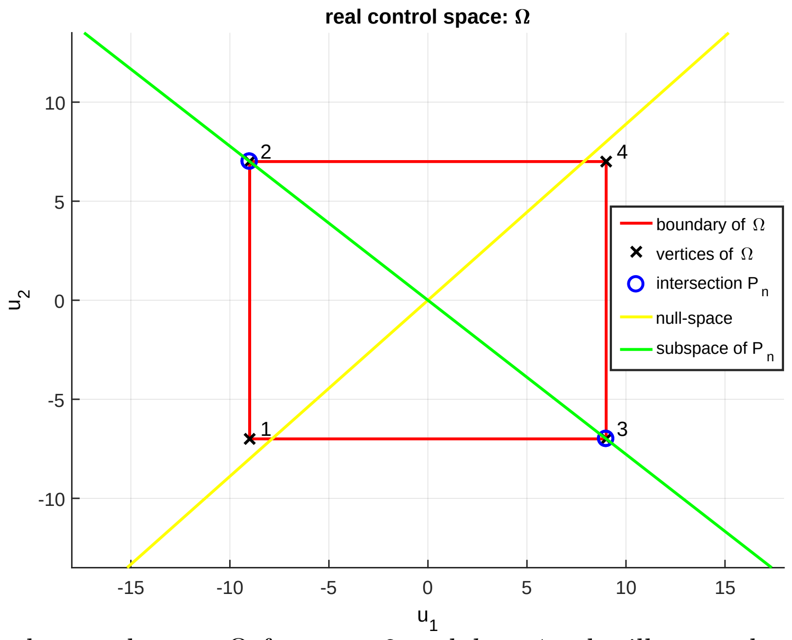

Figure 4.1.: Real control space Ω for $m = 2$ and $k = 1$: the illustrated generalized inverse enables the usage of the entire AMS (see Figure 4.2). This can be seen because there is no point on the green line which lies outside Ω and can be translated back into Ω by following the nullspace direction.

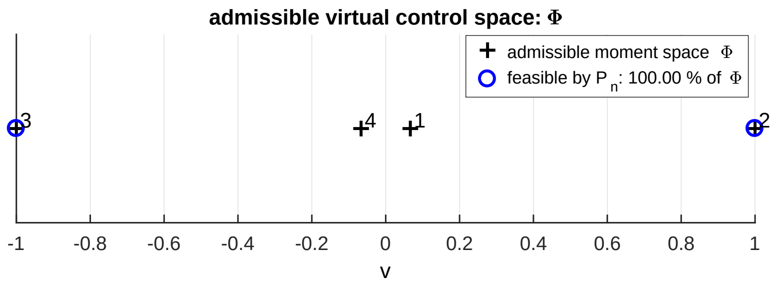

Figure 4.2.: Admissible virtual control space Φ belonging to Figure 4.1: $\Pi = \Phi$.

Theorem 4.3.1. *Consider an input matrix* $\boldsymbol{B_u} \in \mathbb{R}^{n \times m}$ *with* $\operatorname{rank}(\boldsymbol{B_u}) = 1$, *i.e.*

$$\boldsymbol{B_u} = \begin{bmatrix} \boldsymbol{b}_{\boldsymbol{u1}}^T \\ c_2 \boldsymbol{b}_{\boldsymbol{u1}}^T \\ \vdots \\ c_n \boldsymbol{b}_{\boldsymbol{u1}}^T \end{bmatrix} \tag{4.8}$$

where $\boldsymbol{b}_{\boldsymbol{u1}}^T = [b_{u11} \; \dots \; b_{u1m}]$ *is the first row of the input matrix and* $c_i \in \mathbb{R}$ *are scalar factors with* $i \in \{2, ..., n\}$. *Then*

$$\boldsymbol{P_n} = \operatorname{diag}\left[\operatorname{sign}(\boldsymbol{b}_{\boldsymbol{u1}}^T)\right] \boldsymbol{u_{max}} \tag{4.9}$$

is an optimal generalized inverse for factorization (4.3) which maximizes the volume ratio (4.1) [25].

Proof. Initially it is shown that the assumption $\operatorname{rank}(\boldsymbol{B_u P_n}) = \operatorname{rank}(\boldsymbol{B_{v,n}}) = 1$ in Theorem 4.2.1 is satisfied. Because of $\boldsymbol{B_{v,n}} \in \mathbb{R}^{n \times 1}$ this assumption is equivalent to the demand on

$\boldsymbol{B_{v,n}}$ not having all entries set to zero. Combining (4.8) and (4.9) the virtual input matrix can be written as [25]

$$\boldsymbol{B_{v,n}} = \begin{bmatrix} 1 & c_2 & \dots & c_n \end{bmatrix}^T \sum_{i=1}^{m} |b_{u1i}| u_{i,max}. \tag{4.10}$$

At least one entry of $\boldsymbol{b_{u1}^T}$ must be different from zero (otherwise there were no plant inputs) and so $\operatorname{rank}(\boldsymbol{B_{v,n}}) = 1$ is guaranteed. The next step is showing that (4.9) is the optimal solution. The set of Ω's vertices reads as [25]

$$\Upsilon = \{\boldsymbol{p_1}(\Omega), \dots, \boldsymbol{p_{2^m}}(\Omega)\} \tag{4.11}$$

where $\boldsymbol{p_j}(\Omega)^T = [\pm u_{1,max} \dots \pm u_{m,max}]$ is the j-th vertex corresponding to one specific sign combination. The AMS Φ_n is bounded by the convex hull of the projection of Υ into k-D virtual control space ([3], [4]), i.e. $\partial(\Phi_n) = \operatorname{conv}\{\boldsymbol{B_n}\Upsilon\}$. Since $k = 1$ one can simplify that to get its interval representation

$$\Phi_{n,1} = [\min(\boldsymbol{B_n}\Upsilon), \max(\boldsymbol{B_n}\Upsilon)]. \tag{4.12}$$

Inserting (4.8) and (4.10) into (4.3c) results in

$$\begin{aligned} \boldsymbol{B_n} &= \frac{\left(1 + c_2^2 + \dots + c_n^2\right)\left(\sum_{i=1}^{m} |b_{u1i}| u_{i,max}\right)}{\left(1 + c_2^2 + \dots + c_n^2\right)\left(\sum_{i=1}^{m} |b_{u1i}| u_{i,max}\right)^2} \boldsymbol{b_{u1}^T} \\ &= \frac{1}{\sum_{i=1}^{m} |b_{u1i}| u_{i,max}} \boldsymbol{b_{u1}^T}. \end{aligned} \tag{4.13}$$

The projection of an arbitrary vertex of Υ by means of (4.13) reads as

$$\boldsymbol{B_n}\boldsymbol{p_j}(\Omega) = \frac{\pm |b_{u11}| u_{1,max} \pm \dots \pm |b_{u1m}| u_{m,max}}{\sum_{i=1}^{m} |b_{u1i}| u_{i,max}}. \tag{4.14}$$

Equation (4.14) reaches its minimum (maximum) if all signs are negative (positive). Thus evaluating (4.12) yields $\Phi_{n,1} = [-1, 1]$. A direct implication from (4.9) is that the subset of $\Phi_{n,1}$ where a multiplication with $\boldsymbol{P_n}$ yields $\boldsymbol{u} \in \Omega$ can also be written as $\Pi_{n,1} = [-1, 1]$. Therefore $\Pi_{n,1} = \Phi_{n,1}$ and (4.9) is the optimal solution [25]. ■

Remark 4.3.1. A necessary condition for 100 % AMS usage is that the number of intersection points that form Π is equal to the number of vertices that define Φ. For higher dimensional virtual control spaces this requirement is in general not met, apart from some degenerated cases.

Remark 4.3.2. As it can be seen in Figure 4.2 v is restricted to values between -1 and +1. This normalization also appears for higher dimensional virtual control spaces (see next Section 4.4) and defines the name of this method.

4.4. Construction of $\boldsymbol{P_n}$ for $k > 1$

In this case a combinatorial search algorithm examines several candidates for $\boldsymbol{P_n}$ and computes their volume ratio in a very efficient way. The remainder of this section deals with the following major issues:

1. How can one efficiently determine shape and volume of Π?
2. Which vertex-combinations of Ω should be used as candidates for $\boldsymbol{P_n}$?
3. How can one efficiently compute the volume of the AMS Φ?

4.4.1. Shape and volume of Π

The candidates are constructed from vertices of Ω in order to normalize the subset Π. As a consequence, there is a finite number of possible shapes with prescribed volume that Π can take. Hence, there is no need for intersection computation and triangulation any more. Each candidate reads as [25]

$$\boldsymbol{P_n} = \begin{bmatrix} \pm u_{1,max} & \cdots & \pm u_{1,max} \\ \vdots & & \vdots \\ \pm u_{m,max} & \cdots & \pm u_{m,max} \end{bmatrix} \in \mathbb{R}^{m \times k} \tag{4.15}$$

and because of its columns being part of Ω's boundary they automatically specify k intersections.

Definition 4.4.1. *Let* $\boldsymbol{r}^T \in \mathbb{R}^{1\times q}$ *be a row of* $\boldsymbol{M} \in \mathbb{R}^{p\times q}$. *Then* $\boldsymbol{r}^T$ *is called* scalarly independent *if it cannot be obtained by a scalar multiplication of any of the* $p-1$ *other rows of* $\boldsymbol{M}$ *[25].*

Theorem 4.4.1. *Let the generalized inverse candidate complying with (4.15) contain* 2^{k-1} *scalarly independent rows. Then the resulting subset* Π *is a k-D cross-polytope with fixed volume [25].*

Proof. In order to compute Π one must project the vertices of $\Theta(\boldsymbol{P_n})$ into virtual control space $\mathbb{R}^k$ by means of the control effectivity matrix (4.3c). The k columns of (4.15) which are also vertices are mapped to unit vectors in $\mathbb{R}^k$, i.e. $\boldsymbol{I_k} = \boldsymbol{B_n}\boldsymbol{P_n}$ because of $\boldsymbol{B_n}$ being constructed to have $\boldsymbol{P_n}$ as right-inverse. Due to the symmetry of Ω one has to consider not only the vertices which form the generalized inverse but also their reflections around the origin (multiplication by -1). Consequently, they are mapped to negative unit vectors, i.e. $-\boldsymbol{I_k} = \boldsymbol{B_n}(-\boldsymbol{P_n})$ which means that at least $2k$ unit vectors are vertices of Π. Assuming that these $2k$ points are the only vertices, the subset which guarantees $\boldsymbol{P_n v} \in \Omega$ can be written as [25]

$$\Pi_X = \left\{ \boldsymbol{v} \in \mathbb{R}^k \middle| \left(\sum_{i=1}^{k} |v_i| \right) \leq 1 \right\}. \tag{4.16}$$

Expression (4.16) is also the definition of a k-D unit cross-polytope [43] whose volume reads as

$$V(\Pi_X) = \frac{2^k}{k!}. \tag{4.17}$$

What remains is to show under which circumstances the columns of $\boldsymbol{P_n}$ and $-\boldsymbol{P_n}$ are indeed the only vertices. According to (3.31) further vertices must lie in the column space $\mathcal{R}(\boldsymbol{P_n})$ of the generalized inverse. This is equivalent to being a linear combination of the columns of $\boldsymbol{P_n}$. Therefore, all intersections $\boldsymbol{s}$ fulfill [25]

$$\boldsymbol{P_n v} = \boldsymbol{s} \quad \text{with} \quad \boldsymbol{s} \in \partial(\Omega). \tag{4.18}$$

Noting from (3.32) that Π is the linear mapping of the convex set $\Theta(\boldsymbol{P_n})$ it is also convex. Hence, the projections $\boldsymbol{v} = [v_1 \dots v_k]^T$ of additional intersection points, that extend (4.16), have to satisfy [25]

$$\left(\sum_{i=1}^{k} |v_i| \right) > 1. \tag{4.19}$$

Considering (4.15) it can be seen that each row of the system of equations (4.18) is scaled by the corresponding component of $\boldsymbol{u_{max}}$. Therefore the entries of $\boldsymbol{P_n}$ and the bounds on the elements of vector $\boldsymbol{s}$ can be normalized to 1. The overall mathematical problem whose solution can yield additional intersections reads as [25]

$$\underbrace{\begin{bmatrix} \pm 1 & \cdots & \pm 1 \\ \vdots & & \vdots \\ \pm 1 & \cdots & \pm 1 \end{bmatrix}}_{\overline{\boldsymbol{P}}_{\boldsymbol{n}} \in \mathbb{R}^{m \times k}} \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} = \begin{bmatrix} s_1 \\ \vdots \\ s_m \end{bmatrix} \tag{4.20a}$$

$$\text{with } -1 \leq s_i \leq 1 \quad \text{for} \quad i = 1, \ldots, m \tag{4.20b}$$

$$\exists\, j \in \{1, \ldots, m\} : |s_j| = 1 \tag{4.20c}$$

$$|v_1| + \ldots + |v_k| > 1 \tag{4.20d}$$

The maximum number of possible unique rows in $\overline{\boldsymbol{P}}_{\boldsymbol{n}}$ is 2^k. They can be generated by taking all binary numbers from 0 to $2^k - 1$ and by replacing 0 by -1. A matrix containing the first half of the resulting combinations is given by [25]

$$\boldsymbol{J} = \begin{bmatrix} -1 & & \ldots & & & -1 \\ -1 & & \ldots & & -1 & 1 \\ -1 & & \ldots & -1 & 1 & -1 \\ -1 & & \ldots & -1 & 1 & 1 \\ & & & \vdots & & \\ -1 & 1 & \ldots & & 1 & -1 \\ -1 & 1 & \ldots & & & 1 \end{bmatrix} \in \mathbb{R}^{2^{k-1} \times k}. \tag{4.21}$$

Multiplying all combinations in (4.21) with -1 yields the remaining second half of possible rows. Consequently, one can compose a matrix $\tilde{\boldsymbol{J}} \in \mathbb{R}^{2^{k-1} \times k}$ of any rows taken from $\boldsymbol{J}$ and $-\boldsymbol{J}$ that are scalarly independent and $-\tilde{\boldsymbol{J}}$ is the other half of all possible rows.
On the other hand the number of possible sign combinations of the elements $v_1, \ldots, v_k$ is also 2^k and can be generated in exactly the same way as stated above. Thus if $\overline{\boldsymbol{P}}_{\boldsymbol{n}}$ contains 2^{k-1} scalarly independent rows from $\boldsymbol{J}$ and $-\boldsymbol{J}$ then one of those rows has equal signs as either $\boldsymbol{v}$ or $-\boldsymbol{v}$. Assuming the related row index being l the corresponding equation of (4.20a) can be written as [25]

$$(\pm 1)\left(|v_1| + \ \ldots \ + |v_k|\right) = s_l. \tag{4.22}$$

Equation (4.22) must now satisfy the conflicting inequalities (4.20b) and (4.20d). For this reason no more intersections can occur under the given circumstances [25]. ■

In order to clarify Theorem 4.4.1 consider the case where $k = 3$. According to (4.21) one obtains

$$\boldsymbol{J} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & 1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{4.23}$$

which contains one half of all possible rows of $\overline{\boldsymbol{P}}_{\boldsymbol{n}}$ and at the same time one half of the possible sign combinations of the elements of $\boldsymbol{v} = [v_1 \ \ v_2 \ \ v_3]^T$. Assume that $\overline{\boldsymbol{P}}_{\boldsymbol{n}}$ comprises all rows from (4.23). Consequently, the entries of one row must have either equal or opposite signs as the elements of any $\boldsymbol{v}$ and (4.22) follows from (4.20a). However, (4.22) is also true if any number of rows in (4.23) is multiplied by -1 as the rows remain scalarly independent.

Corollary 4.4.1. *In 2-D virtual control space the cross-polytope is the only possible geometric shape that the normalized subset* Π *can take (see Figure 4.3) [25].*

Proof. Every generalized inverse $\boldsymbol{P_{n,2}} \in \mathbb{R}^{m\times 2}$ satisfies $\operatorname{rank}(\boldsymbol{P_{n,2}}) = 2$ and so it follows that $\overline{\boldsymbol{P}}_{\boldsymbol{n}}$ in (4.20a) has two linearly independent rows. The corresponding matrix $\boldsymbol{J} \in \mathbb{R}^{2\times 2}$ in (4.21) also contains just two linearly independent rows. Therefore, the assumptions of Theorem 4.4.1 are automatically met [25]. ∎

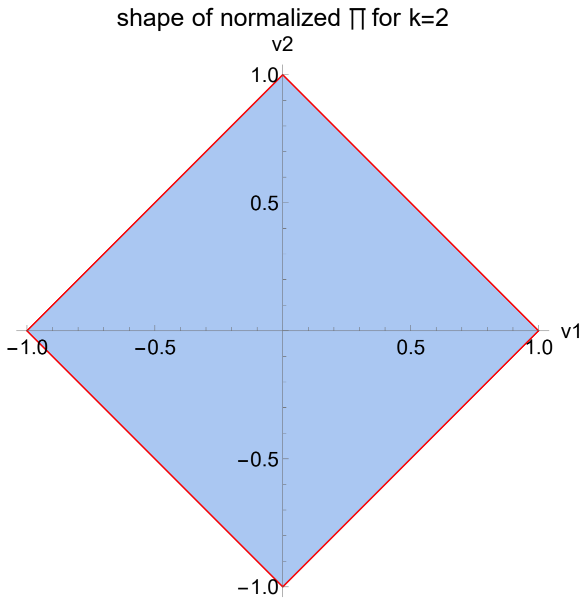

Figure 4.3.: The 2-D cross-polytope is a square and according to (4.17) its volume is $V = 2$. (© 2016 IEEE, [25]).

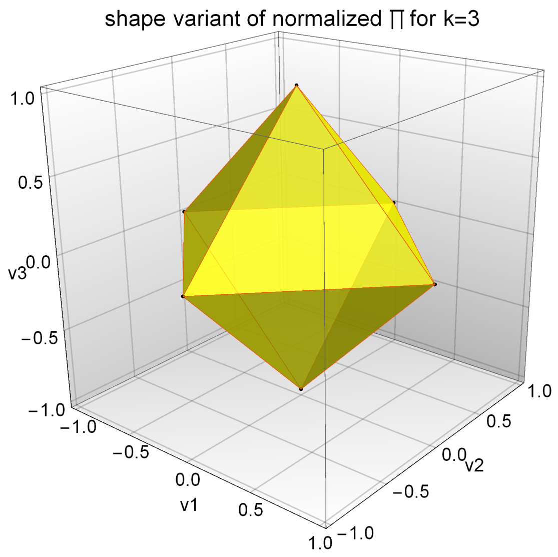

Figure 4.4.: The 3-D cross-polytope is an octahedron and according to (4.17) its volume is $V = \frac{4}{3}$. (© 2016 IEEE, [25]).

Corollary 4.4.2. *In 3-D virtual control spaces the shape of the normalized subset* Π *is either a cross-polytope or a parallelepiped (see Figures 4.4 - 4.8).*

Proof. Evaluating (4.21) for $k = 3$ yields

$$\boldsymbol{J_3} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & 1 \\ -1 & 1 & -1 \\ -1 & 1 & 1 \end{bmatrix}. \tag{4.24}$$

Every generalized inverse $\boldsymbol{P_{n,3}} \in \mathbb{R}^{m\times 3}$ satisfies $\operatorname{rank}(\boldsymbol{P_{n,3}}) = 3$ and so it contains only three linearly independent rows. Thus, it is not necessary for $\overline{\boldsymbol{P}}_{\boldsymbol{n}}$ to involve $2^{k-1} = 4$ scalarly independent rows from $\boldsymbol{J_3}$ and $-\boldsymbol{J_3}$ but only three. If four rows of $\overline{\boldsymbol{P}}_{\boldsymbol{n}}$ are scalarly independent it follows from Theorem 4.4.1 that Π is a cross-polytope (Figure 4.4). In case of only three scalarly independent rows the number of possibilities to choose three out of four without

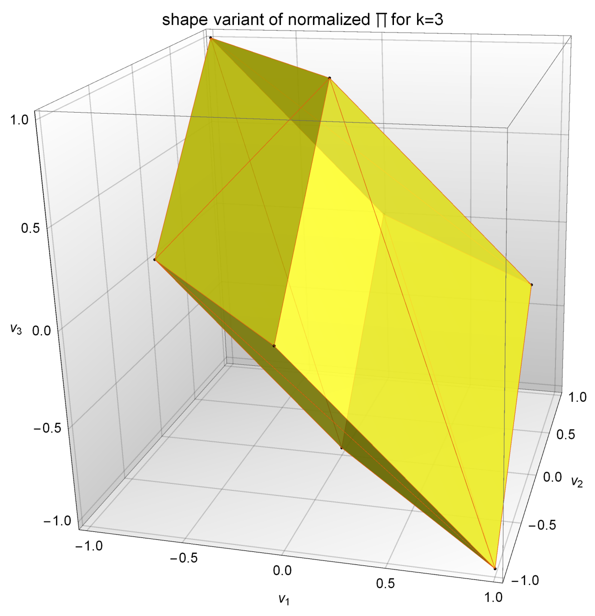

Figure 4.5.: Variant 1 of the normalized parallelepiped (rows 1, 2, and 3). (© 2016 IEEE, [25]).

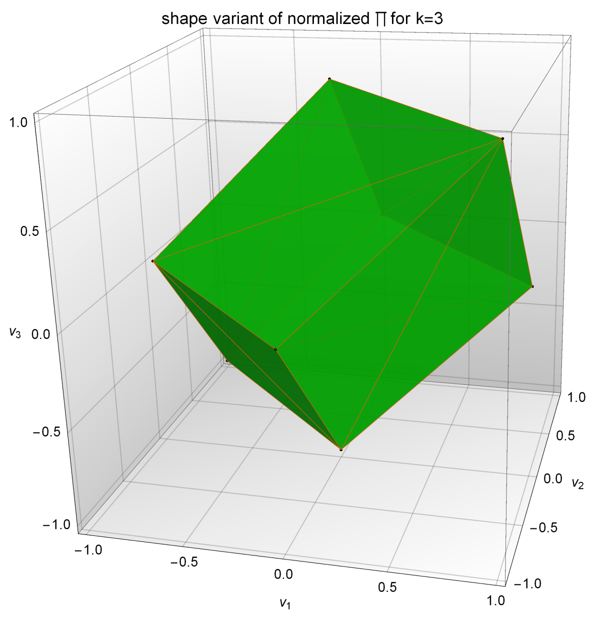

Figure 4.6.: Variant 2 of the normalized parallelepiped (rows 1, 2, and 4). (© 2016 IEEE, [25]).

repetition and ordering is $\binom{4}{3} = 4$. For each of those row combinations additional intersections extend Π which leads to four manifestations of the resulting geometric shape [25]. Assume that all rows of $\overline{\boldsymbol{P}}_{\boldsymbol{n}}$ are scalar multiples of the first three rows of $-\boldsymbol{J_3}$. Equation (4.20a) can now be reduced to

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}}_{\boldsymbol{v_I}} = \underbrace{\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}}_{\boldsymbol{s}} \tag{4.25}$$

because the remaining $m-3$ equations are linearly dependent, i.e. they are automatically fulfilled if a solution for (4.25) is found. Incorporating the solution of (4.25) into (4.20b) - (4.20d) yields

$$|s_2 + s_3| + |s_1 - s_3| + |s_1 - s_2| > 2 \tag{4.26a}$$
$$\text{with } -1 \le s_i \le 1 \quad \text{for} \quad i = 1, 2, 3 \tag{4.26b}$$
$$\exists j \in \{1, 2, 3\} : |s_j| = 1. \tag{4.26c}$$

Considering (4.26b) it can be seen that the maximum of the left-hand-side of (4.26a) is reached if $\boldsymbol{s}^T = [\ \ 1 -1 -1]$ or $\boldsymbol{s}^T = [-1 \quad 1 \quad 1]$. The projections of these solutions into virtual control space are $\boldsymbol{v_I^T} = [-1 \quad 1 \quad 1]$ and $-\boldsymbol{v_I^T} = [\ \ 1 \ -1 \ -1]$ respectively. It can be shown that all other solutions of (4.26) lie either on or inside of

$$\partial\left(\Pi_{\Diamond}\right) = \operatorname{conv}\left\{\boldsymbol{I_k}, -\boldsymbol{I_k}, \boldsymbol{v_I}, -\boldsymbol{v_I}\right\}. \tag{4.27}$$

Figure 4.5 shows the visualization of (4.27) and one can recognize a parallelepiped which is just a 3-D unit cross-polytope that is augmented by two unit simplices. The volume of a k-D unit simplex S (see [43]) is

$$V(S) = \frac{\sqrt{k+1}}{k!}. \tag{4.28}$$

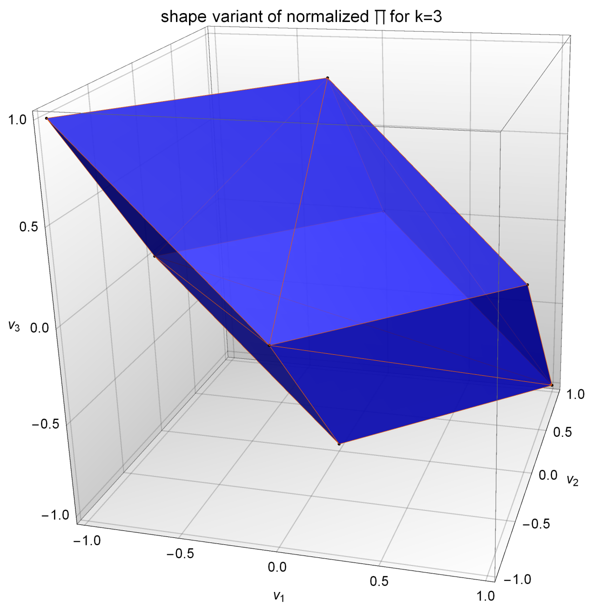

Figure 4.7.: Variant 3 of the normalized parallelepiped (rows 1, 3, and 4).

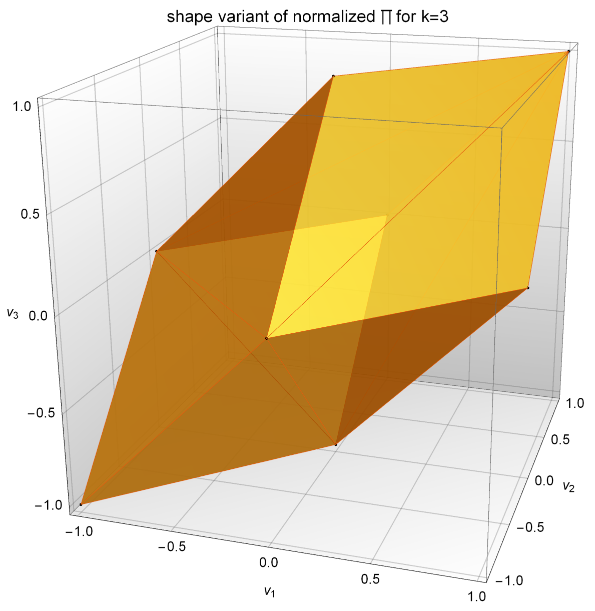

Figure 4.8.: Variant 4 of the normalized parallelepiped (rows 2, 3, and 4).

Using (4.17) and (4.28) the volume of the 3-D parallelepiped is [25]

$$V(\Pi_{\Diamond}) = V(\Pi_X) + 2V(S) = 2. \tag{4.29}$$

The other three possible row combinations of (4.25) reveal parallelepipeds with different orientations but the same volume (4.29) as it can be seen in Figures 4.6 - 4.8. ■

For higher virtual control space dimensions $k > 3$ the amount of possible shapes and volumes increases and the number of scalarly independent rows is not the only distinguishing factor any more.

4.4.2. Generating candidates for P_n

Definition 4.4.2. *An* n-flat *is an n-D subspace which might not contain the origin [4]. It is determined by $n+1$ points. In $\mathbb{R}^m$ (with $m > n$) a point on an n-flat ω has n DOFs and must fulfill $m - n = \mathrm{c_{geo}}(\omega)$ geometric constraints [41].*

Definition 4.4.3. *An* object *is a closed subset of an n-flat [4].*

The actuator constraints (1.31) form an m-D hyperrectangle which in turn is bounded by $2m$ so-called (m-1)-flats [43]. On each of those (m-1)-flats one actuator is either at its minimum or maximum value, i.e. a point on an (m-1)-flat has (m-1) DOFs and one geometric constraint which also corresponds to an actuator constraint (red faces in Figure 4.9). The intersection of two (m-1)-flats is an (m-2)-D flat where both of the corresponding two actuators are at their extremal values. So in general c (m-1)-flats on the boundary of Ω intersect in an (m-c)-flat ω_c where $c = \mathrm{c_{sat}}(\omega_c)$ actuators are permanently at their extremal values.

Definition 4.4.4. *The number of actuators which are at their extremal values everywhere on a given n-flat ω reads as $\mathrm{c_{sat}}(\omega) \in \mathbb{N}_+$ and is called* saturation number *[25].*

Definition 4.4.5. *The set of indices of permanently saturated actuators on a certain n-flat ω contains $c = \mathrm{c}_{\mathrm{sat}}(\omega)$ elements and is denoted as* saturation indices

$$\mathcal{S} = \{s_1, \ldots, s_c\} = r_{fix}(\omega). \tag{4.30}$$

Definition 4.4.6. *Consider a set of saturation indices $\mathcal{S}$ with c elements. $\omega_{\mathcal{S}}$ is the intersection of those c (m-1)-flats on $\partial(\Omega)$ whose permanently saturated actuators correspond to $\mathcal{S}$. A point on $\omega_{\mathcal{S}}$ has $m - c$ DOFs and must satisfy $c = \mathrm{c}_{\mathrm{geo}}(\omega_{\mathcal{S}})$ constraints [4]. Hence, the (m-c)-flat $\omega_{\mathcal{S}}$ exhibits the special relationship*

$$\mathrm{c}_{\mathrm{sat}}(\omega_{\mathcal{S}}) = \mathrm{c}_{\mathrm{geo}}(\omega_{\mathcal{S}}). \tag{4.31}$$

$\boldsymbol{P_n}$ consists of k column vectors of dimension m which are vertices of Ω and span a k-D subspace of $\mathbb{R}^m$ known as column space $\mathcal{R}(\boldsymbol{P_n})$. The intersection with the set of actuator constraints is $\Theta(\boldsymbol{P_n}) = \mathcal{R}(\boldsymbol{P_n}) \cap \Omega$ which is consequently bounded by (k-1)-flats $\overline{\vartheta}_i$ each of them requiring k points for its definition. Every (k-1)-flat is obtained from

$$\overline{\vartheta}_i = \mathcal{R}(\boldsymbol{P_n}) \cap \omega_{\mathcal{S}} \quad \text{with} \quad r_{fix}(\overline{\vartheta}_i) = r_{fix}(\omega_{\mathcal{S}}). \tag{4.32}$$

The feasible area on each n-flat is given by

$$\vartheta_i = \{\boldsymbol{u} \in \overline{\vartheta}_i \big| \boldsymbol{u} \in \partial(\Omega)\}. \tag{4.33}$$

Thus, $\boldsymbol{P_n}$ not only describes a k-D subspace Θ but also a (k-1)-flat $\overline{\vartheta}_i$ and a (k-1)-D object ϑ_i which is located on both boundaries, i.e. [25]

$$\vartheta_i \subset \partial\left[\Theta(\boldsymbol{P_n})\right] \subset \partial(\Omega) \quad \text{with}\, i = 1, \ldots, 2n_{\vartheta}. \tag{4.34}$$

The projections of (4.34) into virtual control space delimit the feasible set $\Pi\left(\boldsymbol{P_n}\right)$ for a particular candidate. Note that the number of bounding objects in (4.34) must be even because due to the symmetry of constraints an intersection with ϑ_i also implies $-\vartheta_i \subset \partial\left[\Theta(\boldsymbol{P_n})\right]$ where $-\vartheta_i$ represents its reflection in the origin. Therefore, $\partial\left[\Theta(\boldsymbol{P_n})\right]$ and $\partial\left[\Pi(\boldsymbol{P_n})\right]$ are uniquely determined by n_ϑ objects, provided that none of them is a reflection of another one. In case of $k = 2$ it can be seen from Figure 4.3 that $n_\vartheta = 2$. In contrast there are two possibilities if $k = 3$, namely $n_\vartheta = 3$ in case of Π being a parallelepiped (e.g. Figure 4.5) and $n_\vartheta = 4$ if Π is a cross-polytope (Figure 4.4). Matrix representations of each $\vartheta_i \in \partial\left[\Theta(\boldsymbol{P_n})\right]$ can be chosen as different candidates, but the resulting sets $\Theta \subset \mathbb{R}^m$ and $\Pi \subset \mathbb{R}^k$ remain the same as well as the corresponding volume ratio (4.1). This fact is exploited to reduce the number of vertex-combinations to check [25].

Figure 4.9 illustrates the previous definitions by means of a 3-D example. It also emphasizes the distinction between geometric (Definition 4.4.2) and actuator constraints (saturation number, Definition 4.4.4) of ϑ_i.

Lemma 4.4.1. *The saturation number $c = \mathrm{c}_{\mathrm{sat}}(\vartheta_i)$ of a (k-1)-D object ϑ_i in (4.34) satisfies*

$$1 \leq c \leq m - k + 1 = \mathrm{c}_{\mathrm{geo}}(\vartheta_i). \tag{4.35}$$

Proof. Since ϑ_i is a (k-1)-D object a point on ϑ_i has $k - 1$ DOFs and $\mathrm{c}_{\mathrm{geo}}(\vartheta_i) = m - k + 1$ geometric constraints. Hence, the maximum possible saturation number is $m - k + 1$ otherwise ϑ_i were a p-flat with $p < k - 1$. As $\vartheta_i \in \partial(\Omega)$ it follows that at least one actuator must be at an extremal value. ■

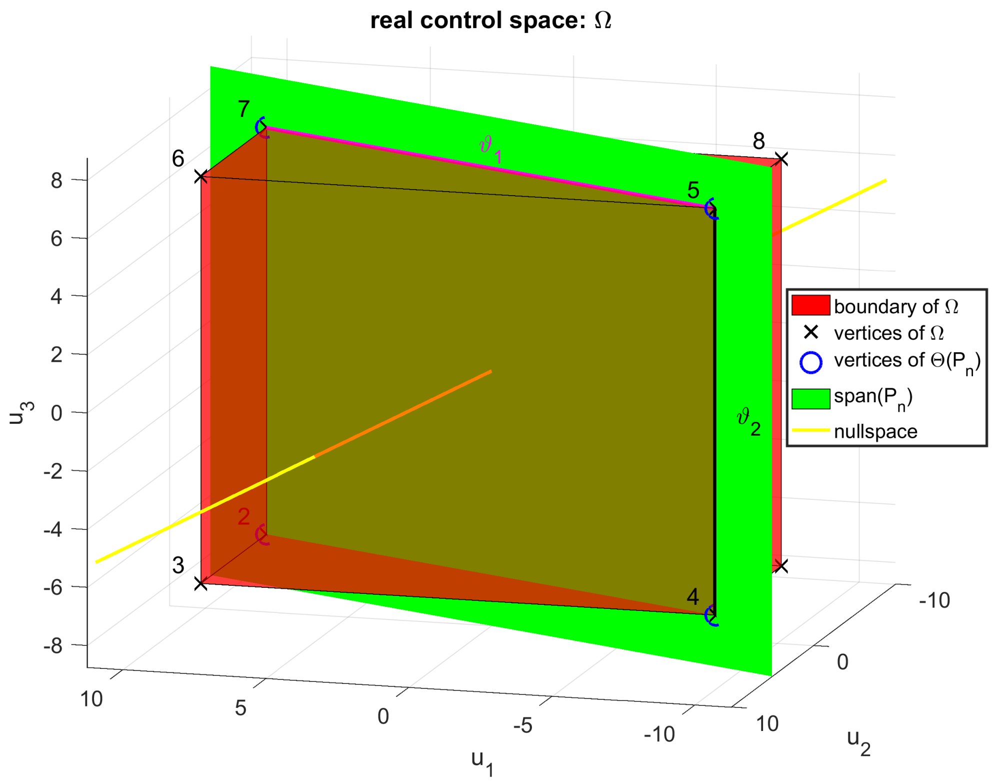

Figure 4.9.: Visualization of Ω and Θ for problem dimensions $m = 3$ and $k = 2$: Each red face is a 2-flat corresponding to one saturated actuator. The green plane depicts the k-D subspace spanned by one possible $\boldsymbol{P_n} \in \mathbb{R}^{3\times 2}$. The darker area inside Ω represents $\Theta(\boldsymbol{P_n})$. The boundary of $\Theta(\boldsymbol{P_n})$ consists of $2n_\vartheta = 4$ (k-1)-D objects which are lines in this case. Any of them is determined by two vertices of Ω which can be chosen as columns for a $\boldsymbol{P_n}$ candidate without changing $\Theta(\boldsymbol{P_n})$, $\Pi(\boldsymbol{P_n})$, and the volume ratio. Only two of them, either $\{\vartheta_1, \vartheta_2\}$ or $\{-\vartheta_1, -\vartheta_2\}$, are required to define the boundary $\partial\left[\Theta(\boldsymbol{P_n})\right]$ due to the symmetry. Their saturation numbers are $1 = \mathrm{c_{sat}}(\vartheta_1)$ and $2 = \mathrm{c_{sat}}(\vartheta_2)$. A point on any ϑ has one DOF and must satisfy two geometric constraints in a 3-D space. However, on ϑ_1 only one of them is due to actuator bounds as opposed to ϑ_2 which has two active actuator constraints.

Definition 4.4.7. *Each ϑ_i in (4.34) can be described by several matrix representations $\boldsymbol{P_{n,1}}$, $\boldsymbol{P_{n,2}}, \ldots$ which span the same column space. Since all $\boldsymbol{P_{n,j}}$ are composed of vertices of Ω it follows that those elements corresponding to saturated actuators are identical in all columns of the matrices. Thus, all $\boldsymbol{P_{n,j}}$ related to a specific ϑ_i contain the same $\mathrm{c_{sat}}(\vartheta_i)$ rows with identical elements which are called* fixed rows *and in accordance with (4.30) denoted as*

$$\forall j : r_{fix}(\boldsymbol{P_{n,j}}) = r_{fix}(\vartheta_i). \tag{4.36}$$

Remark 4.4.1. Note that every candidate matrix $\boldsymbol{P_n} \in \mathbb{R}^{m\times k}$ must satisfy $\mathrm{rank}(\boldsymbol{P_n}) = k$. Otherwise, at least one column were a linear combination of the others. As a consequence, only $k-1$ points were necessary to determine ϑ_i meaning it were a (k-2)-D object. Furthermore, it were no right-inverse of $\boldsymbol{B_n}$ (see Section 4.2).

Lemma 4.4.2. *Let ϑ_i be a (k-1)-D object in (4.34) and $\mathcal{S} = r_{fix}(\vartheta_i)$ with $c = \mathrm{c_{sat}}(\vartheta_i)$. Then $-\vartheta_i$ is the only other (k-1)-D object on $\partial\left[\Theta(\boldsymbol{P_n})\right]$ with actuators contained in $\mathcal{S}$ being at their extremal values permanently.*

Proof. Every ϑ_i has at least one matrix representation via $\boldsymbol{P_n}$ (see Definition 4.4.7). Assume their is another (k-1)-D object $\vartheta_j \in \partial\left[\Theta(\boldsymbol{P_n})\right]$ with $\mathcal{S} = r_{fix}(\vartheta_j)$. It follows that all of its matrix representations $\boldsymbol{P_{n,j}}$ must span the same column space as $\boldsymbol{P_n}$, i.e. $\forall j : \mathcal{R}(\boldsymbol{P_n}) = \mathcal{R}(\boldsymbol{P_{n,j}})$. A matrix $\boldsymbol{P_{n,j}}$ with the same column space as $\boldsymbol{P_n}$ must be representable as linear combination of its columns. Thus, the right-multiplication

$$\boldsymbol{P_n V} = \boldsymbol{P_{n,j}} \tag{4.37}$$

with $\boldsymbol{V} \in \mathbb{R}^{k\times k}$ and $\operatorname{rank}(\boldsymbol{V}) = k$ has to yield the sought matrix. W.l.o.g. assume that $\mathcal{S} = \{1,\dots,c\}$, i.e. the set consists of the first c row indices. According to (4.15) the elements in each row of $\boldsymbol{P_n}$ and $\boldsymbol{P_{n,j}}$ have the same magnitude. Therefore, a left-multiplication of (4.37) with a suitable invertible diagonal matrix normalizes all elements of $\boldsymbol{P_n}$ and $\boldsymbol{P_{n,j}}$ to ± 1. Note that it does not matter that this normalization changes the column spaces of $\boldsymbol{P_n}$ and $\boldsymbol{P_{n,j}}$ as this proof is based on the relationship between them. This is equivalent to assuming that $\boldsymbol{u_{max}} = \boldsymbol{i_m}$ with $\boldsymbol{i_m^T} = [1\dots 1] \in \mathbb{R}^{1\times m}$ and so the matrix representation for ϑ_i can be written as

$$\boldsymbol{P_n} = \begin{bmatrix} \pm\boldsymbol{i_k^T} \\ \vdots \\ \pm\boldsymbol{i_k^T} \\ \boldsymbol{p_{c+1}^T} \\ \vdots \\ \boldsymbol{p_m^T} \end{bmatrix} \begin{array}{l} \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} \tilde{\boldsymbol{P}}_{\boldsymbol{n}} \in \mathbb{R}^{c\times k} \text{ with } \operatorname{rank}(\tilde{P}_n) = 1 \\ \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} \overline{\boldsymbol{P}}_{\boldsymbol{n}} \in \mathbb{R}^{(m-c)\times k} \text{ with } k-1 \le \operatorname{rank}(\overline{\boldsymbol{P}}_{\boldsymbol{n}}) \le k \end{array} \tag{4.38}$$

with $\boldsymbol{p_j^T} = [\pm 1 \dots \pm 1] \in \mathbb{R}^{1\times k}\ \forall j = (c+1),\dots,m$ and at least one element of each row $\boldsymbol{p_j^T}$ has a different sign than the others. In general there are three possibilities[1] that have to be considered:

1. Exactly the same set of actuator constraints is permanently active on ϑ_j. It follows from (4.32) that $\overline{\vartheta}_j = \mathcal{R}(\boldsymbol{P_n}) \cap \omega_{\mathcal{S}} = \overline{\vartheta}_i$ and so (4.33) implies identical objects $\vartheta_j = \vartheta_i$.
2. The inverse constraints are permanently active on ϑ_j. According to (4.32) one obtains $\overline{\vartheta}_j = \mathcal{R}(\boldsymbol{P_n}) \cap -\omega_{\mathcal{S}} = -\overline{\vartheta}_i$ which means $\vartheta_j = -\vartheta_i$. Thus, ϑ_j is the reflection of ϑ_i in the origin.
3. ϑ_j and ϑ_i have equal saturation indices but $1 \le d \le (c-1)$ of those actuators have different saturation types (min. or max). Considering (4.37) and (4.38) this is equivalent to the modification of d rows of $\tilde{\boldsymbol{P}}_{\boldsymbol{n}}$ by right-multiplying with $\boldsymbol{V}$ without affecting the remaining $c-d$ rows. Since $\operatorname{rank}(\tilde{\boldsymbol{P}}_{\boldsymbol{n}}) = 1$ this is impossible.

Now assume that $r_{fix}(\vartheta_j) \subset \mathcal{S}$. This also requires changing certain rows of $\tilde{\boldsymbol{P}}_{\boldsymbol{n}}$ in (4.38) by right-multiplying with $\boldsymbol{V}$ without affecting the remaining ones. Hence, this scenario cannot occur.

Finally, consider the case where $\mathcal{S} \subset r_{fix}(\vartheta_j)$. Suppose there is a matrix $\boldsymbol{P_{n,j}}$ in $\mathcal{R}(\boldsymbol{P_n})$ obtained from (4.37) with $r_{fix}(\boldsymbol{P_{n,j}}) = \{1,\dots,(c+1)\}$. Considering only the first $c+1$ rows

[1] Note that $\mathcal{S}$ only contains saturation indices. It does not provide any information about the type of saturation (min. or max.).

of (4.37) gives

$$\underbrace{\begin{bmatrix} \pm\boldsymbol{i}_{\boldsymbol{k}}^T \\ \vdots \\ \pm\boldsymbol{i}_{\boldsymbol{k}}^T \\ \boldsymbol{p}_{\boldsymbol{c}}^T \end{bmatrix}}_{\hat{\boldsymbol{P}}_{\boldsymbol{n}} \in \mathbb{R}^{(c+1)\times k}} \cdot \boldsymbol{V} = \underbrace{\begin{bmatrix} \pm\boldsymbol{i}_{\boldsymbol{k}}^T \\ \vdots \\ \pm\boldsymbol{i}_{\boldsymbol{k}}^T \\ \pm\boldsymbol{i}_{\boldsymbol{k}}^T \end{bmatrix}}_{\hat{\boldsymbol{P}}_{\boldsymbol{n},\boldsymbol{j}} \in \mathbb{R}^{(c+1)\times k}} \tag{4.39}$$

with $\operatorname{rank}(\hat{\boldsymbol{P}}_{\boldsymbol{n}}) = 2$ and $\operatorname{rank}(\hat{\boldsymbol{P}}_{\boldsymbol{n},\boldsymbol{j}} = 1)$. Applying (A.3) yields

$$\begin{aligned} \operatorname{rank}(\hat{\boldsymbol{P}}_{\boldsymbol{n}}) + \operatorname{rank}(\boldsymbol{V}) - k \leq \operatorname{rank}(\hat{\boldsymbol{P}}_{\boldsymbol{n}}\boldsymbol{V}) \leq \min\left[\operatorname{rank}(\hat{\boldsymbol{P}}_{\boldsymbol{n}}), \operatorname{rank}(\boldsymbol{V})\right] \\ 2 + \operatorname{rank}(\boldsymbol{V}) - k \leq 1 \leq \min\left[2, \operatorname{rank}(\boldsymbol{V})\right]. \end{aligned} \tag{4.40}$$

Inequality (4.40) is satisfied under the assumption $\operatorname{rank}(\boldsymbol{V}) = 1$ and $k \geq 2$ but it can be seen from (4.37) and (A.3) that $\operatorname{rank}(\boldsymbol{P}_{\boldsymbol{n},\boldsymbol{j}}) \neq k$. On the other hand assuming that $\operatorname{rank}(\boldsymbol{V}) > 1$ one obtains from (4.40)

$$\begin{aligned} 2 + \operatorname{rank}(\boldsymbol{V}) - k \leq 1 \leq 2 \\ \operatorname{rank}(\boldsymbol{V}) \leq k - 1 \end{aligned} \tag{4.41}$$

which also renders $\operatorname{rank}(\boldsymbol{P}_{\boldsymbol{n},\boldsymbol{j}}) = k$ impossible. Therefore ϑ_i and $-\vartheta_i$ are the only (k-1)-D objects in $\partial\left[\Theta(\boldsymbol{P}_{\boldsymbol{n}})\right]$ which contain saturation indices from $\mathcal{S}$. ■

In summary, Lemma 4.4.2 states that it is impossible to change only a subset of the fixed rows of $\boldsymbol{P}_{\boldsymbol{n}}$ or to preserve them while modifying the rest of the matrix.

Theorem 4.4.2. *Let $\boldsymbol{P}_{\boldsymbol{n}}$ be a generalized inverse according to (4.15). Consider a set of (k-1)-D bounding objects of $\Theta(\boldsymbol{P}_{\boldsymbol{n}})$*

$$\Theta_{\partial/2} = \left\{\vartheta_i \middle| \vartheta_i \neq \pm\vartheta_j \; i \in \mathcal{I},\, j \in \mathcal{I} \setminus i\right\} \tag{4.42}$$

with $\mathcal{I} = \{1, \ldots, n_\vartheta\}$, that uniquely *defines $\partial\left[\Theta(\boldsymbol{P}_{\boldsymbol{n}})\right]$, i.e. contains no reflections of other included objects. Then the sum of saturation numbers fulfills*

$$n_\vartheta \leq \sum_{i=1}^{n_\vartheta} \mathrm{c}_{\mathrm{sat}}(\vartheta_i) \leq m \quad \textit{with} \quad \vartheta_i \in \Theta_{\partial/2} \tag{4.43}$$

and

$$n_\vartheta \geq k. \tag{4.44}$$

Proof. The columns $\boldsymbol{q}_{\boldsymbol{j}}$ with $j = 1, \ldots, k$ of $\boldsymbol{P}_{\boldsymbol{n}}$ are vertices of $\Theta(\boldsymbol{P}_{\boldsymbol{n}})$. Each vertex $\boldsymbol{q}_{\boldsymbol{j}}$ is the intersection of n_o (k-1)-flats lying on $\partial\left[\Theta(\boldsymbol{P}_{\boldsymbol{n}})\right]$. Hence, on $\boldsymbol{q}_{\boldsymbol{j}}$ every constraint of the surrounding (k-1)-flats is satisfied. However, not all of them are actuator constraints but rather general geometric constraints that a point must fulfill in order to be on the corresponding flat. From Lemma 4.4.1 we know that the number of permanently saturated actuators on the corresponding (k-1)-D objects is given by $1 \leq \mathrm{c}_{\mathrm{sat}}(\vartheta_i) \leq m - k + 1$. Lemma 4.4.2 states that each actuator is permanently saturated only on *one* object in $\Theta_{\partial/2}$. It follows that

$$n_o \leq \sum_{i=1}^{n_o} \mathrm{c}_{\mathrm{sat}}(\vartheta_i) \leq m. \tag{4.45}$$

On the other hand, the column vectors $\boldsymbol{q_j}$ are also a vertices of Ω. A vertex can also be referred to as 0-flat which means that it is the intersection of m (m-1)-flats that bound the m-D hyperrectangle Ω. This is exactly one half of all existing (m-1)-flats of Ω and due to the symmetry of constraints a reflection around the origin yields the remaining ones. Each ϑ_i is a subset of at least one (m-1)-flat otherwise it were not part of $\partial(\Omega)$. Due to the symmetry Ω and $\Theta(\boldsymbol{P_n})$ a vertex must also be the intersection of one half of all objects on $\partial[\Theta(\boldsymbol{P_n})]$, i.e. $n_o = n_\vartheta$. The objects ϑ_i are part of a k-D subspace and so one needs at least the intersection of k (k-1)-flats to obtain a 0-flat, which means $n_\vartheta \geq k$. ■

The rather abstract result of Theorem 4.4.2 is illustrated by means of an example. Let $m = 5$ and $k = 3$. According to (4.43) the bounding objects of Θ resulting in a parallelepiped fulfill

$$3 \leq \underbrace{\sum_{i=1}^{3} \mathrm{c}_{\mathrm{sat}}(\vartheta_i)}_{s_c} \leq 5. \tag{4.46}$$

Assuming that $s_c = 5$ there are two possibilities to combine the corresponding saturation numbers (the ordering does not matter), namely

$$s_c = 5 = 1 + 1 + 3$$
$$s_c = 5 = 1 + 2 + 2.$$

The other possible outcomes of (4.46) are $s_c = 4$ and $s_c = 3$ which yield

$$s_c = 4 = 1 + 1 + 2 \quad \text{and}$$
$$s_c = 3 = 1 + 1 + 1.$$

In case of Θ's projection into virtual control space being a cross-polytope (4.46) changes to

$$4 \leq \underbrace{\sum_{i=1}^{4} \mathrm{c}_{\mathrm{sat}}(\vartheta_i)}_{s_c} \leq 5 \tag{4.47}$$

which is achievable by two combinations

$$s_c = 5 = 1 + 1 + 1 + 2 \quad \text{and}$$
$$s_c = 4 = 1 + 1 + 1 + 1.$$

Each $\boldsymbol{P_n}$ uniquely describes a ϑ and its corresponding $\Theta(\boldsymbol{P_n})$. Since every possible sum of saturation numbers contains "1" it is sufficient to use only candidates with $1 = \mathrm{c}_{\mathrm{sat}}(\vartheta_i)$. In contrast if $m = 6$ one possibility to gain a Θ that leads to a parallelepiped is $s_c = 6 = 2+2+2$. Hence, for 6-D control spaces candidates with $2 = \mathrm{c}_{\mathrm{sat}}(\vartheta_i)$ have to be considered additionally. This observation is formalized in the following corollary for the purpose of reducing the computational effort of the algorithm.

Corollary 4.4.3. *Given the dimensions m of real control space and k of virtual control space the candidates $\boldsymbol{P_n}$ can be restricted to those whose corresponding object's saturation number complies with*

$$1 \leq \mathrm{c}_{\mathrm{sat}}(\vartheta) \leq \left\lfloor \frac{m}{k} \right\rfloor . \tag{4.48}$$

Proof. Due to the fact that all summands of (4.43) are integer there is a finite number of possible combinations. As the number of summands in (4.43) is fixed it follows that the highest saturation numbers are required for the greatest sum. Hence, considerations are restricted to the upper bound of (4.43), i.e. $m = \sum_{i=1}^{n_\vartheta} \mathrm{c}_{\mathrm{sat}}(\vartheta_i)$. Suppose that $(m \bmod n_\vartheta) = 0$ one can write (4.43) as

$$m = \mathrm{c}_{\mathrm{sat}}(\vartheta_1) + \ldots + \mathrm{c}_{\mathrm{sat}}(\vartheta_{n_\vartheta}) = \frac{m}{n_\vartheta} + \ldots + \frac{m}{n_\vartheta}. \tag{4.49}$$

So in this case $\frac{m}{n_\vartheta}$ is the upper bound for the saturation number of the candidates because each $\mathrm{c}_{\mathrm{sat}}(\vartheta_i) > \frac{m}{n_\vartheta}$ implies that (some of) the remaining summands are decreased. When $(m \bmod n_\vartheta) > 0$ it follows that

$$m = n_\vartheta \left\lfloor \frac{m}{n_\vartheta} \right\rfloor + \underbrace{m \bmod n_\vartheta}_{=:\sigma < n_\vartheta}. \tag{4.50}$$

Since the maximum value that σ can take is $n_\vartheta - 1$ there is at least one $\mathrm{c}_{\mathrm{sat}}(\vartheta_i) = \left\lfloor \frac{m}{n_\vartheta} \right\rfloor$ if σ is equally distributed among all summands. Inequality (4.44) can be used to define an upper bound $\left\lfloor \frac{m}{k} \right\rfloor$ for $\mathrm{c}_{\mathrm{sat}}(\vartheta_i)$ which is independent of the unknown n_ϑ. ■

All candidates with $\mathrm{c}_{\mathrm{sat}}(\vartheta) > \left\lfloor \frac{m}{k} \right\rfloor$ are part of a k-D subspace of another candidate that satisfies (4.48) and so they can be ignored. The algorithm iterates over all (k-1)-D objects $\vartheta \subset \partial(\Omega)$ fulfilling (4.48). Each of these objects is represented as matrix $\overline{\boldsymbol{P}}_{\boldsymbol{n}}$ as defined in (4.20a) whereby the signs of the object's vertices specify the matrix columns. The volume of the corresponding set Π in virtual control space is defined by the scalarly independent rows of the resulting matrix $\overline{\boldsymbol{P}}_{\boldsymbol{n}}$ [25].

4.4.3. Efficient computation of the AMS volume

For the purpose of evaluating (4.1) one must also determine the volume of the AMS Φ. At the beginning of the algorithm an arbitrary input matrix factorization $\boldsymbol{B_u} = \boldsymbol{B_{v0}} \boldsymbol{B_0}$ is computed. After that the volume of Π_0 related to the weighted pseudoinverse with $\boldsymbol{W} = \mathrm{diag}(\frac{1}{u_{1,max}}, ..., \frac{1}{u_{m,max}})$ is calculated together with the AMS-volume $V(\Phi_0)$. Only generalized inverses that have a greater volume ratio than this standard solution are considered as possible results. For each candidate $\boldsymbol{P_n}$ there is an appropriate factorization of $\boldsymbol{B_u}$ according to (4.3). Using (3.9) and (4.3b) the transformation matrix from the initial to the candidate's factorization reads as [25]

$$\boldsymbol{T_{0c}} = \left(\boldsymbol{B_{v0}}^T \boldsymbol{B_{v0}}\right)^{-1} \boldsymbol{B_{v0}}^T \boldsymbol{B_u} \boldsymbol{P_n}. \tag{4.51}$$

Now (3.36) is applied to obtain the AMS volume in the candidate's factorization [25]

$$V(\Phi) = \frac{V(\Phi_0)}{\left|\det\left(\boldsymbol{T_{0c}}\right)\right|}. \tag{4.52}$$

4.4.4. Selecting the best candidate

The algorithm uses the candidates' prescribed volume and (4.52) to find the generalized inverse with the greatest volume ratio (4.1) [25].

Remark 4.4.2. Note that (4.51) is the first occasion where the actual problem data ($\boldsymbol{B_u}$ and $\boldsymbol{u_{max}}$) appear in the computations. Therefore all preceding steps could be done offline provided that k and m are known. The online-part of the algorithm would only consist of evaluating (4.51), (4.52), and (4.1) for all candidates. But this separation has not been done in the current implementation [25].

4.5. Results

The first example demonstrates the NINV computation for an input matrix

$$\boldsymbol{B_u} = \begin{bmatrix} 5 & -7 & 4 \\ -15 & 21 & -12 \end{bmatrix}$$

with $\text{rank}(\boldsymbol{B_u}) = 1$ and input constraints $\boldsymbol{u_{max}^T} = [5 \quad 8 \quad 8]$. According to (4.9) the NINV reads as $\boldsymbol{P_n^T} = [5 \; -8 \quad 8]$ and the resulting factorization (4.3) yields [25]

$$\boldsymbol{B_{v,n}} = \begin{bmatrix} 113 \\ -339 \end{bmatrix} \quad \text{and} \quad \boldsymbol{B_n} = [\tfrac{5}{113} \quad \tfrac{-7}{113} \quad \tfrac{4}{113}].$$

Figure 4.10 shows the real control space for this example. It can be seen that there is no point in $\mathcal{R}(\boldsymbol{P_n})$ (green line) which lies outside Ω and from where Ω could be reached by moving along the nullspace directions. Hence the entire AMS Φ is utilized. The next example deals with the input matrix [25]

$$\boldsymbol{B_u} = \begin{bmatrix} -1 & -1 & 3 & -2 & 5 & 2 \\ 3 & 4 & -1 & 3 & -1 & 0 \end{bmatrix}$$

with $\text{rank}(\boldsymbol{B_u}) = 2$ and the constraints are $\boldsymbol{u_{max}^T} = [8 \quad 8 \quad 5 \quad 8 \quad 8 \quad 7]$. The resulting NINV is

$$\boldsymbol{P_n} = \begin{bmatrix} -8 & -8 \\ -8 & -8 \\ -5 & 5 \\ -8 & -8 \\ -8 & 8 \\ -7 & 7 \end{bmatrix}$$

and the saturation number of the corresponding (k-1)-D object is three. The input matrix factorization according to (4.3) gives

$$\boldsymbol{B_{v,n}} = \begin{bmatrix} -37 & 101 \\ -67 & -93 \end{bmatrix} \quad \text{and} \quad \boldsymbol{B_n} = \begin{bmatrix} \frac{-41}{1993} & \frac{-164}{5383} & \frac{-23}{1319} & \frac{-117}{10208} & \frac{-91}{2552} & \frac{-17}{933} \\ \frac{-23}{1319} & \frac{-48}{2279} & \frac{55}{2359} & \frac{-245}{10208} & \frac{93}{2552} & \frac{39}{2971} \end{bmatrix}.$$

Figure 4.11 shows the usable subsets of Φ for the NINV (Π_1) and the weighted pseudoinverse (Π_2) with $\boldsymbol{W} = \text{diag}(\frac{1}{u_{1,max}}, ..., \frac{1}{u_{6,max}})$. The achieved volume ratio of NINV is nearly 86 % which is significantly higher than that of the weighted pseudoinverse [25]. In order to demonstrate the performance of the proposed algorithm it is compared with TINV. The evaluation procedure randomly generates input matrices and constraints and compares computation time as well as the resulting volume ratio of both methods. The number of virtual controls k is varied between 2 and 3 and the number of real controls m takes values from k+1 to 9. For each combination of m and k the average results of 500 tests are shown in Tables 4.1 and 4.2. It can be seen that the average computation time of NINV is significantly lower than

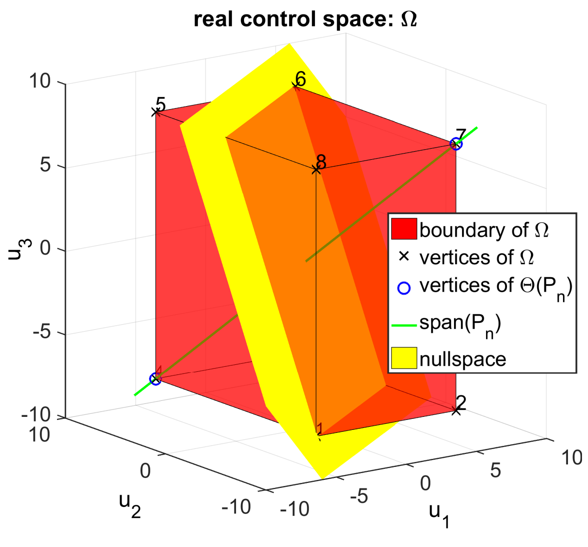

Figure 4.10.: Real control space for example 1. The green line represents $\boldsymbol{P_n}$. (© 2016 IEEE, [25]).

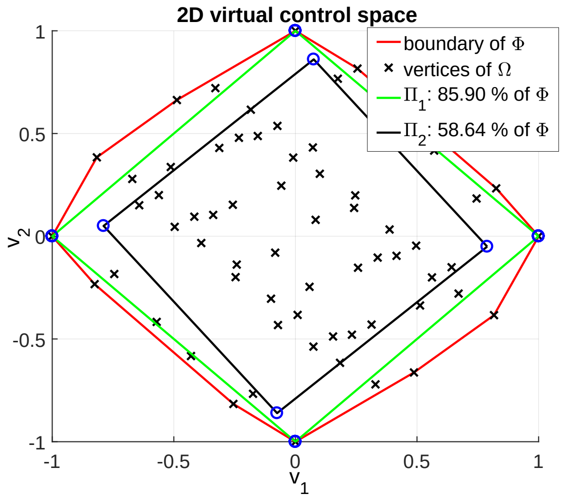

Figure 4.11.: Virtual control space for example 2. The green square is the feasible area for the NINV. (© 2016 IEEE, [25]).

Table 4.1.: Computation times and quality of results of NINV and TINV are shown for 2-D virtual control spaces. (© 2016 IEEE, [25]).

m	Avg. time TINV	Avg. time NINV	equal results	NINV better	Avg. diff. in remaining cases
3	0.236 s	0.004 s	76.0 %	5.4 %	1.4 %
4	1.074 s	0.006 s	46.6 %	36.6 %	1.1 %
5	2.402 s	0.008 s	32.8 %	52.0 %	1.4 %
6	4.388 s	0.015 s	23.2 %	64.2 %	0.9 %
7	7.526 s	0.026 s	17.8 %	69.0 %	1.1 %
8	12.092 s	0.083 s	16.4 %	73.4 %	1.0 %
9	15.850 s	0.194 s	12.8 %	79.6 %	0.8 %

that of TINV. Furthermore, the tables show that with increasing m NINV's results tend to be better than TINV. The reason for that is the increasing number of degrees of freedom in (4.2) which makes it more likely for TINV to get stuck in a local maximum. The last column in these tables deals with those tests where TINV obtained a higher volume ratio than NINV. It contains the average volume ratio benefit of TINV in these cases and one recognizes that it is always below 2 % which bears no relation with the computational overhead. The test machine is a notebook equipped with an Intel Core i5-3320M CPU, 4 GB RAM, Windows 10 64-Bit, and Matlab R2015b 64-bit. Both algorithms are implemented in Matlab. Convex hull and volume calculations in both algorithms are done by means of "convhulln", which is an implementation of the Quickhull algorithm [44]. TINV uses "fminunc" with a quasi-newton algorithm to search for the maximum volume ratio. The computation of $\Theta(\boldsymbol{P_{tail}})$ is carried out as described in [4] during the iterations of TINV [25].

Table 4.2.: Comparison of computation times and result quality of NINV and TINV for 3-D virtual control spaces. (© 2016 IEEE, [25]).

m	Avg. time TINV	Avg. time NINV	equal results	NINV better	Avg. diff. in remaining cases
4	0.420 s	0.006 s	57.8 %	3.8 %	1.5 %
5	2.978 s	0.018 s	23.4 %	22.0 %	1.6 %
6	8.396 s	0.122 s	15.2 %	36.6 %	1.5 %
7	19.246 s	0.775 s	12.8 %	45.6 %	1.2 %
8	38.430 s	5.182 s	8.2 %	50.0 %	1.2 %
9	68.760 s	49.507 s	10.6 %	58.4 %	1.1 %

4.6. Conclusion

A novel computation algorithm (NINV) for generalized inverses, which takes actuator constraints into account, is introduced in this chapter. It outperforms the existing approach for small and medium problem sizes in terms of computation time. Beside its speed one of the main advantages of this method is the fixed computation time which facilitates an implementation on an electronic control unit. Another benefit is the lack of tuning parameters which enables the usage of this algorithm without extensive adjustments on the problem. The search process among the candidates consists of numerous independent calculations. Therefore, further speedup could be achieved by parallelization. Moreover, a separation of the algorithm in offline and online parts could reduce the computation time considerably [25].

5. Enhanced Redistributed Pseudoinverse

The factorization impact on Redistributed Pseudoinverse (RPINV) is reduced to the number of active actuator constraints in Section 3.6.1. It has been proven that in general the rank-reduction of the pseudoinverse entails deviating allocation results for different factorizations. Above a certain number of active actuator constraints the desired control effort is generally not reached by RPINV. In order to address this issue an enhanced version of that algorithm is proposed. Therein one can specify allocation error components which should preferably vanish. This objective is met by a change of factorization during execution [30].

5.1. Optimal factorization

In accordance with Section 3.6 the number of free actuators during one RPINV iteration is labeled as j. Suppose $j < k$ and $\boldsymbol{W} = \boldsymbol{I_m}$ (see Lemma 3.6.1). Due to $\boldsymbol{B}^{\#}$ lacking full column rank $\boldsymbol{v_{des}}$ will generally not be reached any more because $\boldsymbol{B_0}\boldsymbol{B}^{\#} \neq \boldsymbol{I_k}$. Of course this raises the question whether there is an optimal factorization of $\boldsymbol{B_u}$ which makes the error between the desired virtual controls $\boldsymbol{v_{des}}$ and the actually achieved $\boldsymbol{v_{act}}$ as small as possible [30]. Using (2.20) this error can be written as

$$\boldsymbol{v_{des}} - \boldsymbol{v_{act}} = \left(\boldsymbol{I_k} - \boldsymbol{B_0}\boldsymbol{B}^{\#}\right)\left(\boldsymbol{v_{des}} + \boldsymbol{B_0}\boldsymbol{c}\right). \tag{5.1}$$

The dependence on the transformation parameters is introduced into (5.1) via (3.11) and (3.21) which yields

$$\begin{aligned} \overline{\boldsymbol{e}}_{\boldsymbol{v}} &= \left(\boldsymbol{I_k} - \boldsymbol{T}^{-1}\boldsymbol{B_0}\boldsymbol{B}^{\#}_{\boldsymbol{T}}\right)\boldsymbol{T}^{-1}\left(\boldsymbol{v_{des}} + \boldsymbol{B_0}\boldsymbol{c}\right) \\ &= \boldsymbol{T}^{-1}\left(\boldsymbol{I_k} - \boldsymbol{B_0}\boldsymbol{B}^{\#}_{\boldsymbol{T}}\boldsymbol{T}^{-1}\right)\left(\boldsymbol{v_{des}} + \boldsymbol{B_0}\boldsymbol{c}\right) \end{aligned} \tag{5.2}$$

with short notation $\boldsymbol{B}^{\#}_{\boldsymbol{T}}$ for (3.57), i.e.

$$\boldsymbol{B}^{\#}_{\boldsymbol{T}} = \boldsymbol{R}\left(\widetilde{\boldsymbol{B}}^{T}\boldsymbol{T}^{-T}\boldsymbol{T}^{-1}\widetilde{\boldsymbol{B}}\right)^{-1}\widetilde{\boldsymbol{B}}^{T}\boldsymbol{T}^{-T}. \tag{5.3}$$

An intuitive problem formulation could be

$$\min_{\boldsymbol{T}} \|\overline{\boldsymbol{e}}_{\boldsymbol{v}}\| = \|\boldsymbol{T}^{-1}\left(\boldsymbol{I_k} - \boldsymbol{B_0}\boldsymbol{B}^{\#}_{\boldsymbol{T}}\boldsymbol{T}^{-1}\right)\left(\boldsymbol{v_{des}} + \boldsymbol{B_0}\boldsymbol{c}\right)\| \tag{5.4a}$$

$$\boldsymbol{u_{min}} \leq -\boldsymbol{c} + \boldsymbol{B}^{\#}_{\boldsymbol{T}}\boldsymbol{T}^{-1}\left(\boldsymbol{v_{des}} + \boldsymbol{B_0}\boldsymbol{c}\right) \leq \boldsymbol{u_{max}} \tag{5.4b}$$

where it is recognizable that this cost function can be made arbitrarily small by simple scaling. Furthermore, small virtual error components can have a great impact on the plant dynamics if the corresponding entries of the virtual input matrix $\boldsymbol{B_v}$ are sufficiently large. Therefore, it is more reasonable to minimize the deviation between desired and actual effect on the plant [30]:

$\boldsymbol{e_v} = \boldsymbol{B_v}\left(\boldsymbol{v_{des}} - \boldsymbol{v_{act}}\right)$. Considering (3.10) results in the multiplication of (5.2) with $\boldsymbol{B_v T}$ and one can formulate the optimization problem [30]

$$\min_{\boldsymbol{T}} \left\|\boldsymbol{e_v}\right\| = \min_{\boldsymbol{T}} \left\|\boldsymbol{M_T}\underbrace{\left(\boldsymbol{v_{des}} + \boldsymbol{B_0 c}\right)}_{\tilde{\boldsymbol{v}}}\right\| \tag{5.5a}$$

$$\boldsymbol{M_T} = \boldsymbol{B_v T}\left(\boldsymbol{I_k} - \boldsymbol{T}^{-1}\boldsymbol{B_0 B_T^{\#}}\right)\boldsymbol{T}^{-1} \tag{5.5b}$$

$$\boldsymbol{u_{min}} \leq -\boldsymbol{c} + \boldsymbol{B_T^{\#}}\boldsymbol{T}^{-1}\left(\boldsymbol{v_{des}} + \boldsymbol{B_0 c}\right) \leq \boldsymbol{u_{max}} \tag{5.5c}$$

If $\tilde{\boldsymbol{v}} \in \mathcal{N}_{\mathrm{r}}\left(\boldsymbol{M_T}\right)$, the desired value $\boldsymbol{v_{des}}$ could be reached, although $\boldsymbol{B_0 B^{\#}} \neq \boldsymbol{I_k}$ provided that $\boldsymbol{v_{des}} \in \Phi$. It turns out that both $\boldsymbol{M_T}$ (because of $\widetilde{\boldsymbol{B}}$ in (5.3)) and $\tilde{\boldsymbol{v}}$ depend on $\boldsymbol{v_{des}}$ and so there is no universally best factorization that could be computed offline. Problem (5.5) is a nonlinear constrained optimization problem which makes online solving quite sophisticated. RPINV is intended to be a low-effort method for considering actuator constraints and so this is not a reasonable option [30].

5.2. Effect prioritization

Instead of minimizing the total effect error $\boldsymbol{e_v}$ it is easier to focus on certain components. The basic idea of the approach is to exploit the rank deficiency of $\boldsymbol{M_T}$ by introducing zero rows. In this subsection $n = k$ and w.l.o.g. $\boldsymbol{B_v} = \boldsymbol{I_k}$ are assumed and therefore (5.5b) simplifies to [30]

$$\boldsymbol{M_T} = \boldsymbol{I_k} - \boldsymbol{B_0 B_T^{\#}}\boldsymbol{T}^{-1}. \tag{5.6}$$

Lemma 5.2.1. *For all $j < k$ and all invertible transformations $\boldsymbol{T} \in \mathbb{R}^{k\times k}$ the rank of (5.6) is $k - j$ [30].*

Proof. Define $\widetilde{\boldsymbol{B}}_{\boldsymbol{T}} = \boldsymbol{T}^{-1}\widetilde{\boldsymbol{B}}$ and (A.3) in Appendix A.2 yields $\operatorname{rank}(\widetilde{\boldsymbol{B}}_{\boldsymbol{T}}) = j$. According to (3.55) one obtains $\boldsymbol{B_T^{\#}} = \boldsymbol{R}\widetilde{\boldsymbol{B}}_{\boldsymbol{T}}^{\dagger}$. From Lemma 1 of [27] it follows that $\operatorname{rank}(\widetilde{\boldsymbol{B}}_{\boldsymbol{T}}^{\dagger}) = \operatorname{rank}(\widetilde{\boldsymbol{B}}_{\boldsymbol{T}}) = j$ and repeatedly applying (A.3) results in [30]

$$j = \operatorname{rank}(\boldsymbol{R}\widetilde{\boldsymbol{B}}_{\boldsymbol{T}}^{\dagger}) = \operatorname{rank}(\boldsymbol{B_T^{\#}}\boldsymbol{T}^{-1}). \tag{5.7}$$

Note that $\boldsymbol{B_0 B_T^{\#}}\boldsymbol{T}^{-1} = \widetilde{\boldsymbol{B}}\widetilde{\boldsymbol{B}}_{\boldsymbol{T}}^{\dagger}\boldsymbol{T}^{-1}$ and once again (A.3) reveals that

$$\operatorname{rank}(\boldsymbol{B_0 B_T^{\#}}\boldsymbol{T}^{-1}) = j. \tag{5.8}$$

Appendix A.3 provides the rank of (5.6) by means of (A.4). Recall that $\boldsymbol{B_0 B_T^{\#}} = \widetilde{\boldsymbol{B}}\widetilde{\boldsymbol{B}}_{\boldsymbol{T}}^{\dagger}$ and $\boldsymbol{I_k^{\dagger}} = \boldsymbol{I_k}$. Thus $\boldsymbol{B_0 B_T^{\#}}\boldsymbol{T}^{-1}\boldsymbol{I_k^{\dagger}}\boldsymbol{B_0 B_T^{\#}}\boldsymbol{T}^{-1} = \widetilde{\boldsymbol{B}}\underbrace{\widetilde{\boldsymbol{B}}_{\boldsymbol{T}}^{\dagger}\boldsymbol{T}^{-1}\widetilde{\boldsymbol{B}}}_{\boldsymbol{I_j}}\widetilde{\boldsymbol{B}}_{\boldsymbol{T}}^{\dagger}\boldsymbol{T}^{-1} = \boldsymbol{B_0 B_T^{\#}}\boldsymbol{T}^{-1}$.

Let $\overline{\boldsymbol{C}} = [\boldsymbol{I_k} \;\; \boldsymbol{B_0 B_T^{\#}}\boldsymbol{T}^{-1}] \in \mathbb{R}^{k\times 2k}$, $\underline{\boldsymbol{C}} = \begin{bmatrix}\boldsymbol{I_k}\\ \boldsymbol{B_0 B_T^{\#}}\boldsymbol{T}^{-1}\end{bmatrix} \in \mathbb{R}^{2k\times k}$. Due to $\operatorname{rank}(\boldsymbol{I_k}) = k$ it follows that $\overline{\boldsymbol{C}}$ has full row rank and $\underline{\boldsymbol{C}}$ full column rank. Consequently, (5.8) and (A.4) yields $\operatorname{rank}(\boldsymbol{I_k} - \boldsymbol{B_0 B_T^{\#}}\boldsymbol{T}^{-1}) = \operatorname{rank}(\boldsymbol{I_k}) - \operatorname{rank}(\boldsymbol{B_0 B_T^{\#}}\boldsymbol{T}^{-1}) = k - j$ [30]. ■

Theorem 5.2.1. *Let $\widetilde{\boldsymbol{B}}_{\boldsymbol{j}} \in \mathbb{R}^{j\times j}$ be an invertible submatrix of $\widetilde{\boldsymbol{B}}$ consisting of rows with indices $\{r_1, ..., r_j\}$. Then there exist invertible matrices $\boldsymbol{T} \in \mathbb{R}^{k\times k}$ which transform (5.6) such that its rows with indices $\{r_1, ..., r_j\}$ contain only zeros [30].*

Proof. $\operatorname{rank}(\boldsymbol{M_T}) = k - j$ due to Lemma 5.2.1 which means it can contain up to j zero rows. W.l.o.g. it is assumed that the first j rows of (5.6) should be zeroed. Using the partitioned matrices [30]

$$\widetilde{\boldsymbol{B}} = \begin{bmatrix} \widetilde{\boldsymbol{B}}_{\boldsymbol{j}} \\ \widetilde{\boldsymbol{B}}_{\boldsymbol{k-j}} \end{bmatrix} \text{ with } \widetilde{\boldsymbol{B}}_{\boldsymbol{j}} \in \mathbb{R}^{j\times j} \text{ and } \widetilde{\boldsymbol{B}}_{\boldsymbol{k-j}} \in \mathbb{R}^{(k-j)\times j} \tag{5.9}$$

and

$$\widetilde{\boldsymbol{B}}^{\dagger}_{\boldsymbol{T}}\boldsymbol{T}^{-1} = \begin{bmatrix} \widetilde{\boldsymbol{A}}_{\boldsymbol{1}} & \widetilde{\boldsymbol{A}}_{\boldsymbol{2}} \end{bmatrix} \text{ with } \widetilde{\boldsymbol{A}}_{\boldsymbol{1}} \in \mathbb{R}^{j\times j} \text{ and } \widetilde{\boldsymbol{A}}_{\boldsymbol{2}} \in \mathbb{R}^{j\times(k-j)} \tag{5.10}$$

the first j rows of $\boldsymbol{M_T} = \boldsymbol{I_k} - \widetilde{\boldsymbol{B}}\widetilde{\boldsymbol{B}}^{\dagger}_{\boldsymbol{T}}\boldsymbol{T}^{-1}$ read as

$$\boldsymbol{0}_{\boldsymbol{j\times k}} \overset{!}{=} \begin{bmatrix} \boldsymbol{I_j} - \widetilde{\boldsymbol{B}}_{\boldsymbol{j}}\widetilde{\boldsymbol{A}}_{\boldsymbol{1}} & \widetilde{\boldsymbol{B}}_{\boldsymbol{j}}\widetilde{\boldsymbol{A}}_{\boldsymbol{2}} \end{bmatrix} \tag{5.11}$$

where $\boldsymbol{0}_{\boldsymbol{j\times k}} \in \mathbb{R}^{j\times k}$ is a zero-matrix. It follows that $\widetilde{\boldsymbol{A}}_{\boldsymbol{1}} = \widetilde{\boldsymbol{B}}_{\boldsymbol{j}}^{-1}$ and $\widetilde{\boldsymbol{A}}_{\boldsymbol{2}} = \boldsymbol{0}_{\boldsymbol{j\times(k-j)}}$. Defining $\overline{\boldsymbol{T}} = \boldsymbol{T}^{-1}$ one obtains [30]

$$\widetilde{\boldsymbol{B}}^{\dagger}_{\boldsymbol{T}}\overline{\boldsymbol{T}} = \begin{bmatrix} \widetilde{\boldsymbol{B}}_{\boldsymbol{j}}^{-1} & \boldsymbol{0}_{\boldsymbol{j\times(k-j)}} \end{bmatrix} = \left(\widetilde{\boldsymbol{B}}^{T}\overline{\boldsymbol{T}}^{T}\overline{\boldsymbol{T}}\widetilde{\boldsymbol{B}}\right)^{-1}\widetilde{\boldsymbol{B}}^{T}\overline{\boldsymbol{T}}^{T}\overline{\boldsymbol{T}}. \tag{5.12}$$

After partitioning $\overline{\boldsymbol{T}} = [\overline{\boldsymbol{T}}_{\boldsymbol{j}} \;\; \overline{\boldsymbol{T}}_{\boldsymbol{k-j}}]$ with $\overline{\boldsymbol{T}}_{\boldsymbol{j}} \in \mathbb{R}^{k\times j}$ and $\overline{\boldsymbol{T}}_{\boldsymbol{k-j}} \in \mathbb{R}^{k\times(k-j)}$ one can split (5.12) into two equations. One of them is [30]

$$\boldsymbol{0}_{\boldsymbol{j\times(k-j)}} = \underbrace{\left(\widetilde{\boldsymbol{B}}^{T}\overline{\boldsymbol{T}}^{T}\overline{\boldsymbol{T}}\widetilde{\boldsymbol{B}}\right)^{-1}\widetilde{\boldsymbol{B}}^{T}\overline{\boldsymbol{T}}^{T}}_{\widetilde{\boldsymbol{B}}^{\dagger}_{\boldsymbol{T}}}\overline{\boldsymbol{T}}_{\boldsymbol{k-j}} \tag{5.13}$$

where $\operatorname{rank}(\widetilde{\boldsymbol{B}}^{\dagger}_{\boldsymbol{T}}) = j$ (see Lemma 5.2.1). Hence, the bracket term in (5.13) is full rank. Considering the left nullspace $\mathcal{N}_l(\widetilde{\boldsymbol{B}}^{T}) = \boldsymbol{0}^{T}$ and partitioning of $\widetilde{\boldsymbol{B}}^{T}$ and $\overline{\boldsymbol{T}}^{T}$, equation (5.13) simplifies to [30]

$$\boldsymbol{0}_{\boldsymbol{j\times(k-j)}} = \begin{bmatrix} \widetilde{\boldsymbol{B}}_{\boldsymbol{j}}^{T} & \widetilde{\boldsymbol{B}}_{\boldsymbol{k-j}}^{T} \end{bmatrix} \begin{bmatrix} \overline{\boldsymbol{T}}_{\boldsymbol{j}}^{T} \\ \overline{\boldsymbol{T}}_{\boldsymbol{k-j}}^{T} \end{bmatrix} \overline{\boldsymbol{T}}_{\boldsymbol{k-j}}. \tag{5.14}$$

Rewriting (5.14) yields

$$\boldsymbol{0}_{\boldsymbol{j\times(k-j)}} = \underbrace{\left(\widetilde{\boldsymbol{B}}_{\boldsymbol{j}}^{T}\overline{\boldsymbol{T}}_{\boldsymbol{j}}^{T} + \widetilde{\boldsymbol{B}}_{\boldsymbol{k-j}}^{T}\overline{\boldsymbol{T}}_{\boldsymbol{k-j}}^{T}\right)}_{\in\,\mathbb{R}^{j\times k}}\overline{\boldsymbol{T}}_{\boldsymbol{k-j}} \tag{5.15}$$

Note that $\dim\left[\mathcal{N}_l(\overline{\boldsymbol{T}}_{\boldsymbol{k-j}})\right] = j$, which coincides with the number of rows in the bracket term of (5.15). Thus, one can choose an arbitrary $\overline{\boldsymbol{T}}_{\boldsymbol{k-j}}$ with $\operatorname{rank}(\overline{\boldsymbol{T}}_{\boldsymbol{k-j}}) = k - j$ and determine a matrix $\boldsymbol{N_T} \in \mathbb{R}^{j\times k}$ such that $\operatorname{span}(\boldsymbol{N_T}) = \mathcal{N}_l(\overline{\boldsymbol{T}}_{\boldsymbol{k-j}})$. Finally, the remaining columns of $\overline{\boldsymbol{T}}$ are given by [30]

$$\overline{\boldsymbol{T}}_{\boldsymbol{j}} = \left(\boldsymbol{N_T} - \widetilde{\boldsymbol{B}}_{\boldsymbol{k-j}}^{T}\overline{\boldsymbol{T}}_{\boldsymbol{k-j}}^{T}\right)^{T}\widetilde{\boldsymbol{B}}_{\boldsymbol{j}}^{-1}. \tag{5.16}$$

Now that $\overline{\boldsymbol{T}}$ is completely determined it must be shown that it satisfies

$$\widetilde{\boldsymbol{B}}_{\boldsymbol{j}}^{-1} = \left(\widetilde{\boldsymbol{B}}^{T}\overline{\boldsymbol{T}}^{T}\overline{\boldsymbol{T}}\widetilde{\boldsymbol{B}}\right)^{-1}\widetilde{\boldsymbol{B}}^{T}\overline{\boldsymbol{T}}^{T}\overline{\boldsymbol{T}}_{\boldsymbol{j}}, \tag{5.17}$$

i.e. the second matrix equation in (5.12). Finally, please observe that the last $k-j$ columns of $\widetilde{\boldsymbol{B}}^T\overline{\boldsymbol{T}}^T\left[\overline{\boldsymbol{T}}_{\boldsymbol{j}}\ \overline{\boldsymbol{T}}_{\boldsymbol{k-j}}\right]$ are all zeros due to (5.14). Hence it follows that

$$\widetilde{\boldsymbol{B}}^T\overline{\boldsymbol{T}}^T\overline{\boldsymbol{T}}\widetilde{\boldsymbol{B}} = \widetilde{\boldsymbol{B}}^T\overline{\boldsymbol{T}}^T\overline{\boldsymbol{T}}_{\boldsymbol{j}}\widetilde{\boldsymbol{B}}_{\boldsymbol{j}} \tag{5.18}$$

and inserted into (5.17) results indeed in [30]

$$\left(\widetilde{\boldsymbol{B}}^T\overline{\boldsymbol{T}}^T\overline{\boldsymbol{T}}_{\boldsymbol{j}}\widetilde{\boldsymbol{B}}_{\boldsymbol{j}}\right)^{-1}\widetilde{\boldsymbol{B}}^T\overline{\boldsymbol{T}}^T\overline{\boldsymbol{T}}_{\boldsymbol{j}} = \widetilde{\boldsymbol{B}}_{\boldsymbol{j}}^{-1}\left(\widetilde{\boldsymbol{B}}^T\overline{\boldsymbol{T}}^T\overline{\boldsymbol{T}}_{\boldsymbol{j}}\right)^{-1}\widetilde{\boldsymbol{B}}^T\overline{\boldsymbol{T}}^T\overline{\boldsymbol{T}}_{\boldsymbol{j}} = \widetilde{\boldsymbol{B}}_{\boldsymbol{j}}^{-1}. \tag{5.19}$$

■

Based on Theorem 5.2.1 an Enhanced Redistributed Pseudoinverse (ERPINV) algorithm is proposed. This RPINV extension supports the specification of a priority list containing the numbers of $k-1$ components of $\boldsymbol{e_v}$ which should be made zero. As soon as $j<k$ it takes the following measures [30]:

5.2.1. Enhanced Redistributed Pseudoinverse

1. The priority list items $1,...,j$ determine the rows of $\widetilde{\boldsymbol{B}}$ which form $\widetilde{\boldsymbol{B}}_{\boldsymbol{j}}$. The remaining rows of $\widetilde{\boldsymbol{B}}$ are called $\widetilde{\boldsymbol{B}}_{\boldsymbol{k-j}} \in \mathbb{R}^{(k-j)\times j}$.
2. If $\det(\widetilde{\boldsymbol{B}}_{\boldsymbol{j}}) = 0$ then abort and continue with ordinary RPINV calculations.
3. Choose an *arbitrary* matrix $\overline{\boldsymbol{T}}_{\boldsymbol{k-j}}$ with $\operatorname{rank}(\overline{\boldsymbol{T}}_{\boldsymbol{k-j}}) = k-j$ and compute a basis of its left nullspace $\boldsymbol{N_T} \in \mathbb{R}^{j\times k}$.
4. Evaluate $\overline{\boldsymbol{T}}_{\boldsymbol{j}} = \left(\boldsymbol{N_T} - \widetilde{\boldsymbol{B}}_{\boldsymbol{k-j}}^T\overline{\boldsymbol{T}}_{k-j}^T\right)^T\widetilde{\boldsymbol{B}}_{\boldsymbol{j}}^{-1}$ to get the inverse transformation $\boldsymbol{T}^{-1} = \overline{\boldsymbol{T}} = [\overline{\boldsymbol{T}}_{\boldsymbol{j}}\ \overline{\boldsymbol{T}}_{\boldsymbol{k-j}}]$.
5. Use $\overline{\boldsymbol{T}}$ to transform $\boldsymbol{B}$, $\boldsymbol{B_0}$, and $\boldsymbol{v_{des}}$ via (3.11) and (3.21). Now, (2.19) is reformulated as
$$\hat{\boldsymbol{u}}^N = -\boldsymbol{c}^N + \left(\overline{\boldsymbol{T}}\boldsymbol{B_m}\right)^{\#}\overline{\boldsymbol{T}}\left(\boldsymbol{v}+\boldsymbol{B}\boldsymbol{c}^N\right). \tag{5.20}$$
6. If $\hat{\boldsymbol{u}}^N$ contains no new constraint violations or all actuators are saturated: STOP.
7. Otherwise, update the offset vector $\boldsymbol{c}^N$ as well as $\boldsymbol{B_m}$ and go to step (1) again.

Note that these extensions do not require any modifications of the controller as the desired virtual control vector is transformed according to the new factorization [30].

Remark 5.2.1. Apparently zeroing components of (5.6) only works if the resulting $\boldsymbol{u}^N \in \Omega$ because otherwise (2.20) is not fulfilled which is the basis for the derivation of (5.6) [30].

5.3. Simulation results

This section demonstrates the results for ERPINV. The control goal is to stabilize the origin $\boldsymbol{x}=0$. The state space representation of the plant reads as [30]

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 2 & 0 \\ -1 & 1 & 2 \end{bmatrix}\boldsymbol{x} + \underbrace{\begin{bmatrix} -6 & 1 & 1 & 10 \\ -4 & 0 & 9 & -5 \\ -1 & 8 & 3 & 4 \end{bmatrix}}_{\boldsymbol{B_u}}\boldsymbol{u} \tag{5.21}$$

and the input constraints are $-\boldsymbol{u_{max}} \leq \boldsymbol{u} \leq \boldsymbol{u_{max}}$ with $\boldsymbol{u}^T_{\boldsymbol{max}} = [1\ 13\ 13\ 12]$. Three factorizations of $\boldsymbol{B_u}$ are tested:

Factorization 1:

$$\boldsymbol{B_{v1}} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{2}{3} & -\frac{4}{47} & 1 \\ \frac{1}{6} & 1 & 0 \end{bmatrix} \qquad \boldsymbol{B_1} = \begin{bmatrix} -6 & 1 & 1 & 10 \\ 0 & \frac{47}{6} & \frac{17}{6} & \frac{7}{3} \\ 0 & 0 & \frac{403}{47} & -\frac{539}{47} \end{bmatrix}$$

Factorization 2:

$$\boldsymbol{B_{v2}} = \begin{bmatrix} -7 & 4 & 4 \\ -\frac{142}{141} & \frac{1516}{141} & \frac{139}{141} \\ \frac{17}{6} & -\frac{1}{3} & \frac{26}{3} \end{bmatrix} \qquad \boldsymbol{B_2} = \begin{bmatrix} \frac{14}{27} & \frac{73}{243} & \frac{5447}{11421} & -\frac{15022}{11421} \\ -\frac{8}{27} & -\frac{23}{486} & \frac{19673}{22842} & -\frac{7625}{11421} \\ -\frac{8}{27} & \frac{200}{243} & \frac{2551}{11421} & \frac{9889}{11421} \end{bmatrix}$$

Factorization 3:

$$\boldsymbol{B_{v3}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{B_3} = \begin{bmatrix} -6 & 1 & 1 & 10 \\ -4 & 0 & 9 & -5 \\ -1 & 8 & 3 & 4 \end{bmatrix}$$

The transformation matrix describing the relationship between factorizations 1 and 2 has orthogonal columns and reads as

$$\boldsymbol{T_{12}} = \begin{bmatrix} -7 & 4 & 4 \\ 4 & -1 & 8 \\ 4 & 8 & -1 \end{bmatrix}. \tag{5.22}$$

The weighting matrix is $\boldsymbol{W} = \mathrm{diag}(\frac{1}{u_{max,1}}, ..., \frac{1}{u_{max,4}})$. A linear state-controller is chosen for stabilization. For each factorization the controller gain matrix $\boldsymbol{K_i}$ is chosen in order to place all eigenvalues of the closed-loop system matrices $(\boldsymbol{A} - \boldsymbol{B_{vi}K_i})$ for $i = 1, 2, 3$ at -12. The initial state at $t = 0$ is $\boldsymbol{x_0^T} = [-7\ \ -2\ \ \ 14]$. In Figures 5.1 and 5.2 one can see a different behavior depending on the factorization because the number of free controls is smaller than $k = 3$. Due to the special transformation matrix $\boldsymbol{T_{12}}$ there is no deviation between factorizations 1 and 2 [30]. Figures 5.3 and 5.4 show desired and actually achieved effect on the plant as well as the effect error $\boldsymbol{e_v}$ for RPINV and ERPINV respectively. Initially ERPINV uses factorization 3 because $\boldsymbol{B_{v3}} = \boldsymbol{I_3}$ is required. It is configured to prefer vanishing error components $e_{v,1}$ and $e_{v,2}$ as recognizable in Figure 5.4. This also influences the performance in bringing the corresponding state variables to zero as illustrated by the mean squared errors (MSE) in Figure 5.1. The required transformation matrix is computed online and depends on which actuators saturate. In this example an approximation of one possible result is [30]

$$\boldsymbol{T}^{-1} \approx \begin{bmatrix} -\frac{3155}{10844} & -\frac{5680}{33393} & \frac{1190}{3253} \\ -\frac{3397}{8324} & -\frac{1410}{9977} & \frac{16862}{22085} \\ -\frac{8543}{30064} & -\frac{4075}{19103} & \frac{9647}{15364} \end{bmatrix}. \tag{5.23}$$

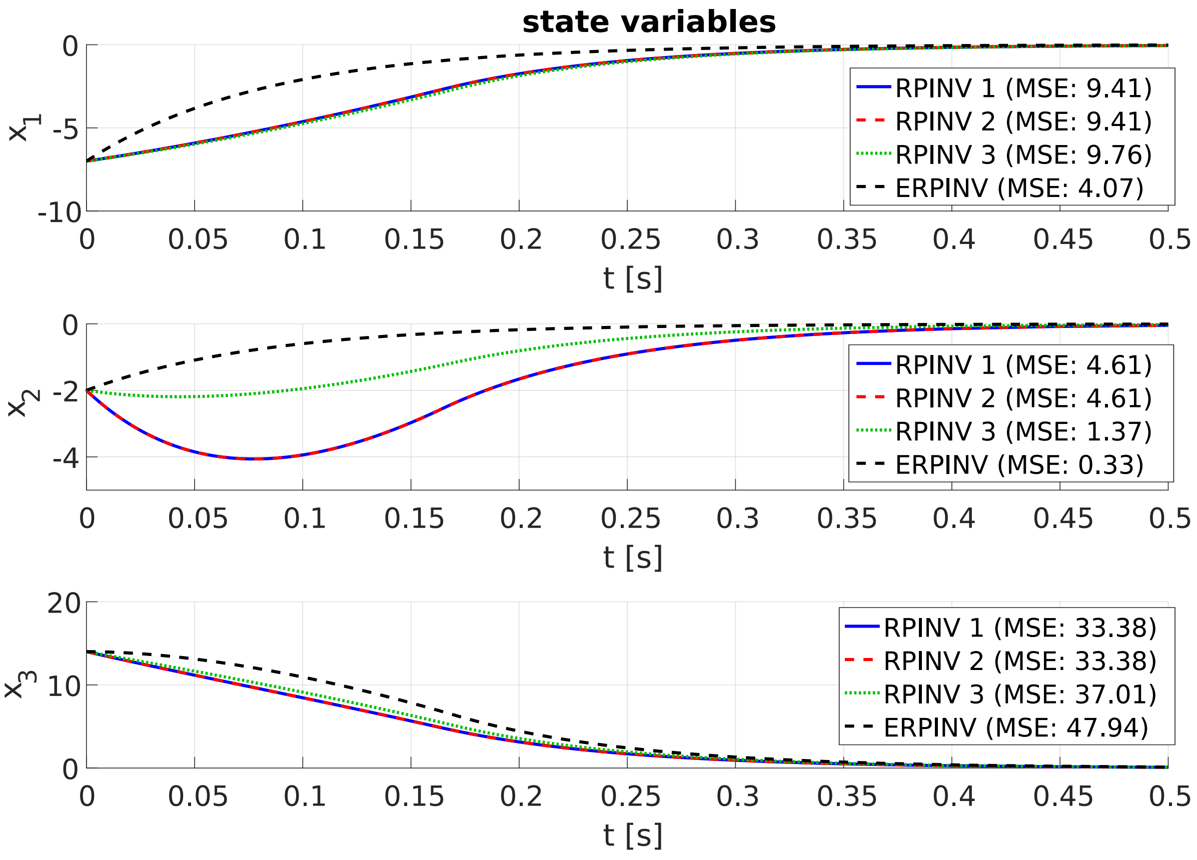

Figure 5.1.: States over time using RPINV for factorizations 1 - 3 and ERPINV. (© 2018 IEEE, [30]).

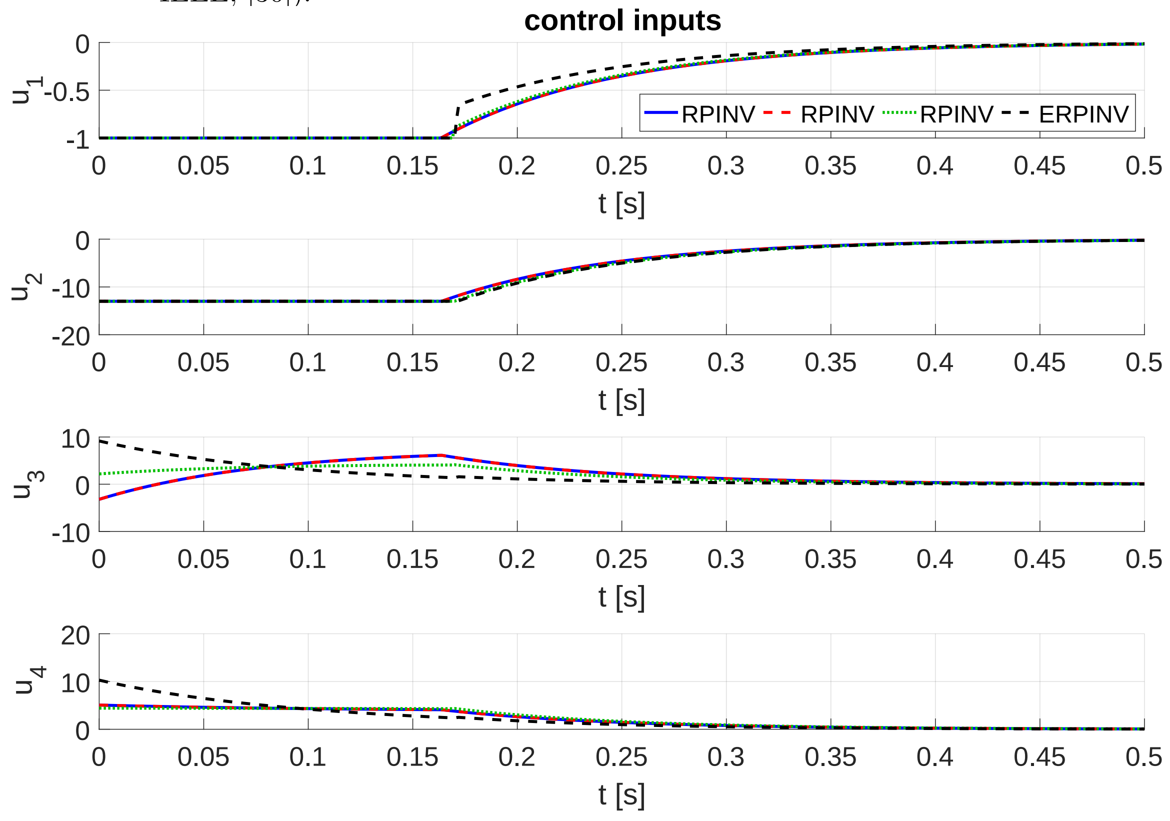

Figure 5.2.: Inputs over time using RPINV for factorizations 1 - 3 and ERPINV. (© 2018 IEEE, [30]).

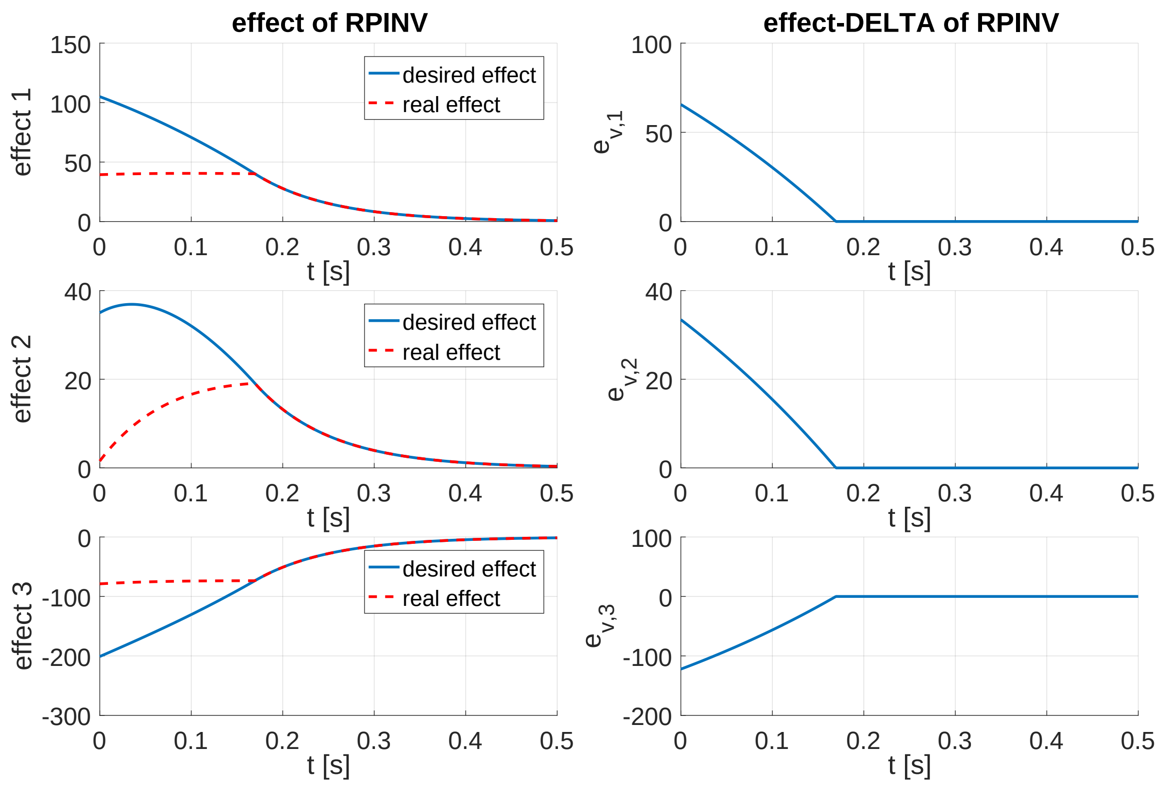

Figure 5.3.: Desired and achieved effect on plant for RPINV with factorization 3. (© 2018 IEEE, [30]).

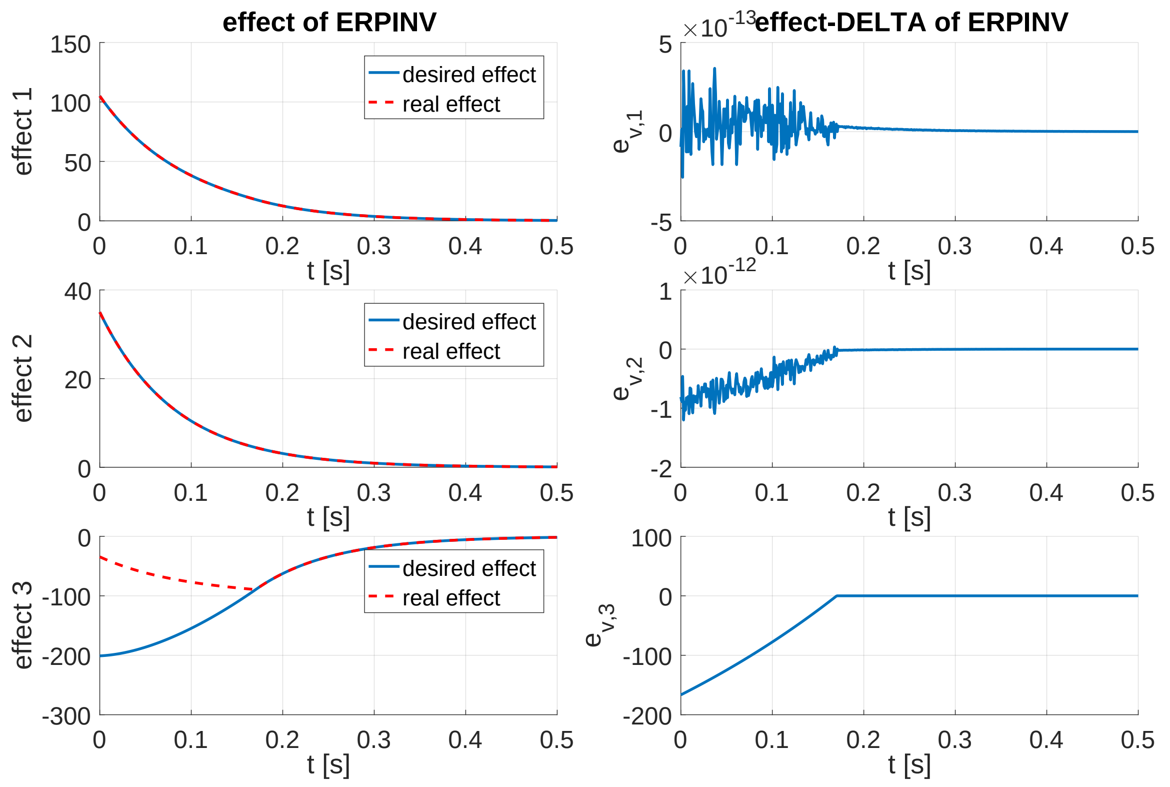

Figure 5.4.: Desired and achieved effect on plant for ERPINV with prioritization of $e_{v,1}$ and $e_{v,2}$. (© 2018 IEEE, [30]).

5.4. Conclusion

In this chapter it is shown that by changing the factorization online depending on the desired virtual control one could reduce the potential virtual control error originating from the rank-reduction of the pseudoinverse. However, the complex structure of the optimization problem would stand in contrast to the simplicity of RPINV. Rather than solving the optimization problem (5.5) online it is possible to make a certain number of error components zero (ER-PINV) depending on the rank of the reduced pseudoinverse. This is accomplished by a change of input matrix factorization, which is completely transparent to the controller. It allows a prioritization of error components in case of a rank deficient pseudoinverse originating from actuator saturations [30].

6. Exterior Point Algorithm[1]

Several numerical optimization methods have already been successfully applied to solve the CA problem [6] [31] [33] [35]. A popular problem formulation yields a constrained QP which is solved in each execution step of the controller. The fixed-point method [34] is conceptually simple and easy to implement. It finds approximate solutions and works well if the optimum is not too close to the boundary of the feasible region. Alternatives are active-set strategy based algorithms. They are computationally very efficient for small problem sizes, but do not scale very well with an increasing number of actuators. A great advantage of active-set strategies is their hot-start ability, i.e. an initial guess of the working set from previous execution cycles can be used [6]. In contrast, interior-point methods have more computational overhead than active-set algorithms at lower problem dimensions whereas they outperform them as the number of variables grows. The algorithm presented in this chapter is based on the penalty function method [45] [46] and gradient projection [32]. It restates the constrained optimization problem to an unconstrained one and generates a sequence of exterior points that converges to the solution of the original problem if it is consistent. The proposed Exterior Point Algorithm (EPA) shares some similarities with [47] but there are also some important differences. First of all, it is specially designed not only to handle inequality-type but also equality-type constraints. A crucial point of algorithms based on penalty functions is the adaption of the penalty parameter during the iterations. In [47] this is done by multiplication with a constant factor whereas the proposed approach takes the progress towards the feasible region into account. The resulting algorithm scales very well with problem dimensions which makes it applicable not only for CA but also in domains where larger problems arise. The general benefits of the proposed algorithm are its fast computation time and its effectiveness at problems with dense weighting matrices. Advantages which are especially useful in a CA context are the efficient way of handling equality constraints and the fact that the algorithm features hot-start ability, i.e. its computation time can be reduced by taking previous result into account.

6.1. Problem statement

The relationship between virtual and real controls is described by the under-determined system of equations (2.2) which has to be solved for $\boldsymbol{u}$ by CA. Consequently, the nullspace of $\boldsymbol{B}$ is non-trivial and its dimension $\dim[\mathcal{N}_\mathrm{r}(\boldsymbol{B})] = m - k$ follows from the rank-nullity theorem [24]. Typically, actuator constraints (1.31) limit the range of values that the components of $\boldsymbol{u}$ can take to a subset of the m-dimensional (m-D) control space. The projection of this m-D hyperbox into virtual control space $\mathbb{R}^k$ is the AMS given by (1.34). It contains all vectors $\boldsymbol{v}$ which can be achieved by feasible $\boldsymbol{u} \in \Omega$. Considering (2.2), (1.31), and $\mathrm{rank}(\boldsymbol{B}) = k$ it is possible that there is no, exactly one, or an infinite number of valid control vectors $\boldsymbol{u}$ that map to a specific vector $\boldsymbol{v}$. In the latter case, secondary objectives can be incorporated leading to a

[1]Parts of this chapter have been submitted to the 57th IEEE Conference on Decision and Control. At print time the decision has been pending.

notation of the CA task as an optimization problem. A commonly used formulation is known as *Weighted Least Squares* (WLS) method (see Section 2.1.6 and [6]) and reads as

$$\min_{\boldsymbol{u}} \; \|\boldsymbol{W_u}\left(\boldsymbol{u}-\boldsymbol{u_p}\right)\|_2^2 + \gamma\|\boldsymbol{W_v}\left(\boldsymbol{B}\boldsymbol{u}-\boldsymbol{v}\right)\|_2^2 \tag{6.1a}$$

$$\text{subject to } \; \boldsymbol{u}\in\Omega \tag{6.1b}$$

with $\boldsymbol{W_u}\in\mathbb{R}^{m\times m}$ and $\boldsymbol{W_v}\in\mathbb{R}^{k\times k}$ being full rank weighting matrices, $\boldsymbol{u_p}\in\mathbb{R}^m$ containing the preferred actuator positions, and $\gamma>0$ being a weighting factor. Instead of explicitly including (2.2) as equation-type constraints it is part of the cost function and for large values of γ one can expect the solution of (6.1) $\boldsymbol{u}^*_{\boldsymbol{W}}$ to comply with it. Note that with $\boldsymbol{u_p}=\boldsymbol{0}$ and $\boldsymbol{W_u}=\boldsymbol{I_m}$ denoting the m-D identity matrix (6.1) seeks to minimize the actuator energy consumption.

6.1.1. Feasible virtual controls

Assuming $\boldsymbol{v}\in\Phi$ the second weighting term in (6.1a) is zero at the minimum and (6.1) can be rewritten as

$$\min_{\boldsymbol{u}} \; \underbrace{\boldsymbol{u}^T\boldsymbol{W}\boldsymbol{u}+\boldsymbol{u}^T\boldsymbol{c_u}}_{f_u(\boldsymbol{u})} \tag{6.2a}$$

$$\text{subject to } \; \boldsymbol{v}=\boldsymbol{B}\boldsymbol{u} \tag{6.2b}$$

$$\text{and } \; \boldsymbol{u}\in\Omega \tag{6.2c}$$

with positive definite $\boldsymbol{W}=\boldsymbol{W_u}^T\boldsymbol{W_u}$, $\boldsymbol{c_u}=-2\boldsymbol{W}\boldsymbol{u_p}$ and neglecting the constant offset $\boldsymbol{u_p}^T\boldsymbol{W}\boldsymbol{u_p}$ in (6.2a). Because of $\boldsymbol{\mathcal{H}}_{f_u}=2\boldsymbol{W}\succ 0$ on $\mathbb{R}^m$ the cost function $f_u(\boldsymbol{u})$ is strictly convex. Since (6.2b) and (6.2c) describe convex subsets of $\mathbb{R}^m$ problems (6.1) and (6.2) have a common strict global minimizer [45] denoted as $\boldsymbol{u}^*$.

6.1.2. Infeasible virtual controls

An infeasible virtual control vector means that $\boldsymbol{v}\notin\Phi \;\Leftrightarrow\; \nexists\boldsymbol{u}\in\Omega:\boldsymbol{v}=\boldsymbol{B}\boldsymbol{u}$. Due to the fact that WLS incorporates hard actuator constraints via (6.1b) the corresponding minimizer $\boldsymbol{u}^*_{\boldsymbol{W}}$ is feasible. Inevitably, (2.2) is not satisfied, but $\boldsymbol{u}^*_{\boldsymbol{W}}$ minimizes the weighted virtual control deviation

$$d(\boldsymbol{u},\boldsymbol{v})=\|\boldsymbol{W_v}\left(\boldsymbol{B}\boldsymbol{u}-\boldsymbol{v}\right)\|_2^2. \tag{6.3}$$

Problem (6.2) is inconsistent in this case because (6.2b) and (6.2c) cannot both be fulfilled.

6.2. Algorithm concept

The suggested algorithm is based on the following principles:

1. Neglecting (6.2c) the optimization problem has a closed-form solution which involves the computation of a generalized inverse of $\boldsymbol{B}$. If the obtained solution happens to be feasible regarding (6.2c) no further steps are required.

2. The equality constraints (6.2b) reduce the degrees of freedom of the vector of optimization variables $\boldsymbol{u}$ in (6.2a) in the feasible case.

3. The constrained optimization problem is transformed into an unconstrained one using the idea of penalty functions (see [47], [45], and [32]).
4. If $\boldsymbol{v} \notin \Phi$ additional steps are taken subsequently to ensure a feasible result.

6.2.1. Initial solution

The solution of (6.2a)-(6.2b) can be derived from the method of Lagrange multipliers (e.g. [32]) in closed-form and reads as

$$\boldsymbol{u_0} = -\frac{1}{2}\boldsymbol{W}^{-1}\boldsymbol{c_u} + \boldsymbol{B}^{\#}\left(\boldsymbol{v} + \frac{1}{2}\boldsymbol{B}\boldsymbol{W}^{-1}\boldsymbol{c_u}\right) \tag{6.4a}$$

$$\boldsymbol{B}^{\#} = \boldsymbol{W}^{-1}\boldsymbol{B}^{T}\left(\boldsymbol{B}\boldsymbol{W}^{-1}\boldsymbol{B}^{T}\right)^{-1} \tag{6.4b}$$

where $\boldsymbol{B}^{\#} \in \mathbb{R}^{m\times k}$ is a right-inverse of $\boldsymbol{B}$ (see [25]). In case of $\boldsymbol{u_0} \in \Omega$ the algorithm terminates.

6.2.2. Problem dimension reduction

Assume that $\boldsymbol{v} \in \Phi$. If the algorithm has to continue one can exploit the fact that $\boldsymbol{u_0}$ already fulfills (6.2b). Therefore the search directions for finding $\boldsymbol{u}^*$ can be restricted to those which are part of the nullspace of the control effectivity matrix $\boldsymbol{B}$. Suppose that a matrix $\boldsymbol{N} \in \mathbb{R}^{m\times(m-k)}$ is known whose columns span the nullspace, i.e.

$$\operatorname{span}(\boldsymbol{N}) = \mathcal{N}_{\mathrm{r}}\left(\boldsymbol{B}\right). \tag{6.5}$$

The minimizer of (6.2) $\boldsymbol{u}^*$ coincides with

$$\boldsymbol{u_E^*} = \boldsymbol{u_0} + \boldsymbol{N}\boldsymbol{x}^* \tag{6.6}$$

where $\boldsymbol{x}^* \in \mathbb{R}^{m-k}$ is the solution of a related problem in nullspace. The distinction between $\boldsymbol{u}^*$ and $\boldsymbol{u_E^*}$ is made because the problem in nullspace is chosen such that it is consistent for all virtual controls, i.e. (6.6) always exists while $\boldsymbol{u}^*$ does not. Note that $\forall \boldsymbol{x} \in \mathbb{R}^{m-k} : \boldsymbol{v} = \boldsymbol{B}\boldsymbol{u_E^*}$ because $\boldsymbol{B}\boldsymbol{N} = \boldsymbol{0} \in \mathbb{R}^{k\times(m-k)}$. The optimization process takes now place in nullspace which only has $(m-k)$ degrees of freedom. Hence, the problem's number of optimization variables is reduced. The feasible region in nullspace reads as

$$\Psi(\boldsymbol{u_0}) = \left\{\boldsymbol{x} \in \mathbb{R}^{m-k} \middle| \boldsymbol{u_{min}} \leq \boldsymbol{u_0} + \boldsymbol{N}\boldsymbol{x} \leq \boldsymbol{u_{max}}\right\} \tag{6.7}$$

which is also convex. By means of the short notation $\boldsymbol{u_x}(\boldsymbol{x}) := (\boldsymbol{u_0} + \boldsymbol{N}\boldsymbol{x})$ one can rewrite (6.2) as

$$\min_{\boldsymbol{x}\in\Psi(\boldsymbol{u_0})} \quad \boldsymbol{u_x}(\boldsymbol{x})^T\boldsymbol{W}\boldsymbol{u_x}(\boldsymbol{x}) + \boldsymbol{u_x}(\boldsymbol{x})^T\boldsymbol{c_u}. \tag{6.8}$$

For convenience (6.8) is represented in standard form of quadratic programming as

$$\min_{\boldsymbol{x}} \quad \underbrace{\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{c} + c_0}_{f(\boldsymbol{x})} \tag{6.9a}$$

$$\text{subject to } \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{6.9b}$$

$$\text{with } \boldsymbol{A}^T = [\boldsymbol{N}^T \quad -\boldsymbol{N}^T] \tag{6.9c}$$

$$\boldsymbol{b}^T = [(\boldsymbol{u_{max}} - \boldsymbol{u_0})^T \quad (\boldsymbol{u_0} - \boldsymbol{u_{min}})^T], \tag{6.9d}$$

$\boldsymbol{Q} = \boldsymbol{N}^T\boldsymbol{W}\boldsymbol{N}$ being positive definite and symmetric (see [24]), $\boldsymbol{c} = \boldsymbol{N}^T\left(2\boldsymbol{W}\boldsymbol{u_0} + \boldsymbol{c_u}\right)$, and $c_0 = \left(\boldsymbol{u_0}^T\boldsymbol{W} + \boldsymbol{c_u}^T\right)\boldsymbol{u_0}$. If $\boldsymbol{v} \in \Phi$ then (6.9) has a minimizer denoted as $\boldsymbol{x_Q^*}$ which corresponds with $\boldsymbol{x}^*$ in (6.6).

6.2.3. Penalty function

Instead of proceeding with the constrained problem (6.9) a sequence of unconstrained ones is stated with solutions converging to $\boldsymbol{x}^*_{\boldsymbol{Q}}$ under the assumption that $\boldsymbol{v} \in \Phi$. Violations of (6.9b) are penalized by means of an additional quadratic term in the cost function

$$v(\boldsymbol{x}) = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^T \boldsymbol{V}(\boldsymbol{x})\,(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{6.10}$$

with diagonal matrix

$$\boldsymbol{V}(\boldsymbol{x}) = \operatorname{diag}(w_1(\boldsymbol{x}), ..., w_{2m}(\boldsymbol{x})) \tag{6.11a}$$

$$w_i(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{A_{i\cdot}}\boldsymbol{x} - b_i > 0 \\ 0 & \text{else} \end{cases} . \tag{6.11b}$$

The new cost function combines the original $f(\boldsymbol{x})$ with (6.10) and is defined as

$$P_\alpha(\boldsymbol{x}) = v(\boldsymbol{x}) + \alpha f(\boldsymbol{x}) \tag{6.12}$$

with penalty parameter $\alpha > 0$. Starting from $\boldsymbol{u_0}$ given by (6.4a) with initial values $\boldsymbol{x_1} = \boldsymbol{0}$ and α_1 the algorithm performs a sequence of unconstrained minimizations

$$\min_{\boldsymbol{x}} P_\alpha(\boldsymbol{x}) \tag{6.13}$$

while adapting α which ultimately tends to zero. The minimizer of (6.13) for the final penalty parameter α_{fin} is denoted as

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} P_{\alpha_{fin}}(\boldsymbol{x}). \tag{6.14}$$

Note that (6.14) always exists, even if $\boldsymbol{v} \notin \Phi$ as opposed to (6.2) and (6.9). In this case only $v(\boldsymbol{x})$ is minimized.

6.2.4. Infeasibility handling

EPA indirectly includes the actuator constraints via $v(\boldsymbol{x})$ in (6.13) while only considering solution candidates which automatically fulfill (2.2). Hence, $\boldsymbol{u}^*_{\boldsymbol{E}} \notin \Omega$ follows and additional measures have to be taken. There are two possible strategies considered which differ in their computational complexity.

6.2.4.1. Projection onto Ω

Performing the Euclidean projection of the optimization result $\boldsymbol{u}^*_{\boldsymbol{E}}$ on the feasible set Ω is the simplest way of infeasibility handling. The projection operator on an m-D box is the component-wise saturation function (1.35) [29], i.e.

$$\operatorname{proj}_\Omega(\boldsymbol{u}^*_{\boldsymbol{E}}) = \operatorname{sat}_\Omega(\boldsymbol{u}^*_{\boldsymbol{E}}). \tag{6.15}$$

Please note that in case of $\boldsymbol{v} \in \Phi$ (6.15) yields the unchanged $\boldsymbol{u}^*_{\boldsymbol{E}}$. In order to investigate the properties of the projection of $\boldsymbol{u}^*_{\boldsymbol{E}}$ onto Ω one can express it as

$$\operatorname{proj}_\Omega(\boldsymbol{u}^*_{\boldsymbol{E}}) = \boldsymbol{u}^*_{\boldsymbol{E}} + \boldsymbol{u}_{\boldsymbol{\Delta}} \tag{6.16}$$

where $\boldsymbol{u_\Delta} = [u_{\Delta,1} \, \dots \, u_{\Delta,m}]^T$ represents shift due to the projection. Using the short notation w_i for (6.11b) one can define the auxiliary matrix

$$\overline{\boldsymbol{V}}(\boldsymbol{x}) = [-\operatorname{diag}(w_1, ..., w_m) \;\; \operatorname{diag}(w_{m+1}, ..., w_{2m})]. \tag{6.17}$$

From (6.6), (6.9c) - (6.9d), (6.15), and (6.16) one can verify that

$$u_{\Delta,i} = \begin{cases} b_i - \boldsymbol{A_i}.\boldsymbol{x}^* & \text{if } w_i > 0 \\ \boldsymbol{A_{(i+m)}}.\boldsymbol{x}^* - b_{i+m} & \text{if } w_{i+m} > 0 \\ 0 & \text{else} \end{cases} \tag{6.18}$$

with $i = 1, ..., m$. Note that w_i and w_{i+m} are never both equal to one because maximum and minimum constraints cannot be violated by the same variable. It follows from (6.17) and (6.18) that

$$\boldsymbol{u_\Delta} = \overline{\boldsymbol{V}}(\boldsymbol{x}^*)\,(\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}) \tag{6.19}$$

and considering $\boldsymbol{V}(\boldsymbol{x}) = \overline{\boldsymbol{V}}(\boldsymbol{x})^T\overline{\boldsymbol{V}}(\boldsymbol{x})$ together with (6.10) results in

$$v(\boldsymbol{x}^*) = \|\boldsymbol{u_\Delta}\|_2^2. \tag{6.20}$$

EPA minimizes $v(\boldsymbol{x}^*)$ and so $\boldsymbol{u_E^*}$ has minimum distance to its projection on Ω, while still satisfying $\boldsymbol{v} = \boldsymbol{B}\boldsymbol{u_E^*}$. Conversely, $\operatorname{proj}_\Omega(\boldsymbol{u_E^*})$ is the feasible control vector lying closest to a vector satisfying (2.2). The rest of this section shows under which circumstances the results of EPA and WLS are identical.

Lemma 6.2.1. *The vector $\boldsymbol{u_\Delta}$ connecting the optimization result $\boldsymbol{u_E^*}$ with its projection on Ω satisfies*

$$\boldsymbol{N}^T\boldsymbol{u_\Delta} = \boldsymbol{0}. \tag{6.21}$$

Proof. A necessary condition for minimizing $v(\boldsymbol{x})$ is

$$\boldsymbol{\nabla}_v(\boldsymbol{x}^*) = \boldsymbol{0}. \tag{6.22}$$

Evaluating (6.22) and considering (6.19) yields

$$\underbrace{\boldsymbol{A}^T\overline{\boldsymbol{V}}(\boldsymbol{x}^*)^T}_{\overline{\boldsymbol{N}}^T \in \mathbb{R}^{(m-k)\times m}} \; \underbrace{\overline{\boldsymbol{V}}(\boldsymbol{x}^*)\,(\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b})}_{\boldsymbol{u_\Delta}} = \boldsymbol{0}. \tag{6.23}$$

$\overline{\boldsymbol{N}}$ contains rows from $\boldsymbol{N}$ and zero rows. Thus, $\overline{\boldsymbol{N}}^T\boldsymbol{u_\Delta} = \boldsymbol{0}$ implies $\boldsymbol{N}^T\boldsymbol{u_\Delta} = \boldsymbol{0}$. ■

Assumption 6.2.1. $\boldsymbol{W_v} = \boldsymbol{I_k}$.

Assumption 6.2.2. The singular values of $\boldsymbol{B}$ are all equal, i.e. $\sigma_i(\boldsymbol{B}) = \sigma \;\; \forall i = 1, ..., k$.

Theorem 6.2.1. *Under Assumptions 6.2.1 and 6.2.2, WLS and EPA yield identical results, i.e. $\boldsymbol{u_W^*} = \operatorname{proj}_\Omega(\boldsymbol{u_E^*})$.*

Proof. Because $\boldsymbol{v} \notin \Phi$ and γ being very large the first summand in (6.1a) is neglected and the WLS cost function reduces to (6.3). By inserting (6.16) into (6.3) and considering Assumption 6.2.1 one obtains

$$\begin{aligned} d(\operatorname{proj}_\Omega(\boldsymbol{u_E^*}), \boldsymbol{v}) &= \|\boldsymbol{W_v}\big(\underbrace{\boldsymbol{B}\boldsymbol{u_E^*} - \boldsymbol{v}}_{\boldsymbol{0}} + \boldsymbol{B}\boldsymbol{u_\Delta}\big)\|_2^2 \\ &= \|\boldsymbol{B}\boldsymbol{u_\Delta}\|_2^2. \end{aligned} \tag{6.24}$$

The singular value decomposition (SVD) of $\boldsymbol{B} = \boldsymbol{USV}^T$ is inserted in (6.24) leading to

$$d(\text{proj}_\Omega(\boldsymbol{u}^*_{\boldsymbol{E}}), \boldsymbol{v}) = \boldsymbol{u}^T_{\boldsymbol{\Delta}} \boldsymbol{V} \boldsymbol{S}^T \boldsymbol{S} \underbrace{\boldsymbol{V}^T \boldsymbol{u}_{\boldsymbol{\Delta}}}_{\boldsymbol{v}_{\boldsymbol{u}_{\boldsymbol{\Delta}}}}. \tag{6.25}$$

It is worth noting that the last $m-k$ columns of $\boldsymbol{V}$ span $\mathcal{N}_\text{r}(\boldsymbol{B})$ because they are the eigenvectors belonging to the $m-k$ zero-eigenvalues of $\boldsymbol{B}^T\boldsymbol{B}$. Hence,

$$\boldsymbol{v}^T_{\boldsymbol{u}_{\boldsymbol{\Delta}}} = [* \ \dots \ * \ \underbrace{0 \ \dots \ 0}_{m-k}] \tag{6.26}$$

follows from Lemma 6.2.1. Assumption 6.2.2 entails

$$\boldsymbol{S}^T\boldsymbol{S} = \sigma^2 \begin{bmatrix} \boldsymbol{I}_{\boldsymbol{k}} & \boldsymbol{0}_{\boldsymbol{k}\times(\boldsymbol{m}-\boldsymbol{k})} \\ \boldsymbol{0}_{(\boldsymbol{m}-\boldsymbol{k})\times\boldsymbol{k}} & \boldsymbol{0}_{(\boldsymbol{m}-\boldsymbol{k})\times(\boldsymbol{m}-\boldsymbol{k})} \end{bmatrix} \tag{6.27}$$

and from (6.25), (6.26), and (6.27) one obtains

$$d(\text{proj}_\Omega(\boldsymbol{u}^*_{\boldsymbol{E}}), \boldsymbol{v}) = \|\sigma^2 \boldsymbol{V}^T \boldsymbol{u}_{\boldsymbol{\Delta}}\|_2^2 = \sigma^2 \|\boldsymbol{u}_{\boldsymbol{\Delta}}\|_2^2. \tag{6.28}$$

Consequently, (6.20) and (6.28) imply

$$\min_{\boldsymbol{u}\in\Omega} \gamma d\left(\boldsymbol{u}, \boldsymbol{v}\right) = \min_{\boldsymbol{x}} v(\boldsymbol{x}) \Rightarrow \boldsymbol{u}^*_{\boldsymbol{W}} = \text{proj}_\Omega(\boldsymbol{u}^*_{\boldsymbol{E}}).$$

■

Remark 6.2.1. Input matrix factorization is not unique and so suppose $\boldsymbol{B_u} = \boldsymbol{B_{v1}}\boldsymbol{B_1} = \boldsymbol{B_{v2}}\boldsymbol{B_2}$ is given. The initial solution $\boldsymbol{u_0}$ in

$$\text{proj}_\Omega(\boldsymbol{u}^*_{\boldsymbol{E}}) = \boldsymbol{u_0} + \boldsymbol{N}\boldsymbol{x}^* + \boldsymbol{u}_{\boldsymbol{\Delta}} \tag{6.29}$$

is computed via the generalized inverse in (6.4a). It is shown in Section 3.5 and [25] that CA via generalized inverses is not influenced by factorization and so is $\boldsymbol{u_0}$. Since $\boldsymbol{B_{v1}}$ and $\boldsymbol{B_{v2}}$ must have full column rank it follows that

$$\mathcal{N}_\text{r}(\boldsymbol{B_u}) = \mathcal{N}_\text{r}(\boldsymbol{B_1}) = \mathcal{N}_\text{r}(\boldsymbol{B_2}). \tag{6.30}$$

As a consequence (6.9), its minimizer $\boldsymbol{x}^*_{\boldsymbol{Q}}$, (6.13) with minimizer $\boldsymbol{x}^*$, and finally $\text{proj}_\Omega(\boldsymbol{u}^*_{\boldsymbol{E}})$ remain the same or in other words EPA with projection onto Ω is unaffected by input matrix factorization regardless of whether $\boldsymbol{v} \in \Phi$ or not.

The factorization influence on WLS depends on the feasibility of $\boldsymbol{v}$. If $\boldsymbol{v} \in \Phi$ all factorization-dependent parts of (6.1) are zero and so there is no effect on the minimizer. In case of $\boldsymbol{v} \notin \Phi$ (6.1a) effectively reduces to $\min\limits_{\boldsymbol{u}} d(\boldsymbol{u}, \boldsymbol{v})$. Given the transformation matrix $\boldsymbol{T} \in \mathbb{R}^{k\times k}$ from factorization 1 to 2 the cost functions are related by means of (see Section 3.1 or [25])

$$\|\boldsymbol{W_v}\left(\boldsymbol{B_2 u} - \boldsymbol{v_2}\right)\|_2^2 = \|\boldsymbol{W_v}\boldsymbol{T}^{-1}\left(\boldsymbol{B_1 u} - \boldsymbol{v_1}\right)\|_2^2. \tag{6.31}$$

Therefore, if $\boldsymbol{v} \notin \Phi$ factorization influences WLS in the same way as changing the virtual control weighting matrix.

Remark 6.2.2. One possibility to gain a factorization $\boldsymbol{B_{v1}B_1} = \boldsymbol{B_u}$ satisfying Assumption 6.2.2 reads as follows. Let $\boldsymbol{U_{B_u}S_{B_u}V_{B_u}^T} = \boldsymbol{B_u}$ be the result of a SVD and define the auxiliary matrices $\tilde{\boldsymbol{S}} \in \mathbb{R}^{k\times k}$ consisting of the first k rows and columns of $\boldsymbol{S_{B_u}}$ and $\tilde{\boldsymbol{V}} \in \mathbb{R}^{m\times k}$ comprising all rows from the first k columns of $\boldsymbol{V_{B_u}}$. The sought factorization of $\boldsymbol{B_u}$ is

$$\boldsymbol{B_{v1}} = \boldsymbol{U_{B_u}}\tilde{\boldsymbol{S}} \tag{6.32a}$$

$$\boldsymbol{B_1} = \tilde{\boldsymbol{V}}^T \tag{6.32b}$$

with all singular values of $\boldsymbol{B_1}$ being equal to one. Please note that the factorization (6.32) causes the WLS minimizer $\boldsymbol{u_W^*}$ to coincide with $\mathrm{proj}_\Omega(\boldsymbol{u_E^*})$ which is identical in all factorizations (see Remark 6.2.1).

6.2.4.2. Gradient projection

As deflections in nullspace directions of $\boldsymbol{B}$ do not change the resulting virtual controls the minimization of (6.3) for arbitrary factorizations must be done in original control space $\mathbb{R}^m$. Proceeding with $\mathrm{proj}_\Omega(\boldsymbol{u_E^*})$ as starting point a subsequent QP obtained by expanding (6.3) as

$$\min_{\boldsymbol{u}\in\Omega} J_G(\boldsymbol{u}) \tag{6.33a}$$

$$J_G(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{G}\boldsymbol{u} + \boldsymbol{u}^T\boldsymbol{c_G} + g_0 \tag{6.33b}$$

with $\boldsymbol{G} = \boldsymbol{B}^T\boldsymbol{W_v}^T\boldsymbol{W_v}\boldsymbol{B}$, $\boldsymbol{c_G} = -\boldsymbol{B}^T\boldsymbol{W_v}^T\boldsymbol{W_v}\boldsymbol{v}$, and constant $g_0 = \frac{1}{2}\boldsymbol{v}^T\boldsymbol{W_v}^T\boldsymbol{W_v}\boldsymbol{v}$ is solved. Due to the fact that the projection operation onto the m-D box Ω is very simple the utilization of a gradient projection algorithm [32] is facilitated. This method comprises two phases in each iteration. In phase one the search direction is chosen as the negative of the gradient

$$\boldsymbol{g}(\boldsymbol{u}) := \boldsymbol{\nabla}_{J_G} = \boldsymbol{G}\boldsymbol{u} + \boldsymbol{c_G} \tag{6.34}$$

of the cost function in (6.33). As soon as $-\boldsymbol{g}(\boldsymbol{u})$ hits a constraint the corresponding component of $\boldsymbol{u}$ is fixed causing a kink of the search direction. The result is a piecewise-linear feasible path which is examined for the first local minimizer $\boldsymbol{u_c}$ [32]. The active constraints of $\boldsymbol{u_c}$ are fixed during the next phase of gradient projection. Thus, the search is restricted to a particular subspace of $\mathbb{R}^m$ with lower dimension. However, this subproblem is a constraint QP itself whose exact solution is too expensive. The requirements on the result of the second phase $\boldsymbol{u_H}$ to obtain convergence are relatively modest, namely $J_G(\boldsymbol{u_H}) \leq J_G(\boldsymbol{u_c})$ [32]. Thus, the subspace minimization problem is only solved approximately by means of a single step in the direction obtained from Newton's method [32]. If the calculated point violates the constraints the step length is reduced until a feasible solution is found. After phase two $\boldsymbol{g}(\boldsymbol{u_H})$ is evaluated and compared with the previous gradient. The procedure terminates if the absolute values of the deviations are below a small threshold $\epsilon_g > 0$.

6.3. Implementation

6.3.1. Algorithm structure

The proposed method's basic structure is given as pseudo-code in Algorithm 1. It is an iterative process whose maximum iteration number is specified by i_{max}. The search direction $\boldsymbol{p}(\boldsymbol{x}, \alpha_i)$

is computed with Newton's method (see Section 6.3.3). If the norm of the gradient is below a small threshold $0 < \epsilon < 1$ the current result is considered as minimizer of (6.12) for the current penalty parameter α_i. The choice of α_{i+1} depends on whether a solution has been found for the current α_i. If the minimizer is found the penalty parameter can be further decreased. Otherwise, α_{i+1} is selected closer to that value where the last minimizer has been found (see Section 6.3.4). In this way the corresponding minimizer of (6.12) for α_{i+1} lies nearer to the current position facilitating a successful minimization in the next iteration. The stopping criteria in step 4) are thresholds for the absolute values of the gradient of (6.12) and for the minimum progress during the last $N > 1$ iterations. After step 4) the resulting $\boldsymbol{x}^*$ is used

Algorithm 1 Exterior Point Algorithm

```
1) compute initial solution (6.4a)
2) determine nullspace matrix $\boldsymbol{N}$
3) initialize $\boldsymbol{x_1}$ and $\alpha_1$
4) iterative minimization:
   for $i \leftarrow 1, 2, \dots, i_{max}$ do
      4.1) calculate Newton search direction $\boldsymbol{p}(\boldsymbol{x}, \alpha_i)$
      4.2) adapting penalty parameter:
         if $\|\boldsymbol{\nabla}_{P_\alpha}(\boldsymbol{x_i} + \boldsymbol{p}(\boldsymbol{x}, \alpha_i))\| < \epsilon$ then
            $\boldsymbol{x_{i+1}} \leftarrow \boldsymbol{x_i} + \boldsymbol{p}(\boldsymbol{x}, \alpha_i)$
            reduce $(\alpha_{i+1})$
         else
            $\boldsymbol{x_{i+1}} \leftarrow \boldsymbol{x_i}$
            increase $(\alpha_{i+1})$
         end if

      4.3) check stopping criteria
   end for

5) Result & infeasibility handling
   if $\boldsymbol{u_E^*} \in \Omega$ then
      $\boldsymbol{u} \leftarrow \boldsymbol{u_E^*}$
   else
      $\boldsymbol{u} \leftarrow f_{infhandling}(\boldsymbol{u_E^*})$
   end if
```

compute $\boldsymbol{u_E^*}$ via (6.6). If $\boldsymbol{u_E^*} \in \Omega$ this is the minimizer of (6.2) $\boldsymbol{u}^*$. Otherwise, the infeasibility handling takes place which guarantees that $\boldsymbol{u} \in \Omega$.

6.3.2. Initial solution and nullspace matrix

Because of $\boldsymbol{W}$ being positive definite one can use Cholesky decomposition [40] to obtain $\boldsymbol{W} = \hat{\boldsymbol{W}}^T \hat{\boldsymbol{W}}$ and introduce an auxiliary matrix

$$\boldsymbol{B_W} = \boldsymbol{B}\hat{\boldsymbol{W}}^{-1}. \tag{6.35}$$

Using (6.4) and (6.35) the unconstrained minimum reads as

$$\boldsymbol{u_0} = -\frac{1}{2}\boldsymbol{W}^{-1}\boldsymbol{c_u} + \underbrace{\hat{\boldsymbol{W}}^{-1}\boldsymbol{B}^{\dagger}}_{\boldsymbol{B}^{\#}}\left(\boldsymbol{v} + \frac{1}{2}\boldsymbol{B}\boldsymbol{W}^{-1}\boldsymbol{c_u}\right) \tag{6.36}$$

where $\boldsymbol{B}^{\dagger} = \boldsymbol{B}_{\boldsymbol{W}}^{T}\left(\boldsymbol{B_W}\boldsymbol{B}_{\boldsymbol{W}}^{T}\right)^{-1}$ is the Moore-Penrose pseudoinverse [40] of $\boldsymbol{B_W}$. However, (6.36) is not directly calculated to reduce computation time. By means of QR factorization [40] of $\boldsymbol{B}_{\boldsymbol{W}}^{T}$ one gets

$$\boldsymbol{B_W} = \underbrace{\left[\hat{\boldsymbol{R}}^{\boldsymbol{T}} \mid \boldsymbol{0}\right]}_{\boldsymbol{R}^T} \underbrace{\begin{bmatrix} \hat{\boldsymbol{Q}}^{\boldsymbol{T}} \\ \hline \overline{\boldsymbol{Q}}^{\boldsymbol{T}} \end{bmatrix}}_{\boldsymbol{Q}^T \in \mathbb{R}^{m\times m}} \tag{6.37}$$

with $\hat{\boldsymbol{R}} \in \mathbb{R}^{k\times k}$ being an upper triangle matrix and the orthogonal matrix $\boldsymbol{Q}$ consisting of $\hat{\boldsymbol{Q}} \in \mathbb{R}^{m\times k}$ and $\overline{\boldsymbol{Q}} \in \mathbb{R}^{m\times(m-k)}$. It can be seen that a right multiplication of (6.37) with $\overline{\boldsymbol{Q}}$ (the last $m-k$ columns of $\boldsymbol{Q}$) results in $\boldsymbol{0} \in \mathbb{R}^{m\times(m-k)}$. Thus, they span the nullspace of $\boldsymbol{B_W}$ which is utilized to determine the sought nullspace matrix

$$\boldsymbol{N} = \hat{\boldsymbol{W}}^{-1}\overline{\boldsymbol{Q}}. \tag{6.38}$$

From (6.37) one obtains $\boldsymbol{B}^{\#} = \hat{\boldsymbol{W}}^{-1}\hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}^{-T}$. So the first step to efficiently compute the initial solution is solving

$$\boldsymbol{v} + \frac{1}{2}\boldsymbol{B}\boldsymbol{W}^{-1}\boldsymbol{c_u} = \hat{\boldsymbol{R}}^{T}\boldsymbol{y} \tag{6.39}$$

for $\boldsymbol{y}$ by forward substitution. Now

$$\boldsymbol{u_0} = -\frac{1}{2}\boldsymbol{W}^{-1}\boldsymbol{c_u} + \hat{\boldsymbol{W}}^{-1}\hat{\boldsymbol{Q}}\boldsymbol{y} \tag{6.40}$$

follows. Please note that $\hat{\boldsymbol{W}}^{-1}$ in the preceding expressions is never computed explicitly, instead (column-wise) backward substitution is used because $\hat{\boldsymbol{W}}$ is an upper triangular matrix. The main advantage of this approach is that $\boldsymbol{N}$ is computed without any additional effort (solving a homogeneous system of equations is not required).

6.3.3. Search direction

In the i-th iteration EPA determines a search direction $\boldsymbol{p}$ in order to minimize (6.12) for a given α_i by means of Newton's method [32], i.e.

$$\boldsymbol{H}\boldsymbol{p} = -\boldsymbol{\nabla}_{P_\alpha}(\boldsymbol{x_i}, \alpha_i) \tag{6.41}$$

is solved for $\boldsymbol{p}$ with $\boldsymbol{H} \in \mathbb{R}^{(m-k)\times(m-k)}$ usually being $\boldsymbol{\mathcal{H}}_{P_\alpha}(\boldsymbol{x_i}, \alpha)$. Its inverse is not explicitly computed, but instead Gaussian elimination with partial pivoting is used [40] to obtain $\boldsymbol{p}$. As $\boldsymbol{x_i}$ approaches $\Psi(\boldsymbol{u_0})$ and α_i gets very small the condition number κ of (6.41) can get very large. Therefore, $\boldsymbol{H}$ is chosen as

$$\boldsymbol{H} = \begin{cases} \boldsymbol{\mathcal{H}}_{P_\alpha}(\boldsymbol{x_i}, \alpha) & \text{if } \kappa\left[\boldsymbol{\mathcal{H}}_{P_\alpha}\right] < \kappa_{max} \\ \boldsymbol{\mathcal{H}}_{P_\alpha}(\boldsymbol{x_i}, \alpha) + \gamma_H \boldsymbol{I_{m-k}} & \text{else} \end{cases} . \tag{6.42}$$

with $\gamma_H > 0$ (Hessian Modification, [32]).

6.3.4. Penalty parameter α

Let i be the iteration number, α_i the corresponding penalty parameter of the i-th iteration, and $\boldsymbol{x}_{\boldsymbol{\alpha_i}} = \arg\min\limits_{\boldsymbol{x}} P_{\alpha_i}(\boldsymbol{x})$. In general the sequence $\{\alpha_i\}$ must decrease in order that $\{\boldsymbol{x}_{\boldsymbol{\alpha_i}}\}$ converges to $\boldsymbol{x}^*_{\boldsymbol{Q}}$. An adaption law that works very well reads as

$$\alpha_{i+1} = \min\left(\frac{1}{2}\alpha_i, \frac{v(\boldsymbol{x}_{\boldsymbol{\alpha_i}})}{f^+(\boldsymbol{x}_{\boldsymbol{\alpha_i}})}\right) \tag{6.43}$$

with $f^+(\boldsymbol{x}) = f(\boldsymbol{x}) + f_{offset}$ and $f_{offset} > 0$ ensuring $\forall \boldsymbol{x} : f^+(\boldsymbol{x}) > 0$. However, if Newton's method does not find $\boldsymbol{x}_{\boldsymbol{\alpha_i}}$ during the current iteration then α_{i+1} is increased by

$$\alpha_{i+1} = \frac{1}{2}\left(\alpha_{ok} + \alpha_i\right) \tag{6.44}$$

with α_{ok} being the last penalty parameter value where a solution was found (see Section 6.4).

6.3.5. Initial values and hot-start

Usually, EPA is initialized with $\boldsymbol{x}_{\boldsymbol{\alpha_0}} = \boldsymbol{0}$ (corresponds with solution from generalized inverse), $\alpha_0 = 2$, and evaluating (6.43). In case of a hot-start, values based on the last execution of the CA-algorithm are used for initialization instead. The basic idea behind this strategy is that the virtual controls of consecutive time steps do not differ very much. Hence, the same is expected to be true for the real controls $\boldsymbol{u}$ and nullspace vector $\boldsymbol{x}$ (although there is of course no guarantee). The hot-start initialization is $\boldsymbol{x}_{\boldsymbol{\alpha_0}} = \overline{\boldsymbol{x}}^*\delta_1$ and $\alpha_1 = \overline{\alpha}\delta_2$ with $\overline{\boldsymbol{x}}^*$ and $\overline{\alpha}$ being the final values from the last time step, $0 < \delta_1 < 1$, and $\delta_2 > 1$.

6.3.6. Gradient projection

The gradient projection method comprises two stages. The first one starts with the calculation of a piecewise-linear path from the steepest descent direction (the negative gradient (6.34)) of the cost function. Introducing the scalar parameter $l \geq 0$ and the initial value as $\boldsymbol{u}_{\boldsymbol{0}} = \mathrm{proj}_\Omega(\boldsymbol{u}^*_{\boldsymbol{E}})$ the path is described by

$$\boldsymbol{u}(l) = \mathrm{proj}_\Omega\left[\boldsymbol{u}_{\boldsymbol{0}} - l\boldsymbol{g}(\boldsymbol{u}_{\boldsymbol{0}})\right]. \tag{6.45}$$

The next step is the computation of those values of l where the boundary of Ω is reached for each vector component of (6.45). After sorting the results in ascending order and removing duplicates one obtains a list $\{l_1, ..., l_N\}$ for examining the path interval-wise. On every path segment $[l_{i-1}, l_i]$ for $i = 1, ..., N$ with $l_0 := 0$ one can express (6.33b) as a scalar quadratic function of $l_\Delta \in [0, l_i - l_{i-1}]$. Setting its gradient to zero results in l^*_Δ and if $0 \leq l^*_\Delta < (l_i - l_{i-1})$ the first local minimizer is given as

$$\boldsymbol{u}_{\boldsymbol{c}} = \boldsymbol{u}(l_{i-1} + l^*_\Delta). \tag{6.46}$$

Otherwise, the next interval on the path is examined in the same way. The implementation of this first phase of gradient projection is done as described in [32].

Let the active set of constraints at $\boldsymbol{u}_{\boldsymbol{c}}$ be given as

$$\mathcal{A}(\boldsymbol{u}_{\boldsymbol{c}}) := \big\{i \in \{1, ..., m\} \big| (u_{c,i} = u_{min,i}) \vee (u_{c,i} = u_{max,i})\big\} \tag{6.47}$$

and the corresponding set of free variables as

$$\mathcal{F}(\boldsymbol{u_c}) := \big\{ i \in \{1, ..., m\} \big| i \notin \mathcal{A}(\boldsymbol{u_c}) \big\} \, . \tag{6.48}$$

The second stage of gradient projection performs an approximate subspace minimization with (6.47) kept constant [32]. Hence, the new vector of optimization variables reads as $\boldsymbol{x} = [x_1, ..., x_{N_f}]^T$ with $N_f := |\mathcal{F}(\boldsymbol{u_c})|$. The relationship to the optimization vector from stage one reads as

$$\boldsymbol{u} = \boldsymbol{u_c} + \boldsymbol{N_H x} \tag{6.49}$$

where $\boldsymbol{N_H} \in \mathbb{R}^{m \times N_f}$. Considering $\mathcal{F}(\boldsymbol{u_c}) = \{f_1, ..., f_{N_f}\}$ each column $\boldsymbol{h_i}$ of $\boldsymbol{N_H}$ contains only zeros except for the element with row index f_i which is "1". Note that (6.49) is basically a transformation into nullspace similar to Section 6.2.2. Neglecting g_0 in (6.33) and using (6.49) yields[2]

$$\min_{\boldsymbol{x}} J_H(\boldsymbol{x}) \tag{6.50a}$$

$$J_H(\boldsymbol{x}) = \big[\frac{1}{2}\boldsymbol{x}^T \underbrace{\boldsymbol{N_H^T G N_H}}_{\boldsymbol{G_H}} \boldsymbol{x} + \boldsymbol{x}^T \underbrace{\big(\boldsymbol{N_H^T G u_c} + \boldsymbol{N_H^T c_G}\big)}_{\boldsymbol{c_H}} \big] \tag{6.50b}$$

$$\text{subject to} \quad \boldsymbol{N_H^T}(\boldsymbol{u_{min}} - \boldsymbol{u_c}) \leq \boldsymbol{x} \leq \boldsymbol{N_H^T}(\boldsymbol{u_{max}} - \boldsymbol{u_c}). \tag{6.50c}$$

The search direction $\boldsymbol{p_H}$ is now computed with Newton's method from

$$\overline{\boldsymbol{H}} \boldsymbol{p_H} = -\boldsymbol{\nabla}_{J_H}(\boldsymbol{x}) \tag{6.51}$$

with $\overline{\boldsymbol{H}} \in \mathbb{R}^{N_f \times N_f}$. It should be pointed out that $\text{rank}(\boldsymbol{\mathcal{H}}_{J_H}) = \text{rank}(\boldsymbol{G_H}) = \min(N_f, k)$ which follows from $\text{rank}(\boldsymbol{G}) = k$. Therefore, the following Hessian modification

$$\overline{\boldsymbol{H}} = \begin{cases} \boldsymbol{\mathcal{H}}_{J_H} & \text{if } N_f \leq k \\ \boldsymbol{\mathcal{H}}_{J_H} + \epsilon_H \boldsymbol{I_{N_f}} & \text{else} \end{cases} \tag{6.52}$$

with $0 < \epsilon_H < 1$ is carried out. Finally, the step length l_H is reduced until $\boldsymbol{x} = l_H \boldsymbol{p_H}$ satisfies (6.50c) and the result of stage two reads as

$$\boldsymbol{u_H} = \boldsymbol{u_c} + l_H \boldsymbol{N_H p_H}. \tag{6.53}$$

Denote the gradient projection result of the last iteration as $\boldsymbol{u_H^{(j-1)}}$ and the current result as $\boldsymbol{u_H^{(j)}}$. The algorithm terminates if

$$|\boldsymbol{u_H^{(j)}} - \boldsymbol{u_H^{(j-1)}}| < \epsilon \boldsymbol{1} \tag{6.54}$$

with $0 < \epsilon < 1$ and $\boldsymbol{1} \in \mathbb{R}^m$ having all entries set to "1". Otherwise, the next iteration is started using $\boldsymbol{u_H^{(j)}}$ as initial value in (6.45) instead of $\boldsymbol{u_0}$.

6.4. Convergence

The penalty function approach is well-known in optimization literature. Proofs of convergence can be found for example in [45], [46], and [32] where instead of (6.12) a slightly modified unconstrained cost function

$$\overline{P}_\alpha(\boldsymbol{x}) = \frac{1}{\alpha} v(\boldsymbol{x}) + f(\boldsymbol{x}) \tag{6.55}$$

is used with $\frac{1}{\alpha} \to \infty$.

[2]The resulting constant terms in (6.50) have already been omitted.

Assumption 6.4.1. Problem (6.9) has a solution $\boldsymbol{x}^*_{\boldsymbol{Q}}$.

Assumption 6.4.2. $f(\boldsymbol{x})$ and $\boldsymbol{A_i}.\boldsymbol{x} - b_i$ with $i = 1, ..., 2m$ are continuous functions on $\mathbb{R}^{m-k}$.

Assumption 6.4.3. $f(\boldsymbol{x})$ is coercive, i.e. $\lim\limits_{\|\boldsymbol{x}\|\to\infty} f(\boldsymbol{x}) = +\infty$.

Lemma 6.4.1. *Under assumptions 6.4.1 - 6.4.3 it follows that*

$$\forall\alpha > 0, \exists\, \boldsymbol{x_\alpha} : \; P_\alpha(\boldsymbol{x_\alpha}) = \min_{\boldsymbol{x}} P_\alpha(\boldsymbol{x}) \tag{6.56a}$$

$$\lim_{\alpha\to 0} \{\boldsymbol{x_\alpha}\} = \boldsymbol{x}^*_{\boldsymbol{Q}} \tag{6.56b}$$

$$\{\boldsymbol{x_\alpha}\} \textit{ bounded.} \tag{6.56c}$$

Proof. Assumption 6.4.2 is fulfilled because $f(\boldsymbol{x})$ and $\boldsymbol{A_i}.\boldsymbol{x} - b_i$ are polynomials which are continuous on their domain and $\boldsymbol{Q} \succ 0$ guarantees Assumption 6.4.3. The expression $P_\alpha(\boldsymbol{x}) = \alpha\left[f(\boldsymbol{x}) + \frac{1}{\alpha}v(\boldsymbol{x})\right] = \alpha\overline{P}_\alpha(\boldsymbol{x})$ is just (6.55) scaled by a positive value which does not change any properties of the optimization problem. Multiplying (6.56a) by $\frac{1}{\alpha}$ yields exactly the same statements as in Corollary 6.2.4 from [45] which completes the proof. ■

Using (6.9a), (6.10), (6.12), and the abbreviation $\boldsymbol{V_x} = \boldsymbol{V}(\boldsymbol{x})$ one obtains the overall cost function

$$\begin{aligned} P_\alpha(\boldsymbol{x}) &= (\boldsymbol{Ax} - \boldsymbol{b})^T \boldsymbol{V_x} (\boldsymbol{Ax} - \boldsymbol{b}) + \alpha\left(\boldsymbol{x}^T\boldsymbol{Qx} + \boldsymbol{x}^T\boldsymbol{c} + c_0\right) \\ &= \boldsymbol{x}^T\left(\boldsymbol{A}^T\boldsymbol{V_x}\boldsymbol{A} + \alpha\boldsymbol{Q}\right)\boldsymbol{x} + \boldsymbol{x}^T\left(\alpha\boldsymbol{c} - 2\boldsymbol{A}^T\boldsymbol{V_x}\boldsymbol{b}\right) + \boldsymbol{b}^T\boldsymbol{V_x}\boldsymbol{b} + \alpha c_0. \end{aligned} \tag{6.57}$$

Lemma 6.4.2. *For all $\alpha > 0$ expression (6.57) has a global minimizer at $\boldsymbol{x_\alpha}$ which is a critical point, i.e. $\boldsymbol{\nabla}_{P_\alpha}(\boldsymbol{x_\alpha}) = \boldsymbol{0}$.*

Proof. From $\boldsymbol{V_x} \succeq 0$ it follows that $\boldsymbol{A}^T\boldsymbol{V_x}\boldsymbol{A} \succeq 0$ and because of $\boldsymbol{Q} \succ 0$ one obtains $\boldsymbol{A}^T\boldsymbol{V_x}\boldsymbol{A} + \alpha\boldsymbol{Q} \succ 0$ (see [24]). This implies that $P_\alpha(\boldsymbol{x})$ is coercive and therefore $\boldsymbol{x_\alpha}$ is a global minimizer (see [45], Theorem 1.4.4). The gradient

$$\begin{aligned} \boldsymbol{\nabla}_{P_\alpha}(\boldsymbol{x}, \alpha) &= 2\boldsymbol{A}^T\boldsymbol{V_x}(\boldsymbol{Ax} - \boldsymbol{b}) + 2\alpha\boldsymbol{Qx} + \alpha\boldsymbol{c} \\ &= 2\left(\boldsymbol{A}^T\boldsymbol{V_x}\boldsymbol{A} + \alpha\boldsymbol{Q}\right)\boldsymbol{x} - 2\boldsymbol{A}^T\boldsymbol{V_x}\boldsymbol{b} + \alpha\boldsymbol{c} \end{aligned} \tag{6.58}$$

is a continuous function which exists on all of $\mathbb{R}^{m-k}$ and so it follows from the same theorem in [45] that $\boldsymbol{x_\alpha}$ must be a critical point. ■

Remark 6.4.1. Note that Lemma 6.4.2 remains true if $\boldsymbol{v} \notin \Phi$, i.e. (6.14) always exists.

Remark 6.4.2. $\forall\boldsymbol{x} \in \mathbb{R}^{m-k} \setminus \mathcal{D} : \frac{\partial\boldsymbol{V_x}}{\partial\boldsymbol{x}} = \boldsymbol{0}$ with

$$\mathcal{D} = \left\{\boldsymbol{x} \in \mathbb{R}^{m-k} | \exists (i \leq 2m) \in \mathbb{N}^+, \boldsymbol{A_i}.\boldsymbol{x} - b_i = 0\right\}, \tag{6.59}$$

i.e. $\mathcal{D}$ contains the discontinuities of (6.11b). However, due to the quadratic form of the penalty function (6.10) the gradient (6.58) nevertheless exists and is continuous on all of $\mathbb{R}^{m-k}$. The Hessian on the other hand side

$$\boldsymbol{\mathcal{H}}_{P_\alpha}(\boldsymbol{x}, \alpha) = 2\left(\boldsymbol{A}^T\boldsymbol{V_x}\boldsymbol{A} + \alpha\boldsymbol{Q}\right) \tag{6.60}$$

only exists on $\mathbb{R}^{m-k} \setminus \mathcal{D}$. Despite that (6.60) has no singularities (see Section 6.3.3) and can be used for all $\boldsymbol{x}$ as discussed in [47].

The region where the identical set of constraints is violated as at a given position $\boldsymbol{x_0}$ is denoted as

$$\mathcal{M}_{\boldsymbol{x_0}} = \left\{ \boldsymbol{x} \in \mathbb{R}^{m-k} \middle| \boldsymbol{V_x} = \boldsymbol{V_{x_0}} \right\}. \tag{6.61}$$

Lemma 6.4.3. *Let $\alpha_i > 0$ be the value of the penalty parameter in the i-th iteration and assume that the minimum for α_{i-1} was found during the previous iteration at $\boldsymbol{x_{\alpha_{i-1}}} \notin \mathcal{D}$. Suppose $\boldsymbol{x_n}$ is the result of a single Newton step. If $\boldsymbol{x_n} \in \mathcal{M}_{\boldsymbol{x_{\alpha_{i-1}}}}$ holds it follows that $\boldsymbol{x_n} = \boldsymbol{x_{\alpha_i}}$.*

Proof. The Newton direction is given as $\boldsymbol{x_n} = \boldsymbol{x_{\alpha_{i-1}}} - \boldsymbol{\mathcal{H}}_{P_\alpha}(\boldsymbol{x_{\alpha_{i-1}}}, \alpha_i)^{-1} \boldsymbol{\nabla}_{P_\alpha}(\boldsymbol{x_{\alpha_{i-1}}}, \alpha_i)$ and using (6.58), (6.60) and $\boldsymbol{S_{x,\alpha}} := \boldsymbol{A}^T \boldsymbol{V_x} \boldsymbol{A} + \alpha \boldsymbol{Q}$ results in

$$\boldsymbol{x_n} = \boldsymbol{S}^{-1}_{\boldsymbol{x_{\alpha_{i-1}}},\boldsymbol{\alpha_i}} \left(\boldsymbol{A}^T \boldsymbol{V}_{\boldsymbol{x_{\alpha_{i-1}}}} \boldsymbol{b} - \frac{\alpha_i}{2} \boldsymbol{c} \right) \tag{6.62}$$

The gradient evaluated at the new position $\boldsymbol{x_n}$ is

$$\begin{gathered} \boldsymbol{\nabla}_{P_\alpha}(\boldsymbol{x_n}, \alpha_i) = -2\boldsymbol{A}^T \boldsymbol{V_{x_n}} \boldsymbol{b} + \alpha_i \boldsymbol{c} + \\ 2\boldsymbol{S}_{\boldsymbol{x_n},\boldsymbol{\alpha_i}} \underbrace{\boldsymbol{S}^{-1}_{\boldsymbol{x_{\alpha_{i-1}}},\boldsymbol{\alpha_i}} \left(\boldsymbol{A}^T \boldsymbol{V}_{\boldsymbol{x_{\alpha_{i-1}}}} \boldsymbol{b} - \frac{\alpha_i}{2} \boldsymbol{c} \right)}_{\boldsymbol{x_n}}. \end{gathered} \tag{6.63}$$

$\boldsymbol{x_n} \in \mathcal{M}_{\boldsymbol{x_{\alpha_{i-1}}}}$ is equivalent to $\boldsymbol{V_{x_n}} = \boldsymbol{V}_{\boldsymbol{x_{\alpha_{i-1}}}}$ which implies (6.63) is zero and this yields $\boldsymbol{x_n} = \boldsymbol{x_{\alpha_i}}$. ■

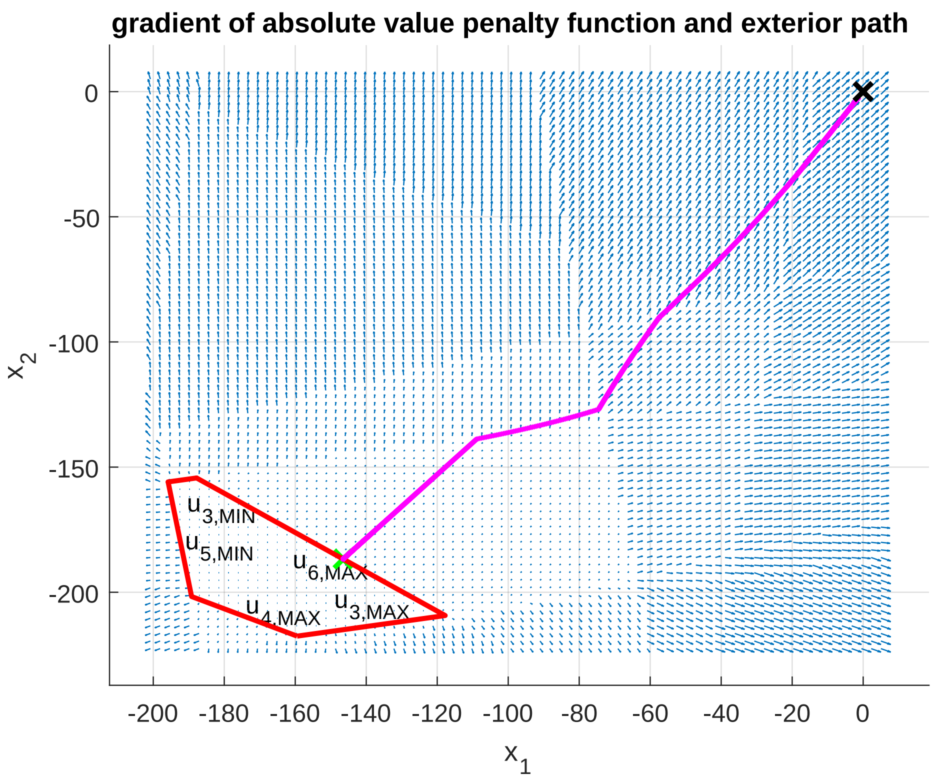

Figure 6.1.: Visualization of the exterior path: The gradient of an absolute value penalty function reveals those regions in nullspace with identical constraints violated as areas with the same arrow directions. The red polygon represents the feasible region $\Psi(\boldsymbol{u_0})$. The continuous exterior path is shown as sequence of purple 'x' connecting $\mathbf{0}$ and $\boldsymbol{x}^*_{\boldsymbol{Q}}$ (labeled with a green 'x').

The sequence of points $\{\boldsymbol{x_\alpha}\}$ starting at $\mathbf{0}$ and converging to $\boldsymbol{x}^*_{\boldsymbol{Q}}$ (or $\boldsymbol{x}^*$ in case of an infeasible $\boldsymbol{v} \notin \Phi$) is called *exterior path* [47]. It is defined by the implicit function $\boldsymbol{\nabla}_{P_\alpha}(\boldsymbol{x},\alpha) = \mathbf{0}$. Since on $\mathbb{R}^{m-k} \setminus \mathcal{D}$ the gradient $\boldsymbol{\nabla}_{P_\alpha}$ is continuously differentiable and $\boldsymbol{\mathcal{H}}_{P_\alpha}$ is invertible it can be concluded from the implicit function theorem [32] that a continuous function $\boldsymbol{x_\alpha}(\alpha)$ exists. In case of a 2-D nullspace the exterior path and the feasible region can be visualized (see Figure 6.1) by plotting the solutions of (6.13) for α decreasing in tiny steps. The gradient of an absolute value penalty function [45] is used in this figure to illustrate the areas where the same constraints are violated by arrows pointing in the same direction. The exterior path continuously connects the origin and $\boldsymbol{x}^*_{\boldsymbol{Q}}$.

The following discussion deals with $\boldsymbol{x_n} \notin \mathcal{M}_{\boldsymbol{x}_{\alpha_{i-1}}}$. EPA repeats computing Newton steps for a number of times $N > 1$. In case of reaching $\boldsymbol{x_{\alpha_i}}$ the penalty parameter α_{i+1} is reduced as described in Section 6.3.4. If $\boldsymbol{x_{\alpha_i}}$ is not found the failed Newton steps are discarded, $\boldsymbol{x_{\alpha_{i-1}}}$ is chosen as starting point for iteration $i+1$, and α_{i+1} is increased to a value $\alpha_i < \alpha_{i+1} \leq \alpha_{i-1}$. Now three scenarios are possible:

scenario 1: $\boldsymbol{x_n} \in \mathcal{M}_{\boldsymbol{x}_{\alpha_{i-1}}}$. It follows from Lemma 6.4.3 that $\boldsymbol{x_{\alpha_{i+1}}}$ is found in one step. Consequently, the penalty parameter of the next iteration α_{i+2} is decreased. This corresponds with going from $\boldsymbol{x_{\alpha_0}}$ to $\boldsymbol{x_{n_1}}$ in Figure 6.2.

scenario 2: $\boldsymbol{x_n} \notin \mathcal{M}_{\boldsymbol{x}_{\alpha_{i-1}}}$ and $\boldsymbol{x_n} \notin \mathcal{M}_{\boldsymbol{x}_{\alpha_{i+1}}}$. There is still no guarantee that $\boldsymbol{x_{\alpha_{i+1}}}$ is found by the Newton step. This is represented by the step from $\boldsymbol{x_{\alpha_1}}$ to $\boldsymbol{x_{n_2}}$ in Figure 6.2.

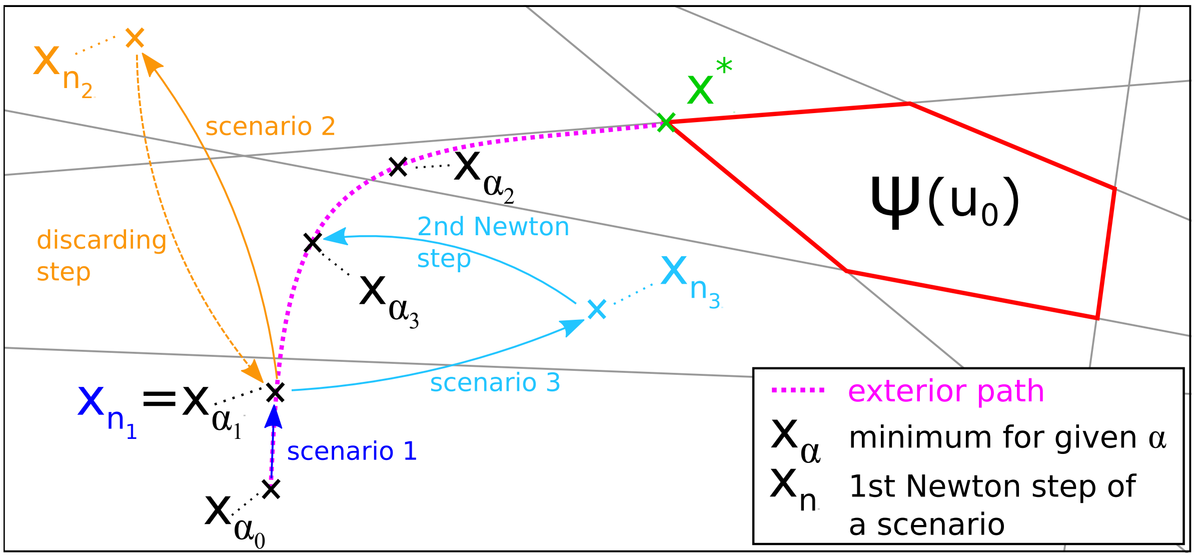

Figure 6.2.: Principal behavior of EPA when the set of violated constraints changes on the exterior path. The red polygon depicts the feasible area in nullspace and the gray-framed polygons outline regions where the same constraints are violated. Initially, scenario 1 applies because the sets of violated constraints at $\boldsymbol{x}_{\boldsymbol{\alpha_0}}$ and $\boldsymbol{x}_{\boldsymbol{\alpha_1}}$ are identical (see Lemma 6.4.3) whereas in case of scenarios 2 and 3 those sets differ from each other. Thus, the first Newton steps in scenarios 2 and 3 will not reach the minimum for the current α_i in general. In scenario 2 the set of violated constraints at $\boldsymbol{x}_{\boldsymbol{n_2}}$ is different from the one at $\boldsymbol{x}_{\boldsymbol{\alpha_2}}$. Therefore, it is not certain that subsequent Newton steps reach $\boldsymbol{x}_{\boldsymbol{\alpha_2}}$ and so α_3 possibly has to be increased. This reduces the distance between the current position on the exterior path $\boldsymbol{x}_{\boldsymbol{\alpha_1}}$ and $\boldsymbol{x}_{\boldsymbol{\alpha_3}}$. In the third scenario $\boldsymbol{x}_{\boldsymbol{n_3}} \in \mathcal{M}_{\boldsymbol{x}_{\boldsymbol{\alpha_3}}}$ and so the second Newton step yields the current minimum $\boldsymbol{x}_{\boldsymbol{\alpha_3}}$.

EPA potentially has to discard the failed steps, use $\boldsymbol{x}_{\boldsymbol{\alpha_{i-1}}}$ again as starting point, and increase α_{i+2} such that $\alpha_{i+1} < \alpha_{i+2} \leq \alpha_{i-1}$.

scenario 3: $\boldsymbol{x}_{\boldsymbol{n}} \notin \mathcal{M}_{\boldsymbol{x}_{\boldsymbol{\alpha_{i-1}}}}$ and $\boldsymbol{x}_{\boldsymbol{n}} \in \mathcal{M}_{\boldsymbol{x}_{\boldsymbol{\alpha_{i+1}}}}$. EPA reaches the correct position $\boldsymbol{x}_{\boldsymbol{\alpha_{i+1}}}$ with the second Newton step and α_{i+2} is decreased. This is indicated by moving from $\boldsymbol{x}_{\boldsymbol{\alpha_1}}$ to $\boldsymbol{x}_{\boldsymbol{n_3}}$ in Figure 6.2.

To put it briefly, EPA reduces the distance on the continuous exterior path between the starting point of an iteration and the sought position. Thereby $|\boldsymbol{\nabla}_{P_\alpha}|$ at the starting point is decreased and as a consequence the same is true for the step-size which facilitates the occurrence of scenario 3 from the list above.

6.5. Results

EPA is implemented in C++ using the BLAS and LAPACK[3] routines of the mathematical programming library Intel MKL 2018 [48]. MEX functions provide the interface to Matlab. Execution time is measured by means of the Matlab-commands *tic* and *toc*. The test machine

[3]**B**asic **L**inear **A**lgebra **S**ubprograms and **L**inear **A**lgebra **Pack**age define standardized programming interfaces for numerous mathematical operations.

is a notebook computer coming with an Intel Core i7-6700HQ CPU with 2.6 GHz, 16 GB RAM, Windows 10 64-bit, and Matlab R2016b 64-bit.

6.5.1. Small-scale problems

Quadratic programming in CA has mainly to deal with low-dimensional problems with $k, m \leq 10$. At first this section contrasts the performance of EPA with QCAT [49] which is a collection of WLS solvers for CA. In this comparison the tested algorithms are

- wlsc_alloc (WLA): active-set strategy [6] implemented in C, using the BLAS and LAPACK routines of Intel MKL 2018
- fxp_alloc (FXP): fixed-point algorithm [34]
- ip_alloc (IPA): interior-point algorithm [35].

The considered optimization task is (6.1) with problem data

$$\begin{aligned}
\boldsymbol{B} &= \begin{bmatrix} 10 & 8 & 2 & 1 & 0 \\ -8 & 10 & -1 & 2 & 0 \\ -\frac{2366}{1171} & -\frac{869}{2060} & \frac{91128}{7709} & -\frac{149}{2393} & 5 \end{bmatrix} \\
\boldsymbol{u_{max}} &= \begin{bmatrix} 1 & 2 & 2 & 5 & 1 \end{bmatrix}^T \\
\boldsymbol{u_{min}} &= \begin{bmatrix} -1 & -1 & -4 & -4 & -4 \end{bmatrix}^T \\
\boldsymbol{u_p} &= \boldsymbol{0},\ \boldsymbol{W} = \boldsymbol{I_m},\ \boldsymbol{W_v} = \boldsymbol{I_k} \\
\boldsymbol{v_{fea}} &= \begin{bmatrix} 20 & 28 & 27 \end{bmatrix}^T,\ \boldsymbol{v_{inf}} = \begin{bmatrix} 30 & -25 & 25 \end{bmatrix}^T
\end{aligned} \tag{6.64}$$

Initially, the desired virtual control vector $\boldsymbol{v_{fea}}$ lies inside the AMS (1.34). The quality of a result is measured by means of the cost function value $f_u(\boldsymbol{u})$, the maximum constraint violation $v(\boldsymbol{u})$, and the norm of the virtual control error (6.3). In order to determine the average computation time each algorithm has to solve the problem 10000 times. The results are condensed in Table 6.1. One can see that WLA requires the least computation time closely followed by EPA. In terms of accuracy in reaching the desired $\boldsymbol{v_{fea}}$ EPA achieves the best result followed by WLA. However, WLA is the only algorithm that features a small constraint violation. The fixed-point method entails the greatest error in matching the desired virtual control. By increasing its maximum number of iterations, which was set to 200 during the tests, one could reduce this error at the price of rising execution time. The worst result regarding computation time and cost function value comes from IPA which is moreover very sensitive for variations of the weighting parameter γ. The nullspace dimension of $\boldsymbol{B}$ in (6.64) is two, which allows a comprehensible visualization. Figures 6.3 and 6.4 show the optimization steps of EPA together with the cost function $f(\boldsymbol{x})$ and the constraint violation $v(\boldsymbol{x})$. One recognizes that already the result of the fourth iteration lies very close to the optimal value, i.e. the computation time could be further reduced by adjusting the tolerance of the stopping criteria. The next test deals with the infeasible virtual control vector $\boldsymbol{v_{inf}}$. As it can be seen in Figure 6.5 EPA minimizes the constraint violation in such cases, i.e. the optimization result $\boldsymbol{u_E^*}$ is as close to Ω as possible. However, EPA does not return $\boldsymbol{u_E^*}$ in this case but its projection on Ω. Because of all singular values of $\boldsymbol{B}$ in (6.64) being identical EPA is able to minimize the deviation between desired and actual virtual controls without the subsequent gradient projection stage as it can be seen in the last line of Table 6.2. Finally, the virtual

Table 6.1.: Results for solving (6.64) with $\boldsymbol{v} = \boldsymbol{v_{fea}}$.

	EPA	WLA	FXP	IPA
time	22 μs	12 μs	155 μs	304 μs
iter.	6	3	200	11
$f_u(\boldsymbol{u})$	21.4309878	21.4309841	20.5597369	21.6729754
$v(\boldsymbol{u})$	0	$1.43 \cdot 10^{-8}$	0	0
$d(\boldsymbol{u}, \boldsymbol{v})$	$7.59 \cdot 10^{-9}$	$1.64 \cdot 10^{-6}$	0.27	$1.67 \cdot 10^{-3}$

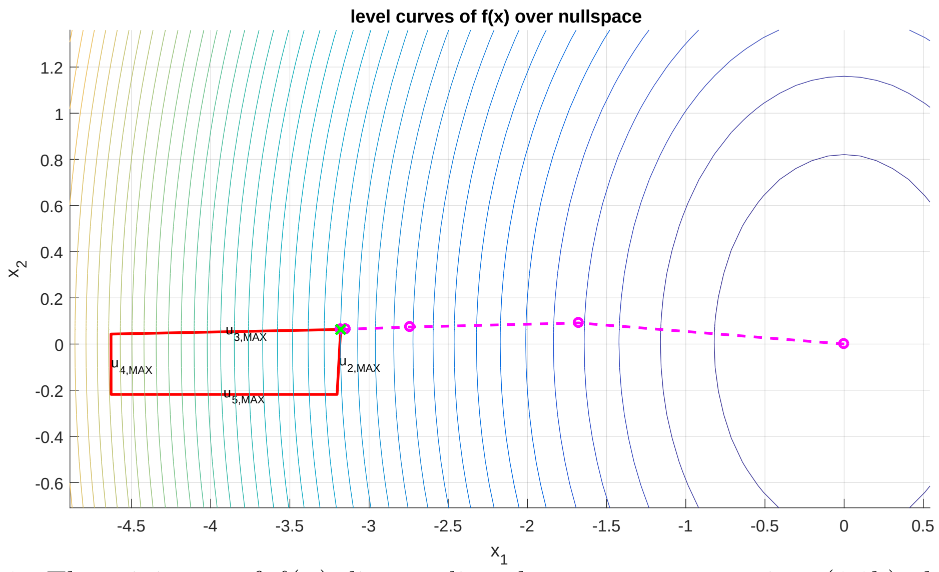

Figure 6.3.: The minimum of $f(\boldsymbol{x})$ disregarding the actuator constraints (6.9b) always lies in the center of nullspace because this corresponds with the solution $\boldsymbol{u_0}$ from the generalized inverse (6.4a). The red polygon depicts the feasible area $\Psi(\boldsymbol{u_0})$ where the actuator constraints are satisfied. The exterior path is shown as dashed, purple line and ends at the global constrained minimizer $\boldsymbol{x_Q^*} \approx [-3.18\ 0.06]^T$.

control weighting matrix in (6.64) is changed to $\boldsymbol{W_v} = \text{diag}(100, 3, 52)$ while the desired virtual control vector is still the infeasible $\boldsymbol{v_{inf}}$. Thus, gradient projection has to be applied after determining $\text{proj}_\Omega(\boldsymbol{u_E^*})$ by EPA. The related results can be seen in Table 6.3. EPA needs two gradient projection iterations after its first optimization stage in nullspace. EPA and WLA achieve effectively the same results for the weighted virtual control error $d(\boldsymbol{u}, \boldsymbol{v})$. In order to assess the performance of EPA for a wider range of problems it has to solve randomly generated test cases. Its results and execution times are not only compared to the algorithms from QCAT but also to the following reference algorithms:

- Quadprog with *interior-point-convex* alg. (QUA)
- Gurobi 7.02 64-bit ([50]) (GUR)
- IBM ILOG CPLEX 12.7 64-bit (CPL)
- qpOASES 3.2.0 ([51]) (QPO)

Quadprog is Matlab's built-in solver for quadratic programming. Gurobi and CPLEX are separate software packages which come with interfaces that allow access from Matlab. qpOASES is an active-set based open-source solver for QPs written in C++ which can be compiled for

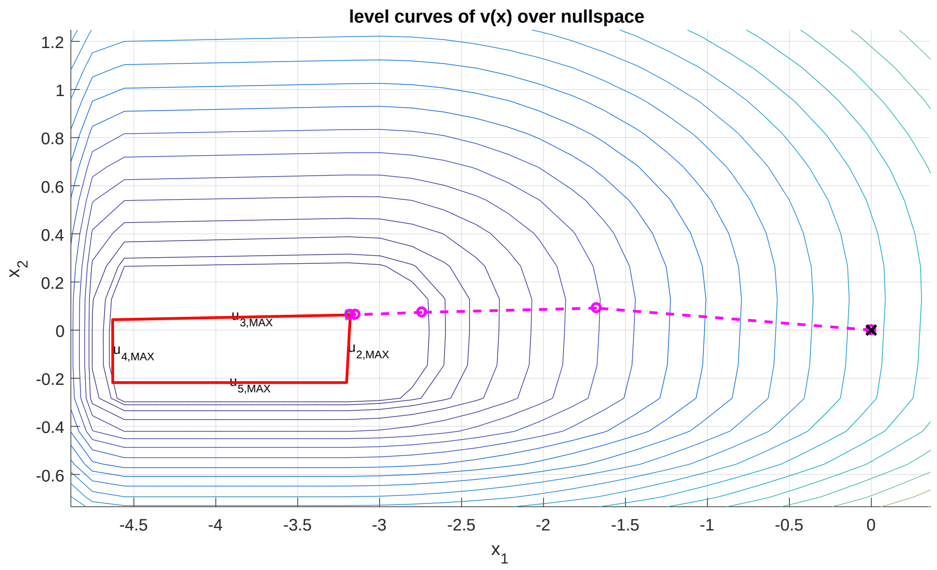

Figure 6.4.: The costs for constraint violation are zero only inside the feasible region $\Psi(\boldsymbol{u_0})$ (red polygon).

Table 6.2.: Results for solving (6.64) with $\boldsymbol{v} = \boldsymbol{v_{inf}}$.

	EPA	WLA	FXP	IPA
time	39 μs	13 μs	153 μs	258 μs
iter.	11	4	200	10
$f_u(\boldsymbol{u})$	21.7529785	21.7530014	21.7529998	21.7529593
$v(\boldsymbol{u})$	0	$4.26 \cdot 10^{-8}$	0	0
$d(\boldsymbol{u}, \boldsymbol{v})$	19.990242	19.990242	19.990242	19.990242

use in Matlab as MEX-File. All algorithms are used with their standard parameters. The results can be found in Table 6.4 which is horizontally split into three parts. The first eight rows contain the results for each algorithm in case of feasible virtual control vectors, the next eight rows are dedicated to infeasible ones with identity weighting, and the remaining rows are related to infeasible $\boldsymbol{v}$ with random $\boldsymbol{W_v}$. In either case three types of actuator weighting matrices are tested: identity, diagonal, and dense. The preferred control position is the zero vector in all generated problems, the number of optimization variables (subject to the box-like constraints Ω) is $m = 10$, and the number of equality constraints is $k = 5$. Table 6.4 not only shows the average computation time and number of iterations for each algorithm but also how many of the problems could be solved. In the feasible cases a result is as considered correct if its maximum constraint violation regarding (6.2b) and (6.2c) is below 0.01 % and the cost function value does not exceed the best solution by more than 0.01 %. A correct result for infeasible problems requires the maximum violation of (6.2c) to be below 0.01 % and the virtual control error (6.3) to be at most 0.01 % higher than the best one. The current implementation of IPA can only handle identity weighting matrices which is why merely the first three entries of Table 6.4 are filled for feasible problems. In the infeasible case the dominating part of the cost function is the virtual control error. Therefore, IPA is able to solve those problems for all types of weighting matrices. Another peculiarity of Table 6.4 is the fact that the iteration count of of CPL cannot be shown because its Matlab interface does not return a valid number. The essence of Table 6.4 is:

- QUA, GUR, and CPL are supposed to be used for larger problem sizes. Their compu-

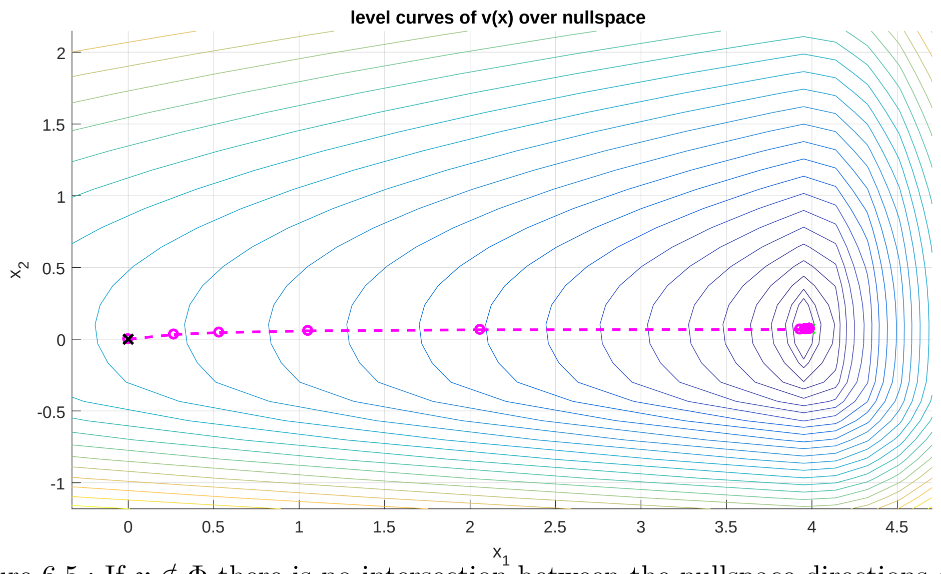

Figure 6.5.: If $\boldsymbol{v} \notin \Phi$ there is no intersection between the nullspace directions and Ω.

Table 6.3.: Results for (6.64) with $\boldsymbol{W_v} = \text{diag}(100, 3, 52)$ and $\boldsymbol{v} = \boldsymbol{v_{inf}}$.

	EPA	WLA	FXP	IPA
time	43 μs	15 μs	154 μs	396 μs
iter.	11 (2)	6	200	-
$f_u(\boldsymbol{u})$	9.7157025	9.7157024	9.6325387	9.7157025
$v(\boldsymbol{u})$	0	$1.08 \cdot 10^{-8}$	0	0
$d(\boldsymbol{u}, \boldsymbol{v})$	104.811509	104.811511	105.414924	104.811509

tation times are much higher in the low-dimensional examples considered here.

- FXP and IPA are not very reliable for solving feasible problems. Especially IPA depends very much on the selection of the weighting parameter γ. In the infeasible cases on the other hand they work satisfactorily. The maximum number of FXP-iterations has been set to 300 in the infeasible test cases. Increasing this value would also increase the number of solved problems but with higher computation time.

- Active-set based algorithms (WLA and QPO) are well-suited for low problem dimensions. However, WLA has difficulties solving problems with dense weighting matrices. The weighting parameter γ in (6.1a) has been increased to get more feasible results.

- EPA has a computation time similar to WLA for small-scale feasible examples, but it can also handle dense weighting matrices.

- In the infeasible cases with non-identity $\boldsymbol{W_v}$ the active-set algorithms (WLA and QPO) perform best for identity and diagonal actuator weighting matrices $\boldsymbol{W_u}$, followed by EPA. However, for dense $\boldsymbol{W_u}$ EPA is able to solve more problems than WLA and QPO.

- Since changing the virtual weighting matrix $\boldsymbol{W_v}$ is equivalent with changing the factorization and the actuator weighting matrix has effectively no influence on the infeasible cases IPA can also be tested.

Table 6.4.: Randomized testing of 8000 test cases for each actuator weighting matrix type with $m = 10$, $k = 5$, and $\boldsymbol{u_p} = \mathbf{0}$. The last eight rows contain the results where gradient projection has been applied. Its iteration number can be seen in brackets beside the exterior point iteration number.

	solver	identity $\boldsymbol{W_u}$			diag. $\boldsymbol{W_u}$			dense $\boldsymbol{W_u}$		
		time [ms]	*iterations*	*solved*	*time [ms]*	*iterations*	*solved*	*time [ms]*	*iterations*	*solved*
feasible	EPA	0.088	5.2	7999	0.088	5.0	8000	0.104	7.5	8000
	WLA	0.081	3.3	7999	0.081	3.3	8000	0.079	3.0	2244
	FXP	0.300	300	3915	0.296	300	153	0.324	300	4
	IPA	0.501	12.4	5882	-	-	-	-	-	-
	QUA	2.726	5.3	8000	2.794	6.2	7994	2.871	6.7	8000
	GUR	1.550	8.5	8000	1.378	8.9	8000	1.710	9.6	8000
	CPL	4.126	?	7994	4.195	?	8000	4.358	?	7998
	QPO	0.251	17.5	8000	0.252	17.9	8000	0.284	21.6	8000
infeasible, $\boldsymbol{W_v} = \boldsymbol{I}$	EPA	0.124	12.6	8000	0.128	12.7	8000	0.164	17.5	8000
	WLA	0.100	11.4	8000	0.101	11.5	8000	0.101	11.8	7985
	FXP	0.391	400	7999	0.397	400	7999	0.395	400	7998
	IPA	0.393	9.3	8000	0.397	9.3	8000	0.396	9.4	7999
	QUA	3.007	8.5	8000	3.020	8.5	8000	3.022	8.5	7985
	GUR	1.847	20.3	8000	1.935	20.2	8000	1.853	20.0	7985
	CPL	4.464	?	8000	4.881	?	8000	4.736	?	7985
	QPO	0.128	14.8	8000	0.128	14.9	8000	0.121	13.2	7985
infeasible, rand. $\boldsymbol{W_v}$	EPA	0.168	13.6 (6.9)	8000	0.162	12.6 (6.9)	8000	0.203	18.5 (7.3)	7999
	WLA	0.097	8.2	8000	0.096	8.1	8000	0.099	9.2	7756
	FXP	0.393	400	3707	0.391	400	3706	0.395	400	3716
	IPA	0.498	12.5	7994	0.497	12.5	7994	0.500	12.5	7990
	QUA	2.911	7.4	8000	2.904	7.4	8000	2.911	7.4	7756
	GUR	1.776	19.4	8000	1.785	19.8	8000	1.803	21.6	7755
	CPL	4.356	?	7997	4.341	?	7996	4.357	?	7753
	QPO	0.134	14.9	8000	0.131	14.4	8000	0.139	16.3	7756

6.5.2. Medium-scale problems

With rising m and k the benefit of the proposed approach becomes clearer. The outcome of randomized testing with $m = 50$ and $k = 25$ is contained in Table 6.5. EPA is the fastest solver in these examples. FXP would require more iterations for feasible virtual controls and infeasible ones with random $\boldsymbol{W_v}$ whereas its results for $\boldsymbol{W_v} = \boldsymbol{I}$ are nearly as good as those from EPA. Ignoring the fact that IPA cannot handle non-identity $\boldsymbol{W_u}$ it is very effective for infeasible $\boldsymbol{v}$. One can summarize Table 6.5 as follows:

- The computation time depends on the feasibility of $\boldsymbol{v}$ for all tested algorithms. Except for QPO and IPA, all solvers need more time for the infeasible problems.
- The maximum iteration number (300) of IPA in the feasible test cases is too low and so none of the problems has been solved. The test is not repeated as more iterations require

more computation time and make FXP less competitive to the other solvers. The same observation can be made for FXP's performance in case of infeasible $\boldsymbol{v}$ and random $\boldsymbol{W_v}$.

- Considering computation time and number of solved problems EPA achieves very good results.

Table 6.6 shows how the considered algorithms deal with problems which are ten times larger than those in the last section. It gives an estimate on how well they scale with an increasing number of optimization variables and constraints. In contrast to the examples in Tables 6.4 and 6.5 the preferred control vector is different from zero in all of the randomly generated test cases, i.e. the linear part of the cost function (6.2a) does not vanish. The key aspects of Table 6.6 are:

- FXP cannot handle cost functions with linear terms (for lower dimensional problems this is also true). However, in case of infeasible problems it is applicable (because of the high weighting of the virtual control error) and together with EPA it requires the least computation time.

- The number of iterations of active-set based algorithms rises significantly. For feasible problems QPO requires the most computation time. WLA is only applicable for identity and diagonal weighting matrices whereas most of the problems with dense $\boldsymbol{W_u}$ could not be solved although γ has been increased significantly. The average computation time of WLA is roughly 100 times higher than in Table 6.4 although the problem size is only 10 times greater.

- EPA achieves the best results regarding computation time and number of solved problems.

Figure 6.6 demonstrates the advantage of transforming the optimization into nullspace if there are equality constraints involved. It depicts the average execution times for problems with identity weighting matrix, non-zero linear term in the cost function, a constant nullspace dimension of $m-k = 25$, and an increasing number of optimization variables m. One recognizes that the increase of EPA's execution time is clearly below the other algorithms. The results in the remainder of this section demonstrate the capabilities of EPA in solving problems larger than they usually occur in CA. QPs appear in various engineering applications and so there are lots of possibilities to apply EPA in other areas apart from CA. Hence, the considered problem type is now extended by general inequality constraints which have to be fulfilled additionally to the box-like constraints in the examples before. Table 6.7 contains the average results for problems with $m = 500$ optimization variables, box-like inequality constraints, no equality constraints ($k = 0$), and $i = 500$ extra inequality constraints which can be written in matrix form as

$$\boldsymbol{A_{iq}u} \leq \boldsymbol{b_{iq}}. \tag{6.65}$$

Note that only four algorithms are tested because the others require significantly more computation time and do not find a solution in the majority of cases anyway. In summary, Table 6.7 shows that EPA is able to solve the problems significantly faster than the other algorithms, especially for dense weighting matrices. In these cases CPL had some difficulties as indicated by the small number of solved problems.

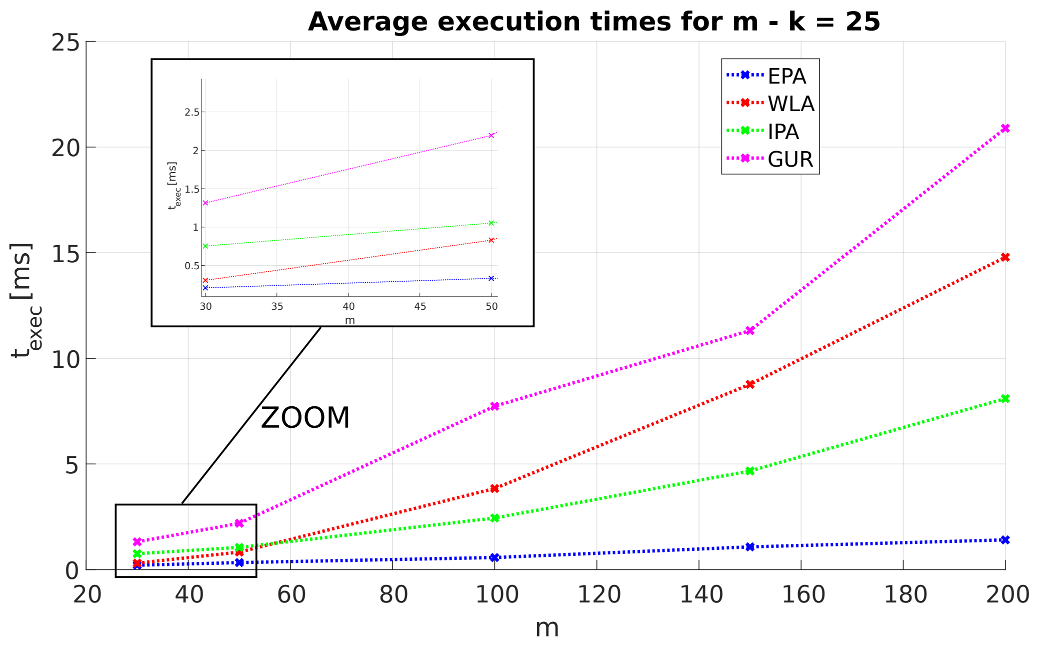

Figure 6.6.: The average execution time of EPA for problems with $\boldsymbol{v} \in \Phi$ and $m - k = 25$ scales very well with an increasing number of optimization variables m.

6.5.3. Simulation results

In this section EPA is tested in a closed-loop simulation with CA. Given the linear plant model

$$\dot{\boldsymbol{x}}_{\boldsymbol{st}} = \underbrace{\begin{bmatrix} 0 & 2 & 2 \\ 0 & 2 & 1 \\ 1 & -1 & -2 \end{bmatrix}}_{\boldsymbol{A_{st}}} \boldsymbol{x}_{\boldsymbol{st}} + \underbrace{\begin{bmatrix} -7 & -3 & 6 & 4 & 1 \\ 0 & 2 & -4 & 8 & -7 \\ 10 & -5 & 1 & 10 & -7 \end{bmatrix}}_{\boldsymbol{B_u}} \boldsymbol{u} \tag{6.66}$$

subject to input constraints $-\boldsymbol{u_{max}} \leq \boldsymbol{u} \leq \boldsymbol{u_{max}}$ with $\boldsymbol{u_{max}}^T = [2\ 7\ 14\ 5\ 9]$ the control goal is the stabilization of the origin. The initial state reads as $\boldsymbol{x}_{\boldsymbol{st,0}}^T = [7\ -2\ -4]$ and the input matrix factorization $\boldsymbol{B_v B} = \boldsymbol{B_u}$ is conducted according to (6.32). A linear state controller is developed for $\dot{\boldsymbol{x}}_{\boldsymbol{st}} = \boldsymbol{A_{st} x_{st}} + \boldsymbol{B_v v}$ such that all eigenvalues of the closed-loop system matrices are at -12. The weighting matrix is chosen as $\boldsymbol{W} = \mathrm{diag}(\frac{1}{u_{max,1}}, ..., \frac{1}{u_{max,m}})$ and the preferred control positions are $\boldsymbol{u_p} = \boldsymbol{0}$. Furthermore a sinusoidal disturbance

$$d(t) = 70\mathrm{si}_{10t} \tag{6.67}$$

affects the second state variable in order to increase the necessary control action during the simulation. The CA problem is solved with WLA (with $\boldsymbol{W_v} = \boldsymbol{I_k}$), EPA, and EPA-HS (hot-start). Figures 6.7 and 6.8 show states and control inputs over time. One recognizes that EPA and EPA-HS yield the same solution as WLA. The benefit of exploiting the hot-start ability of EPA can be recognized in Figure 6.9.

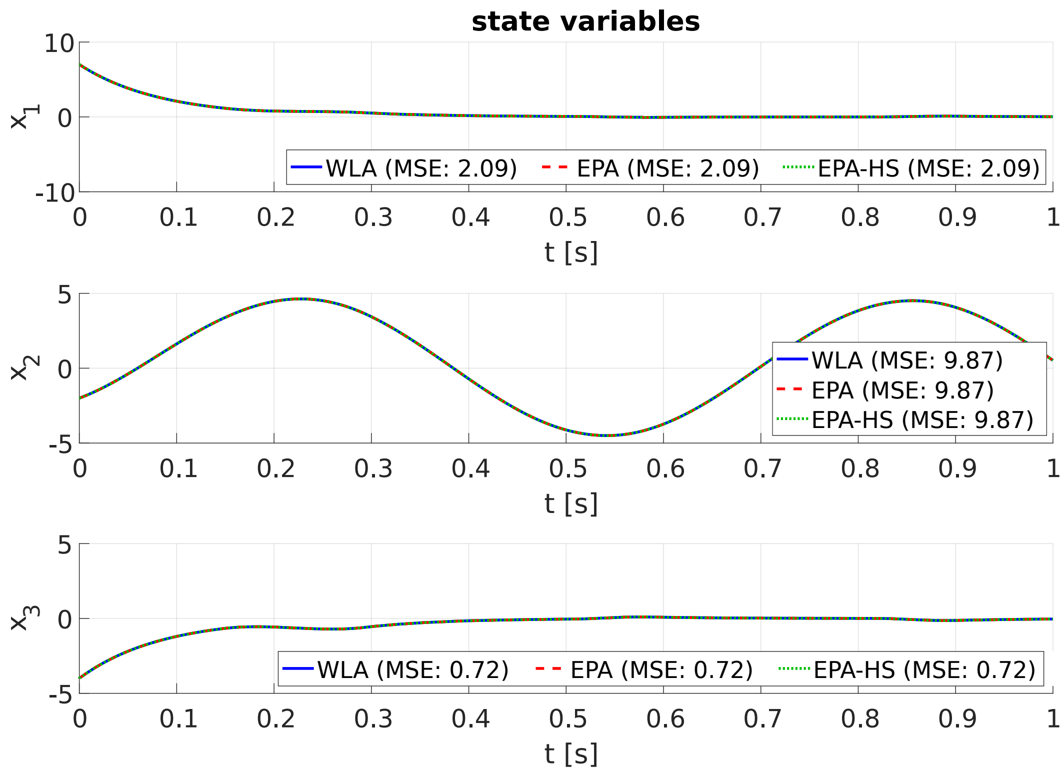

Figure 6.7.: As expected the state curves are identical for all algorithms. The second state variable shows the influence of the disturbance (6.67).

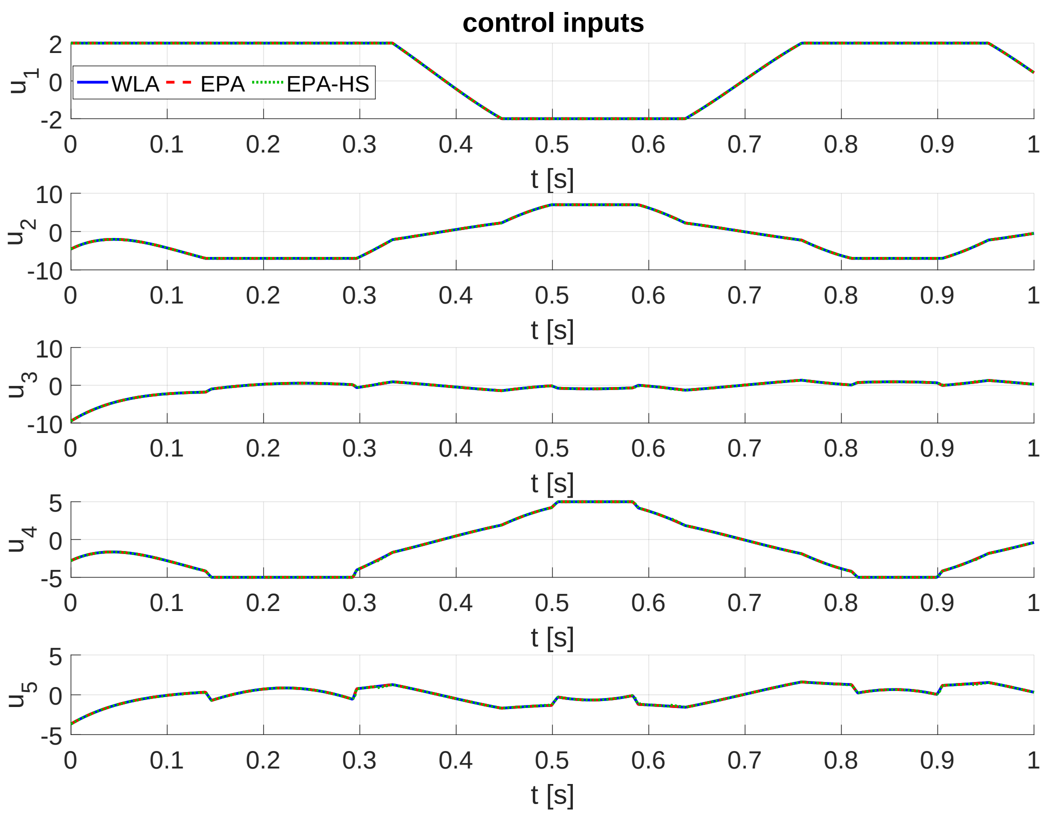

Figure 6.8.: These actuator signals are the results of the three CA-algorithms.

Table 6.5.: Randomized testing of 8000 test cases for each actuator weighting matrix type with $m = 50$, $k = 25$, and $\boldsymbol{u_p} = \mathbf{0}$. The last eight rows contain the results where gradient projection (iteration number in brackets) has been applied. The weighting factor γ for WLA is increased by a factor of $1e4$ in case of dense actuator weighting matrices in order to obtain feasible results.

		identity $\boldsymbol{W_u}$			diag. $\boldsymbol{W_u}$			dense $\boldsymbol{W_u}$		
	solver	*time [ms]*	*iterations*	*solved*	*time [ms]*	*iterations*	*solved*	*time [ms]*	*iterations*	*solved*
feasible	EPA	0.365	6.0	7999	0.355	5.7	8000	0.453	8.2	7999
feasible	WLA	0.741	13.1	7993	0.675	12.1	7998	0.931	18.9	7788
feasible	FXP	-	-	0	-	-	0	-	-	0
feasible	IPA	1.011	14.6	6537	-	-	0	-	-	0
feasible	QUA	6.240	6.9	7999	7.438	8.1	7993	8.857	8.5	7999
feasible	GUR	2.275	9.9	7999	2.300	10.4	8000	5.681	14.7	8000
feasible	CPL	5.152	?	7999	4.563	?	7999	9.214	?	7996
feasible	QPO	8.122	121.8	7999	9.319	144.9	7999	8.996	118.5	7999
infeasible, $\boldsymbol{W_v} = \boldsymbol{I}$	EPA	0.430	13.0	8000	0.466	13.6	8000	0.640	16.7	8000
infeasible, $\boldsymbol{W_v} = \boldsymbol{I}$	WLA	1.911	54.8	8000	1.915	55.4	8000	1.914	54.0	7966
infeasible, $\boldsymbol{W_v} = \boldsymbol{I}$	FXP	0.628	400	8000	0.626	400	8000	0.631	400	8000
infeasible, $\boldsymbol{W_v} = \boldsymbol{I}$	IPA	0.750	9.7	8000	0.749	9.7	8000	0.753	9.7	8000
infeasible, $\boldsymbol{W_v} = \boldsymbol{I}$	QUA	7.951	9.8	8000	7.872	9.8	8000	7.919	9.9	7966
infeasible, $\boldsymbol{W_v} = \boldsymbol{I}$	GUR	5.394	30.8	8000	5.469	30.9	8000	4.708	24.1	7966
infeasible, $\boldsymbol{W_v} = \boldsymbol{I}$	CPL	6.860	?	8000	6.755	?	8000	6.817	?	7966
infeasible, $\boldsymbol{W_v} = \boldsymbol{I}$	QPO	2.457	104.8	8000	2.496	107.2	8000	1.265	56.4	7966
infeasible, rand. $\boldsymbol{W_v}$	EPA	0.947	15.2 (26.4)	7994	0.961	15.4 (26.4)	7994	1.197	21.9 (26.3)	7997
infeasible, rand. $\boldsymbol{W_v}$	WLA	1.588	38.6	8000	1.603	39.3	8000	1.747	45.0	7905
infeasible, rand. $\boldsymbol{W_v}$	FXP	0.620	400	1006	0.615	400	1006	0.613	400	940
infeasible, rand. $\boldsymbol{W_v}$	IPA	1.144	15.9	7999	1.144	15.9	7999	1.136	15.9	7999
infeasible, rand. $\boldsymbol{W_v}$	QUA	7.542	9.2	8000	7.589	9.2	8000	7.336	9.3	7905
infeasible, rand. $\boldsymbol{W_v}$	GUR	5.420	26.4	7999	5.421	26.4	7999	7.827	48.5	7903
infeasible, rand. $\boldsymbol{W_v}$	CPL	6.559	?	7991	6.602	?	7989	6.583	?	7903
infeasible, rand. $\boldsymbol{W_v}$	QPO	2.374	93.2	8000	2.574	103.3	8000	2.867	114.5	7905

Table 6.6.: Randomized testing of 5000 test cases for each weighting matrix type with $m = 100$, $k = 50$, and random $\boldsymbol{u_p}$. The last eight rows contain the results where gradient projection (iteration number in brackets) has been applied. The weighting factor γ for WLA is increased by a factor of $1e4$ in case of dense actuator weighting matrices in order to obtain feasible results.

	solver	identity $\boldsymbol{W_u}$			diag. $\boldsymbol{W_u}$			dense $\boldsymbol{W_u}$		
		time [ms]	*iter.*	*solved*	*time [ms]*	*iter.*	*solved*	*time [ms]*	*iter.*	*solved*
feasible	EPA	1.066	7.8	4999	1.306	10	4999	1.419	10.6	4998
	WLA	7.369	36.2	5000	8.953	50.1	5000	7.503	34.0	698
	FXP	-	-	0	-	-	0	-	-	0
	IPA	2.554	17.5	4479	-	-	0	-	-	0
	QUA	15.934	8.0	4999	17.313	8.3	5000	32.541	9.9	5000
	GUR	6.011	10.6	5000	6.426	11.2	5000	23.091	20.6	5000
	CPL	9.416	?	5000	9.428	?	5000	9.418	?	4998
	QPO	62.949	291	5000	67.484	317	5000	69.201	258	4992
infeasible, $\boldsymbol{W_v} = \boldsymbol{I}$	EPA	1.244	13.0	5000	1.337	13.0	5000	1.783	16.8	5000
	WLA	12.821	98.5	4739	12.763	98.8	4655	13.140	99.4	4093
	FXP	1.659	400	4538	-	-	0	1.683	400	4400
	IPA	1.655	10.0	5000	1.632	10	4999	1.678	10.0	4989
	QUA	18.416	10.1	5000	18.265	10.1	5000	18.309	10.2	4983
	GUR	20.544	34.5	5000	20.733	34.5	5000	19.299	31.4	4983
	CPL	11.177	?	5000	10.951	?	5000	11.240	?	4983
	QPO	18.063	235	5000	18.231	239	5000	9.376	128	4983
infeasible, rand. $\boldsymbol{W_v}$	EPA	3.338	16.3 (48.5)	5000	3.577	18.4 (48.9)	5000	4.076	23.3 (48.4)	5000
	WLA	11.710	78.0	4993	12.269	84.2	4866	12.448	88.1	4702
	FXP	-	-	0	-	-	0	1.637	400	17
	IPA	2.745	17.6	5000	2.756	17.6	4996	2.719	17.6	4993
	QUA	15.379	10.3	5000	15.473	10.3	5000	15.300	10.3	4917
	GUR	20.792	37.1	4998	18.285	37.0	4998	30.152	57.5	4917
	CPL	12.868	?	4998	12.828	?	4996	12.909	?	4917
	QPO	16.269	197	5000	16.625	203	5000	20.997	259	4917

Table 6.7.: Randomized testing of 2500 test cases for each weighting matrix type with $m = 500$, $k = 0$, $i = 500$, and $\boldsymbol{c_u} \neq \boldsymbol{0}$

solver	identity $\boldsymbol{W_u}$			diag. $\boldsymbol{W_u}$			dense $\boldsymbol{W_u}$		
	time [ms]	*iter.*	*solved*	*time [ms]*	*iter.*	*solved*	*time [ms]*	*iter.*	*solved*
EPA	63.49	5.1	2500	84.97	5.3	2500	157.3	16.2	2499
QUA	1272	9.3	2500	1538	10.1	2500	3945	15.2	2500
GUR	182.4	14.9	2500	180.1	15	2500	6453	124	2500
CPL	236.0	?	2500	243.6	?	2500	1078	?	21

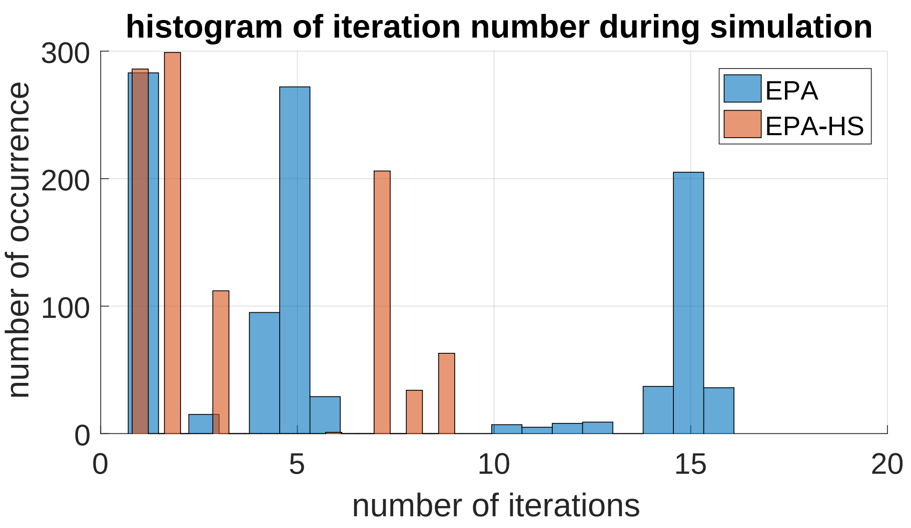

Figure 6.9.: The hot-start feature of EPA-HS reduces the number of iterations to find the solution.

6.6. Conclusion

In this chapter an efficient solver (EPA) for convex quadratic programs is presented. The underlying principles are a transformation of the problem into nullspace for efficiently addressing equality constraints and the augmentation of the cost function with an additional term dedicated to the inequality constraints. The adaption law of the penalty parameter ensures the convergence of the the algorithm to the minimum. For infeasible virtual controls EPA computes a feasible vector $\boldsymbol{u}$ which has either minimum distance to the desired value in real control space $\mathbb{R}^m$ or alternatively minimizes the weighted distance in virtual control space $\mathbb{R}^k$. It is shown that for certain problems both properties can be achieved at the same time. Randomized testing shows the competitiveness of EPA with established solvers for small (up to 10 optimization variables) and medium-scale (several hundreds of optimization variables) problems. Similar to active-set based algorithms EPA-HS facilitates hot-starts, i.e. the initialization with results from previous executions for similar problems. This property leads to a reduction of the required iteration number and makes it particularly useful as a CA-method. However, due to the fact that its computational effort scales well with the problem size it has potential to be used in other applications too.

Part III.

Applications

7. ICE start-up Control for HEV Powertrains

In this chapter the problem of starting the ICE of a HEV while driving (see Section 1.2 for more details) is solved with the aid of a CA-based control structure. The main goal during the starting process of the ICE is to increase its speed by means of a clutch which connects it to the rest of the powertrain. Once the desired speed is reached combustion is started. The first presented approach utilizes pure speed control to harmonize the shaft speeds. The powertrain acceleration is kept constant until ω_1 and ω_2 match. The second method is designed in order to simultaneously meet the overall driving torque demand of the driver. While a speed controller operates on ω_1, the desired torque related to the accelerator pedal position is maintained by torque control. Both approaches are tested in simulations where the plant dynamics are governed by (1.5) - (1.10) and (1.12). The required model parameters for simulating the driving shaft dynamics are condensed in Table 7.1. The actuator limits forming the feasible subset of

parameter	value	parameter	value
J_1	$0.25\,kgm^2$	μ_{slip}	0.2
J_2	$0.2\,kgm^2$	μ_{stick}	0.3
k_1	$0.5\,\frac{kgm^2}{s}$	r_a	$0.125\,m$
k_2	$0.5\,\frac{kgm^2}{s}$	n_s	8
τ_1	$0.12\,s$	τ_3	$0.01\,s$
τ_2	$0.05\,s$	-	-

Table 7.1.: Driving shaft model parameters.

controls $\Omega \subset \mathbb{R}^3$ are

$$\boldsymbol{u_{min}} = [0 \;\; -250 \;\; 0]^T \tag{7.1a}$$

$$\boldsymbol{u_{max}} = [140 \;\; 250 \;\; 3000]^T \tag{7.1b}$$

The start-up procedure can be divided into three phases:

1. acceleration phase: The ICE speed is increased by means of EM and clutch.
2. ignition and torque blending: The ICE is started and its torque starts to increase.
3. clutch lock-up: The clutch actuator's axial force is chosen such that the shafts remain permanently connected, i.e. $|\overline{T}_c| \leq T_{c,max}$. The system dynamics reduce to (1.11) and the clutch is no torque source any more. The majority of the required torque should come from the ICE.

In a CA-based control system it has to be guaranteed that the commanded controls $\boldsymbol{u}$ are realized properly. This is the responsibility of *low-level control* (see Figure 1.1) which consists of individual actuator controllers. In the present case the existence of motor and clutch controllers is presumed, i.e. each components and its controller are considered as single system with input and output torques. The interested reader is referred to Appendix C where the development of a clutch actuator controller is exhaustively explained.

7.1. Pure speed control

In this approach a speed controller is used during start-up to align both shaft speeds. The speed reference value depends on the driving situation, i.e., the current speed and whether the car is accelerating or not. The starting point of considerations is model (1.14). The first steps are conducting an input matrix factorization of $\boldsymbol{B_{u1}}$ and introducing the virtual control vector $\boldsymbol{v}$. In the present case the factorization is chosen as

$$\boldsymbol{B_{u1}}(\boldsymbol{\omega}) = \boldsymbol{B_v B} = \begin{bmatrix} \frac{1}{J_1} & 0 \\ 0 & \frac{1}{J_2} \end{bmatrix} \begin{bmatrix} 1 & 0 & -\mu(\boldsymbol{\omega}) \\ 0 & 1 & \mu(\boldsymbol{\omega}) \end{bmatrix}. \tag{7.2}$$

This leads to a redundancy-free model with two virtual controls

$$\begin{aligned} \dot{\boldsymbol{\omega}} = \boldsymbol{A\omega} + \boldsymbol{B_v v} &= \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} -\frac{k_1}{J_1} & 0 \\ 0 & -\frac{k_2}{J_2} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{J_1} & 0 \\ 0 & \frac{1}{J_2} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\ \boldsymbol{y} = \boldsymbol{C\omega} &= \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} \end{aligned} \tag{7.3}$$

which is used for developing a linear state feedback controller

$$\boldsymbol{v} = -\boldsymbol{K\omega} + \boldsymbol{V r} \tag{7.4}$$

with controller parameters $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{2\times 2}$ and the reference signal $\boldsymbol{r} \in \mathbb{R}^2$. The gain matrix $\boldsymbol{K}$ is computed such that the eigenvalues of the closed-loop system matrix $(\boldsymbol{A} - \boldsymbol{B_v K})$ lie at -10 and -13. Due to $\boldsymbol{A}$'s diagonal structure $\boldsymbol{K}$ is also diagonal and reads as

$$\boldsymbol{K} = \begin{bmatrix} k_{11} & 0 \\ 0 & k_{22} \end{bmatrix} = \begin{bmatrix} 10J_1 - k_1 & 0 \\ 0 & 13J_2 - k_2 \end{bmatrix}. \tag{7.5}$$

Vanishing steady-state errors for step reference signals $\boldsymbol{r}$ are achieved by

$$\boldsymbol{V} = -\left[\boldsymbol{C}\left(\boldsymbol{A} - \boldsymbol{B_v K}\right)^{-1} \boldsymbol{B_v}\right]^{-1}. \tag{7.6}$$

Next, the virtual control computed by (7.4) is distributed among ICE, EM, and the clutch actuator by CA. The reference signals for speed control depend on the current driving situation. If the current speed is more or less constant it is kept constant during start up (Figure 7.1) whereas in an acceleration phase the speed gradient is kept constant (Figure 7.2). The time instant where the ICE reference starts to rise is denoted as T_{start} and the rise time as T_{ramp}. During the pre start-up phase ($t < T_{start}$) the car is exclusively powered by the EM.

7.1.1. Pseudoinverse-based approach

Since it is already roughly known in advance which actuators should dominate in which start-up phase the CA algorithm should offer the possibility to incorporate those ideas. WPINV is one of the simplest CA methods meeting this criterion by means of the weighting matrix $\boldsymbol{W}$ whose composition is described in the following. In this case a variable weighting matrix has proven successful. Its structure is

$$\boldsymbol{W} = \begin{bmatrix} c_{11}e^{c_{12}(\omega_1-\omega_2)^2} & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & c_{31} - c_{32}\sigma(T_{start}, t) \end{bmatrix} \tag{7.7}$$

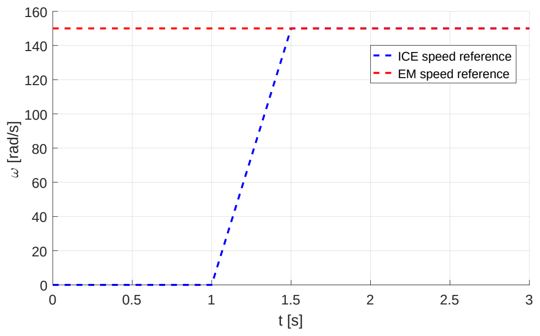

Figure 7.1.: Speed reference for ICE start-up in constant speed phase.

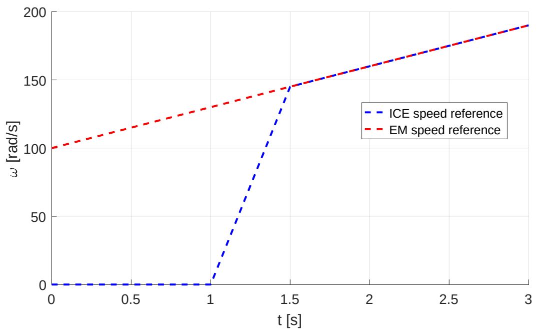

Figure 7.2.: Speed reference for start-up during acceleration.

with scalar constants $0 < c_{11} < c_2$, $0 < c_{32} < c_{31} < c_2$, and $0 < c_{12}$. The exponential term in (7.7) is large during EM-only operation and gets smaller for a decreasing speed deviation leading to a gradually increasing ICE torque. Parameter c_{31} has to be large in pre start-up phase to prevent clutch usage. At the beginning of start-up the step function reduces the weight for clutch actuation. The parameters yielding the results of this section are specified in Table 7.2. The results for constant and ramp EM speed references are shown in Figures 7.3

parameter	value	parameter	value
c_{11}	1	c_{12}	0.001
c_2	5	-	-
c_{31}	1000	c_{32}	999.9

Table 7.2.: Weighting matrix parameters for pseudoinverse computation.

and 7.4 respectively. One recognizes that in both cases the acceleration phase of the ICE is

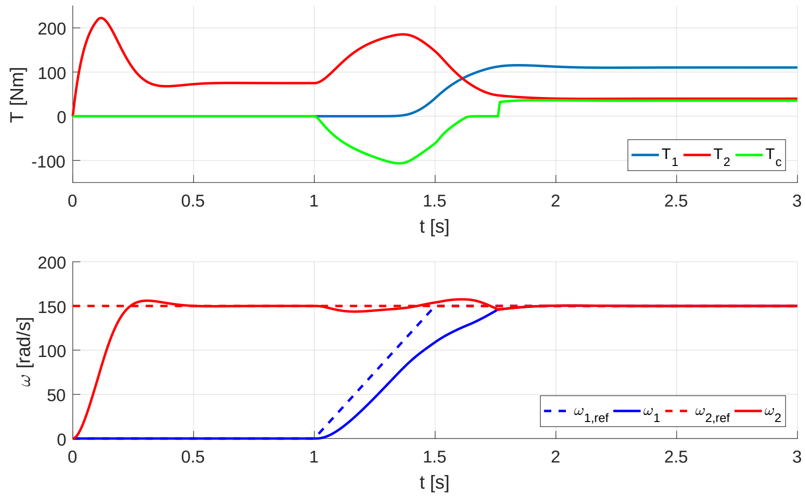

Figure 7.3.: Allocation results using WPINV with (7.7) for constant speed reference.

done with EM and clutch only as requested. The clutch torque T_c is negative if the EM side of the shaft propels the ICE side. ICE torque starts to rise not until $t \approx 1.4\,s$. The parameters where chosen such that at the end of speed matching ($t \approx 1.75$ s) an increasing part of the torque comes from the ICE resulting in $T_c > 0$. Start-up is completed after about 750 ms in both cases. The subsequent lock-up phase is explained in Section 7.1.3.

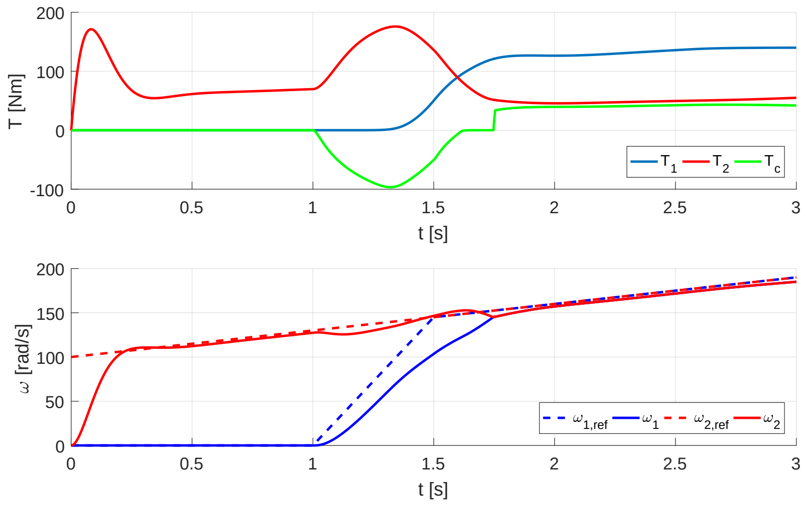

Figure 7.4.: Allocation results using WPINV with (7.7) in case of acceleration.

7.1.2. Direct allocation with weighted nullspace modification

WPINV is not able to take the actuator constraints into account. DA on the other hand is a constrained CA method always able to find a correct allocation, provided that one exists. However, DA exhibits no design parameters which enable influencing the result comparable to the weighting matrix. Therefore, it is natural to seek a combination of DA and weighting matrix. The proposed approach starts with DA and computes an initial solution $\boldsymbol{u_{DA}} = f_{DA}(\boldsymbol{v})$ with $\forall \boldsymbol{v} \in \mathbb{R}^3 : \boldsymbol{u_{DA}} \in \Omega$ and $\forall \boldsymbol{v} \in \Phi : \boldsymbol{v} = \boldsymbol{B}\boldsymbol{u_{DA}}$. Let $\boldsymbol{N} \in \mathbb{R}^{3\times 1}$ span the right nullspace of the control effectivity matrix, i.e. $\mathrm{span}(\boldsymbol{N}) = \mathcal{N}_\mathrm{r}(\boldsymbol{B})$ implying $\boldsymbol{BN} = \boldsymbol{0}$. Now one can formulate an optimization problem in nullspace as

$$\min_x \left(\boldsymbol{u_{DA}} + \boldsymbol{N}x\right)^T \boldsymbol{W} \left(\boldsymbol{u_{DA}} + \boldsymbol{N}x\right) \tag{7.8}$$

with $\boldsymbol{W} \succ 0$ being a weighting matrix such as (7.7). Considering that $\left(\boldsymbol{N}^T\boldsymbol{W}\boldsymbol{N}\right) \succ 0$ [24], the solution of (7.8) reads as

$$x = -\left(\boldsymbol{N}^T\boldsymbol{W}\boldsymbol{N}\right)^{-1} \boldsymbol{N}^T\boldsymbol{W}\boldsymbol{u_{DA}}. \tag{7.9}$$

The combined allocation result of DA and (7.9) can be written as

$$\boldsymbol{u} = \boldsymbol{u_{DA}} + \boldsymbol{N}\alpha x \tag{7.10}$$

with

$$\alpha = \max_{0\leq\alpha\leq 1} \alpha \quad \text{such that} \quad \left(\boldsymbol{u_{DA}} + \boldsymbol{N}\alpha x\right) \in \Omega. \tag{7.11}$$

Due to the convexity of (7.8) and the fact that x is just a scalar, (7.11) is approximately solved by a simple iterative line search algorithm starting with $\alpha = 1$. The results of this approach are presented in Figures 7.5 and 7.6. In principle the torque and speed curves are similar to the WPINV results. The overall start-up process takes slightly longer than in Figures 7.3 and 7.4. But on the other hand the behavior of T_c is much smoother. Comparisons of the effective torque (1.6) in Figures 7.7 and 7.8 reveal that the magnitude of torque jerks is significantly reduced by the use of the modified DA strategy. At $t \approx 1.8\,s$ the lock-up phase begins which is described in the following Section 7.1.3.

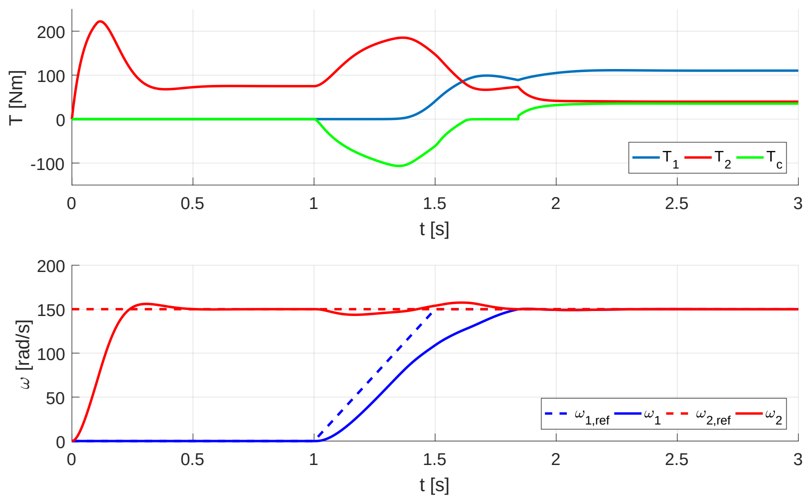

Figure 7.5.: Allocation results using modified DA with (7.7) for constant speed reference.

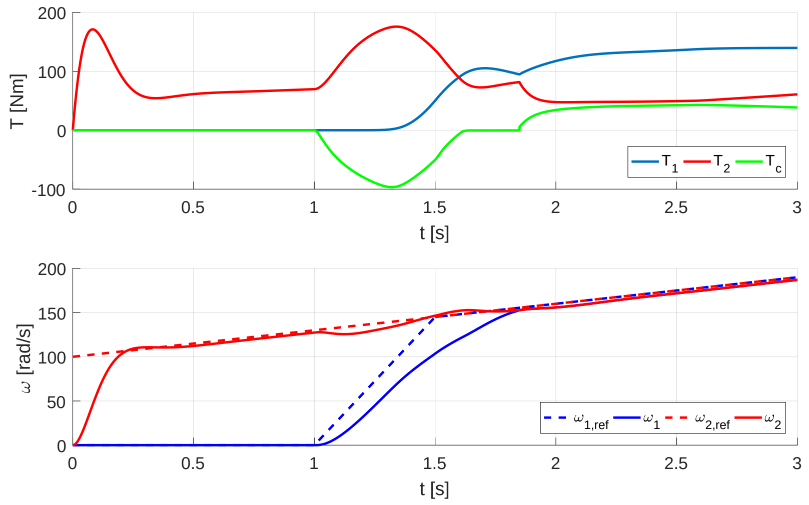

Figure 7.6.: Allocation results using modified DA with (7.7) in case of acceleration.

7.1.3. Control in lock-up phase

When the clutch enters locked mode the degrees of freedom of the considered system reduce to one, i.e. (1.11) governs its dynamics and $\omega_1 \equiv \omega_2$ (see Section 1.2.2). The clutch actuator's axial force does not directly influence the shaft speed any more. Instead, it determines how much torque can be transmitted over the clutch via (1.8). One possibility to deal with this situation is switching to another state controller developed for

$$\dot{\omega} = \frac{1}{J_1 + J_2} \left[-(k_1 + k_2)\omega + v\right] \tag{7.12a}$$

$$y = \omega \tag{7.12b}$$

which calculates a scalar virtual control v and solving the CA task $v = \tilde{\boldsymbol{B}}\tilde{\boldsymbol{u}}$ with $\tilde{\boldsymbol{B}} = [1 \;\; 1]$ and $\tilde{\boldsymbol{u}} = [T_1 \;\; T_2]^T$ afterwards. However, this switching is evitable because it can be shown that in locked mode and for equal speed references $r_1 = r_2$ the closed-loop dynamics of the coupled second-order system and the reduced first-order system with corresponding linear state controllers are the same (see Appendix B for more details). Hence, only CA has to explicitly

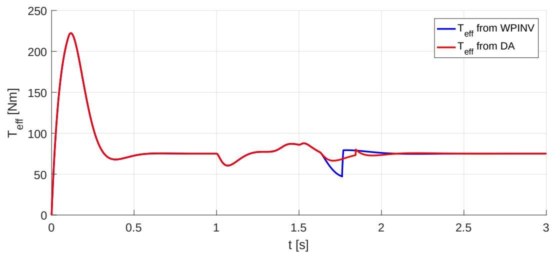

Figure 7.7.: Comparison of the effective torque (1.6) resulting from WPINV and modified DA in case of constant speed.

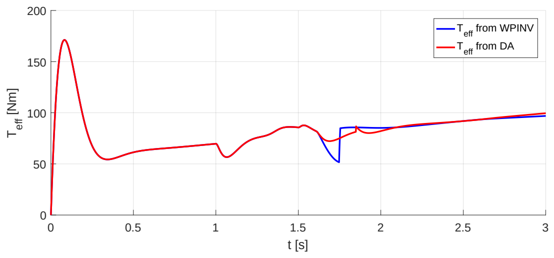

Figure 7.8.: Comparison of the effective torque (1.6) resulting from WPINV and modified DA in case of acceleration.

consider the loss of F_{ax} as actuator in the narrower sense. The necessary F_{ax} for maintaining locked mode of the clutch is computed from the estimated transmitted torque (1.9) and not from CA.

At first, WPINV and DA without the modifications of Section 7.1.2 use $\boldsymbol{B}$ from (7.2) to determine an initial solution $\boldsymbol{u_0} = [u_{0,1} \quad u_{0,2} \quad u_{0,3}]^T$. Because of the last column of $\boldsymbol{B}$ being zero in locked mode the weighted pseudoinverse reads as

$$\boldsymbol{B}^{\#} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}. \tag{7.13}$$

As a consequence, both WPINV and DA calculate $\boldsymbol{u_0} = [u_{0,1} \quad u_{0,2} \quad 0]^T$. In order to achieve the desired torque distribution between ICE and EM the same principle as in Section 7.1.2 is applied *after* the computation of $\boldsymbol{u_0}$. Instead of (7.8) the optimization problem

$$\min_{\tilde{x}} \left(\tilde{\boldsymbol{u}}_{\boldsymbol{0}} + \tilde{\boldsymbol{N}}\tilde{x}\right)^T \tilde{\boldsymbol{W}} \left(\tilde{\boldsymbol{u}}_{\boldsymbol{0}} + \tilde{\boldsymbol{N}}\tilde{x}\right) \tag{7.14}$$

with $\tilde{\boldsymbol{u}}_{\boldsymbol{0}} = [u_{0,1} \quad u_{0,2}]^T$, $\tilde{\boldsymbol{N}} = [1 \quad -1]^T$, and $\tilde{\boldsymbol{W}}$ being the top-right submatrix of (7.7) is solved. Note that $\text{span}(\tilde{\boldsymbol{N}}) = \mathcal{N}_{\text{r}}([1 \quad 1]) = \mathcal{N}_{\text{r}}(\tilde{\boldsymbol{B}})$, i.e. $\tilde{\boldsymbol{N}}$ spans the right nullspace of the reduced control effectivity matrix $\tilde{\boldsymbol{B}}$. Denoting the first two dimensions of Ω as $\tilde{\Omega}$ the commanded controls for ICE and EM are

$$\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{u}}_{\boldsymbol{0}} + \tilde{\boldsymbol{N}}\tilde{\alpha}\tilde{x} \tag{7.15}$$

with $\tilde{\alpha} = \max\limits_{0 \leq \tilde{\alpha} \leq 1} \tilde{\alpha}$ such that $\left(\tilde{\boldsymbol{u}}_{\boldsymbol{0}} + \tilde{\boldsymbol{N}}\tilde{\alpha}\tilde{x}\right) \in \tilde{\Omega}$.

7.2. Combining torque and speed control

Usually, a car's powertrain is torque-controlled with the reference signal coming from the driver via the accelerator pedal position. The velocities of wheels, differential gears, transmission, and driving shaft are not explicitly controlled, instead they are results of the torque T_{eff} and the current driving resistance. Another option for realizing ICE-startup incorporates a combination of torque and speed control. At ICE-startup a speed controller is activated to ensure that $\omega_1 \to \omega_2$ while T_{eff} still follows the reference by means of the torque controller. In further consequence a concept based on the dynamic allocation approach from [2] is presented.

In case of a slipping clutch the dynamics of shafts and actuators are summarized in

$$\dot{\boldsymbol{x}}_{\boldsymbol{s}} = \underbrace{\begin{bmatrix} -\frac{1}{\tau_1} & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{\tau_2} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\tau_{ax}} & 0 & 0 \\ \frac{1}{J_1} & 0 & -\frac{\mu(\boldsymbol{\omega})}{J_1} & -\frac{k_1}{J_1} & 0 \\ 0 & \frac{1}{J_2} & \frac{\mu(\boldsymbol{\omega})}{J_2} & 0 & -\frac{k_2}{J_2} \end{bmatrix}}_{\boldsymbol{A_s}} \boldsymbol{x_s} + \underbrace{\begin{bmatrix} \frac{1}{\tau_1} & 0 & 0 \\ 0 & \frac{1}{\tau_2} & 0 \\ 0 & 0 & \frac{1}{\tau_{ax}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{B_{u,s}}} \boldsymbol{u_s} \tag{7.16a}$$

$$\boldsymbol{y_s} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & -\mu_{slip} & 0 & 0 \end{bmatrix}}_{\boldsymbol{C_s}} \boldsymbol{x_s} \tag{7.16b}$$

with state vector $\boldsymbol{x_s} = [T_1 \;\; T_2 \;\; F_{ax} \;\; \omega_1 \;\; \omega_2]^T$, input vector $\boldsymbol{u_s} = [T_{1,c} \;\; T_{2,c} \;\; F_{ax,c}]^T$ containing the desired actuator commands, and output $\boldsymbol{y_s} = [\omega_1 \;\; T_{eff}]^T$. The corresponding dc-gain matrix $\lim\limits_{s \to 0} \boldsymbol{C_s} \left(s\boldsymbol{I_5} - \boldsymbol{A_s}\right)^{-1} \boldsymbol{B_{u,s}}$ depends on the sign of the speed deviation $s_\omega = \text{sign}(x_4 - x_5)$ and reads as

$$\boldsymbol{P}^*_{slip}(s_\omega) = \begin{bmatrix} \frac{1}{k_1} & 0 & \frac{-\mu_{slip} s_\omega}{k_1} \\ 0 & 1 & \mu_{slip} s_\omega \end{bmatrix}. \tag{7.17}$$

From $\mathcal{N}_\mathrm{r}(\boldsymbol{P}^*_{slip}) \neq \boldsymbol{0}$ weak input redundancy follows. Since the system order decreases in case of a locked clutch the state vector reduces to $\boldsymbol{x_l} = [T_1 \;\; T_2 \;\; \omega]^T$ and the plant input is $\boldsymbol{u_l} = [T_{1,c} \;\; T_{2,c}]^T$. The axial force is controlled separately depending on whether locked mode should be maintained or left. The system description is now

$$\dot{\boldsymbol{x}}_{\boldsymbol{l}} = \underbrace{\begin{bmatrix} -\frac{1}{T_1} & 0 & 0 \\ 0 & -\frac{1}{T_2} & 0 \\ \frac{1}{J_1+J_2} & \frac{1}{J_1+J_2} & \frac{-(k_1+k_2)}{J_1+J_2} \end{bmatrix}}_{\boldsymbol{A_l}} \boldsymbol{x_l} + \underbrace{\begin{bmatrix} \frac{1}{T_1} & 0 \\ 0 & \frac{1}{T_2} \\ 0 & 0 \end{bmatrix}}_{\boldsymbol{B_{u,l}}} \boldsymbol{u_l} \tag{7.18a}$$

$$y_l = \underbrace{\begin{bmatrix} \frac{J_2}{J_1+J_2} & 1 - \frac{J_1}{J_1+J_2} & \frac{k_2 J_1 - k_1 J_2}{J_1+J_2} \end{bmatrix}}_{\boldsymbol{C_l}} \boldsymbol{x_l} \tag{7.18b}$$

and the only interesting output is $y_l = T_{eff}$. Evaluating the dc-gain matrix of (7.18)

$$\boldsymbol{P}^*_{lock} = \begin{bmatrix} \frac{k_2}{k_1 + k_2} & \frac{k_2}{k_1 + k_2} \end{bmatrix} \tag{7.19}$$

for the parameters in Table 7.1 yields $\boldsymbol{P}^*_{lock} = [0.5 \quad 0.5]$ which implies weak input redundancy. As described in Section 2.1.7 dynamic allocation is a technique which modifies the output signal of a controller for input redundant systems such that it converges to the solution of an optimization problem with a configurable rate. Hence, the controller has to provide the entire control vector $\boldsymbol{u_s}$ or $\boldsymbol{u_l}$ respectively (see Figure 7.9) and not just a reduced virtual control as in case of static CA methods. In the present case both control tasks could be solved independently with $F_{ax} = 0$, one speed controller using T_1 as control variable and a torque controller only utilizing T_2. The dynamic allocator ensures that the start-up process takes place as specified at the beginning of this chapter, in particular that EM and clutch are initially given preference to the ICE. A PI controller is used for torque control. Since initially

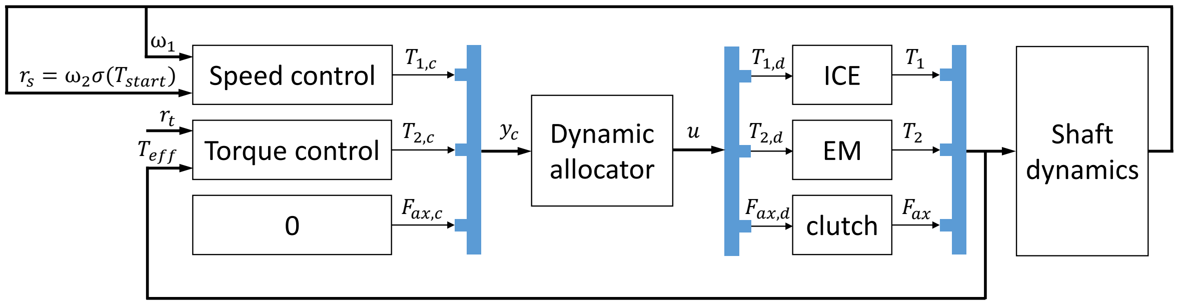

Figure 7.9.: Control scheme for combined torque and speed control during ICE start-up. At time instant $t = T_{start}$ the speed controller's reference signal r_s changes from zero to the speed of the EM. When the shaft speeds coincide and the clutch is in locked mode the speed controller is deactivated, i.e. its output is set to zero.

the ICE is solely accelerated with EM and clutch this results in a disturbance on T_{eff} during start-up which should be kept as small as possible. For this purpose, speed control is also accomplished by a PI-like controller, but the proportional factor depends on the current ICE speed. The error signal for speed matching is

$$e_s := y_{s,1} - r_s = \omega_1 - \omega_2\sigma(T_{start}) \tag{7.20}$$

i.e. the speed controller remains inactive until $t = T_{start}$. The speed control law reads as

$$\begin{aligned} \dot{x}_s &= e_s \\ y_{c,1} &= k_i x_s + k_p|y_{s,1}|e_s. \end{aligned} \tag{7.21}$$

Thus, (7.21) is initially a pure integral controller and at time $t = T_{start}$ its output gradually increases despite the step in its reference signal r_s. Consequently, the impact on T_{eff} is reduced when the speed controller is activated. Alternatively, one could use a ramp reference speed signal in (7.20) (see Section 7.1) instead of the step function. The two controller signals are condensed in

$$\boldsymbol{y_{c,s}} = [T_{1,c} \ \ T_{2,c} \ \ 0]^T \tag{7.22}$$

and are the inputs of the dynamic allocator (see Figure 7.9) as long as the speed deviation is sufficiently large (open clutch or clutch in slipping mode).

The transition from slipping to locked clutch is another major source of unwanted torque jerks. In slipping mode the transmitted clutch torque T_c is proportional to the axial force of the clutch actuator as stated in (1.7). In locked mode (1.9) governs T_c and depends on the torques of EM and ICE as well as on friction and T_{load}. Consequently, these two values differ from each other in general which results in notable jerks. In order to reduce this deviation the following measures are taken:

- Before the speeds of ICE and EM match it follows from (1.7) that $T_c < 0$ because $\omega_1 < \omega_2$. Hence, it must be guaranteed that $\overline{T}_c < 0$ (see Section 1.2.2) when locked mode is entered. Considering (1.9) and $\omega_1 = \omega_2$ implies

$$T_1 J_2 + (k_2 J_1 - k_1 J_2)\omega_1 + T_{load} J_1 < T_2 J_1. \tag{7.23}$$

  Therefore, the weighting parameters of the dynamic allocator are chosen such that T_2 is sufficiently larger than T_1 at the transition.

- Just before the transition from slipping to locked mode the axial force of the clutch actuator is chosen such that the transmitted torque corresponds to (1.9), i.e.

$$F_{trans} = \frac{\delta_{trans}|\overline{T}_c|}{\mu_{slip} r_a n_s} \tag{7.24}$$

  with $\delta_{trans} \approx 1$ being a small tuning parameter.

In locked mode the axial force is chosen in order to remain in this mode as

$$F_{lock} = \max\left(F_{min}, \frac{\delta_{lock}|\overline{T}_c|}{\mu_{stick} r_a n_s}\right) \tag{7.25}$$

with $F_{min} > 0$ specifying the minimum actuation force and tuning parameter $\delta_{lock} > 1$. The speed controller is deactivated as soon as the speed deviation is below a threshold and so the signal entering the dynamic allocator is

$$\boldsymbol{y_{c,l}} = [0 \;\; T_{2,c} \;\; 0]^T. \tag{7.26}$$

Depending on the speed deviation $x_4 - x_5$ the dynamic allocator distinguishes between three different cases and computes the actuator commands as

$$\boldsymbol{u} = \begin{cases} \begin{bmatrix} \boldsymbol{N} x_a \\ F_{lock} \end{bmatrix} + \boldsymbol{y_{c,l}} & \text{if } |x_4 - x_5| \leq \epsilon_{lock} \\ \begin{bmatrix} \boldsymbol{N} x_a \\ F_{trans} \end{bmatrix} + \boldsymbol{y_{c,l}} & \text{if } |x_4 - x_5| \leq \epsilon_{trans} \\ \boldsymbol{N} x_a + \boldsymbol{y_{c,s}} & \text{else} \end{cases} \tag{7.27}$$

with $0 < \epsilon_{lock} < \epsilon_{trans}$, $\boldsymbol{N}$ spans the one-dimensional right nullspace of the dc-gain matrices (7.17) and (7.19)

$$\boldsymbol{N} = \begin{cases} \mathcal{N}_{\mathrm{r}}(\boldsymbol{P}^*_{lock}) & \text{if } |x_4 - x_5| \leq \epsilon_{trans} \\ \mathcal{N}_{\mathrm{r}}(\boldsymbol{P}^*_{slip}(s_\omega)) & \text{else,} \end{cases} \tag{7.28}$$

and the allocator dynamics

$$\dot{x}_a = \begin{cases} -\boldsymbol{K}\boldsymbol{N}^T\boldsymbol{W}\boldsymbol{N} x_a - \boldsymbol{K}\boldsymbol{N}^T\boldsymbol{W}\boldsymbol{y_{c,l}} & \text{if } |x_4 - x_5| \leq \epsilon_{trans} \\ -\boldsymbol{K}\boldsymbol{N}^T\boldsymbol{W}\boldsymbol{N} x_a - \boldsymbol{K}\boldsymbol{N}^T\boldsymbol{W}\boldsymbol{y_{c,s}} & \text{else.} \end{cases} \tag{7.29}$$

The weighting matrix in (7.29) is given by

$$\boldsymbol{W} = \begin{cases} \begin{bmatrix} 100 & 0 \\ 0 & 200 \end{bmatrix} & \text{if } |x_4 - x_5| \leq \epsilon_{lock} \\ \begin{bmatrix} c_{11} - c_{12}\sigma(T_{start} + T_{lag}) & 0 \\ 0 & c_2 \end{bmatrix} & \text{if } |x_4 - x_5| \leq \epsilon_{trans} \\ \begin{bmatrix} c_{11} - c_{12}\sigma(T_{start} + T_{lag}) & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & c_{31} - c_{32}\sigma(T_{start}) \end{bmatrix} & \text{else.} \end{cases} \tag{7.30}$$

If the speed deviation is below $\epsilon_{lock} > 0$ the allocator uses dedicated parameters to enforce the greater portion of the torque coming from the ICE. The else-branch in (7.30) is dedicated to the actual start-up. Initially, all diagonal entries except c_2 are large resulting in an EM preference. At the beginning $t = T_{start}$ the clutch related element is significantly decreased to

parameter	value	parameter	value
c_{11}	10000	c_{12}	9750
T_{lag}	0.4	c_2	200
c_{31}	10000	c_{32}	9999
ϵ_{lock}	0.0001	ϵ_{trans}	3
$\boldsymbol{K}$	1	-	-

Table 7.3.: Parameters of the dynamic allocator

enable the acceleration of the ICE. After a configurable lag T_{lag} the first entry dedicated to the ICE is also reduced. The chosen parameters are given in Table 7.3. The remaining case in (7.30) deals with the transition from slipping to locked mode. The dynamic allocator is tested for constant (Figure 7.10) and variable torque references (Figure 7.11). It can be seen that during the acceleration phase of the ICE the effective torque is not affected. Only at the transition from slipping to locked mode small torque peaks are visible. But due to their short duration and small magnitude they have virtually no effect on the powertrain acceleration.

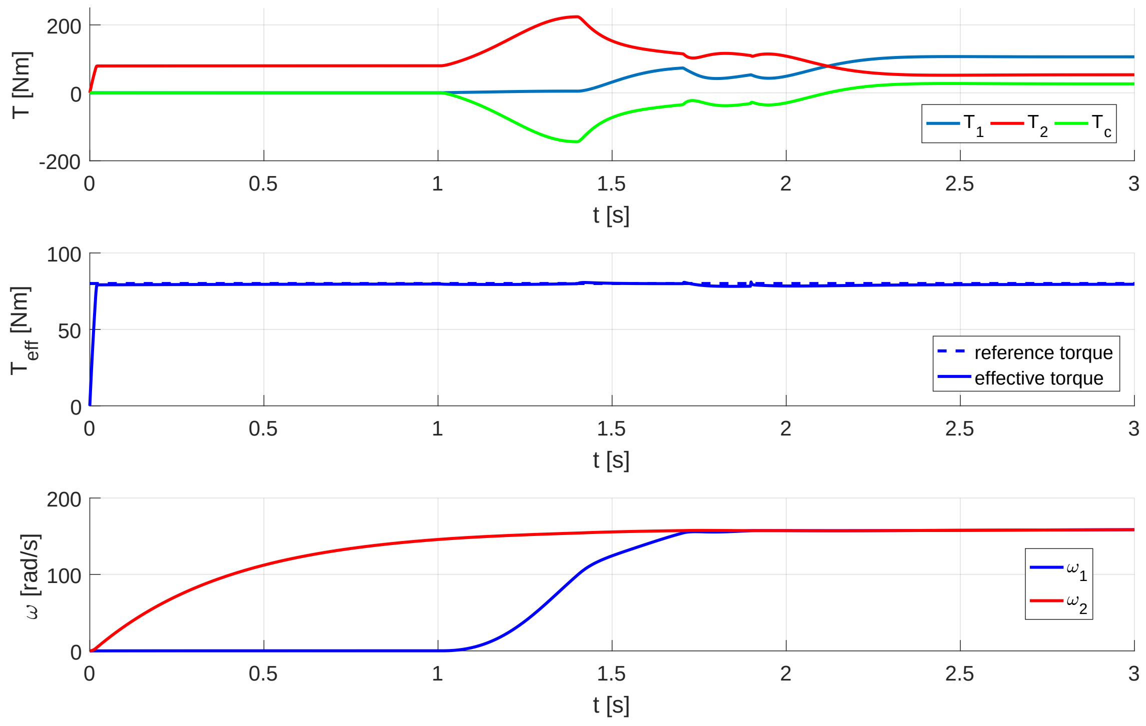

Figure 7.10.: Results of dynamic allocation for constant torque reference.

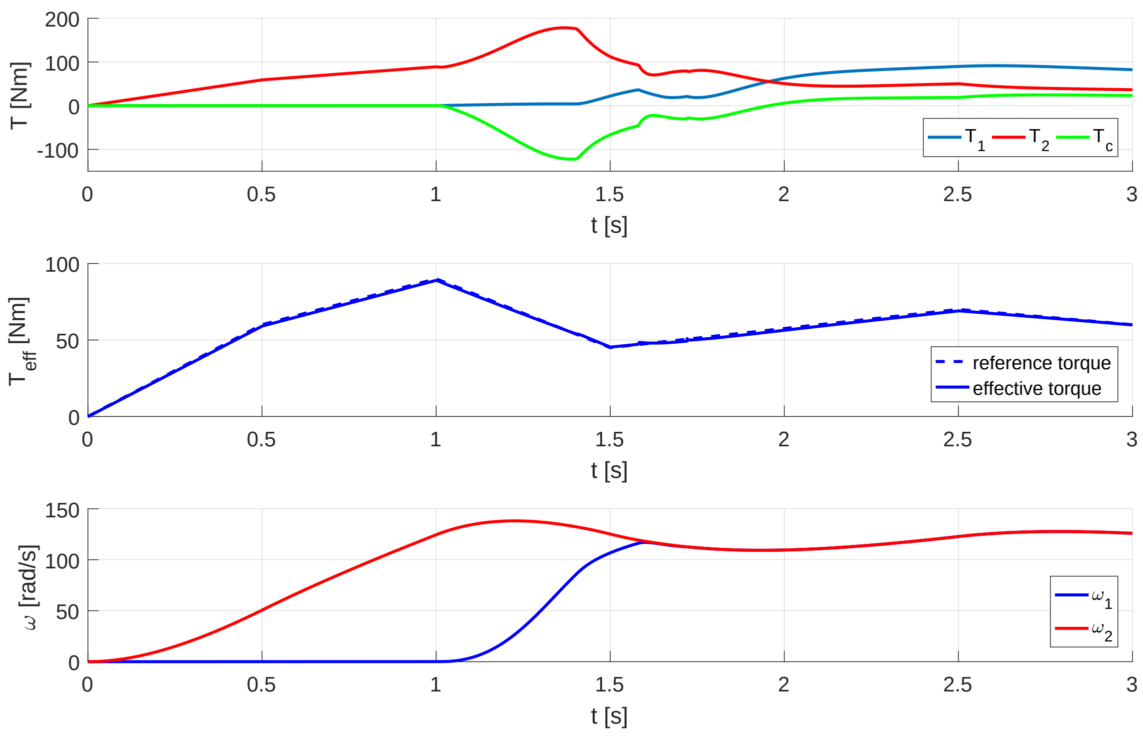

Figure 7.11.: Results of dynamic allocation for variable torque reference.

7.3. Clutch control

The clutch controller plays an important role in the proposed HEV control scheme. On one hand it has to be sufficiently fast to enable the operating strategy to fulfill the driver's torque request as good as possible. On the other hand it should not introduce noticeable jolts during ICE start-up due to jerky actuation. The basic principles of torque transmission with clutches have already been described in Section 1.2. The amount of transferable torque is determined by the friction coefficient of the rotating discs and the normal force generated by the actuator. The usual clutch actuator of today's passenger cars operates on a electrohydraulic or electromechanic basis. Due to their successful application for many years the related control concepts are mature and work very well in practice. Typically a combination of a static feedforward part and a PID controller is used in both cases. An alternative actuation approach uses an electromagnet as direct source for generating the axial force. This has the advantage of requiring less installation space and maintenance than the conventional actuators. On the downside the electromagnet's nonlinear characteristics complicate clutch control significantly.

The foundation of the two suggested control strategies is position control of the clutch actuator. There is a nonlinear mapping from the position of the electromagnet's armature to the axial force of the actuator which directly influences the transmittable torque. Appendix C contains a detailed mathematical model of the actuator, the parameter identification procedure, and the derivation of the position control laws based on feedback linearization and differential flatness. The methods are evaluated in simulations where the influence of uncertain parameters is investigated. Furthermore, the flatness-based approach is implemented on an actuator test bed where it shows very good behavior. Most of this work has already been published in the author's works [52] and [53].

7.4. Conclusion

Two CA-based strategies of ICE start-up in a moving HEV have been presented in this chapter. Apart from harmonizing the two shaft speeds the avoidance of torque jerks during the process is of great importance. The combination of torque and speed control provides the most flexible solution as it does not assume a constant powertrain acceleration during the speed matching phase of the two motors. Thus, as opposed to the pure speed control approach, the ICE start-up can be completed for a wider range of driving maneuvers.

8. Quadrotor Attitude Control[1]

The number of possible application areas of unmanned aerial vehicles (UAV) is vast and increasing. Operational scenarios include search and rescue missions, emergency response, surveillance, building inspection, and mapping [54]. Quadrotors and other vertical takeoff and landing UAVs offer high maneuverability and versatility for relatively low costs. In the majority of practical quadrotor applications a human operator controls the position and specifies a desired attitude using direct sight or an on-board camera, whereas the attitude control is done automatically. An effective attitude controller is not only important for maintaining the pilot's desired orientation, it also stabilizes the UAV and makes it easier to fly [55]. Therefore, quadrotor attitude control has received a lot of attention in the research community and various control approaches have been proposed, e.g. [55], [56], [57], [58]. The typical control scheme consists of three layers: attitude control itself computes virtual torques around the rotational axes, a procedure which translates those torques into actuator commands, and finally motor control. Most of the proposed work focuses on the development of the former and uses a static mapping to obtain the motor commands. However, this approach does not take actuator constraints into account, which can lead to degraded behavior during challenging maneuvers. One possibility to address this issue is the usage of CA algorithms. These techniques not only enable the consideration of constraints but also actuator faults can be incorporated as demonstrated in [9], [59], and [60].

This chapter describes the practical application of CA for quadrotor attitude control and compares the usually applied input mapping with various CA methods. Furthermore, the control task is solved employing LQR and MPC in order to compare the CA-based results with those of the main alternatives for the control of input redundant systems. The experiments are conducted with the Quanser 3 DOF Hover system (see [61]) shown in Figure 8.1. Because of its mounting on a base platform the rotational dynamics are influenced by two additional quantities as opposed to a flying quadrotor: bearing friction and gravitation. Section 8.1 describes the modeling process using Lagrangian dynamics. A Super-Twisting sliding mode controller similar to [58] is chosen to generate the virtual torques for reference angle tracking. Its derivation follows in Section 8.3.1. Subsequently, the input allocation methods are briefly outlined. The experimental results on the Quanser hovering platform demonstrate better performance and more flexibility of the constrained CA methods compared to the conventional input mapping and LQR. The MPC implementation explained in Section 8.5 also achieves good results but requires more computational effort than the CA-based control scheme.

[1]Parts of Sections 8.1 and 8.3 have been submitted to the joint 9th IFAC Symposium on Robust Control Design and 2nd IFAC Workshop on Linear Parameter Varying Systems. At the time of printing the decision has been pending.

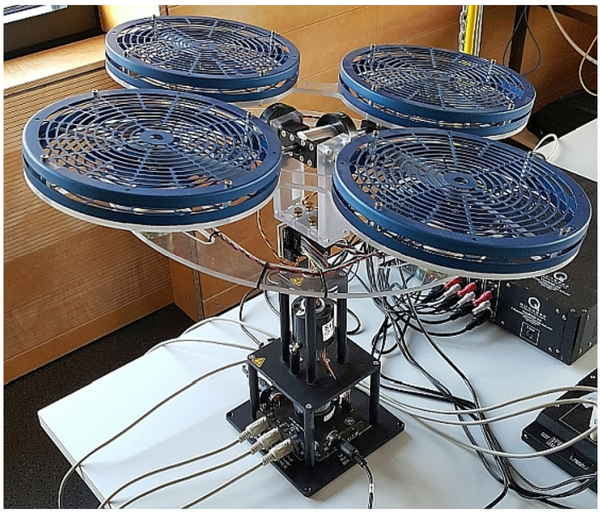

Figure 8.1: The Quanser 3 DOF Hover [61] is used for all control experiments in this chapter.

8.1. Modeling

The Quanser 3 DOF Hover basically consists of a stationary base and a frame with four propellers. The frame is mounted on the base by means of three radial bearings. On each axle there is an encoder measuring the corresponding angles. By definition rotors 1 and 3 are located at the hover's front and rear whereas rotors 2 and 4 lie on the left and right hand sides. The earth-fixed inertial frame is represented by $\mathcal{I} = \{\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{z_i}\}$ and the hover-fixed body frame is called $\mathcal{B} = \{\boldsymbol{x_b}, \boldsymbol{y_b}, \boldsymbol{z_b}\}$. Figure 8.2 shows the unrotated configuration where $\mathcal{B} = \mathcal{I}$ as well as the directions of forces and torques coming from the rotors.

Front and rear rotors rotate anti-clockwise whereas left and right rotors spin in clockwise direction. The resulting torques $\tau_1, \dots, \tau_4$ act into the opposite directions [62]. Differences in rotor speeds lead to unbalanced torques and in further consequence to a rotation around the hover's body yaw axis. Each rotor also produces a lift force $(f_1, \dots, f_4)$ and the force differences between opposite rotors induce torques around body roll and pitch axis. Figures 8.3 - 8.5 show the positive directions of the system's three degrees of freedom yaw (ψ), pitch (θ), and roll (ϕ) together with the orientation of the coordinate frames. Note that due to the fixation on the base there is of course no translational motion and so the origins of $\mathcal{B}$ and $\mathcal{I}$ are always identical. The considered laboratory setup undeniably shares some similarities with a quadrotor. The literature which describes modeling the quadrotor rotational dynamics is vast, examples can be found in [56], [57], [58], [62], and [63] to mention just a few. However, the ground-based nature of the hover leads to some differences between the behavior of the present setup and a flying quadrotor:

- Three radial bearings introduce additional friction into the system dynamics which is notable especially for the yaw rotation.
- The bearing of the hover and the fact that most of its mass is concentrated in the motors and propellers which are substantially far away from the mounting point result in gravitational influence on the rotation dynamics. Although similar quadrotor rotation test beds are used in parts of the referenced literature this phenomenon has been neglected so far.
- On a flying quadrotor typically gyroscopes measure the angular rates with respect to $\mathcal{B}$. Thus, it is natural to specify the system dynamics in $\mathcal{B}$ too. The encoders of the labora-

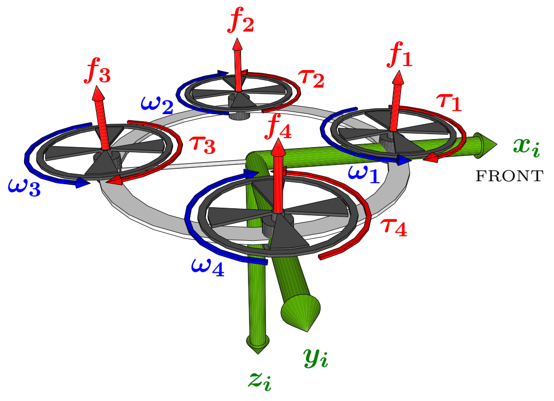

Figure 8.2.: Rotors spin with velocities ω_i and the resulting torques τ_i act in opposite direction.

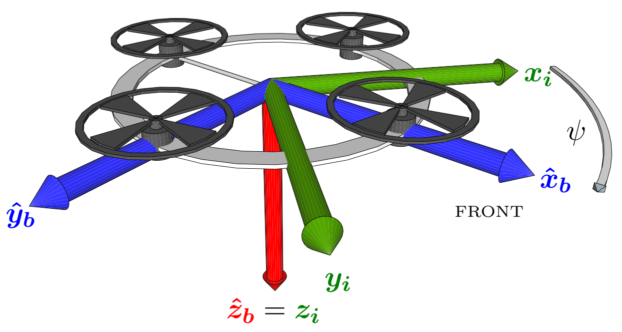

Figure 8.3.: Yaw angle ψ describes rotation around inertial Z-axis z_i.

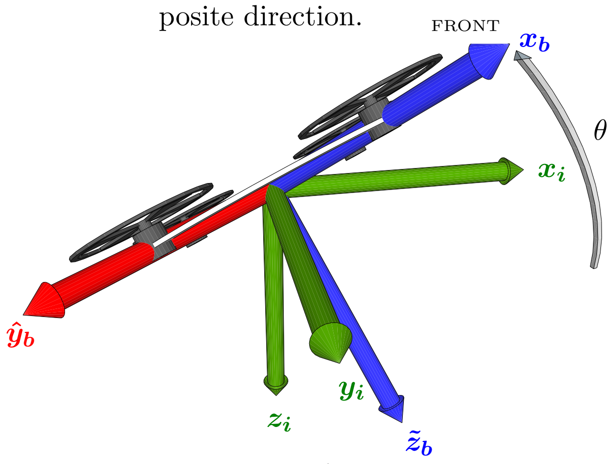

Figure 8.4.: Pitch angle θ describes rotation around current Y-axis $\hat{y}_b$.

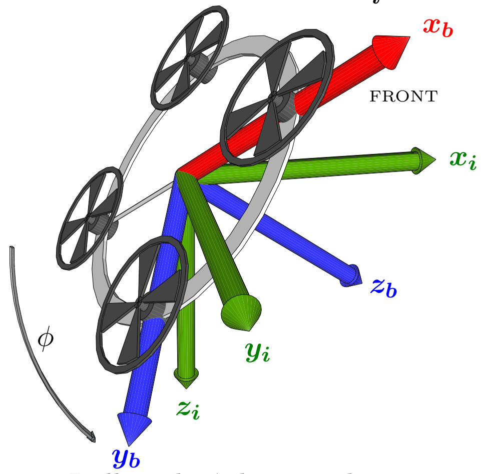

Figure 8.5.: Roll angle ϕ denotes the rotation around the body X-axis x_b.

tory setup measure the roll, pitch, and yaw angles instead and so their time derivatives are chosen to constitute the hover's dynamics.

The presented model takes these issues into account.

8.1.1. Lagrange modeling

The dynamic model of the Quanser 3 DOF Hover is developed by means of Lagrangian dynamics (see [62] and [64]). Initially, an independent set of variables which are able to describe the rigid body motion, the so-called *generalized coordinates*, is chosen as

$$\boldsymbol{q} = \begin{bmatrix} \phi(t) & \theta(t) & \psi(t) \end{bmatrix}^T . \tag{8.1}$$

Let $\boldsymbol{R_\phi}$, $\boldsymbol{R_\theta}$, and $\boldsymbol{R_\psi}$ be the rotation matrices around x, y, and z-axes respectively [64]

$$\boldsymbol{R_\psi} = \begin{bmatrix} \mathrm{co}_\psi & -\mathrm{si}_\psi & 0 \\ \mathrm{si}_\psi & \mathrm{co}_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.2}$$

$$\boldsymbol{R_\theta} = \begin{bmatrix} \mathrm{co}_\theta & 0 & \mathrm{si}_\theta \\ 0 & 1 & 0 \\ -\mathrm{si}_\theta & 0 & \mathrm{co}_\theta \end{bmatrix} \tag{8.3}$$

$$\boldsymbol{R_\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{co}_\phi & -\mathrm{si}_\phi \\ 0 & \mathrm{si}_\phi & \mathrm{co}_\phi \end{bmatrix} \tag{8.4}$$

with $\mathrm{co}_\theta = \cos(\theta)$ and $\mathrm{si}_\theta = \sin(\theta)$. The overall rotation matrix from body to inertial frame is given by $\boldsymbol{R_{b2i}} = \boldsymbol{R_\psi R_\theta R_\phi}$ which yields [63]

$$\boldsymbol{R_{b2i}} = \begin{bmatrix} \mathrm{co}_\theta\mathrm{co}_\psi & \mathrm{co}_\psi\mathrm{si}_\theta\mathrm{si}_\phi - \mathrm{co}_\phi\mathrm{si}_\psi & \mathrm{co}_\phi\mathrm{co}_\psi\mathrm{si}_\theta + \mathrm{si}_\phi\mathrm{si}_\psi \\ \mathrm{co}_\theta\mathrm{si}_\psi & \mathrm{co}_\phi\mathrm{co}_\psi + \mathrm{si}_\theta\mathrm{si}_\phi\mathrm{si}_\psi & \mathrm{co}_\phi\mathrm{si}_\theta\mathrm{si}_\psi - \mathrm{co}_\psi\mathrm{si}_\phi \\ -\mathrm{si}_\theta & \mathrm{co}_\theta\mathrm{si}_\phi & \mathrm{co}_\theta\mathrm{co}_\phi \end{bmatrix}. \tag{8.5}$$

The hover's angular velocities expressed in $\mathcal{B}$ are computed from the following skew symmetric matrix [64]

$$\boldsymbol{\Omega_b} = \begin{bmatrix} 0 & -\omega_{b,z} & \omega_{b,y} \\ \omega_{b,z} & 0 & -\omega_{b,x} \\ -\omega_{b,y} & \omega_{b,x} & 0 \end{bmatrix} = \boldsymbol{R_{b2i}^T \dot{R}_{b2i}} \tag{8.6}$$

and read as

$$\boldsymbol{\omega_b} = \begin{bmatrix} \omega_{b,x} \\ \omega_{b,y} \\ \omega_{b,z} \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \mathrm{si}_\theta\dot{\psi} \\ \mathrm{co}_\phi\dot{\theta} + \mathrm{co}_\theta\mathrm{si}_\phi\dot{\psi} \\ -\mathrm{si}_\phi\dot{\theta} + \mathrm{co}_\theta\mathrm{co}_\phi\dot{\psi} \end{bmatrix}. \tag{8.7}$$

From (8.7) one can determine the *Body Jacobian*

$$\boldsymbol{J_b} = \frac{\partial\boldsymbol{\omega_b}}{\partial\boldsymbol{\dot{q}}} = \begin{bmatrix} 1 & 0 & -\mathrm{si}_\theta \\ 0 & \mathrm{co}_\phi & \mathrm{co}_\theta\mathrm{si}_\phi \\ 0 & -\mathrm{si}_\phi & \mathrm{co}_\theta\mathrm{co}_\phi \end{bmatrix} \tag{8.8}$$

which links the time derivatives of the generalized coordinates with the angular velocities in $\mathcal{B}$ by means of $\boldsymbol{\omega_b} = \boldsymbol{J_b\dot{q}}$. Typically, at this point the assumptions $\phi \approx \theta \approx 0$ motivate the simplification of (8.8) to an identity matrix $\boldsymbol{I_3}$, which results in $\boldsymbol{\omega_b} \approx \boldsymbol{\dot{q}}$. Hence, rotation dynamics are specified in $\mathcal{B}$ and roll, pitch, and yaw angles are computed by integrating the angular velocities around the body axes. However, only the body roll axis $\boldsymbol{x_b}$ coincides with the axis for ϕ-rotation whereas $\boldsymbol{y_b}$ is not the axis for θ-rotation and ψ-rotation is not done around $\boldsymbol{z_b}$ (see Figures 8.3 - 8.5). The torques generated by the spinning rotors are the major influencing factors on the dynamics of the hover. The Body Jacobian is also required to compute the torques $\boldsymbol{\tau_q} \in \mathbb{R}^3$ around the axis of rotation of $\boldsymbol{q}$ from the torques $\boldsymbol{\tau_b} \in \mathbb{R}^3$ expressed in $\mathcal{B}$ employing [64]

$$\boldsymbol{\tau_q} = \boldsymbol{J_b^T\tau_b}. \tag{8.9}$$

The inertia matrix for rotations around the axis of $\mathcal{B}$ is given by

$$\boldsymbol{\mathcal{J}_b} = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix}. \tag{8.10}$$

Now, the kinetic energy $T = \frac{1}{2}\boldsymbol{\omega_b^T\mathcal{J}_b\omega_b}$ is obtained from (8.7) and $\boldsymbol{\mathcal{J}_b}$ and reads as

$$T = \frac{1}{2}\left[J_x\left(\dot{\phi} - \mathrm{si}_\theta\dot{\psi}\right)^2 + J_y\left(\mathrm{co}_\phi\dot{\theta} + \mathrm{co}_\theta\mathrm{si}_\phi\dot{\psi}\right)^2 + J_z\left(\mathrm{si}_\phi\dot{\theta} - \mathrm{co}_\theta\mathrm{co}_\phi\dot{\psi}\right)^2\right]. \tag{8.11}$$

The influence of gravitation on the rotational dynamics is incorporated by means of potential energy. The system's static mechanical equilibrium attitude is denoted as $[\phi_e \;\; \theta_e \;\; *]^T$. Please note that ψ does not affect the static equilibrium. In general the gravitation related potential energy of N point masses is

$$V = \sum_{j=1}^{N} m_j g h_{i,j} \tag{8.12}$$

where g is the acceleration of gravity acting in z-direction of $\mathcal{I}$ and $h_{i,j}$ is the altitude of the j-th mass m_j in $\mathcal{I}$. In order to express (8.12) in terms of the generalized coordinates the altitude is written as $h_{i,j} = -\boldsymbol{e_z}^T \boldsymbol{R_{b2i}} \boldsymbol{r_{b,j}}$ with $\boldsymbol{r_{b,j}} \in \mathbb{R}^3$ being the position of the j-th mass in $\mathcal{B}$ and $\boldsymbol{e_z} = [0 \;\; 0 \;\; 1]^T$. Using $\boldsymbol{h}^T = -\boldsymbol{e_z}^T \boldsymbol{R_{b2i}}$ and the auxiliary vector

$$\boldsymbol{m_{b,g}} = \begin{bmatrix} m_{g,x} \\ m_{g,y} \\ m_{g,z} \end{bmatrix} = \sum_{j=1}^{N} m_j g \boldsymbol{r_{b,j}} \tag{8.13}$$

expression (8.12) changes to

$$V(\phi, \theta) = \boldsymbol{h}^T \boldsymbol{m_{b,g}}. \tag{8.14}$$

The potential energy has a critical point at the static equilibrium attitude, i.e.

$$\boldsymbol{0} = \begin{bmatrix} \frac{\partial V}{\partial \phi} & \frac{\partial V}{\partial \theta} & 0 \end{bmatrix}^T \Bigg|_{\phi=\phi_e, \theta=\theta_e} . \tag{8.15}$$

Using abbreviations for secant and tangent and combining (8.14) and (8.15) yields

$$V = -m_{g,z} \left[\mathrm{se}_{\phi_e} \mathrm{ta}_{\theta_e} \mathrm{si}_\theta + \mathrm{co}_\theta \left(\mathrm{co}_\phi + \mathrm{si}_\phi \mathrm{ta}_{\phi_e} \right) \right]. \tag{8.16}$$

Finally, the Lagrangian $\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = T - V$ is obtained from (8.11) and (8.16). Next, the influence of the actuators is considered. The j-th rotor spins with angular speed ω_j and generates a thrust $F_{rot,j} \propto \omega_j^2$ into the negative z-direction of $\mathcal{B}$ and a torque $T_{rot,j} \propto \omega_j^2$ against its sense of rotation [62]. Therefore, the torques around the three axis of $\mathcal{B}$ arising from actuation are

$$\boldsymbol{\tau_{b,r}} = \begin{bmatrix} \tau_{b,rx} \\ \tau_{b,ry} \\ \tau_{b,rz} \end{bmatrix} = \begin{bmatrix} k_\phi \left(\omega_2^2 - \omega_4^2 \right) \\ k_\theta \left(\omega_1^2 - \omega_3^2 \right) \\ k_\psi \left(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2 \right) \end{bmatrix}. \tag{8.17}$$

The bearing related friction is taken into account with a Stribeck-curve [64] inspired approach

$$\boldsymbol{\tau_{q,f}} = \begin{bmatrix} \tau_{q,fx} \\ \tau_{q,fy} \\ \tau_{q,fz} \end{bmatrix} = - \begin{bmatrix} \left(d_{\phi,1} e^{-d_{\phi,0}|\dot{\phi}|} \right) \dot{\phi} \\ \left(d_{\theta,1} e^{-d_{\theta,0}|\dot{\theta}|} \right) \dot{\theta} \\ \left(d_{\psi,2} + d_{\psi,1} e^{-d_{\psi,0}|\dot{\psi}|} \right) \dot{\psi} \end{bmatrix}. \tag{8.18}$$

Combining (8.9), (8.17), and (8.18) yields the generalized forces (actually torques in this case) $\boldsymbol{\tau_q} = \boldsymbol{J_b}^T \boldsymbol{\tau_{b,r}} + \boldsymbol{\tau_{q,f}}$ which read as

$$\boldsymbol{\tau_q} = \begin{bmatrix} \tau_{b,rx} + \tau_{q,fx} \\ \mathrm{co}_\phi \tau_{b,ry} - \mathrm{si}_\phi \tau_{b,rz} + \tau_{q,fy} \\ \mathrm{co}_\theta \left(\mathrm{si}_\phi \tau_{b,ry} + \mathrm{co}_\phi \tau_{b,rz} \right) - \mathrm{si}_\theta \tau_{b,rx} + \tau_{q,fz} \end{bmatrix}. \tag{8.19}$$

Because of the considerable complexity of the involved expressions the dynamics are derived from the Lagrange formalism

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} = \boldsymbol{\tau_q} \tag{8.20}$$

evaluated by means of the computer algebra system Mathematica [65] which yields three ordinary second-order differential equations.

8.1.2. Model inputs

In the equations obtained from (8.20) the inputs are the angular speeds of the motors ω_i. The motors of the laboratory setup are not equipped with encoders and so it is not possible to measure or control their exact speeds during normal operation. Dedicated measurements with an optical tachometer show that the relation between input voltage u_i and angular speed ω_i is effectively linear in steady state. Since the transients are much faster than the rotational dynamics of the hover they can be neglected. Thus, it is reasonable to use voltages instead of angular velocities as inputs. Due to the fact that only positive motor voltages / speeds are allowed the mapping from u_i^2 to u_i is unique and the input vector can be defined as

$$\boldsymbol{u} = [u_1^2 \;\; u_2^2 \;\; u_3^2 \;\; u_4^2]^T. \tag{8.21}$$

8.1.3. Full state-space model

By introducing the vector of state variables

$$\boldsymbol{x}^T = [x_1\; x_2\; x_3\; x_4\; x_5\; x_6] = [\phi \;\; \theta \;\; \psi \;\; \dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi}] \tag{8.22}$$

one obtains a nonlinear sixth-order state-space model

$$\dot{\boldsymbol{x}} = \boldsymbol{f_1}(\boldsymbol{x}, \boldsymbol{u}) \tag{8.23a}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) \tag{8.23b}$$

with function $\boldsymbol{f_1}(\boldsymbol{x}, \boldsymbol{u})$ as specified in Appendix D and $\boldsymbol{h}(\boldsymbol{x}) = [\boldsymbol{I_3} \;\; \boldsymbol{0}]\boldsymbol{x}$. The elaborate model (8.23) primarily fulfills two purposes:

1. It provides a realistic simulation model for controller testing.

2. The derivation of a state-dependent input matrix $\boldsymbol{B_u}(\boldsymbol{x})$ (see Appendix D) serving as the foundation for CA. Closer examination of (8.23) reveals that it is input affine with respect to (8.22) and exhibits the special structure

$$\dot{\boldsymbol{x}}_{\boldsymbol{123}} = \boldsymbol{x}_{\boldsymbol{456}} \tag{8.24a}$$

$$\dot{\boldsymbol{x}}_{\boldsymbol{456}} = \boldsymbol{a_1}(\boldsymbol{x}) + \boldsymbol{B_u}(\boldsymbol{x})\boldsymbol{u} \tag{8.24b}$$

$$\boldsymbol{y} = \boldsymbol{x}_{\boldsymbol{123}} \tag{8.24c}$$

with $\boldsymbol{x}_{\boldsymbol{123}} = [x_1\; x_2\; x_3]^T$, $\boldsymbol{x}_{\boldsymbol{456}} = [x_4\; x_5\; x_6]^T$, vector function $\boldsymbol{a_1} : \mathbb{R}^6 \to \mathbb{R}^3$, and input matrix $\boldsymbol{B_u}(\boldsymbol{x}) \in \mathbb{R}^{3\times 4}$.

Table 8.1.: Relationship between the parameters of (8.26) and physical values.

roll parameters	pitch parameters	yaw parameters
$a_1 = \frac{J_x+J_y-J_z}{J_x}$	$a_5 = \frac{J_z-J_y-J_x}{J_y}$	$a_9 = \frac{J_x-J_y+J_z}{J_z}$
$a_2 = \frac{d_{\phi,1}}{J_x}$	$a_6 = \frac{d_{\theta,1}}{J_y}$	$a_{10} = \frac{d_{\psi,2}}{J_z}$
$a_3 = d_{\phi,0}$	$a_7 = d_{\theta,0}$	$a_{11} = \frac{d_{d_{\psi,1}}}{J_z}$
$a_4 = \frac{m_{g,z}\mathrm{se}_{\phi_e}}{J_x}$	$a_8 = \frac{m_{g,z}\mathrm{se}_{\phi_e}\mathrm{se}_{\theta_e}}{J_y}$	$a_{12} = d_{\psi,0}$

8.1.4. Small-angle models

The full model (8.23) and its other representation (8.24) are not ideal for controller development due to their complexity. They can be simplified under the small angle assumptions $x_i \approx 0$ for $i \in \{1, 2\}$ which implies $\mathrm{co}_{x_i} \approx 1$ and $\mathrm{si}_{x_i} \approx 0$. Furthermore, (8.8) becomes an identity matrix, i.e. $\boldsymbol{\omega_b} \approx \dot{\boldsymbol{q}}$. Applying those assumptions on the overall model (8.24) results in

$$\dot{\boldsymbol{x}}_{\boldsymbol{123}} = \boldsymbol{x}_{\boldsymbol{456}} \tag{8.25a}$$

$$\dot{\boldsymbol{x}}_{\boldsymbol{456}} = \boldsymbol{a_2}(\boldsymbol{x}) + \overbrace{\begin{bmatrix} 0 & \frac{k_\phi}{J_x} & 0 & -\frac{k_\phi}{J_x} \\ \frac{k_\theta}{J_y} & 0 & -\frac{k_\theta}{J_y} & 0 \\ \frac{k_\psi}{J_z} & -\frac{k_\psi}{J_z} & \frac{k_\psi}{J_z} & -\frac{k_\psi}{J_z} \end{bmatrix}}^{\boldsymbol{B_{u,2}}} \boldsymbol{u} \tag{8.25b}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{x}_{\boldsymbol{123}} \tag{8.25c}$$

with

$$\boldsymbol{a_2}(\boldsymbol{x}) = \begin{bmatrix} a_1 x_5 x_6 - a_2 e^{-a_3|x_4|} x_4 + a_4 \mathrm{si}_{(\phi_e - x_1)} \\ a_5 x_4 x_6 - a_6 e^{-a_7|x_5|} x_5 + a_8 \mathrm{si}_{(\theta_e - x_2)} \\ a_9 x_4 x_5 - \left(a_{10} + a_{11} e^{-a_{12}|x_6|}\right) x_6 \end{bmatrix} \tag{8.26}$$

parameters a_i for $i = 1, \ldots, 12$ summarized in Table 8.1. The second term in each row of (8.26) relates to the bearing friction. The expressions multiplied with a_4 and a_8 in (8.26) respectively describe gravitation's influence on roll and pitch in case of deflections from the equilibrium angles ϕ_e and θ_e. If the small angle assumptions are only utilized to simplify $\boldsymbol{a_1}(\boldsymbol{x})$ in (8.24) one gets

$$\dot{\boldsymbol{x}}_{\boldsymbol{123}} = \boldsymbol{x}_{\boldsymbol{456}} \tag{8.27a}$$

$$\dot{\boldsymbol{x}}_{\boldsymbol{456}} = \boldsymbol{a_2}(\boldsymbol{x}) + \boldsymbol{B_u}(\boldsymbol{x})\boldsymbol{u} \tag{8.27b}$$

$$\boldsymbol{y} = \boldsymbol{x}_{\boldsymbol{123}}. \tag{8.27c}$$

8.1.5. Linearized model

A linear model describing deflections from the equilibrium can be derived by linearization of the full nonlinear model (8.23). Since the resulting model is exclusively used for LQR design more details about it are available in Section 8.4.

8.2. Parameter identification

Before the models developed in Section 8.1 are usable for simulations and controller design their parameters have to be determined. The equilibrium angles can be directly measured

Table 8.2.: Equilibrium angles of the laboratory setup.

parameter	value	parameter	value
ϕ_e	-0.05997 rad	θ_e	0.15064 rad

and are given in Table 8.2. The estimation of the remaining parameters is based on numerical optimization. For each model variant a simulation model is built. The objective function of the optimization algorithm initializes the model with the same initial values as the real laboratory setup, uses the recorded motor voltages as input signals, and computes the sum of squared errors between measured and simulated angles for a predefined set of experiments. The sought model parameters are the optimization variables. As most of the parameters have a physical meaning their possible range of values can be restricted. Hence, parameter identification is done by solving a nonlinear constrained optimization problem which may have multiple local minima. In order to reduce the influence of the starting point of optimization this procedure is carried out for a variety of initial model parameter guesses. In principle the approach consists of two steps:

1. Identification of the unactuated system: The hover is deflected from ϕ_e and θ_e. Then the actuators are deactivated and the measurement is continued until the hover has returned into equilibrium. The previously described optimization procedure is done for the measurements starting with the actuator deactivation. The purpose of these librate experiments is the determination of suitable initial values for the inertias in (8.10), $m_{g,z}$, and the friction coefficients in (8.18) for the subsequent identification step.
2. Identification with actuation: In this category three types of experiments are conducted which are mainly used to identify the three actuator gain parameters.
   - Voltage steps primarily inducing a rotation around a single axis
   - Randomized input signals
   - Inputs generated by an operator with a joystick

Figure 8.6 contains an example of one of the librate experiments used in step 1 of the identification process. The adjacent Figure 8.7 shows a comparison of a measurement which was not used for parameter identification and a simulation with the corresponding inputs for validation purposes. The simulation results shown in those figures originate from the full nonlinear model (8.23). One recognizes good conformity between real plant and model. In the second step of the identification procedure measurements of eleven experiments with active rotors are utilized for estimating the model parameters. One of them is illustrated in Figure 8.8 together with the results for all three model variants including the MSE. The full nonlinear model achieves the best results for all angles in this comparison. Table 8.3 shows the sum over all MSE of those experiments which are used for identification. As expected the full nonlinear model achieves the best results for the data its parameters are optimized for. In order to validate the identified parameters a set of 25 other experiments is used. Figure 8.9 shows one of these validation experiments. In this case, again the conformity between measured and simulated angles is very good for the full nonlinear model. The sum over all MSE of the validation experiments is contained in Table 8.4 where one surprisingly recognizes that the linear model seems to yield the best results. This originates from the fact that the linear model barely shows ψ rotations (see Figures 8.8 - 8.10) whereas the other models rotations in many cases match the real ψ but in some experiments they move to a wrong position. Since ψ is never reset to known values as it happens with ϕ and θ a wrong final value of ψ has a big impact on

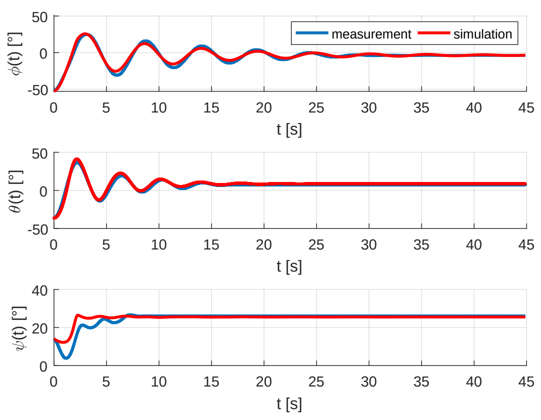

Figure 8.6.: Example of a librate experiment for the identification of the hover's mechanical parameters.

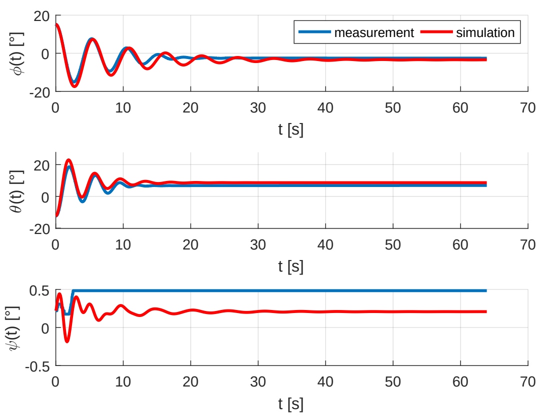

Figure 8.7.: Validation of model and mechanical parameters with another librate experiment.

the MSE. One such experiment is illustrated in Figure 8.10. Whereas the simulation results for ϕ and θ are rather bad for the linear model $\psi \approx 0$ yields better results than the wrong movements from the other two model variants.

Table 8.3.: MSE sum of all experiments used for identification.

model variant	MSE sum
full nonlinear	2631.7
small angle	3349.5
linear	45061.7

Table 8.4.: Total MSE sum of the validation experiments.

model variant	MSE sum
full nonlinear	609552
small angle	795793
linear	576482

Nevertheless, the full nonlinear model provides a better simulation environment than the linear one because the hover dynamics are mainly influenced by ϕ and θ. The identified parameters are listed in Table 8.5. Please note that it contains two entries for parameter $d_{\psi,2}$ as the friction in this bearing shows a dependency on the direction of rotation.

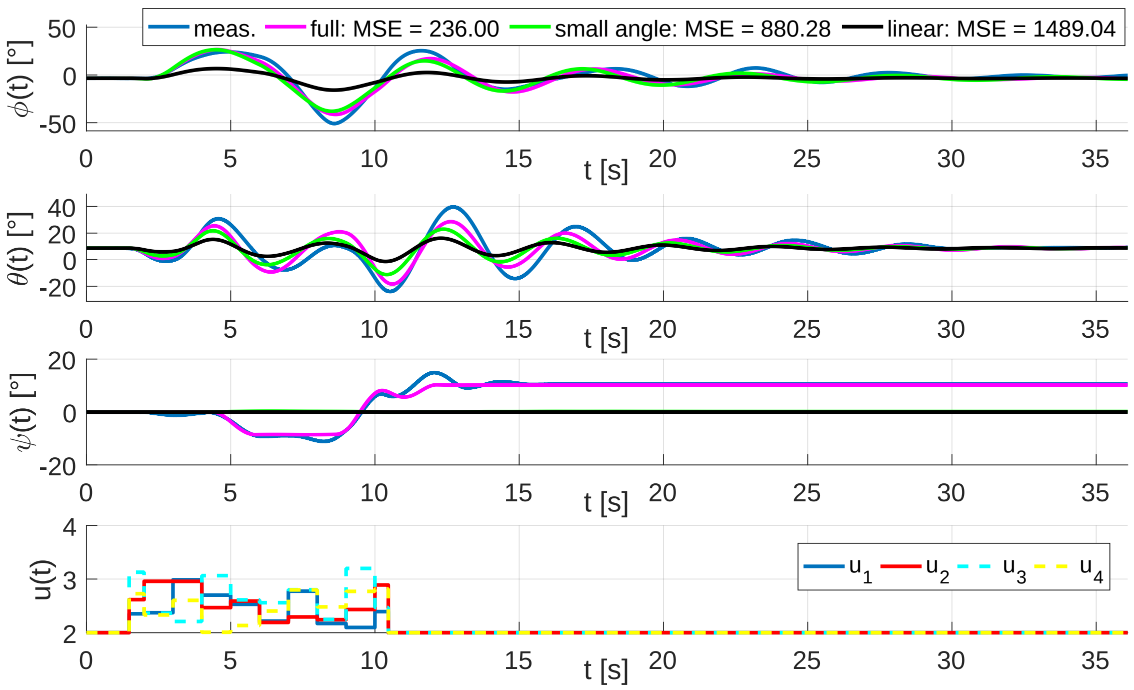

Figure 8.8.: One of the experiments with random input voltages which is used for the parameter identification of the hover.

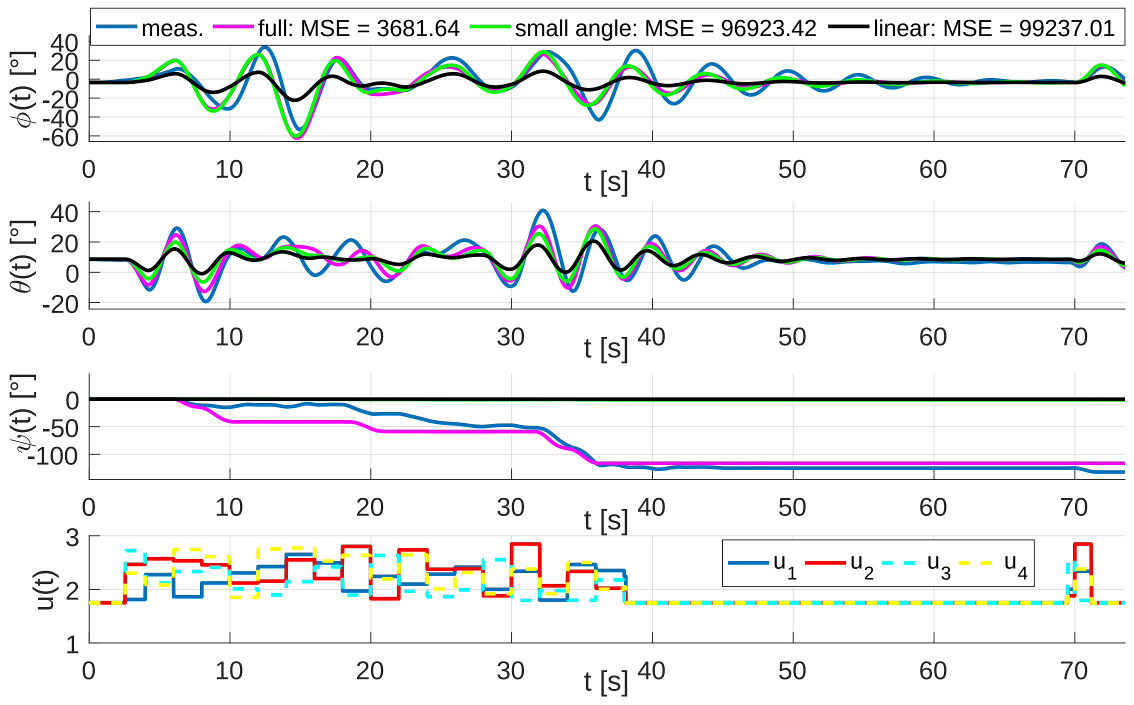

Figure 8.9.: Experiment with random input voltages which is used for validation.

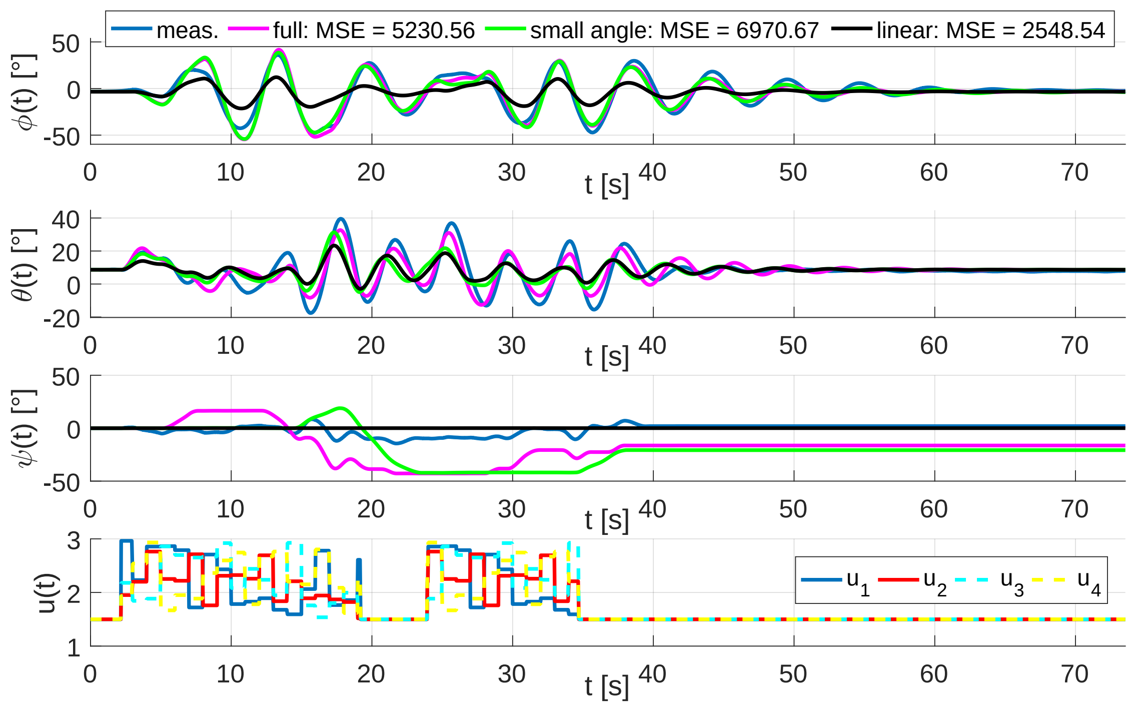

Figure 8.10.: Another validation experiment with random input voltages. Here one recognizes that the linear model keeps $\psi = 0$ which is true on average. On the other hand, the behavior of ϕ and θ is much worse than in case of the other models.

Table 8.5.: Identified parameters for the full nonlinear and small angle models.

parameter	full nonlinear	small angle
$m_{g,z}$	0.4399	0.3345
J_x	0.3302	0.2496
J_y	0.1633	0.1297
J_z	0.4480	0.2259
$d_{\phi,0}$	0.8095	0.0869
$d_{\phi,1}$	0.0997	0.0572
$d_{\theta,0}$	0.001	0.0006
$d_{\theta,1}$	0.0501	0.0431
$d_{\psi,0}$	280.8499	139.1543
$d_{\psi,1}$	87.4773	23.5491
$d_{\psi,2+}$	0.2254	0.1186
$d_{\psi,2-}$	0.1446	0.0712
k_ϕ	0.0503	0.0399
k_θ	0.0196	0.0116
k_ψ	0.0117	0.0063

8.3. Attitude control with control allocation

In order to develop a CA-based control scheme for reference angle tracking (8.27) is adapted. A virtual control vector $\boldsymbol{v} = [v_1\ v_2\ v_3]^T$ is introduced whose elements can be interpreted as torques around the three axis of rotation. The resulting control-oriented model reads as

$$\dot{\boldsymbol{x}}_{\mathbf{123}} = \boldsymbol{x}_{\mathbf{456}} \tag{8.28a}$$
$$\dot{\boldsymbol{x}}_{\mathbf{456}} = \boldsymbol{a}_{\mathbf{2}}(\boldsymbol{x}) + \boldsymbol{v} \tag{8.28b}$$
$$\boldsymbol{y} = \boldsymbol{x}_{\mathbf{123}}. \tag{8.28c}$$

This decoupling of inputs facilitates an independent controller development for roll, pitch, and yaw dynamics [56], [57]. In further consequence, these three virtual control signals have to be distributed among the actually available actuators by CA on the basis of $\boldsymbol{B}_{\boldsymbol{u}}(\boldsymbol{x})$. Assuming a known state vector $\boldsymbol{x}$ (see Section 8.3.6), during each controller execution step one obtains the linear relationship

$$\boldsymbol{v} = \boldsymbol{B}\boldsymbol{u} \tag{8.29}$$

where $\boldsymbol{B} \in \mathbb{R}^{3\times 4}$ is a constant control effectivity matrix with $\text{rank}(\boldsymbol{B}) = 3$, and the elements of $\boldsymbol{u}$ corresponding to squared voltages. The voltages are subject to the constraints[2]

$$u_{min} = 0 \le u_i \le 4 = u_{max} \quad \text{with } i = 1, \dots, 4. \tag{8.30}$$

In a typical real quadrotor application, in addition to the attitude its altitude also has to be controlled. For this purpose the total thrust [55], [56]

$$F_{act} = k_f\left(u_1^2 + u_2^2 + u_3^2 + u_4^2\right) = \underbrace{[k_f\ k_f\ k_f\ k_f]}_{\boldsymbol{k}_{\boldsymbol{f}}^T}\boldsymbol{u} \tag{8.31}$$

generated by all rotors is utilized. In the present laboratory setup the desired thrust is assumed to be constant during each experiment.

8.3.1. High-level controller design

Using the example of tracking a reference roll angle $x_{1,d}$ a Super-Twisting control law similar to [58] is derived. The first steps are the definition of the tracking error $e_1 = x_{1,d} - x_1$ with reference signal $x_{1,d}$ and of the corresponding error dynamics $\dot{e}_1 = \dot{x}_{1,d} - x_4$. The chosen sliding surface and its derivative read as

$$s_1 = \dot{e}_1 + c_\phi e_1 \tag{8.32a}$$
$$\dot{s}_1 = \ddot{x}_{1,d} - \dot{x}_4 + c_\phi \dot{x}_{1,d} - c_\phi x_4 \tag{8.32b}$$

where c_ϕ is a positive design parameter. For the purpose of obtaining a sliding mode on (8.32) the following Super-Twisting approach [58]

$$\dot{s}_1 = -\lambda_\phi |s_1|^{\frac{1}{2}} \operatorname{sign}(s_1) + w_1 \tag{8.33a}$$
$$\dot{w}_1 = -\alpha_\phi \operatorname{sign}(s_1) \tag{8.33b}$$

[2]The amplifier has an internal gain of 3 and so the actual voltage limit at the motors is 12 V.

with additional design parameters $\lambda_\phi > 0$ and $\alpha_\phi > 0$ is applied. By equating (8.32b) and (8.33a) one gets the roll dynamics related control law

$$\begin{aligned} v_1 = {} & \ddot{x}_{1,d} + c_\phi \dot{x}_{1,d} - c_\phi x_4 - a_1 x_5 x_6 - w_1 \\ & + a_2 e^{-a_3|x_4|} x_4 - a_4 \mathrm{si}_{(\phi_e - x_1)} + \lambda_\phi |s_1|^{\frac{1}{2}} \operatorname{sign}(s_1). \end{aligned} \tag{8.34}$$

After defining the tracking errors for the remaining angles θ and ψ as $e_2 = x_{2,d} - x_2$ and $e_3 = x_{3,d} - x_3$ two corresponding sliding manifolds are introduced as $s_2 = \dot{e}_2 + c_\theta e_2$ and $s_3 = \dot{e}_3 + c_\psi e_3$. Following a similar procedure as demonstrated for ϕ's control law one obtains

$$\begin{aligned} v_2 = {} & \ddot{x}_{2,d} + c_\theta \dot{x}_{2,d} - c_\theta x_5 - a_5 x_4 x_6 - w_2 \\ & + a_6 e^{-a_7|x_5|} x_5 - a_8 \mathrm{si}_{(\theta_e - x_2)} + \lambda_\theta |s_2|^{\frac{1}{2}} \operatorname{sign}(s_2) \end{aligned} \tag{8.35}$$

for pitch control and the yaw controller reads as

$$\begin{aligned} v_3 = {} & \ddot{x}_{3,d} + c_\psi \dot{x}_{3,d} - c_\psi x_6 - a_9 x_4 x_5 - w_3 \\ & + \left(a_{10} + a_{11} e^{-a_{12}|x_6|}\right) x_6 + \lambda_\psi |s_3|^{\frac{1}{2}} \operatorname{sign}(s_3). \end{aligned} \tag{8.36}$$

8.3.2. WPINV-based methods

In order to achieve a certain thrust F, a voltage $u_0 = \sqrt{\frac{F}{4k_f}}$ is applied to the motors. CA calculates $\hat{\boldsymbol{u}}$, which are deflections from $\boldsymbol{u_0} = [u_0^2\ u_0^2\ u_0^2\ u_0^2]^T$ such that the desired $\boldsymbol{v}$ is reached. Hence, the constraints considered in CA are altered into

$$\underbrace{-u_0^2}_{\hat{u}_{min}} \leq \hat{u}_i \leq \underbrace{u_{max}^2 - u_0^2}_{\hat{u}_{max}} \quad \text{with } i = 1, \ldots, 4. \tag{8.37}$$

The total input voltages for each motor read as

$$u_i = \begin{cases} \sqrt{\hat{u}_i + u_0^2} & \text{if } \hat{u}_i + u_0^2 \geq 0 \\ 0 & \text{else} \end{cases} \quad \text{with } i = 1, \ldots, 4. \tag{8.38}$$

Considering that $\boldsymbol{u_0}$ is part of $\boldsymbol{B}$'s right nullspace $\mathcal{N}_\mathrm{r}(\boldsymbol{B})$ (see Appendix D) it follows from (8.38) that $\boldsymbol{v} = \boldsymbol{B}\boldsymbol{u} = \boldsymbol{B}\boldsymbol{u_0} + \boldsymbol{B}\hat{\boldsymbol{u}} = \boldsymbol{B}\hat{\boldsymbol{u}}$, i.e. the virtual controls are not influenced by the offset. In summary, the actual CA problem is to solve

$$\boldsymbol{v} = \boldsymbol{B}\hat{\boldsymbol{u}} \tag{8.39}$$

for $\hat{\boldsymbol{u}}$ while satisfying (8.37). As (8.39) is an underdetermined linear system of equations an obvious way of solving is

$$\hat{\boldsymbol{u}} = \boldsymbol{B}^{\#}\boldsymbol{v} \tag{8.40}$$

where $\boldsymbol{B}^{\#} \in \mathbb{R}^{4\times 3}$ is a pseudoinverse (or generalized inverse) of $\boldsymbol{B}$ (see Section 2.1.2 for more details). If no constraints are active (8.40) is the solution of all CA-methods in this section.

Assumption 8.3.1. Since penalizing certain actuators is not useful here, $\boldsymbol{W} = \boldsymbol{I}$.

Assumption 8.3.2. The initial CA-result (8.40) is feasible (without subsequent saturation).

Proposition 8.3.1. *Under assumptions 8.3.1 and 8.3.2 the desired thrust is reached.*

Proof. From (8.31) and (8.38) it follows that the actually achieved thrust reads as

$$F_{act} = \boldsymbol{k_f}^T \boldsymbol{u} = \underbrace{\boldsymbol{k_f}^T \boldsymbol{u_0}}_{F} + \boldsymbol{k_f}^T \hat{\boldsymbol{u}} \tag{8.41}$$

which means that $\boldsymbol{k_f}^T \hat{\boldsymbol{u}} \overset{!}{=} 0$ is required. The initial CA-result is $\hat{\boldsymbol{u}} = \boldsymbol{B}^{\#}\boldsymbol{v}$ with $\boldsymbol{B}^{\#} = \boldsymbol{B}^T \left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}$ due to assumption 8.3.1. From rank $\left(\boldsymbol{B}\boldsymbol{B}^T\right) = k$ and (D.2) one can conclude

$$\mathcal{N}_{\mathrm{l}}(\boldsymbol{B}^{\#}) = \mathcal{N}_{\mathrm{l}}(\boldsymbol{B}^T) = \mathcal{N}_{\mathrm{r}}(\boldsymbol{B})^T = \mathrm{span}\left(\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}\right) \tag{8.42}$$

and considering $\boldsymbol{k_f}^T = k_f[1 \ \ 1 \ \ 1 \ \ 1]$ completes the proof. ■

8.3.2.1. Weighted Pseudoinverse

WPINV computes (8.40) via (2.8b). Its main advantage is the possibility to penalize the usage of certain actuators in case of detected failures [9]. Actuator constraints are not explicitly considered, potential exceedance is just cut off.

8.3.2.2. Redistributed Pseudoinverse

RPINV (see Section 2.1.4 for a description) is a constrained CA method which is based on the successive computation of weighted pseudoinverses [1]. Hence, it enables penalizing the usage of certain actuators (e.g. for incorporating detected actuator faults) in a simple manner. Its advantage is the iterative refinement of the solution which can lead to feasible results in cases where WPINV fails. RPINV maximally requires $m = 4$ iterations.

8.3.2.3. Enhanced Redistributed Pseudoinverse

Let j be the number of *not* saturated (free) controls after conducting RPINV and k the virtual control space dimension ($k = 3$ in the present case). It has been shown in Section 3.6 that if $j < k$ the desired $\boldsymbol{v}$ will generally not be reached. Therefore, in Chapter 5 the RPINV-extension ERPINV is derived which enables influencing the behavior in these situations. One can specify a priority list of $k - 1$ virtual control vector elements more important than the others. Taking the hover example, a priority list of [2 1] means reaching v_2 is more important than v_1, while v_3 is the least relevant element.

8.3.3. WPINV-free algorithms

The following two algorithms do not directly utilize (2.8b) for computing $\hat{\boldsymbol{u}}$. Hence, a different method for gaining the desired thrust is applied *after* CA. Let $\hat{\boldsymbol{u}}$ be the CA result satisfying the original constraints (8.30) and $\boldsymbol{N} = [n_1 \dots n_4]^T$ with span$(\boldsymbol{N}) = \mathcal{N}_{\mathrm{r}}(\boldsymbol{B})$. The resulting thrust is adjusted by a modification in nullspace direction without changing $\boldsymbol{v}$ and reads as

$$F_{act} = \boldsymbol{k_f}^T(\hat{\boldsymbol{u}} + \boldsymbol{N}\overline{x}). \tag{8.43}$$

From $F \overset{!}{=} F_{act}$ it follows that

$$\overline{x} = \frac{F - \boldsymbol{k_f}^T \hat{\boldsymbol{u}}}{\boldsymbol{k_f}^T \boldsymbol{N}}. \tag{8.44}$$

The actual motor input voltages are

$$u_i = \sqrt{\hat{u}_i + n_i x} \quad \text{with} \quad i = 1, \ldots, 4 \tag{8.45}$$

and scalar x is computed such that (8.30) is satisfied as

$$x = \begin{cases} \overline{x} & \text{if } \forall i : u_{min} \leq \sqrt{\hat{u}_i + n_i \overline{x}} \leq u_{max} \\ 0 & \text{else} \end{cases} \tag{8.46}$$

8.3.3.1. Direct allocation

DA is a constrained CA algorithm using geometric principles (see Section 2.1.5). The characteristics of this approach are the preservation of virtual control direction and the inherent utilization of the entire AMS [3]. The algorithm is implemented according to [4].

8.3.3.2. Normalized Generalized Inverse

Note that generalized inverses can compute feasible actuator commands $\hat{\boldsymbol{u}} \in \Omega$ only for a subset $\Pi \subseteq \Phi$. Both Π and Φ are convex k-dimensional (k-D) polytopes and a possible quality measure for generalized inverses is their volume ratio (4.1) [3], [4]. NINV (see Chapter 4) is a method to efficiently search for generalized inverses with high volume ratio. The main idea behind the algorithm is to restrict the search to a finite number of candidates instead of solving a continuous optimization problem. As NINV requires symmetric constraints, an input transformation

$$\underbrace{\hat{\boldsymbol{u}}_{\boldsymbol{min}} + \Delta\boldsymbol{u}}_{-\boldsymbol{u}'_{\boldsymbol{max}}} \leq \underbrace{\hat{\boldsymbol{u}} + \Delta\boldsymbol{u}}_{\boldsymbol{u}'} \leq \underbrace{\hat{\boldsymbol{u}}_{\boldsymbol{max}} + \Delta\boldsymbol{u}}_{\boldsymbol{u}'_{\boldsymbol{max}}} \tag{8.47}$$

is performed. Considering $-\hat{\boldsymbol{u}}_{\boldsymbol{min}} - \Delta\boldsymbol{u} = \hat{\boldsymbol{u}}_{\boldsymbol{max}} + \Delta\boldsymbol{u}$ the offset vector is computed as

$$\Delta\boldsymbol{u} = -\frac{\hat{\boldsymbol{u}}_{\boldsymbol{max}} + \hat{\boldsymbol{u}}_{\boldsymbol{min}}}{2}. \tag{8.48}$$

From $\boldsymbol{v} = \boldsymbol{B}\hat{\boldsymbol{u}} = \boldsymbol{B}\left(\boldsymbol{u}' - \Delta\boldsymbol{u}\right)$ one obtains the transformed CA problem

$$\underbrace{\boldsymbol{v} + \boldsymbol{B}\Delta\boldsymbol{u}}_{\boldsymbol{v}'} = \boldsymbol{B}\boldsymbol{u}' \tag{8.49}$$

which has to satisfy (8.47).

8.3.4. Numerical optimization

During each execution step of the control software the CA task can be formulated as optimization problem. On the laboratory setup quadratic cost functions are employed which generate constrained convex quadratic programs. In the absence of active constraints the solution corresponds with (8.40). Therefore, the demands of total thrust and virtual controls are combined in the same manner as in case of the Weighted Pseudoinverse based methods, in particular (8.37) and (8.38) apply. Due to the fact that CA is executed every $T_s = 1ms$ C-implementations of the algorithms are used to perform the optimization. Both solvers utilize Intel MKL 2018 [48] to speed up mathematical operations.

8.3.4.1. EPA

The principles of this method are the penalty function approach to account for the actuator constraints and a transformation of the problem into nullspace of $\boldsymbol{B}$ (see Chapter 6). The results of the algorithm's iterations approach the optimal values from the outside of the feasible region.

8.3.4.2. WLA

This algorithm solves the mixed optimization problem (2.28) for the 2-norm. The implementation is from QCAT [49] and is based on the active-set strategy [6].

8.3.5. Traditional allocation

The usual way of determining the input signal for quadrotors combines the demands from attitude controllers and (8.31) into a single system of equations [55]

$$\boldsymbol{u} = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_3^2 \\ u_4^2 \end{bmatrix} = \begin{bmatrix} 0 & \frac{k_\phi}{J_x} & 0 & -\frac{k_\phi}{J_x} \\ \frac{k_\theta}{J_y} & 0 & -\frac{k_\theta}{J_y} & 0 \\ \frac{k_\psi}{J_z} & -\frac{k_\psi}{J_z} & \frac{k_\psi}{J_z} & -\frac{k_\psi}{J_z} \\ k_f & k_f & k_f & k_f \end{bmatrix}^{-1} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ F \end{bmatrix} \tag{8.50}$$

with a unique solution. The main disadvantage of this procedure is the inability to take the actuator constraints into account. In the worst case this leads to unattainable negative thrusts. Any violations of (8.30) are cut off and result in deviations from the desired torques and thrust. The solution of (8.50) is called *Traditional allocation* (TA) for the rest of this work and taken as baseline for comparisons.

8.3.6. Implementation issues

The controller is implemented in Matlab/Simulink and QUARC ([61]) is used for real-time execution. The sampling time is set to $T_s = 1ms$. As the high-level control layer requires the time derivatives of the generalized coordinates (x_4, x_5, and x_6), they are estimated from the encoder signals. This is accomplished with a linear low-pass filter followed by difference quotient calculation with an overall transfer function of $D(z) = \frac{0.05(z-1)}{(z-0.95)T_s z}$.

8.3.7. Experimental results

All experiments are carried out with exactly the same parametrization of the Super-Twisting controller, only the method of input mapping is altered. The reference signals are square-waves and sinusoidal signals. They are chosen such that the actuators temporarily operate at their limits as recognizable in Figure 8.11 which depicts the resulting actuation signals from DA and RPINV in case of sinusoidal angle references. The tracking error shown for each angle and allocation method in Figures 8.12 - 8.19 is computed as

$$e_{abs}(x_i, x_{i,d}) = \frac{1}{N} \sum_{j=0}^{N-1} \left| x_{i,d}(jT_s) - x_i(jT_s) \right| \tag{8.51}$$

for $i = 1, \ldots, 3$ with N being the number of samples. The identity matrix is chosen for actuator weighting in all algorithms. TA and WPINV in Figure 8.12 reach the worst results

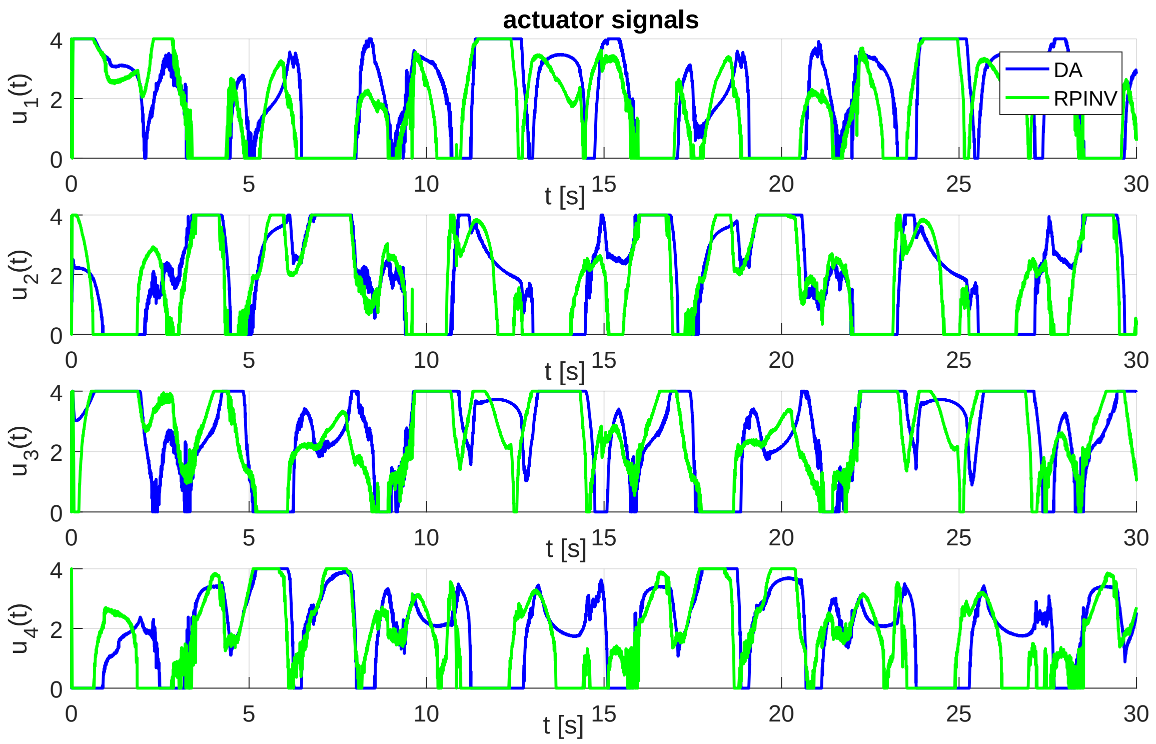

Figure 8.11.: Motor inputs from DA and RPINV during sinusoidal reference tracking (see Figure 8.13).

in the sinusoidal scenario in terms of the total error sum. In contrast NINV is perceptibly better in this case and its total error sum is comparable with that from DA in Figure 8.13. Even better performance is reached by RPINV. The influence of virtual control prioritization is recognizable in Figure 8.14 which depicts the three considered variants of ERPINV. The numbers in the subscripts indicate the prioritized virtual controls. Precedence for v_1 and v_2 yields very good tracking results for ϕ and θ (blue curves) whereas prioritizing v_2 and v_3 improves to results for θ and ψ (green curve). One can see that in this experiment favoring virtual controls 1 and 3 leads to the best overall result (black dashed curve). The results from numerical optimization are presented in Figure 8.15. WLA (with $\boldsymbol{W_v} = \boldsymbol{I}$) leads to similar results as RPINV whereas EPA is better for ψ-tracking at the prize of larger errors at the other angles. The differences between WLA and EPA originate from the fact that $\boldsymbol{B_u}(\boldsymbol{x})$ does not comply with Assumption 6.2.2 and gradient projection has been deactivated for EPA in order to demonstrate the behavior without virtual control weighting. In general, tracking works sufficiently well here because the continuous references come close to a realizable trajectory. The second column in Table 8.6 condenses the sinusoidal experiments by showing the total sum of errors. In this context the top-three results come from ERPINV13, WLA, and RPINV. Due to the actuator saturations the desired $\boldsymbol{v}$ cannot always be achieved. The average relative allocation error is denoted as e_v and given in Table 8.6. It should be noted that ERPINV12 and ERPINV23 produce less allocation errors but more tracking errors than ERPINV13, WLA, and EPA.

Figure 8.16 shows the square-wave results for TA, WPINV, and NINV. First of all, it is apparent that following the square references is more challenging than the sinusoidal ones. One recognizes that TA and WPINV yield very similar results which is not surprising as none of them regards the constraints. NINV is able to achieve better results for ϕ but is less accurate at tracking θ. Altogether, NINV gives no benefit in this test scenario. Figure 8.17 contains the results for DA and conventional RPINV. DA shows very good general performance, i.e. the errors for all angles are moderate. RPINV achieves slightly better results for ϕ and θ tracking but this comes at the price of larger errors for ψ. Once again one can see in Figure 8.18 that favoring virtual controls 1 and 3 leads to the best overall behavior (black dashed curve). Finally, the results from optimization are shown in Figure 8.19. RPINV and WLA are again quite similar. Taking the overall error sum in the fourth column of Table 8.6 as the quality criterion the top three methods are ERPINV13, ERPINV23, and DA. It is also notable that the next best result is achieved by TA lying before WLA although TA has a much greater average error on $\boldsymbol{v}$.

Other important aspects are the efforts for implementation and computation. TA and WPINV are clearly the least demanding to implement and compute. The most challenging part in the NINV implementation is the candidate generation which can be done offline to reduce the computational effort. The online part only consists of a loop iterating through all candidates and evaluating (4.52) and (4.1). DA can easily be realized as Linear Program or with much more effort directly implemented according to [4]. Here, the later approach is chosen which does not rely on a separate solver and is optimized in terms of execution speed. RPINV and ERPINV only differ in the additional steps 1-7 which are explained in Section 5.2.1. Other than that they basically comprise a loop with at most m iterations computing $\boldsymbol{B}^{\#}$ via (2.8b) followed by (2.19). Proceeding from a given solver the approaches based on numerical optimization are straightforward to implement as they only involve the formulation of a QP. The efficient implementation of EPA is more elaborate than in case of WLA. The computation time for both algorithms can be significantly higher than for RPINV and ERPINV especially for greater numbers of actuators than in the present case.

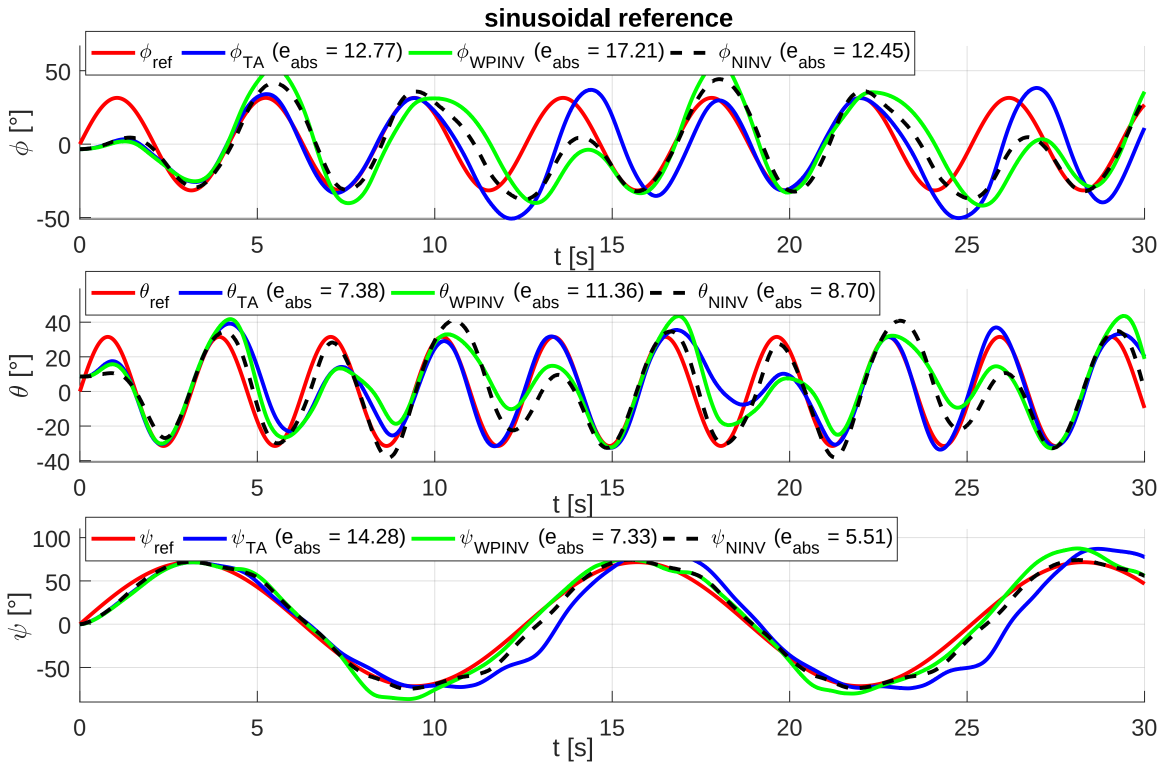

Figure 8.12.: Comparison of TA, WPINV, and NINV.

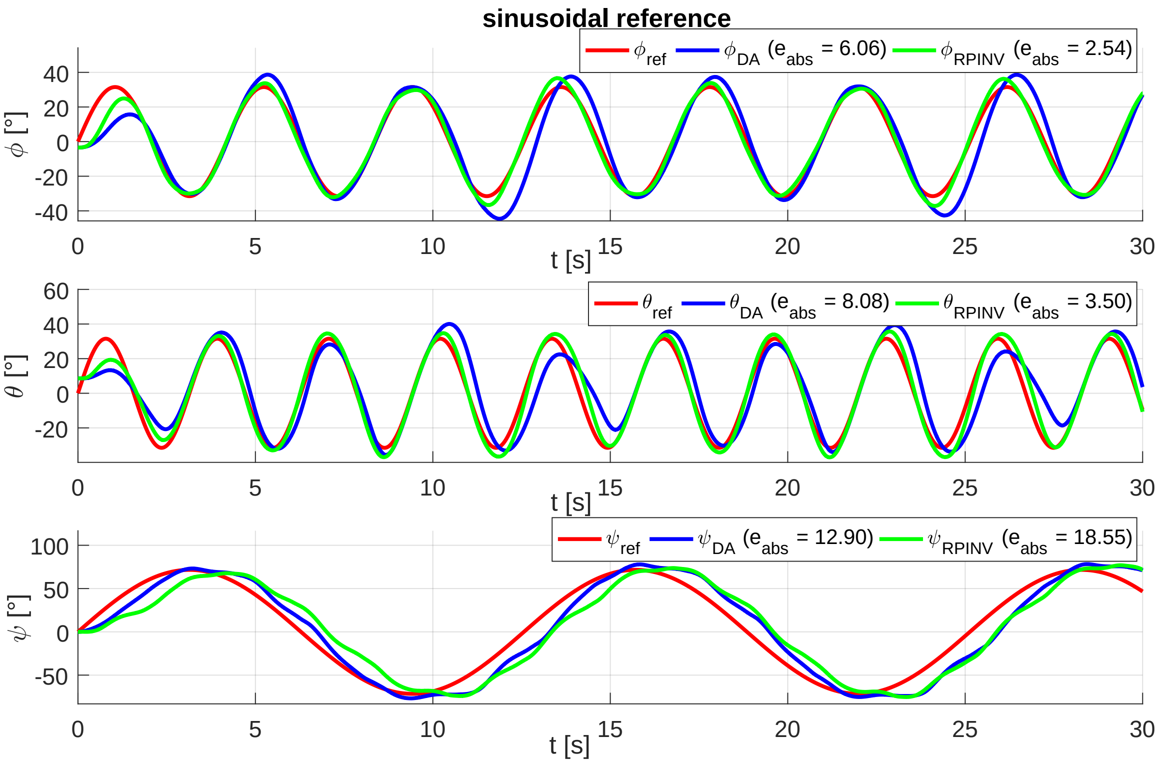

Figure 8.13.: Comparison of DA and RPINV.

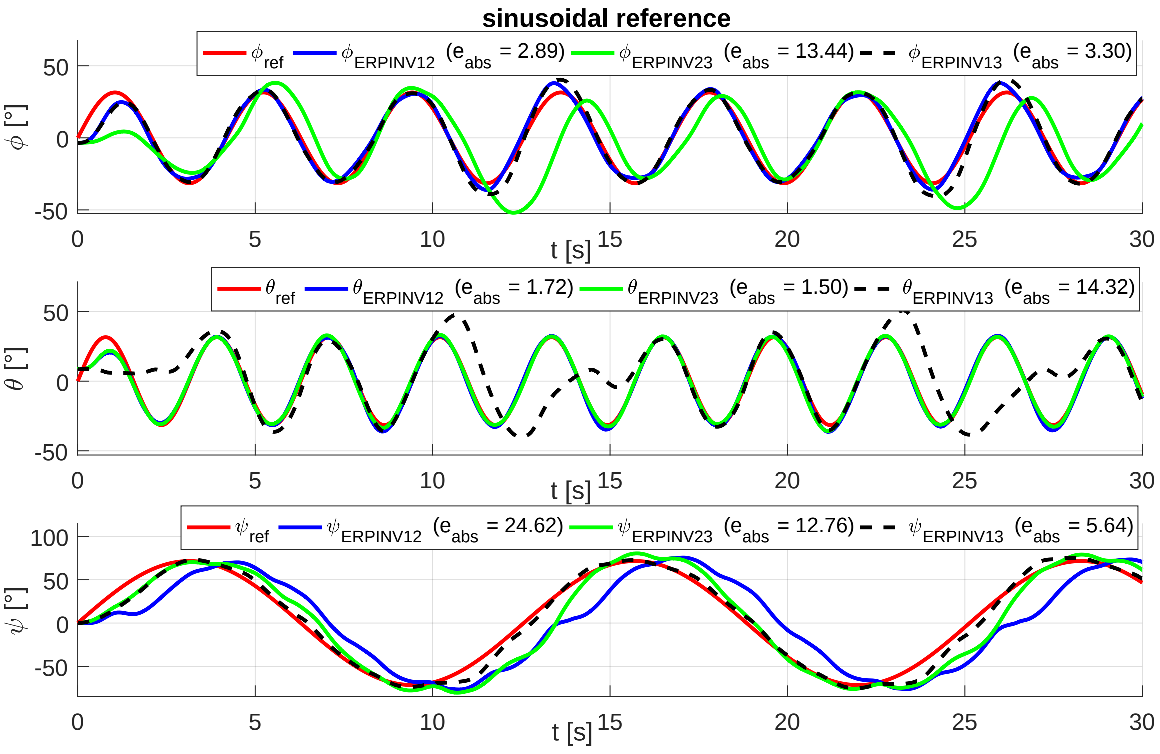

Figure 8.14.: Comparison of ERPINV with differently prioritized virtual controls.

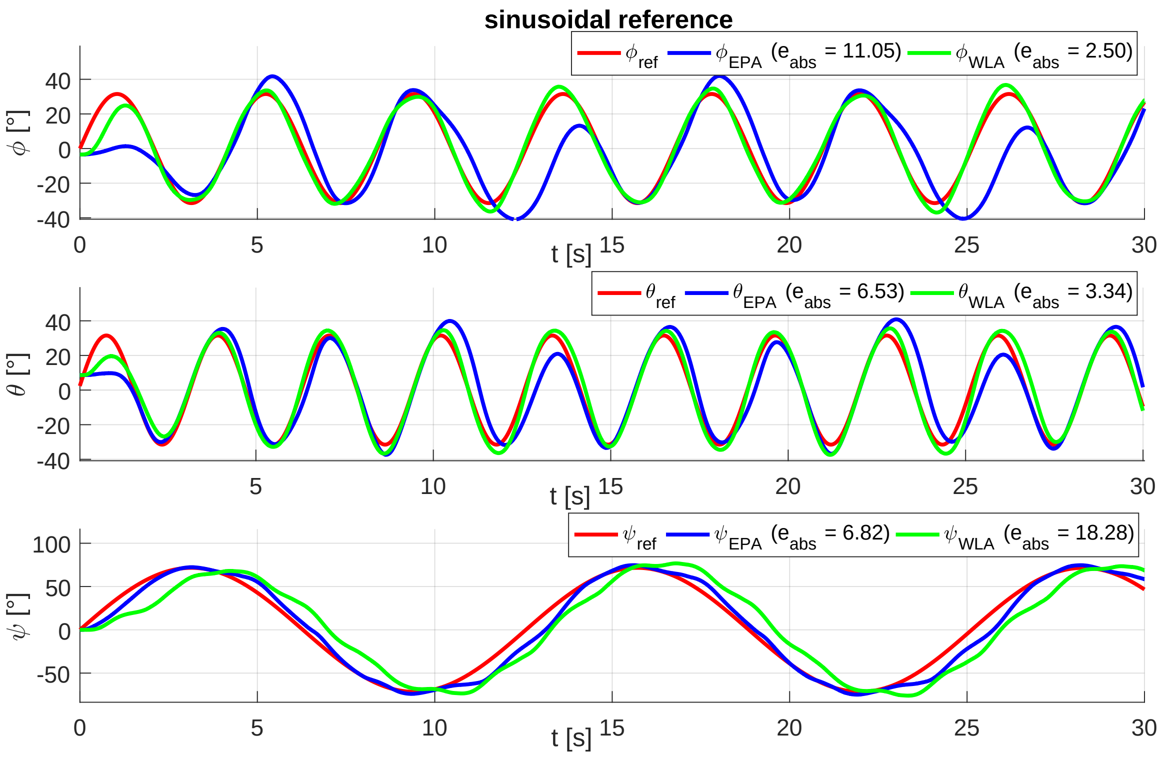

Figure 8.15.: Comparison of WLA and EPA.

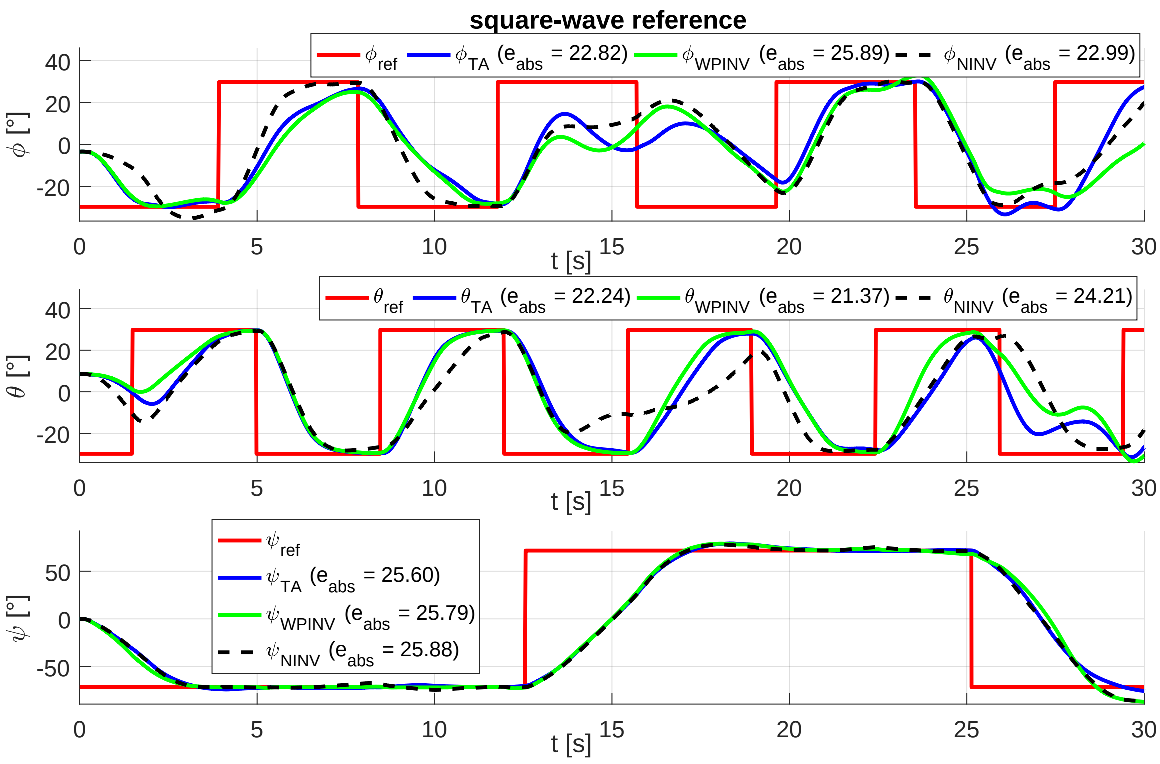

Figure 8.16.: Comparison of TA, WPINV, and NINV.

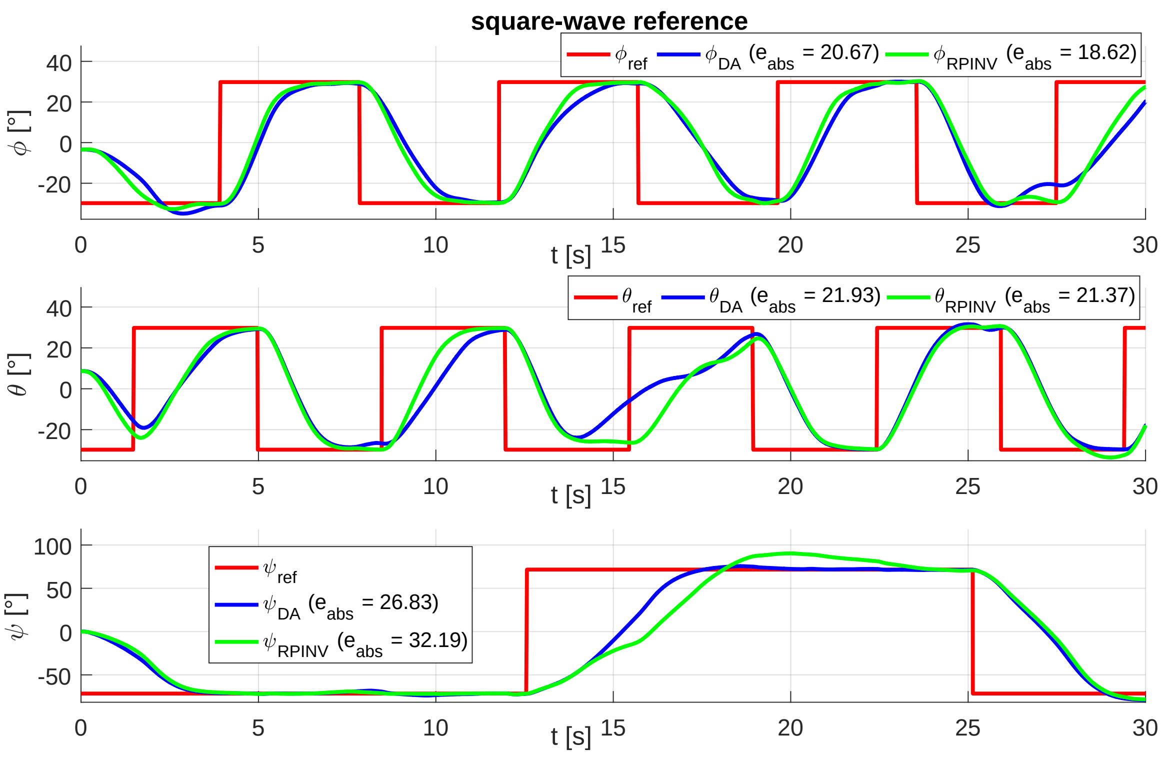

Figure 8.17.: Comparison of DA and RPINV.

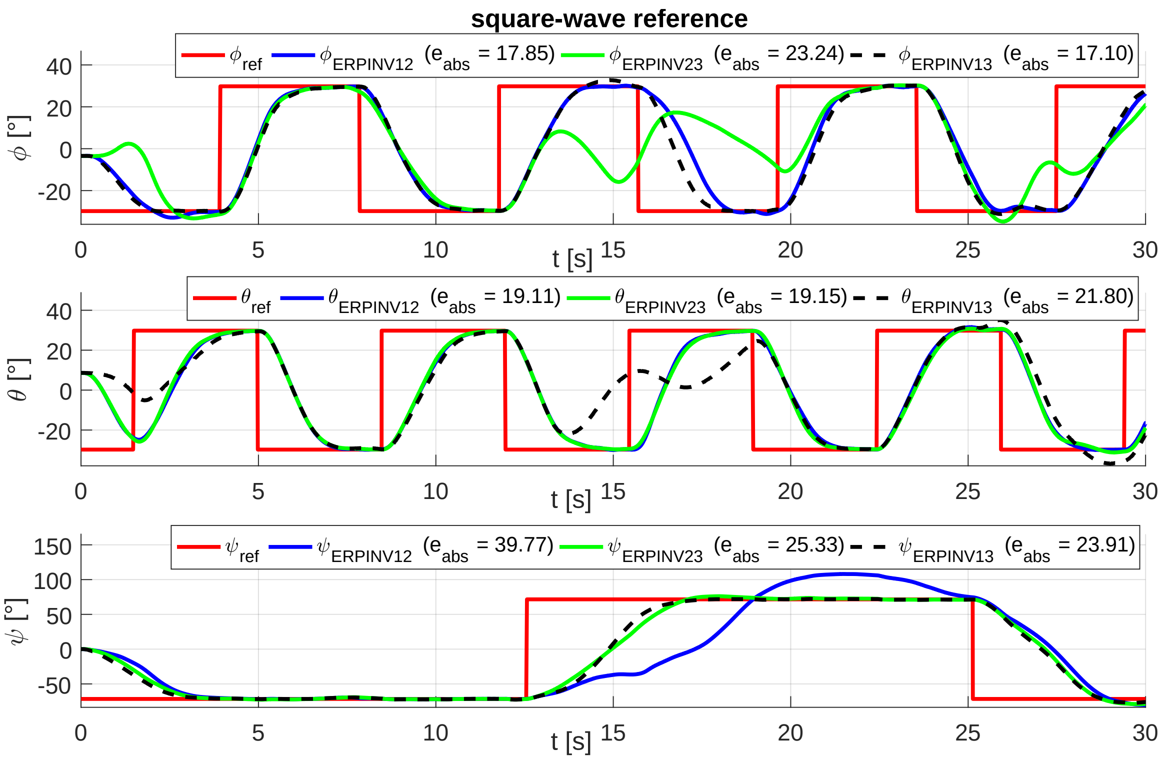

Figure 8.18.: Comparison of ERPINV with differently prioritized virtual controls.

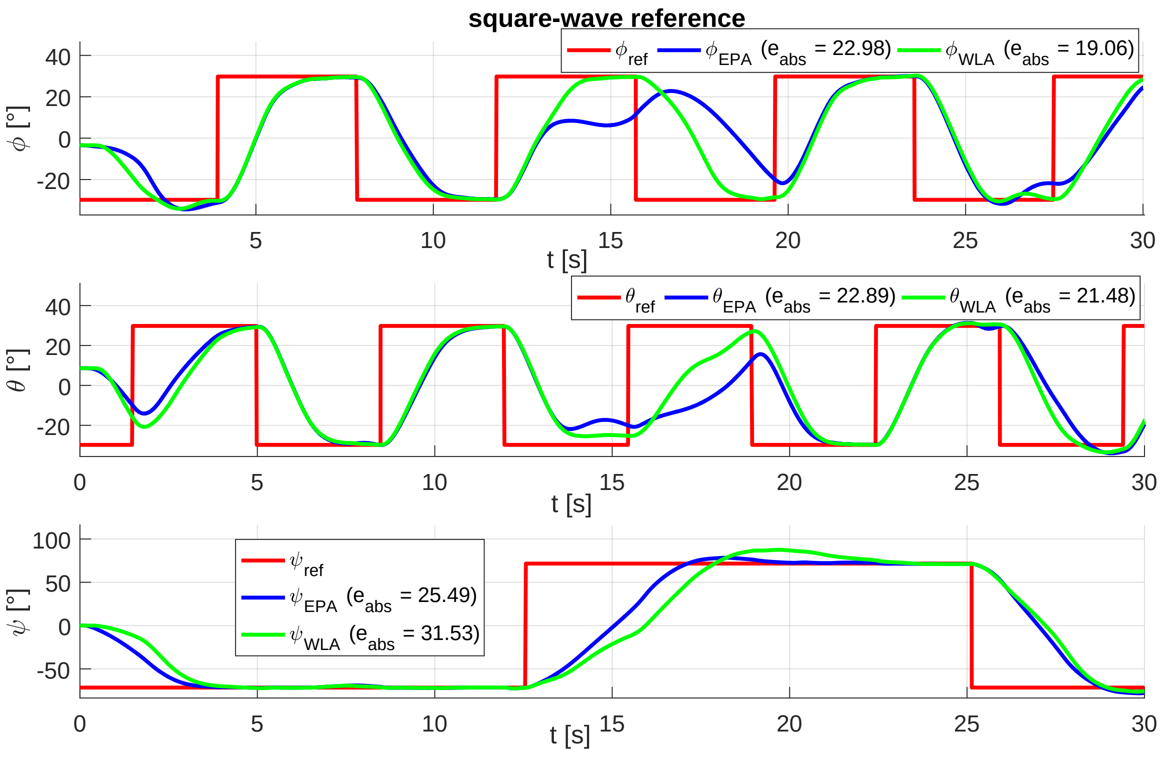

Figure 8.19.: Comparison of WLA and EPA.

Table 8.6.: Comparison in terms of angle error, allocation error, and efforts for implementation and computations.

algorithm	sinus: $\sum e_{abs}$	sinus: e_v	square: $\sum e_{abs}$	square: e_v	impl. effort	comp. effort
TA	34.43	56%	70.66	48%	very low	very low
WPINV	35.90	55%	73.05	38%	very low	very low
NINV	26.66	45%	73.08	31%	medium	online part: low
DA	27.04	44%	69.43	32%	high	medium
RPINV	24.59	41%	72.18	29%	medium	medium
ERPINV12	29.23	37%	76.73	27%	medium	medium
ERPINV23	27.70	35%	67.72	27%		
ERPINV13	23.26	41%	62.81	29%		
WLA	24.12	41%	72.07	28%	without solver: low with solver: medium	high
EPA	24.40	46%	71.36	14%	without solver: low with solver: high	high

8.4. Attitude control with LQR

One alternative approach to solve the attitude control problem builds on the time-invariant LQR design with prefilter as described in Section 2.2.2. The first step is the determination of a linear time-invariant model of the hover dynamics. For this purpose the nonlinear plant model

$$\dot{\boldsymbol{x}} = \boldsymbol{f_1}(\boldsymbol{x}, \underline{\boldsymbol{u}}) \tag{8.52a}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) \tag{8.52b}$$

with input vector[3]

$$\underline{\boldsymbol{u}} = [u_1 \;\; u_2 \;\; u_3 \;\; u_4]^T \tag{8.53}$$

is linearized at $\boldsymbol{x_e} = [\phi_e \;\; \theta_e \;\; 0 \;\; 0 \;\; 0 \;\; 0]^T$ which is an equilibrium, i.e. $\dot{\boldsymbol{x}} = \boldsymbol{0} = \boldsymbol{f_1}(\boldsymbol{x_e}, \underline{\boldsymbol{u}}_{\boldsymbol{e}})$. Since ϕ_e and θ_e are the equilibrium roll and pitch angles of the unactuated system it follows from (8.24) that $\boldsymbol{a_1}(\boldsymbol{x_e}) = \boldsymbol{0}$. Taking (D.2) into account reveals that all input vectors with equal elements maintain this equilibrium. Thus, the desired total thrust (8.31) determines the input vector

$$\underline{\boldsymbol{u}}_{\boldsymbol{e}} = [u_0 \;\; u_0 \;\; u_0 \;\; u_0]^T \quad \text{with} \quad u_0 = \sqrt{\frac{F}{4k_f}}. \tag{8.54}$$

States, inputs, and outputs can be written as deviations from the equilibrium as

$$\boldsymbol{x}(t) = \boldsymbol{x_e} + \hat{\boldsymbol{x}}(t) \tag{8.55a}$$

$$\underline{\boldsymbol{u}}(t) = \underline{\boldsymbol{u}}_{\boldsymbol{e}} + \underline{\hat{\boldsymbol{u}}}(t) \tag{8.55b}$$

$$\boldsymbol{y}(t) = \boldsymbol{y_e} + \hat{\boldsymbol{y}}(t). \tag{8.55c}$$

Taking the time derivative of (8.55a) and using a Taylor series expansion yields

$$\dot{\hat{\boldsymbol{x}}} \approx \underbrace{\boldsymbol{f_1}(\boldsymbol{x_e}, \underline{\boldsymbol{u}}_{\boldsymbol{e}})}_{\boldsymbol{0}} + \underbrace{\left.\frac{\partial \boldsymbol{f_1}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x_e}\\ \underline{\boldsymbol{u}}=\underline{\boldsymbol{u}}_{\boldsymbol{e}}}}}_{\boldsymbol{A}} \cdot (\boldsymbol{x} - \boldsymbol{x_e}) + \underbrace{\left.\frac{\partial \boldsymbol{f_1}}{\partial \underline{\boldsymbol{u}}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x_e}\\ \underline{\boldsymbol{u}}=\underline{\boldsymbol{u}}_{\boldsymbol{e}}}}}_{\boldsymbol{B_u}} \cdot (\underline{\boldsymbol{u}} - \underline{\boldsymbol{u}}_{\boldsymbol{e}}) \tag{8.56}$$

and similarly the output equation reads as

$$\boldsymbol{y} \approx \underbrace{\boldsymbol{h}(\boldsymbol{x_e})}_{\boldsymbol{y_e}} + \underbrace{\left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x_e}}}_{\boldsymbol{C}} \cdot (\boldsymbol{x} - \boldsymbol{x_e}). \tag{8.57}$$

The linear time-invariant model obtained from (8.56) and (8.57) is

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B_u}\underline{\hat{\boldsymbol{u}}} \tag{8.58a}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{C}\hat{\boldsymbol{x}}. \tag{8.58b}$$

with parameter matrices given in Appendix (D.2). The LQR control law which follows from (2.49) and (8.55b) reads as

$$\underline{\boldsymbol{u}} = \underline{\boldsymbol{u}}_{\boldsymbol{e}} - \boldsymbol{K}\hat{\boldsymbol{x}} + \boldsymbol{V}\left(\boldsymbol{r} - \boldsymbol{y_e}\right). \tag{8.59}$$

[3]Note that the input vector definition (8.53) of the LQR approach differs from the one used for CA (8.21) which consists of squared input voltages. This is indicated by the underline.

8.4.1. Simulation results

In order to verify the basic functionality of the LQR controller simulations are carried out. First, the controller is tested together with the linear plant model for which it has been developed. Figures 8.20 and 8.21 show that tracking of step-like angle references does not exhibit steady-state errors as expected. The linearization has been carried out for an input voltage of $u_0 = 1.5$ V. The state and input weighting matrices are chosen as

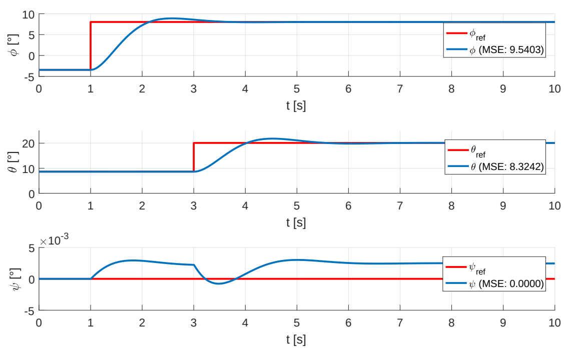

Figure 8.20.: Simulation result of LQR: The simulated plant coincides with the model for LQR design and so there are no steady state errors.

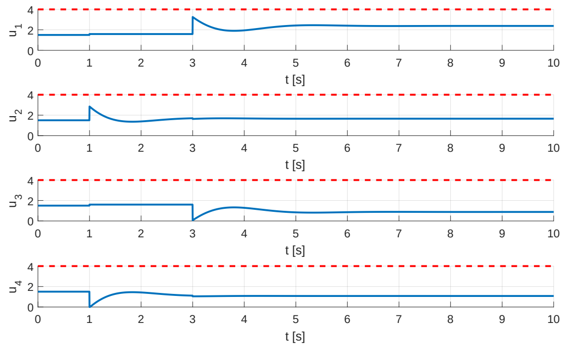

Figure 8.21.: Control signals originating from the LQR simulation in Figure 8.20.

$$\boldsymbol{Q} = \operatorname{diag}(100, 100, 50, 0, 0, 0) \qquad (8.60a) \qquad\qquad \boldsymbol{R} = \boldsymbol{I_4} \qquad (8.60b)$$

respectively. In case of simulations incorporating the full nonlinear plant model (8.24) and angles further away from the equilibrium the steady-state errors remain visible as it can be seen in Figure 8.22. Figure 8.23 shows that in this case input saturation does not have significant influence on the result. Due to the fact that the input constraints are not considered in the LQR design there is a strong suspicion that tracking of reference signals leading to input saturation suffers from further degraded performance. This is confirmed by Figures 8.24 and 8.25 illustrating the behavior for reference signals comprising a sequence of steps.

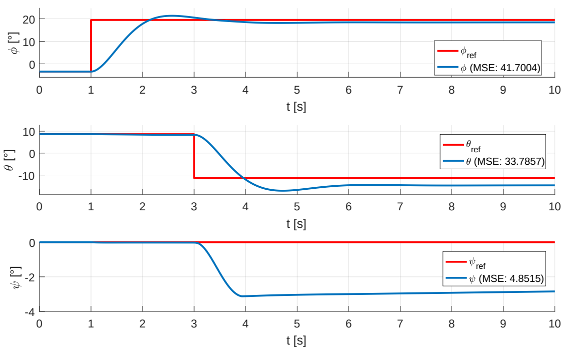

Figure 8.22.: When the nonlinear plant model is used in simulation the LQR cannot prevent steady state errors.

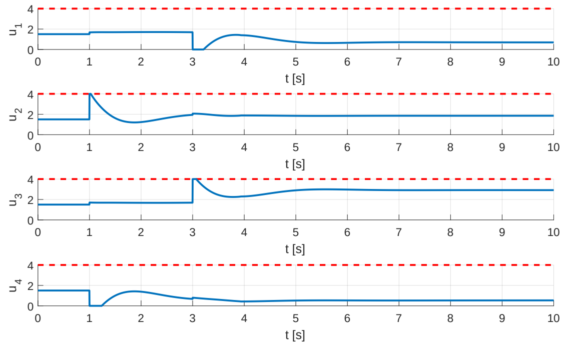

Figure 8.23.: Input signals of the LQR simulation shown in Figure 8.22. The time span where input saturation occurs is very short.

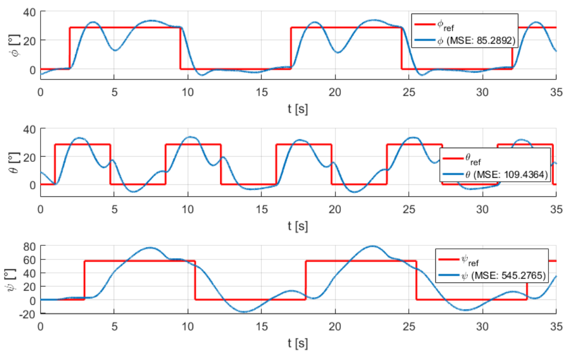

Figure 8.24.: Closed-loop behavior of LQR operating on the nonlinear plant model for angle step sequences.

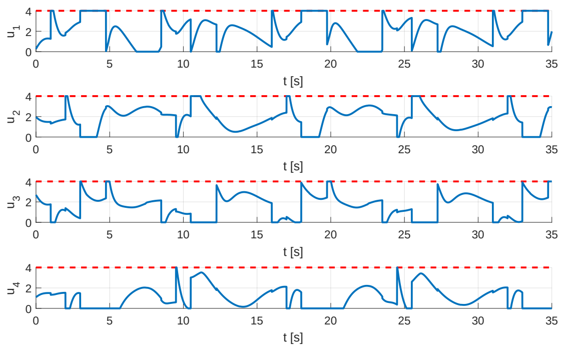

Figure 8.25.: Input signals corresponding to Figure 8.24.

8.4.2. Experimental results

LQR replaces high-level control and CA while the rest of the control software stays the same. The sampling time remains $T_s = 1ms$ as in the CA-based approach. The differentiation of the encoder signals is carried out in the same way as described in Section 8.3.6. The weighting matrices (8.60a) and (8.60b) from simulation are also used in the QUARC implementation of the LQR. Initially an experiment with small reference angle changes is conducted. Figure 8.26 illustrates that the controller works reasonably well in this situation. Unlike in simulations a

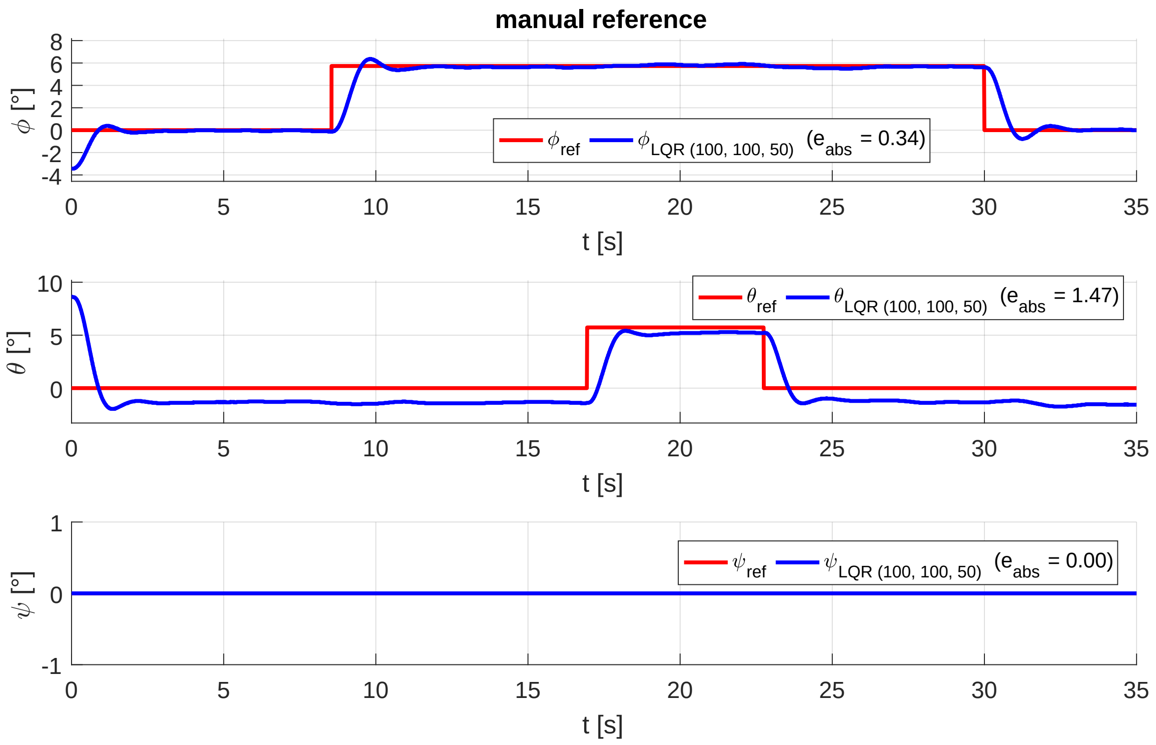

Figure 8.26.: LQR result for small reference steps.

steady-state error occurs due to model mismatches and the fact that the controller exhibits no integral action. The corresponding control signals are shown in Figure 8.27. Due to the fact that the controller has been developed based on a linearized model at the equilibrium one cannot expect it to work very well for large reference signals. Therefore, the experiments from

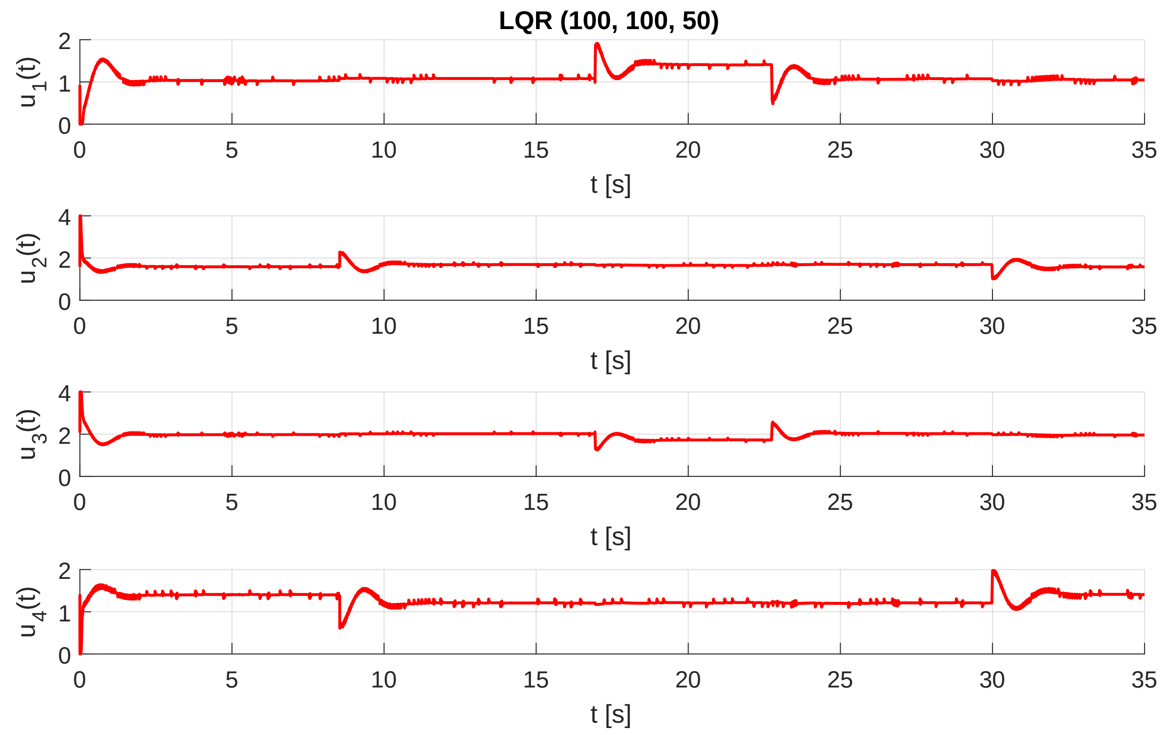

Figure 8.27.: Inputs for the LQR experiment from Figure 8.26.

the CA-based approaches are repeated with a lower amplitude of the reference signals. Figure 8.28 shows the results for a square-wave reference and Figure 8.29 displays the behavior for sinusoidal references.

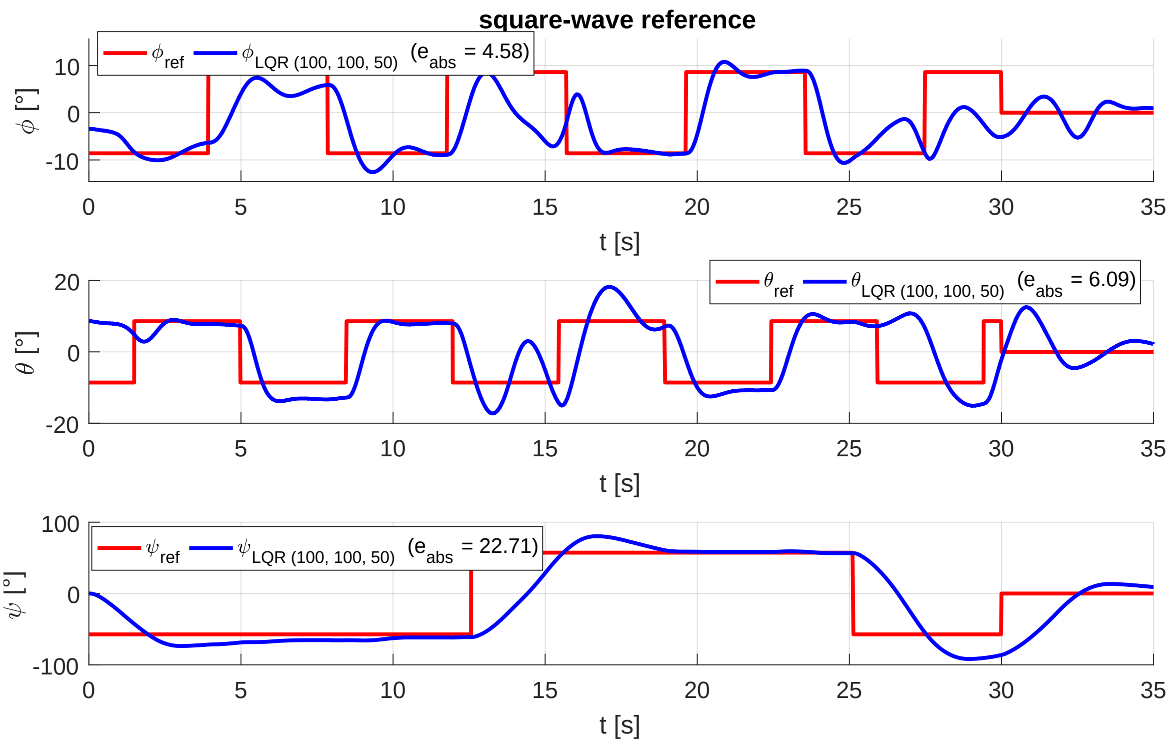

Figure 8.28.: LQR result for square-wave reference signals.

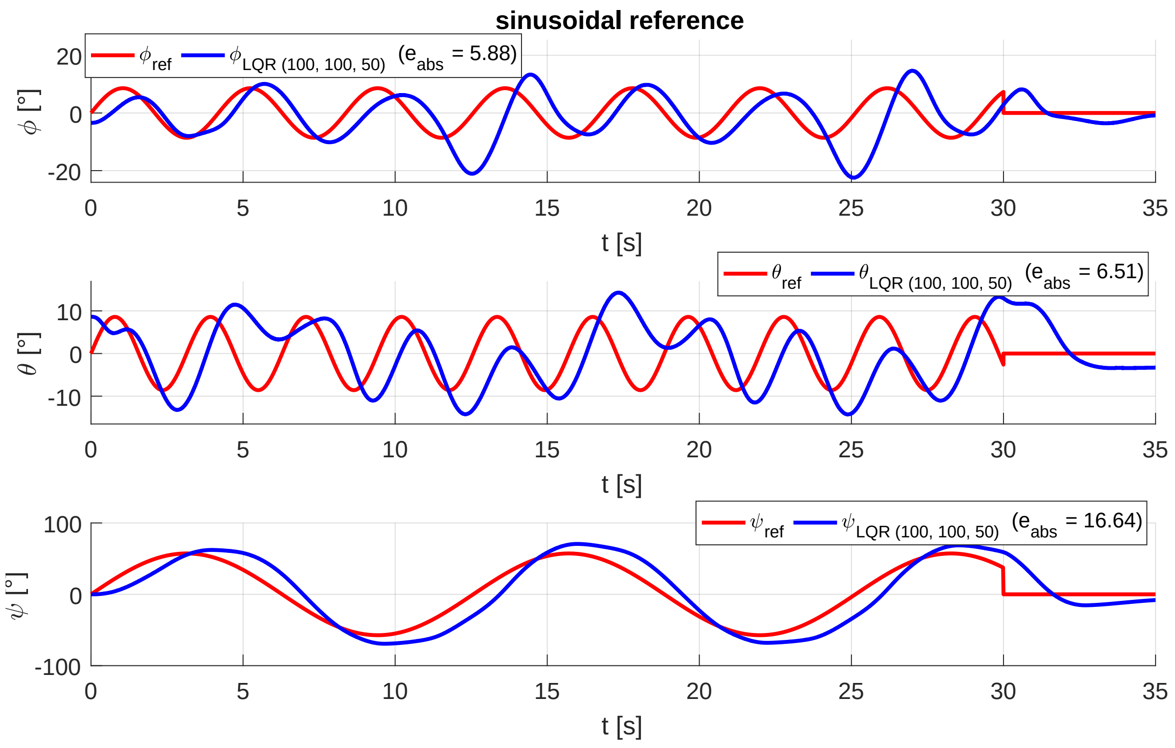

Figure 8.29.: LQR result for sinusoidal reference signals.

8.5. Attitude control with MPC

A controller which builds on the prediction of the plant's behavior apparently benefits from a model as accurate as possible. Hence, intuitively the usage of the full nonlinear model (8.23) promises the best results. Unfortunately, a nonlinear model usually leads to a non-convex optimization problem whose global optimum can hardly be found online [19]. The combination of a linear model with quadratic cost function and linear constraints on the other hand yields a problem which is much easier to solve. Furthermore, the approach should be comparable to the control strategies from Section 8.3 where CA is done on the basis of a linear relationship gained from the evaluation of the nonlinear model at the current state. Thus, the MPC developed in this section rests on a similar idea. In each execution step an affine approximation of (8.25) is computed which depends on the current state and inputs. After determining a discrete-time representation of that model a QP with linear constraints is constructed and solved.

8.5.1. Model

MPC is implemented in a discrete-time environment with higher sampling time T_s than the previously presented approaches owed to the greater computational effort for optimization. Therefore, the time discretization is explicitly considered in the model derivation. At the beginning of the current sampling instant k the states, outputs, and inputs of the hover are given by

$$\boldsymbol{x_k} := \boldsymbol{x}(kT_s) \tag{8.61a}$$
$$\boldsymbol{y_k} := \boldsymbol{y}(kT_s) \tag{8.61b}$$
$$\boldsymbol{u_{k-1}} := \boldsymbol{u}((k-1)T_s). \tag{8.61c}$$

Initially, an approximation of the nonlinear continuous-time plant model (8.25) is calculated which describes the deviations from (8.61) labeled by a hat above the symbol

$$\begin{aligned}\boldsymbol{x}(t) &= \boldsymbol{x_k} + \hat{\boldsymbol{x}}(t)\\ \boldsymbol{y}(t) &= \boldsymbol{y_k} + \hat{\boldsymbol{y}}(t)\\ \boldsymbol{u}(t) &= \boldsymbol{u_{k-1}} + \hat{\boldsymbol{u}}(t).\end{aligned} \tag{8.62}$$

After defining the short notation

$$\boldsymbol{f_2}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} \boldsymbol{x_{456}} \\ \boldsymbol{a_2}(\boldsymbol{x}) + \boldsymbol{B_{u,2}u} \end{bmatrix} \tag{8.63}$$

for the stacked right-hand sides of (8.25a) and (8.25) the first-order terms of a Taylor series expansion read as

$$\dot{\hat{\boldsymbol{x}}} \approx \underbrace{\boldsymbol{f_2}(\boldsymbol{x_k},\boldsymbol{u_{k-1}})}_{\hat{\boldsymbol{z}}_{\boldsymbol{k}}} + \underbrace{\left.\frac{\partial \boldsymbol{f_2}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x_k}\\ \boldsymbol{u}=\boldsymbol{u_{k-1}}}}}_{\hat{\boldsymbol{A}}} \cdot(\boldsymbol{x}-\boldsymbol{x_k}) + \underbrace{\left.\frac{\partial \boldsymbol{f_2}}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x_k}\\ \boldsymbol{u}=\boldsymbol{u_{k-1}}}}}_{\hat{\boldsymbol{B}}} \cdot(\boldsymbol{u}-\boldsymbol{u_{k-1}}) \tag{8.64}$$

and doing the same for the output (8.25c) gives

$$\hat{\boldsymbol{y}} \approx \underbrace{\boldsymbol{h}(\boldsymbol{x_k})}_{\boldsymbol{y_k}} + \underbrace{\left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x_k}}}_{\boldsymbol{C}} \cdot(\boldsymbol{x}-\boldsymbol{x_k}). \tag{8.65}$$

Combining (8.62) - (8.65) results in the model approximation

$$\dot{\hat{\boldsymbol{x}}} = \hat{\boldsymbol{A}}\hat{\boldsymbol{x}}(t) + \hat{\boldsymbol{B}}\hat{\boldsymbol{u}}(t) + \hat{\boldsymbol{z}}_{\boldsymbol{k}} \tag{8.66a}$$
$$\hat{\boldsymbol{y}}(t) = \boldsymbol{C}\hat{\boldsymbol{x}}(t) \tag{8.66b}$$

whose parameters $\hat{\boldsymbol{A}}$, $\hat{\boldsymbol{B}}$, and $\hat{\boldsymbol{z}}_{\boldsymbol{k}}$ are successively recomputed during each execution step while the output matrix remains constant $\boldsymbol{C} = [\boldsymbol{I}\ \ \boldsymbol{0}]$. Since $\hat{\boldsymbol{y}}$ represents the output deviation from its current value $\boldsymbol{y_k}$ an adaption of the reference angles $\boldsymbol{r}$ is required

$$\boldsymbol{r} \overset{!}{=} \boldsymbol{y} = \boldsymbol{y_k} + \hat{\boldsymbol{y}} \quad \Rightarrow \quad \hat{\boldsymbol{r}} := \boldsymbol{r} - \boldsymbol{y_k}. \tag{8.67}$$

The next step is the time-discretization of (8.66). Disregarding $\hat{\boldsymbol{z}}_{\boldsymbol{k}}$ in (8.66) the discrete-time state equations could be written as $\hat{\boldsymbol{x}}_{\boldsymbol{k+1}} = \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{k}}$ with [66]

$$\boldsymbol{A} = \boldsymbol{\Phi}_{\hat{\boldsymbol{A}}}(T_s) \tag{8.68a}$$
$$\boldsymbol{B} = \left[\int_0^{T_s} \boldsymbol{\Phi}_{\hat{\boldsymbol{A}}}(t)dt\right]\hat{\boldsymbol{B}} \tag{8.68b}$$

incorporating the state transition matrix [66]

$$\boldsymbol{\Phi}_{\hat{\boldsymbol{A}}}(t) = \boldsymbol{I} + \hat{\boldsymbol{A}}t + \frac{\left(\hat{\boldsymbol{A}}t\right)^2}{2!} + \frac{\left(\hat{\boldsymbol{A}}t\right)^3}{3!} + \frac{\left(\hat{\boldsymbol{A}}t\right)^4}{4!} + \dots . \tag{8.69}$$

Note that (8.66a) can also be written as

$$\dot{\hat{\boldsymbol{x}}} = \hat{\boldsymbol{A}}\hat{\boldsymbol{x}} + \begin{bmatrix}\hat{\boldsymbol{B}} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{u}} \\ \hat{\boldsymbol{z}}_{\boldsymbol{k}}\end{bmatrix} \tag{8.70}$$

comprising an augmented input matrix and vector. From (8.68) and (8.70) it follows that the discrete-time representation of (8.66) reads as

$$\hat{\boldsymbol{x}}_{\boldsymbol{k+1}} = \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{k}} + \boldsymbol{z}_{\boldsymbol{k}} \tag{8.71a}$$

$$\hat{\boldsymbol{y}}_{\boldsymbol{k}} = \boldsymbol{C}\hat{\boldsymbol{x}}_{\boldsymbol{k}}. \tag{8.71b}$$

with

$$\boldsymbol{z}_{\boldsymbol{k}} = \left[\int_0^{T_s} \boldsymbol{\Phi}_{\hat{\boldsymbol{A}}}(t)dt\right]\hat{\boldsymbol{z}}_{\boldsymbol{k}}. \tag{8.72}$$

The evaluation of the integral in (8.68b) and (8.72) yields

$$\begin{aligned}\int_0^{T_s}\left[\boldsymbol{I} + \hat{\boldsymbol{A}}t + \frac{\left(\hat{\boldsymbol{A}}t\right)^2}{2!} + \frac{\left(\hat{\boldsymbol{A}}t\right)^3}{3!} + \dots\right]dt &= \left[\boldsymbol{I}t + \frac{\hat{\boldsymbol{A}}t^2}{2} + \frac{\hat{\boldsymbol{A}}^2t^3}{6} + \frac{\hat{\boldsymbol{A}}^3t^4}{24} + \dots\right]_0^{T_s} \\ = \boldsymbol{I}T_s + \frac{\hat{\boldsymbol{A}}T_s^2}{2!} + \frac{\hat{\boldsymbol{A}}^2T_s^3}{3!} + \frac{\hat{\boldsymbol{A}}^3T_s^4}{4!} + \dots = \sum_{i=0}^{\infty}\frac{\hat{\boldsymbol{A}}^iT_s^{i+1}}{(i+1)!}.\end{aligned} \tag{8.73}$$

In the MPC implementation (8.73) is approximately calculated as finite sum

$$\int_0^{T_s}\boldsymbol{\Phi}_{\hat{\boldsymbol{A}}}(t)dt \approx \sum_{i=0}^{n_d}\frac{\hat{\boldsymbol{A}}^iT_s^{i+1}}{(i+1)!} \tag{8.74}$$

with $n_d > 1$ chosen such that further increase does not significantly change the outcome of the finite sum in (8.74). In the present case $n_d = 4$ has proven successful.

8.5.2. MPC problem formulation

After the model (8.71) has been determined according to the current state information the actual optimization problem can be stated. The chosen formulation utilizes the control input variations

$$\hat{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} = \hat{\boldsymbol{u}}_{\boldsymbol{k}} - \hat{\boldsymbol{u}}_{\boldsymbol{k-1}} \tag{8.75}$$

as optimization variables. In order to predict the plant's behavior for varying inputs (8.75) and under the assumption of constant $\boldsymbol{z} := \boldsymbol{z}_{\boldsymbol{k}}$ for the duration of the prediction horizon the state equations are recursively evaluated for successive time indices which yields

$$
\begin{aligned}
\hat{\boldsymbol{x}}_{\boldsymbol{k+1}} &= \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{k-1}} + \boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k}} + \boldsymbol{z} \\
\hat{\boldsymbol{x}}_{\boldsymbol{k+2}} &= \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{k+1}} + \boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{k}} + \boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k+1}} + \boldsymbol{z} \\
&= \boldsymbol{A}^2\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{A}\boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{k-1}} + \boldsymbol{A}\boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k}} + \boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}\left(\hat{\boldsymbol{u}}_{\boldsymbol{k-1}} + \hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k}}\right) + \boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k+1}} + \boldsymbol{z} \\
&= \boldsymbol{A}^2\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \left(\boldsymbol{A}\boldsymbol{B} + \boldsymbol{B}\right)\hat{\boldsymbol{u}}_{\boldsymbol{k-1}} + \left(\boldsymbol{A}\boldsymbol{B} + \boldsymbol{B}\right)\hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k}} + \boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k+1}} + \left(\boldsymbol{A} + \boldsymbol{I}\right)\boldsymbol{z} \\
&\ \vdots \\
\hat{\boldsymbol{x}}_{\boldsymbol{k+n_c}} &= \boldsymbol{A}^{n_c}\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \left(\boldsymbol{A}^{n_c-1} + \ldots + \boldsymbol{I}\right)\boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{k-1}} + \sum_{j=1}^{n_c}\left(\boldsymbol{A}^{n_c-j} + \ldots + \boldsymbol{I}\right)\boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k+j-1}} \\
&\quad + \left(\boldsymbol{A}^{n_c-1} + \ldots + \boldsymbol{I}\right)\boldsymbol{z} \\
\hline
&\Rightarrow \text{control horizon ends: from now on } \hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k+i}} = \boldsymbol{0} \\
\hline
\hat{\boldsymbol{x}}_{\boldsymbol{k+n_c+1}} &= \boldsymbol{A}^{n_c+1}\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \left(\boldsymbol{A}^{n_c} + \ldots + \boldsymbol{I}\right)\boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{k-1}} + \sum_{j=1}^{n_c}\left(\boldsymbol{A}^{n_c+1-j} + \ldots + \boldsymbol{I}\right)\boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k+j-1}} \\
&\quad + \left(\boldsymbol{A}^{n_c} + \ldots + \boldsymbol{I}\right)\boldsymbol{z} \\
&\ \vdots \\
\hat{\boldsymbol{x}}_{\boldsymbol{k+n_p}} &= \boldsymbol{A}^{n_p}\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \left(\boldsymbol{A}^{n_p-1} + \ldots + \boldsymbol{I}\right)\boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{k-1}} + \sum_{j=1}^{n_c}\left(\boldsymbol{A}^{n_p-j} + \ldots + \boldsymbol{I}\right)\boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k+j-1}} \\
&\quad + \left(\boldsymbol{A}^{n_p-1} + \ldots + \boldsymbol{I}\right)\boldsymbol{z}.
\end{aligned}
\tag{8.76}
$$

Using these results the predicted output corresponding to the three angles is given by

$$
\begin{aligned}
\hat{\boldsymbol{y}}_{\boldsymbol{k+1}} &= \boldsymbol{C}\hat{\boldsymbol{x}}_{\boldsymbol{k+1}} \\
&\ \vdots \\
\hat{\boldsymbol{y}}_{\boldsymbol{k+n_p}} &= \boldsymbol{C}\hat{\boldsymbol{x}}_{\boldsymbol{k+n_p}}.
\end{aligned}
\tag{8.77}
$$

Input variations and outputs can be lumped together as

$$
\overline{\boldsymbol{y}}_{\boldsymbol{k+1}} = \begin{bmatrix} \hat{\boldsymbol{y}}_{\boldsymbol{k+1}} \\ \vdots \\ \hat{\boldsymbol{y}}_{\boldsymbol{k+n_p}} \end{bmatrix} \quad \text{(8.78a)} \qquad \text{and} \qquad \overline{\boldsymbol{u}}_{\boldsymbol{\Delta,k}} = \begin{bmatrix} \hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k}} \\ \vdots \\ \hat{\boldsymbol{u}}_{\boldsymbol{\Delta,k+n_c-1}} \end{bmatrix}. \quad \text{(8.78b)}
$$

Please note that (8.78a) does not contain $\hat{\boldsymbol{y}}_{\boldsymbol{k}}$ because (8.71) has no direct feedthrough. The evaluation of (8.78a) using (8.76), (8.77), and (8.78b) results in

$$
\overline{\boldsymbol{y}}_{\boldsymbol{k+1}} = \boldsymbol{F}\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \overline{\boldsymbol{G}}\boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{k-1}} + \overline{\boldsymbol{G}}\boldsymbol{z} + \boldsymbol{H}\overline{\boldsymbol{u}}_{\boldsymbol{\Delta,k}} \tag{8.79}
$$

with

$$
\boldsymbol{F} = \begin{bmatrix} \boldsymbol{C}\boldsymbol{A} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^{n_p} \end{bmatrix}, \tag{8.80}
$$

$$\overline{\boldsymbol{G}} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C}\left(\boldsymbol{A}+\boldsymbol{I}\right) \\ \vdots \\ \boldsymbol{C}\left(\boldsymbol{A}^{n_p-1}+\ldots+\boldsymbol{I}\right) \end{bmatrix}, \tag{8.81}$$

and

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{CB} & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{C}\left(\boldsymbol{A}+\boldsymbol{I}\right)\boldsymbol{B} & \boldsymbol{CB} & \ldots & \boldsymbol{0} \\ \vdots & & & \vdots \\ \boldsymbol{C}\left(\boldsymbol{A}^{n_c-1}+\ldots+\boldsymbol{I}\right)\boldsymbol{B} & \boldsymbol{C}\left(\boldsymbol{A}^{n_c-2}+\ldots+\boldsymbol{I}\right)\boldsymbol{B} & \ldots & \boldsymbol{CB} \\ \boldsymbol{C}\left(\boldsymbol{A}^{n_c}+\ldots+\boldsymbol{I}\right)\boldsymbol{B} & \boldsymbol{C}\left(\boldsymbol{A}^{n_c-1}+\ldots+\boldsymbol{I}\right)\boldsymbol{B} & \ldots & \boldsymbol{C}\left(\boldsymbol{A}+\boldsymbol{I}\right)\boldsymbol{B} \\ \vdots & & & \vdots \\ \boldsymbol{C}\left(\boldsymbol{A}^{n_p-1}+\ldots+\boldsymbol{I}\right)\boldsymbol{B} & \boldsymbol{C}\left(\boldsymbol{A}^{n_p-2}+\ldots+\boldsymbol{I}\right)\boldsymbol{B} & \ldots & \boldsymbol{C}\left(\boldsymbol{A}^{n_p-n_c}+\ldots+\boldsymbol{I}\right)\boldsymbol{B} \end{bmatrix}. \tag{8.82}$$

Consider that due to the Taylor series expansion around the current state $\boldsymbol{x_k}$ and input $\boldsymbol{u_{k-1}}$ it follows that the initial deviations are zero, i.e. $\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \boldsymbol{0}$ and also $\hat{\boldsymbol{u}}_{\boldsymbol{k-1}} = \boldsymbol{0}$. Hence, one can define the auxiliary vector

$$\overline{\boldsymbol{g}}_{\boldsymbol{k}} = \boldsymbol{F}\underbrace{\hat{\boldsymbol{x}}_{\boldsymbol{k}}}_{\boldsymbol{0}} + \overline{\boldsymbol{G}}\boldsymbol{B}\underbrace{\hat{\boldsymbol{u}}_{\boldsymbol{k-1}}}_{\boldsymbol{0}} + \overline{\boldsymbol{G}}\boldsymbol{z} \tag{8.83}$$

and from (8.79) and (8.83) one obtains

$$\overline{\boldsymbol{y}}_{\boldsymbol{k+1}} = \overline{\boldsymbol{g}}_{\boldsymbol{k}} + \boldsymbol{H}\overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} \tag{8.84}$$

Similar to (8.78a) and (8.78b) additional auxiliary vectors for reference and total input signals are introduced as

$$\overline{\boldsymbol{r}}_{\boldsymbol{k+1}} = \begin{bmatrix} \boldsymbol{r_{k+1}} \\ \vdots \\ \boldsymbol{r_{k+n_p}} \end{bmatrix} \quad (8.85a) \qquad \text{and} \qquad \overline{\boldsymbol{u}}_{\boldsymbol{k}} = \begin{bmatrix} \boldsymbol{u_k} \\ \vdots \\ \boldsymbol{u_{k+n_p-1}} \end{bmatrix} \quad (8.85b)$$

whereby (8.85b) can be expressed by means of the optimization variables via

$$\begin{aligned} \boldsymbol{u_k} &= \boldsymbol{u_{k-1}} + \hat{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} \\ &\vdots \\ \boldsymbol{u_{k+n_c-1}} &= \boldsymbol{u_{k-1}} + \hat{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} + \ldots + \hat{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k+n_c-1}} \\ &\Rightarrow \text{control horizon ends: from now on } \boldsymbol{u}_{\boldsymbol{\Delta},\boldsymbol{k+i}} = \boldsymbol{0} \\ &\vdots \\ \boldsymbol{u_{k+n_p-1}} &= \boldsymbol{u_{k-1}} + \hat{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} + \ldots + \hat{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k+n_c-1}}. \end{aligned} \tag{8.86}$$

Rewriting (8.86) in a more concise form yields

$$\overline{\boldsymbol{u}}_{\boldsymbol{k}} = \underbrace{\begin{bmatrix} \boldsymbol{I} \\ \vdots \\ \boldsymbol{I} \\ \vdots \\ \boldsymbol{I} \end{bmatrix}}_{\boldsymbol{L_1}\in\mathbb{R}^{(n_p m)\times m}} \boldsymbol{u_{k-1}} + \underbrace{\begin{bmatrix} \boldsymbol{I} & & \\ \vdots & \ddots & \\ \boldsymbol{I} & \ldots & \boldsymbol{I} \\ \vdots & & \vdots \\ \boldsymbol{I} & \ldots & \boldsymbol{I} \end{bmatrix}}_{\boldsymbol{L_2}\in\mathbb{R}^{(n_p m)\times(n_c m)}} \overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}}. \tag{8.87}$$

8.5.2.1. Constraints

The input constraints must be satisfied during the entire control horizon, i.e.

$$\overline{\boldsymbol{u}}_{\boldsymbol{min}} \leq \underbrace{\begin{bmatrix} \boldsymbol{I} \\ \vdots \\ \boldsymbol{I} \end{bmatrix}}_{\tilde{\boldsymbol{L}}_{\boldsymbol{1}} \in \mathbb{R}^{(n_c m) \times m}} \boldsymbol{u}_{\boldsymbol{k-1}} + \underbrace{\begin{bmatrix} \boldsymbol{I} & & 0 \\ \vdots & \ddots & \\ \boldsymbol{I} & \dots & \boldsymbol{I} \end{bmatrix}}_{\tilde{\boldsymbol{L}}_{\boldsymbol{2}} \in \mathbb{R}^{(n_c m) \times (n_c m)}} \overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} \leq \overline{\boldsymbol{u}}_{\boldsymbol{max}} \tag{8.88}$$

with

$$\overline{\boldsymbol{u}}_{\boldsymbol{min}} = \begin{bmatrix} \boldsymbol{u}_{\boldsymbol{min}} \\ \vdots \\ \boldsymbol{u}_{\boldsymbol{min}} \end{bmatrix} \in \mathbb{R}^{n_c m} \qquad (8.89a) \qquad \text{and} \qquad \overline{\boldsymbol{u}}_{\boldsymbol{max}} = \begin{bmatrix} \boldsymbol{u}_{\boldsymbol{max}} \\ \vdots \\ \boldsymbol{u}_{\boldsymbol{max}} \end{bmatrix} \in \mathbb{R}^{n_c m}. \qquad (8.89b)$$

The hover's attainable roll and pitch angles $\boldsymbol{y}_{\boldsymbol{12}} = [y_1 \;\; y_2]^T$ are limited between

$$\boldsymbol{y}_{\boldsymbol{12},\boldsymbol{min}} = \begin{bmatrix} \phi_{min} \\ \theta_{min} \end{bmatrix} \qquad (8.90a) \qquad \text{and} \qquad \boldsymbol{y}_{\boldsymbol{12},\boldsymbol{max}} = \begin{bmatrix} \phi_{max} \\ \theta_{max} \end{bmatrix} \qquad (8.90b)$$

due to the size of the stationary base. Therefore, these limitations are included as output constraints in the MPC problem. First, the predicted roll and pitch angles are given by

$$\tilde{\boldsymbol{y}}_{\boldsymbol{k+1}} = \tilde{\boldsymbol{F}} \underbrace{\hat{\boldsymbol{x}}_{\boldsymbol{k}}}_{\boldsymbol{0}} + \tilde{\boldsymbol{G}} \boldsymbol{B} \underbrace{\hat{\boldsymbol{u}}_{\boldsymbol{k-1}}}_{\boldsymbol{0}} + \tilde{\boldsymbol{G}} \boldsymbol{z} + \tilde{\boldsymbol{H}} \overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} \tag{8.91}$$

with matrices $\tilde{\boldsymbol{F}}$, $\tilde{\boldsymbol{G}}$, and $\tilde{\boldsymbol{H}}$ being constructed in the same way as (8.80), (8.81), and (8.82) except that every third row is skipped because they relate to the yaw angle. Since (8.91) describes the deviations from the current angles $\boldsymbol{y}_{\boldsymbol{12},\boldsymbol{k}}$ the bounds have to be adapted accordingly as

$$\overline{\boldsymbol{y}}_{\boldsymbol{min}} = \begin{bmatrix} \boldsymbol{y}_{\boldsymbol{12},\boldsymbol{min}} - \boldsymbol{y}_{\boldsymbol{12},\boldsymbol{k}} \\ \vdots \\ \boldsymbol{y}_{\boldsymbol{12},\boldsymbol{min}} - \boldsymbol{y}_{\boldsymbol{12},\boldsymbol{k}} \end{bmatrix} \qquad (8.92a) \qquad \text{and} \qquad \overline{\boldsymbol{y}}_{\boldsymbol{max}} = \begin{bmatrix} \boldsymbol{y}_{\boldsymbol{12},\boldsymbol{max}} - \boldsymbol{y}_{\boldsymbol{12},\boldsymbol{k}} \\ \vdots \\ \boldsymbol{y}_{\boldsymbol{12},\boldsymbol{max}} - \boldsymbol{y}_{\boldsymbol{12},\boldsymbol{k}} \end{bmatrix} \qquad (8.92b)$$

with $\overline{\boldsymbol{y}}_{\boldsymbol{min}}, \overline{\boldsymbol{y}}_{\boldsymbol{max}} \in \mathbb{R}^{2n_p}$ leading to the inequality

$$\overline{\boldsymbol{y}}_{\boldsymbol{min}} \leq \tilde{\boldsymbol{y}}_{\boldsymbol{k+1}} \leq \overline{\boldsymbol{y}}_{\boldsymbol{max}}. \tag{8.93}$$

In a final step (8.88) and (8.93) are combined to a single expression

$$\boldsymbol{A}_{\boldsymbol{iq}} \overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} \leq \boldsymbol{b}_{\boldsymbol{iq}} \tag{8.94}$$

where

$$\boldsymbol{A}_{\boldsymbol{iq}} = \begin{bmatrix} -\tilde{\boldsymbol{L}}_{\boldsymbol{2}} \\ \tilde{\boldsymbol{L}}_{\boldsymbol{2}} \\ -\tilde{\boldsymbol{H}} \\ \tilde{\boldsymbol{H}} \end{bmatrix} \qquad (8.95a) \qquad \text{and} \qquad \boldsymbol{b}_{\boldsymbol{iq}} = \begin{bmatrix} \tilde{\boldsymbol{L}}_{\boldsymbol{1}} \boldsymbol{u}_{\boldsymbol{k-1}} - \overline{\boldsymbol{u}}_{\boldsymbol{min}} \\ \overline{\boldsymbol{u}}_{\boldsymbol{max}} - \tilde{\boldsymbol{L}}_{\boldsymbol{1}} \boldsymbol{u}_{\boldsymbol{k-1}} \\ \tilde{\boldsymbol{G}} \boldsymbol{z} - \overline{\boldsymbol{y}}_{\boldsymbol{min}} \\ \overline{\boldsymbol{y}}_{\boldsymbol{max}} - \tilde{\boldsymbol{G}} \boldsymbol{z} \end{bmatrix}. \qquad (8.95b)$$

8.5.2.2. Cost function

The cost function is the place where the actual controller objectives are specified. Apart form following the reference angles the hover should also generate a desired thrust F_{des}. The actual thrust F caused by the rotors is given by (8.31). Combining the resulting thrusts over the control horizon yields

$$\boldsymbol{T} = \underbrace{\begin{bmatrix} k_f & k_f & k_f & k_f & & & & \\ & & & & \ddots & & & \\ & & & & k_f & k_f & k_f & k_f \end{bmatrix}}_{\boldsymbol{O}\in\mathbb{R}^{n_c\times(n_c m)}} \overline{\boldsymbol{u}}_{\boldsymbol{k},\boldsymbol{n_c}} \tag{8.96}$$

with

$$\overline{\boldsymbol{u}}_{\boldsymbol{k},\boldsymbol{n_c}} = \tilde{\boldsymbol{L}}_{\boldsymbol{1}}\boldsymbol{u}_{\boldsymbol{k-1}} + \tilde{\boldsymbol{L}}_{\boldsymbol{2}}\overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}}. \tag{8.97}$$

Using $\boldsymbol{T_{des}} = [F_{des}\dots F_{des}]^T$ deviations from the desired thrust are penalized via the quadratic term

$$J_T = \gamma\left(\boldsymbol{T_{des}} - \boldsymbol{T}\right)^T\left(\boldsymbol{T_{des}} - \boldsymbol{T}\right) \tag{8.98}$$

with weighting factor $\gamma > 0$. The evaluation of (8.98) having regard to (8.96) and (8.97) results in

$$J_T = \gamma\overline{\boldsymbol{u}}^T_{\boldsymbol{\Delta},\boldsymbol{k}}\tilde{\boldsymbol{L}}^T_{\boldsymbol{2}}\boldsymbol{O}^T\boldsymbol{O}\tilde{\boldsymbol{L}}_{\boldsymbol{2}}\overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} + 2\gamma\overline{\boldsymbol{u}}^T_{\boldsymbol{\Delta},\boldsymbol{k}}\left(\tilde{\boldsymbol{L}}^T_{\boldsymbol{2}}\boldsymbol{O}^T\boldsymbol{O}\tilde{\boldsymbol{L}}_{\boldsymbol{1}}\boldsymbol{u}_{\boldsymbol{k-1}} - \tilde{\boldsymbol{L}}^T_{\boldsymbol{2}}\boldsymbol{O}^T\boldsymbol{T_{des}}\right) + c_{J_T} \tag{8.99}$$

where $c_{J_T} > 0$ is a constant scalar offset which can be neglected consequently. Before the remaining parts of the cost function are stated another auxiliary vector is introduced as

$$\overline{\boldsymbol{e}}_{\boldsymbol{k}} = \overline{\boldsymbol{g}}_{\boldsymbol{k}} - \overline{\boldsymbol{r}}_{\boldsymbol{k+1}}. \tag{8.100}$$

The overall quadratic cost function which is minimized online by MPC reads as

$$J(\overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}}) = \left(\overline{\boldsymbol{y}}_{\boldsymbol{k+1}} - \overline{\boldsymbol{r}}_{\boldsymbol{k+1}}\right)^T\boldsymbol{Q}\left(\overline{\boldsymbol{y}}_{\boldsymbol{k+1}} - \overline{\boldsymbol{r}}_{\boldsymbol{k+1}}\right) + \overline{\boldsymbol{u}}^T_{\boldsymbol{k}}\boldsymbol{R_u}\overline{\boldsymbol{u}}_{\boldsymbol{k}} + \overline{\boldsymbol{u}}^T_{\boldsymbol{\Delta},\boldsymbol{k}}\boldsymbol{R_\Delta}\overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} + J_T \tag{8.101}$$

where $\boldsymbol{Q} \in \mathbb{R}^{(n_p p)\times(n_p p)}$ is used for weighting the tracking error, $\boldsymbol{R_u} \in \mathbb{R}^{(n_p m)\times(n_p m)}$ penalizes the energy consumption of the actuators, and $\boldsymbol{R_\Delta} \in \mathbb{R}^{(n_c m)\times(n_c m)}$ is used to reduce the rate of change of the actuation signals. The chosen cost function formulation (8.101) also supports weighting of individual actuators by means of $\boldsymbol{R_u}$ which could be useful in case of detected failures. $\boldsymbol{R_\Delta}$ not only enables the smoothing of the actuation signals and but also the indirect consideration of actuator dynamics. Using (8.84), (8.99), and (8.100) one can rewrite (8.101) as

$$J(\overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}}) = \overline{\boldsymbol{u}}^T_{\boldsymbol{\Delta},\boldsymbol{k}}\boldsymbol{W_\Delta}\overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} + 2\overline{\boldsymbol{u}}^T_{\boldsymbol{\Delta},\boldsymbol{k}}\boldsymbol{c_\Delta} + c_J \tag{8.102}$$

with constant $c_J > 0$ and

$$\boldsymbol{W_\Delta} = \boldsymbol{H}^T\boldsymbol{Q}\boldsymbol{H} + \boldsymbol{L}^T_{\boldsymbol{2}}\boldsymbol{R_u}\boldsymbol{L_2} + \boldsymbol{R_\Delta} + \gamma\tilde{\boldsymbol{L}}^T_{\boldsymbol{2}}\boldsymbol{O}^T\boldsymbol{O}\tilde{\boldsymbol{L}}_{\boldsymbol{2}} \tag{8.103a}$$

$$\boldsymbol{c_\Delta} = \boldsymbol{H}^T\boldsymbol{Q}\overline{\boldsymbol{e}}_{\boldsymbol{k}} + \boldsymbol{L}^T_{\boldsymbol{2}}\boldsymbol{R_u}\boldsymbol{L_1}\boldsymbol{u}_{\boldsymbol{k-1}} + \gamma\left(\tilde{\boldsymbol{L}}^T_{\boldsymbol{2}}\boldsymbol{O}^T\boldsymbol{O}\tilde{\boldsymbol{L}}_{\boldsymbol{1}}\boldsymbol{u}_{\boldsymbol{k-1}} - \tilde{\boldsymbol{L}}^T_{\boldsymbol{2}}\boldsymbol{O}^T\boldsymbol{T_{des}}\right). \tag{8.103b}$$

8.5.3. Implementation issues

The proposed MPC control strategy is realized in Matlab/Simulink which uses automatic code generation to create a binary for execution with Quanser QUARC. The MPC controller is split into two parts. The first one deals with the application specific computations like the determination of the approximated model (8.71) from current measurement data and the evaluation of the expressions from Section 8.5.2. This results in the formulation of the general quadratic program

$$\begin{aligned} &\min_{\overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}}} \left(\overline{\boldsymbol{u}}^T_{\boldsymbol{\Delta},\boldsymbol{k}} \boldsymbol{W}_{\boldsymbol{\Delta}} \overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} + 2\overline{\boldsymbol{u}}^T_{\boldsymbol{\Delta},\boldsymbol{k}} \boldsymbol{c}_{\boldsymbol{\Delta}}\right) \\ &\text{subject to } \ \boldsymbol{A}_{\boldsymbol{iq}} \overline{\boldsymbol{u}}_{\boldsymbol{\Delta},\boldsymbol{k}} \leq \boldsymbol{b}_{\boldsymbol{iq}} \end{aligned} \tag{8.104}$$

which is passed to the subsequent component: the optimization algorithm. C-implementations of QPO and EPA are used to solve the quadratic optimization problems in real-time. Due to the significantly larger problem size compared to the quadratic programs for CA the sample time is increased. Besides the prediction and the control horizon the implementation additionally provides the output constraint horizon n_i which determines the amount of time for which the constraints on ϕ and θ have to be fulfilled. There are two main reasons for introducing this parameter:

1. The model (8.71) used for the predictions is only valid in a small region around the current state. Nevertheless it turned out during the experiments that prediction times below two seconds lead to unsatisfactory results. Within two seconds the hover's state can change dramatically and considering the output constraints for the entire prediction horizon promotes infeasible optimization problems.

2. The computation time is reduced, especially in case of QPO.

The main MPC configuration used for the experiments features a prediction time of $2.8\,s$. All related parameters are shown in Table 8.7. The weighting matrices in (8.101) are chosen

parameter	value	parameter	value
T_s	$80\,ms$	$\boldsymbol{q_{diag}}$	$[1300\ \ 1500\ \ 1000]^T$
n_p	35	$\boldsymbol{r_{u,diag}}$	$[1\ \ 1\ \ 1\ \ 1]^T$
n_c	35	$\boldsymbol{r_{\Delta,diag}}$	$[0.01\ \ 0.01\ \ 0.01\ \ 0.01]^T$
n_i	20	$\boldsymbol{y_{12,min}}$	$[-0.67\ \ -0.67]^T$
γ	1	$\boldsymbol{y_{12,max}}$	$[0.67\ 0.67]^T$

Table 8.7.: MPC configuration 1. The angle constraints are specified in radians and correspond to about 38°. In this case (8.104) has 140 optimization variables and 320 inequality constraints.

as diagonal matrices with uniform parameters over the control horizon. Focusing on the minimization of the output error at the beginning of the prediction by means of decreasing diagonal elements in $\boldsymbol{Q}$ had no positive impact on the results. Non-diagonal positive-definite error weighting matrices did also not lead to better performance. The angular velocities are computed as difference quotients of the encoder signals without additional filtering.

8.5.4. MPC results

The performance of the MPC strategy is evaluated with the same reference signals (square and sinusoidal waves) as the CA-based solution in Section 8.3.7. As already mentioned they are deliberately chosen such that the actuators operate at their limits and perfect tracking is not possible. Figure 8.30 shows the results for QPO and EPA in case of square-wave angle references. As expected the behavior is very similar. The computed actuator signals are depicted in Figure 8.31. On very few occasions the actuator commands show significant changes although the reference signals are constant at that time. In these cases the solver could not find a feasible solution. Nevertheless the computed signals are sent to the actuators. This very simple strategy works as long as not too many infeasible problems occur. Other approaches of handling such situations like using the same commands as in the previous step or using future commands from the last successful optimization did not improve the results or, quite the opposite, deteriorated the behavior. A clear difference between the previous CA-based results and those from MPC is the instant of time where the controller initiates the rotation towards the reference angles. Since the MPC "knows" their future values it starts the movement before the reference signals have changed. This is visible for example at $t \approx 11$ s in the ψ-graph of Figure 8.30. Figure 8.32 illustrates the results for sinusoidal reference angles. Differences between QPO and EPA are more recognizable in this case, e.g. EPA is slightly better in following the ψ reference signal. As a consequence the corresponding actuator commands shown in Figure 8.33 are also more diverse. The number of iterations as well as the execution times of QPO and EPA[4] are presented in Figures 8.34 - 8.37. One can see that EPA requires significantly less time which facilitates testing of MPC configurations with lower sampling time. Finally, the same experiments are carried out in simulations

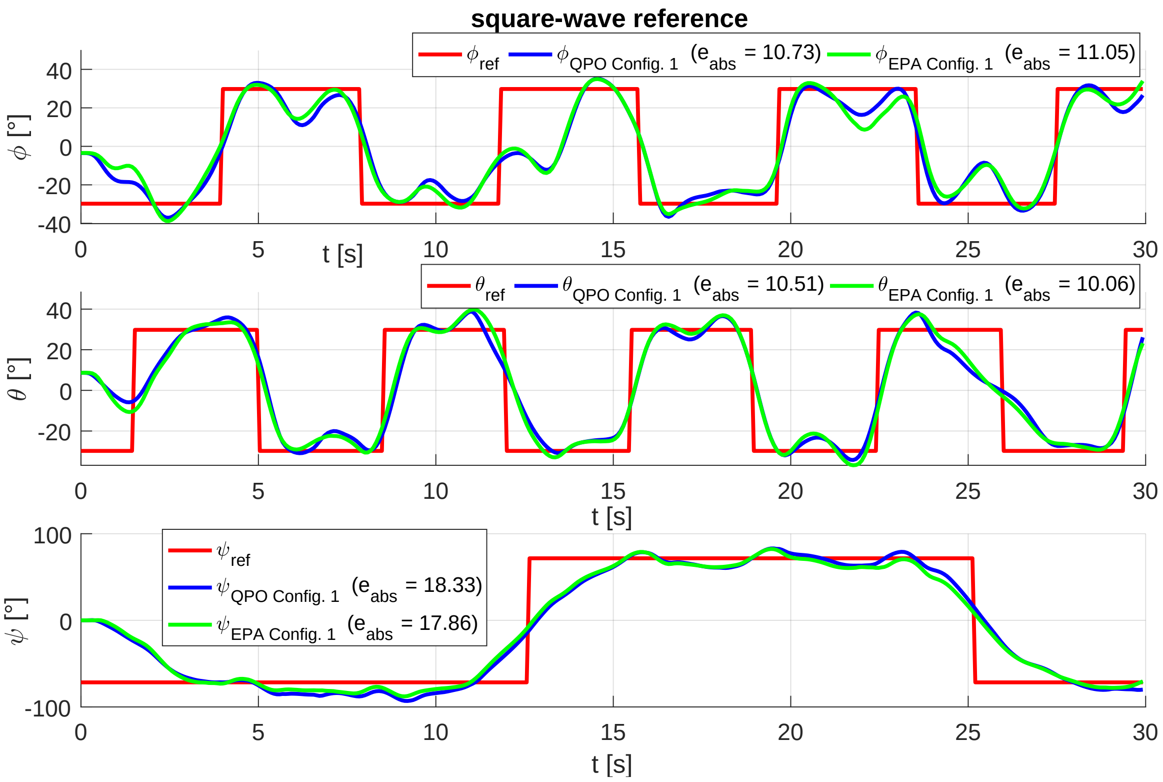

Figure 8.30.: Comparison of MPC via QPO and EPA for square-wave reference signals. These results are obtained from the real experimental setup.

[4]These are execution times for solving (8.104) only. An additional overhead for obtaining the model for the current state, the construction of the MPC-related matrices, and I/O must be added to get the total required CPU time.

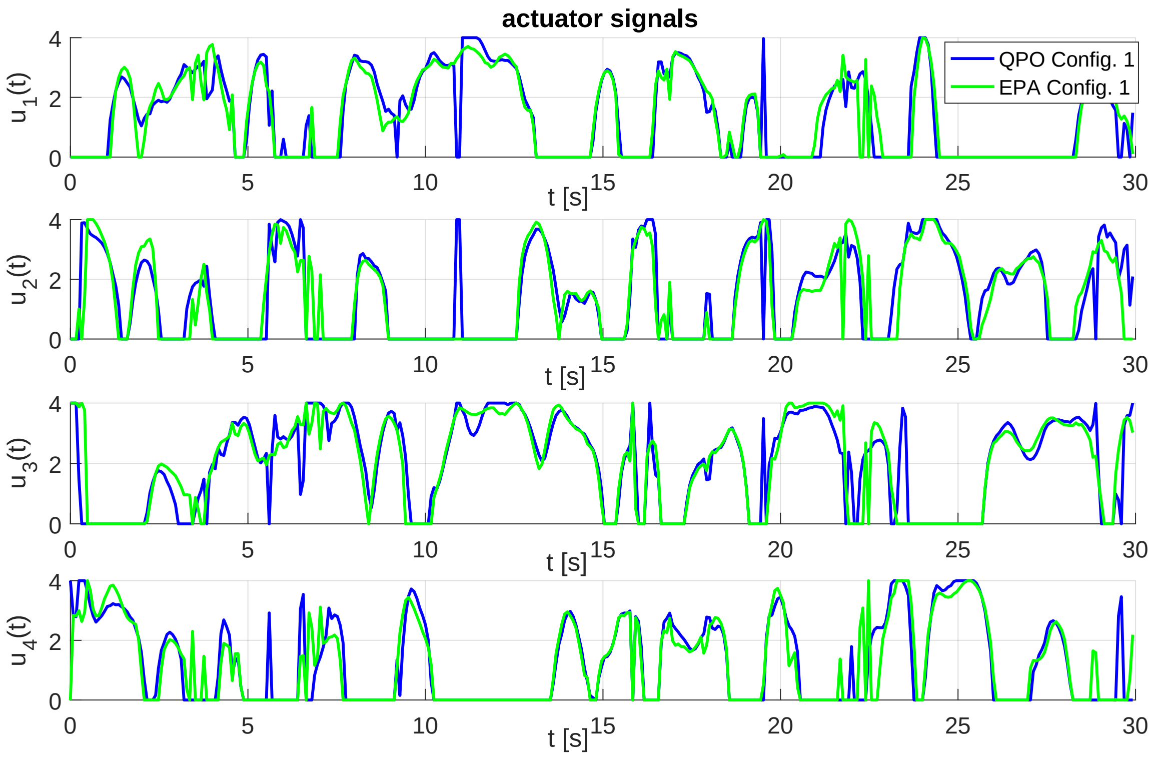

Figure 8.31.: Actuator commands from QPO and EPA related to Figure 8.30.

with the full nonlinear plant model (8.23). Some of the results can be seen in Figures 8.38 and 8.39. The behavior is very similar to the experiments on the real laboratory setup. This is an indication for a good coincidence between model and real plant.

8.5.4.1. Influence of sample time

This section shows how increasing and decreasing the sample time affects the MPC performance. In order to keep the prediction time at $2.8\,s$ the horizons have to be adjusted properly. Two additional MPC configurations are given in Tables 8.8 and 8.9. Due to the decreased sample time of configuration 2 the optimization problem dimensions are increased. On the experimental setup this configuration is only tested with EPA because QPO's execution time is too high. Figures 8.40 and 8.41 contain the resulting angles of the experiments. If compared with the previous results with $T_s = 80\,ms$ one recognizes no benefit from faster sampling. The hover's rotational dynamics are simply not fast enough to take advantage from higher sampling rates. Only for the purpose of external disturbance rejection a faster controller might be favorable. In the present case the illustrated results in Figures 8.40 and 8.41 are even slightly worse than those from the slower MPC configuration 1. The origin of this phenomenon lies in the plant model (8.71) which is computed for the current state. For certain states the resulting matrix $\boldsymbol{A}$ exhibits eigenvalues greater one, i.e. the system is unstable. Matrix powers $\boldsymbol{A}^j$ for $j = 1, \ldots, n_p$ are part of several computation steps of the MPC formulation. In case of unstable systems these matrix powers reach vast magnitudes and increasing n_p tremendously boosts those results. Therefore, the solver faces numerical problems and consequently the number of infeasible optimization problems rises which deteriorates the performance. EPA's execution times for configuration 2 can be seen in Figures 8.42 and 8.43. Except from one outlier in Figure 8.43 the times lie below the sample time.

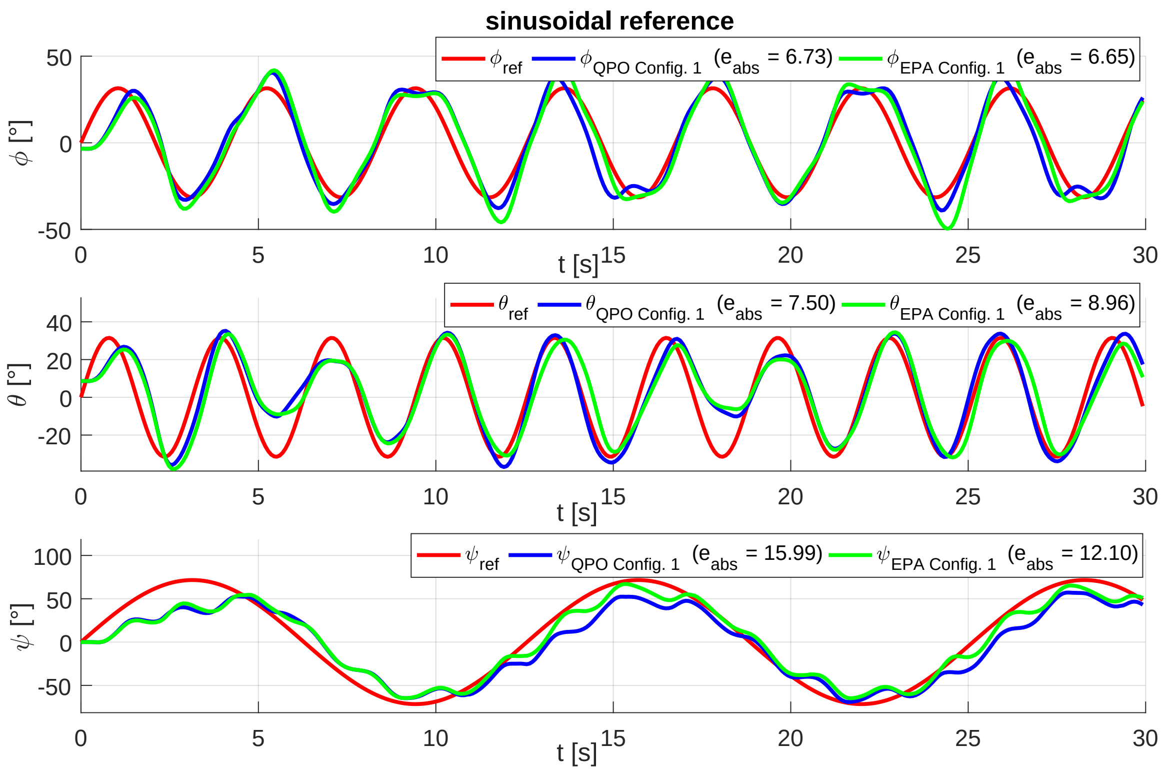

Figure 8.32.: Comparison of MPC via QPO and EPA for sinusoidal reference signals. These results are obtained from the real experimental setup.

Configuration 3 uses a higher sample time resulting in smaller optimization problem dimensions for the same prediction time. The results are illustrated in Figures 8.44 and 8.45. Although the sample time is almost doubled compared to configuration 1 the results are just a little worse especially in case of EPA. The execution times shown in Figures 8.46 and 8.47 are smaller than in the other cases as expected[5]. Altogether MPC configuration 1 achieved the best results.

parameter	value	parameter	value
T_s	$60\,ms$	$\boldsymbol{q_{diag}}$	$[1300\ \ 1300\ \ 1000]^T$
n_p	47	$\boldsymbol{r_{u,diag}}$	$[1\ \ 1\ \ 1\ \ 1]^T$
n_c	47	$\boldsymbol{r_{\Delta,diag}}$	$[0.01\ \ 0.01\ \ 0.01\ \ 0.01]^T$
n_i	27	$\boldsymbol{y_{12,min}}$	$[-0.67\ \ -0.67]^T$
γ	1	$\boldsymbol{y_{12,max}}$	$[0.67\ 0.67]^T$

Table 8.8.: MPC configuration 2. In this case (8.104) has 188 optimization variables and 430 inequality constraints.

8.5.4.2. Influence of prediction time

As already mentioned prediction times below two seconds reduce the MPC performance especially for yaw angle tracking. This effect is demonstrated in simulations using MPC configuration 1 (except for the horizons) whose results are depicted in Figures 8.48 and 8.49. In

[5]For square-wave references the execution times are very similar.

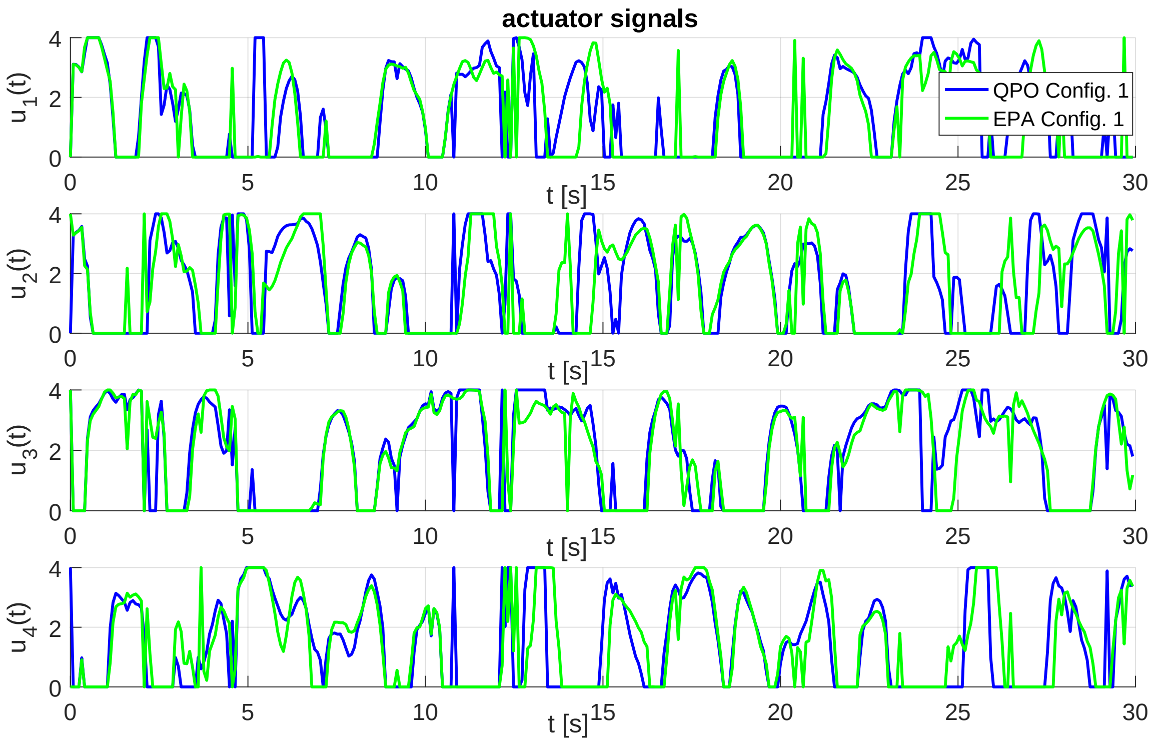

Figure 8.33.: Actuator commands from QPO and EPA related to Figure 8.32.

parameter	value	parameter	value
T_s	$150\,ms$	$\boldsymbol{q_{diag}}$	$[1300 \;\; 1500 \;\; 1000]^T$
n_p	20	$\boldsymbol{r_{u,diag}}$	$[1 \;\; 1 \;\; 1 \;\; 1]^T$
n_c	20	$\boldsymbol{r_{\Delta,diag}}$	$[0.01 \;\; 0.01 \;\; 0.01 \;\; 0.01]^T$
n_i	20	$\boldsymbol{y_{12,min}}$	$[-0.67 \;\; -0.67]^T$
γ	1	$\boldsymbol{y_{12,max}}$	$[0.67 \; 0.67]^T$

Table 8.9.: MPC configuration 3. In this case (8.104) has 80 optimization variables and 200 inequality constraints.

the first scenario the horizons are reduced to 15 which yields a prediction time of $1.2\,s$. As a consequence θ-tracking works better at the cost of diminished performance for ψ. Conversely, increasing the prediction time promotes better results at ψ-tracking but the results for θ are worse.

8.5.4.3. Remark on the choice of the model

In Section 8.5.1 an approximation of the nonlinear small-angle model (8.25) for the current state is proposed. An obvious question that arises is whether the MPC results could be improved if the full nonlinear model (8.23) is used instead. Corresponding experiments have been conducted and reveal that the answer to this question is no. Especially for longer prediction times the results are much worse than with the small angle model. In case of large values of ϕ and/or θ the behavior of the hover and particularly the effect of the individual rotors on the state are very different from the horizontal attitude ($\phi = \theta = 0$). The approximation of (8.23) yields a model that is valid for small deviations from the current state. Large deviations

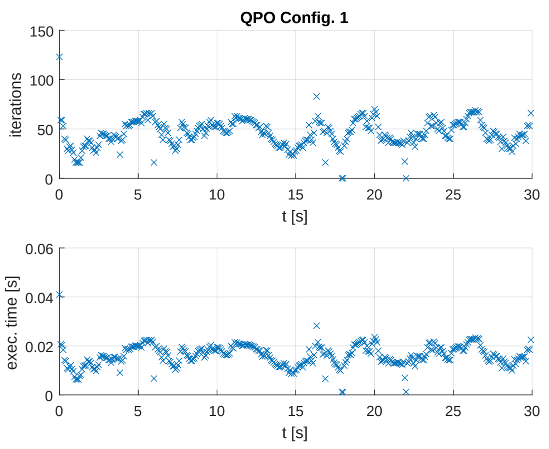

Figure 8.34.: Iteration number and execution times of QPO corresponding to Figure 8.30.

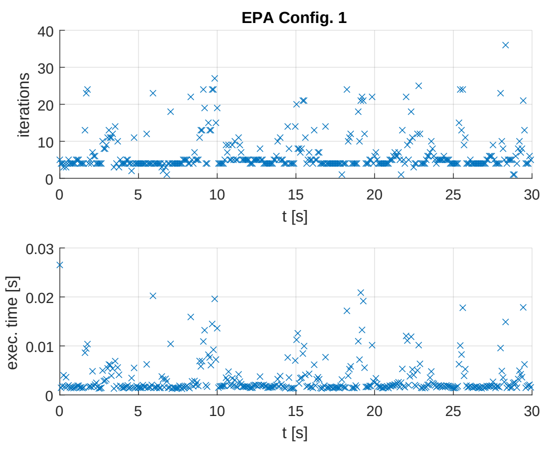

Figure 8.35.: Iteration number and execution times of EPA corresponding to Figure 8.30.

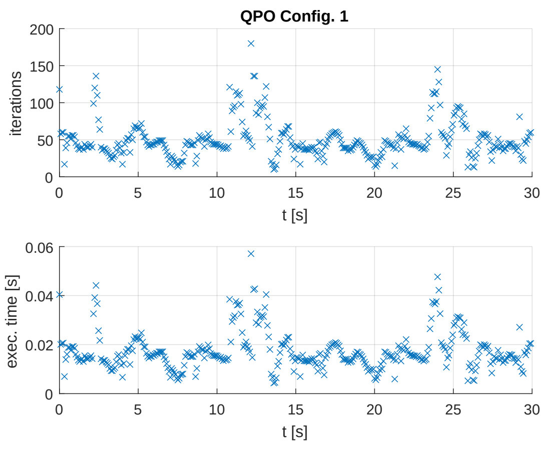

Figure 8.36.: Iteration number and execution times of QPO corresponding to Figure 8.32.

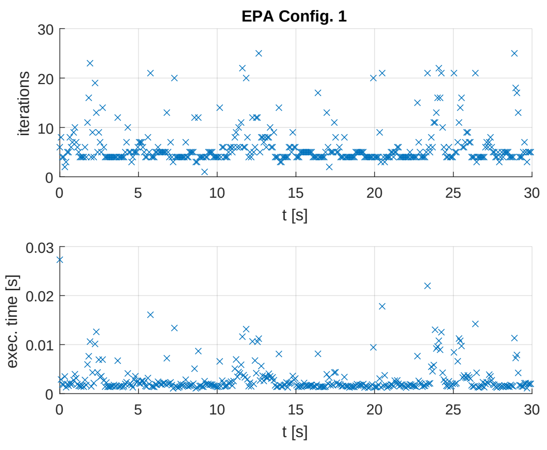

Figure 8.37.: Iteration number and execution times of EPA corresponding to Figure 8.32.

from the current state result in larger model errors. Since the reference signals are offset-free periodic signals the approximation of the small-angle model can be anticipated to describe the plant dynamics better ”on average”.

8.5.4.4. Unpredictable reference signal

Up to now, the reference angles are supposed to be known for the entire prediction horizon. However, if this is not the case MPC can still be used. One possibility to deal with this situation is to assume constant references during the prediction time. A simulation result with MPC configuration 1 referring to this scenario is presented in Figure 8.50. One notices the massive performance decrease.

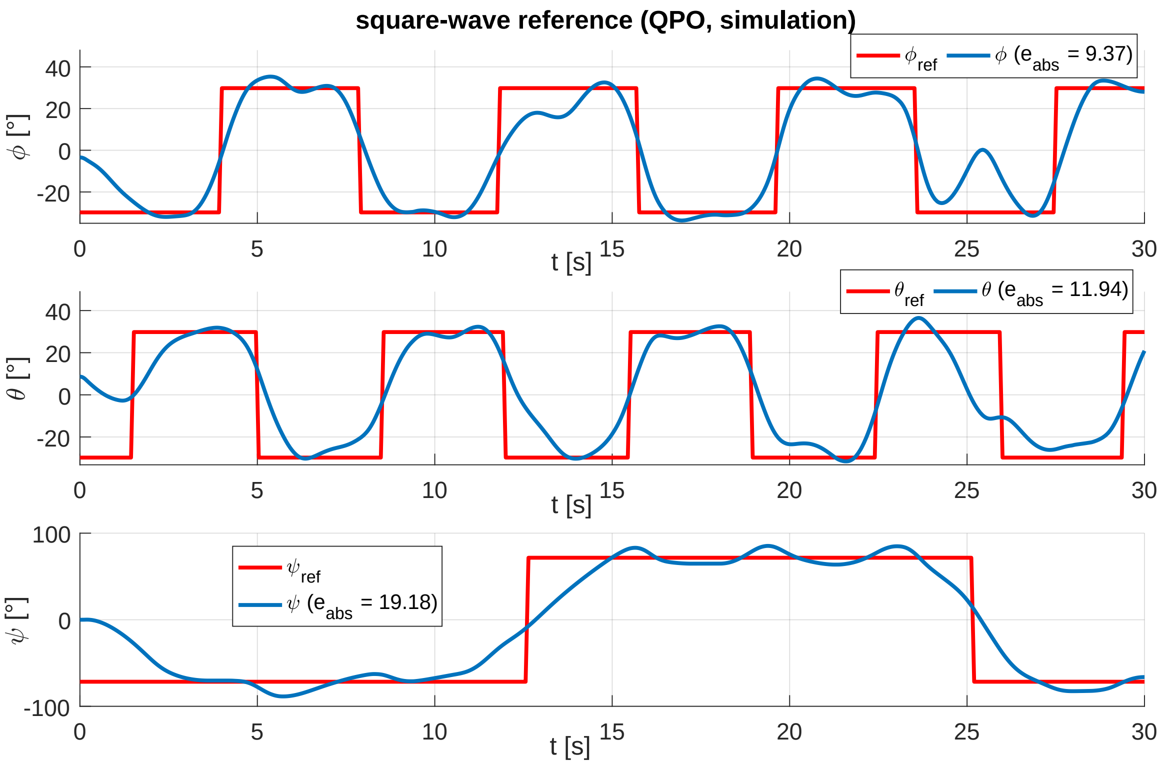

Figure 8.38.: Simulation outcome of MPC via QPO for square-wave reference signals. Comparing with the experimental results in Figure 8.30 exposes very similar behavior. The same is true for EPA which is why its results are omitted here.

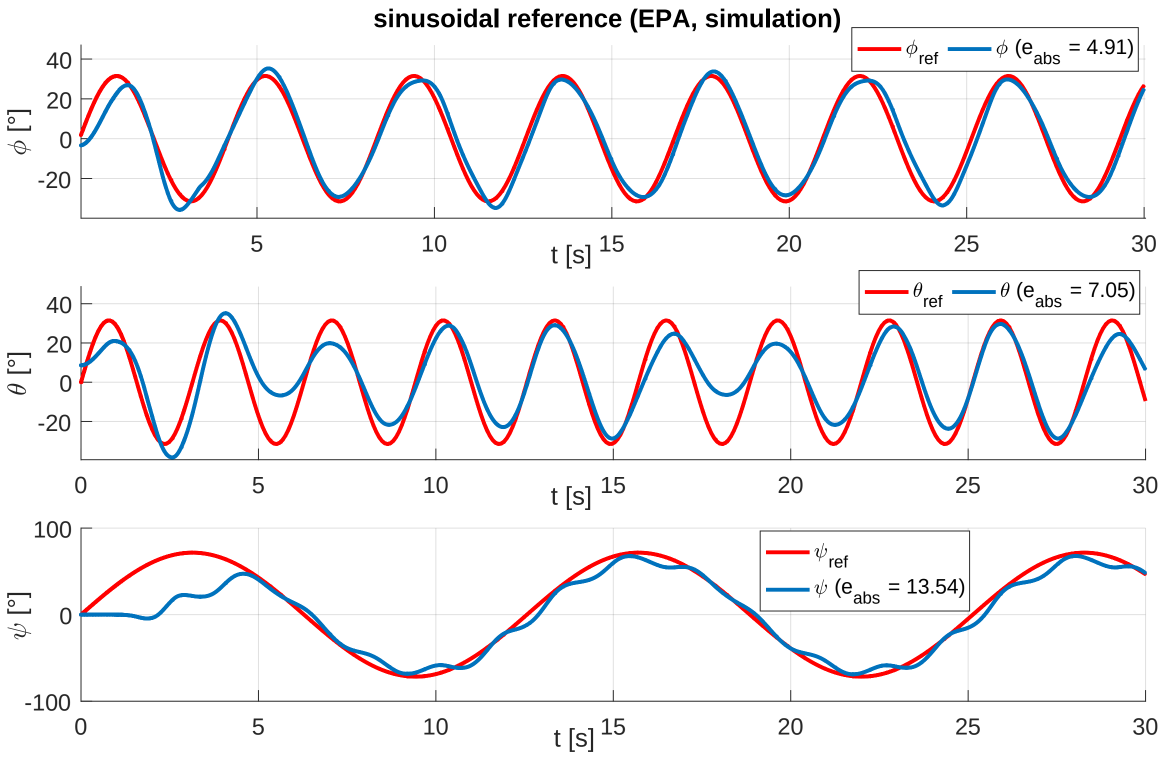

Figure 8.39.: Simulation results of MPC via EPA for sinusoidal reference signals. Compare with experimental results in Figure 8.32. Using QPO causes almost identical behavior and therefore it is omitted here.

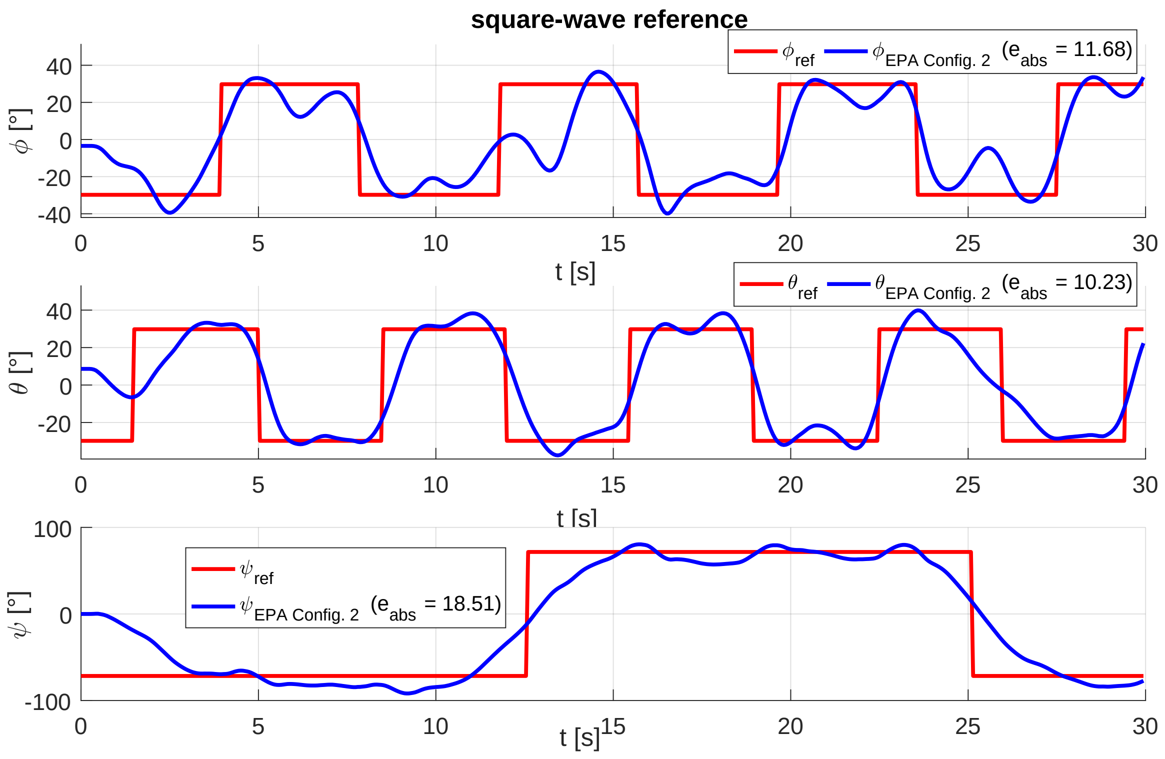

Figure 8.40.: These results of MPC via EPA for square-wave reference signals with configuration 2 (60 ms sample time) are obtained from the real experimental setup.

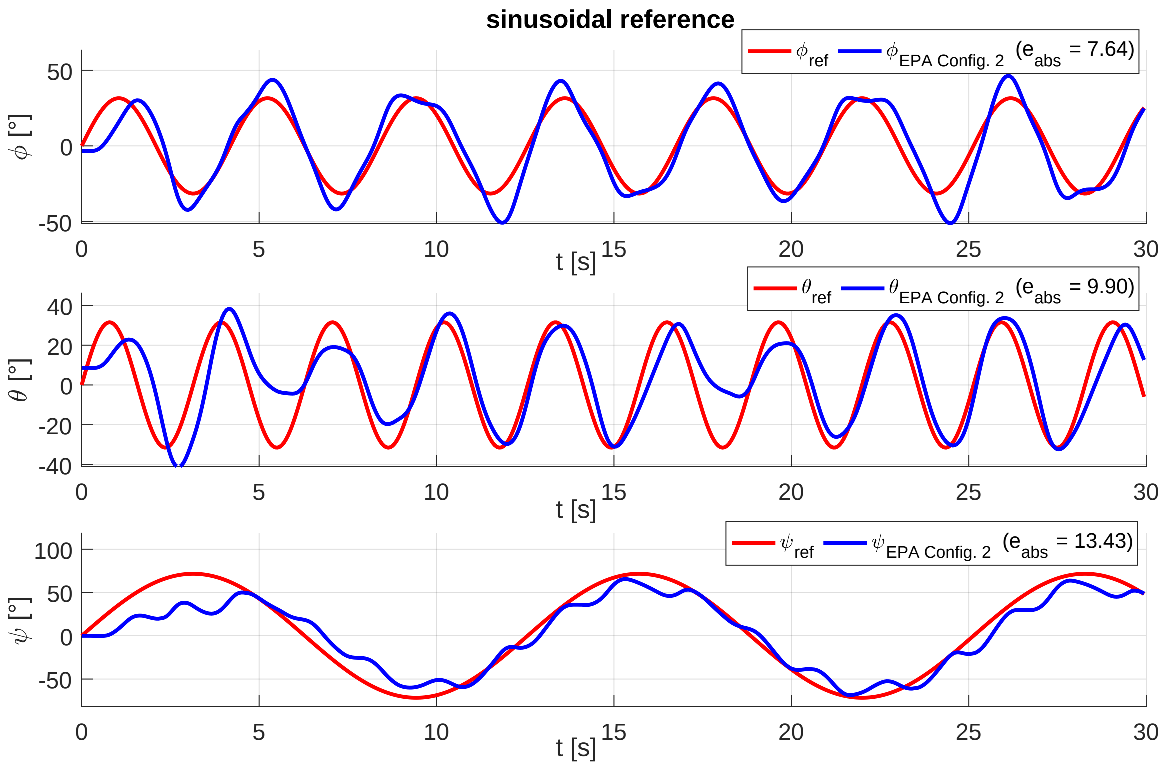

Figure 8.41.: These results of MPC via EPA for sinusoidal reference signals with configuration 2 (60 ms sample time) are obtained from the real experimental setup.

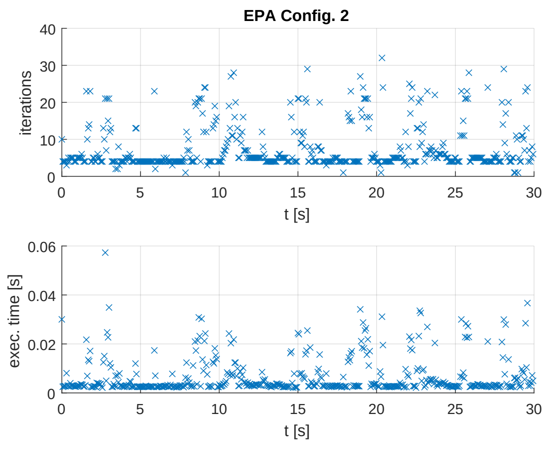

Figure 8.42.: Iteration number and execution times of EPA corresponding to Figure 8.40.

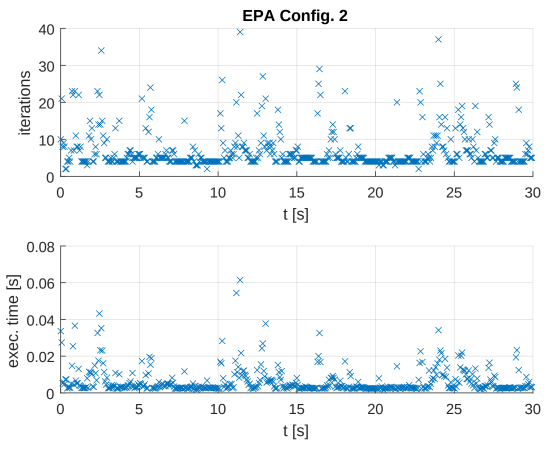

Figure 8.43.: Iteration number and execution times of EPA corresponding to Figure 8.41.

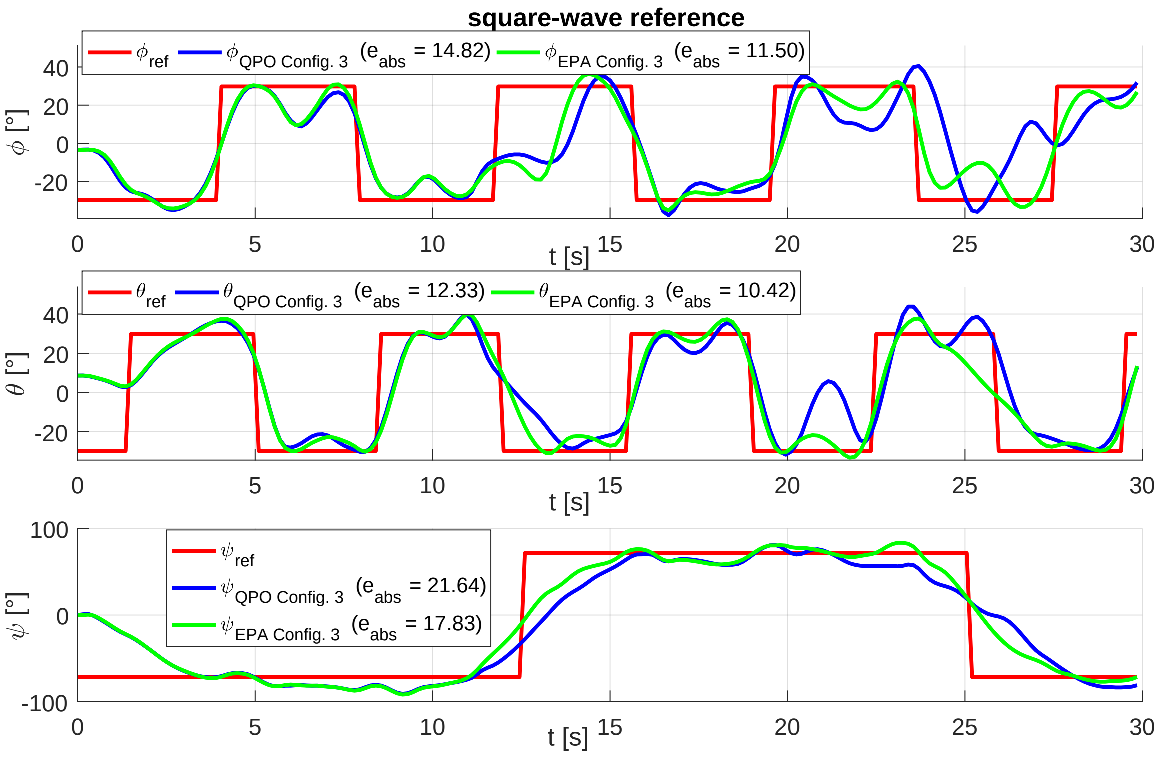

Figure 8.44.: These results of MPC for square-wave reference signals with configuration 3 (150 ms sample time) are obtained from the real experimental setup.

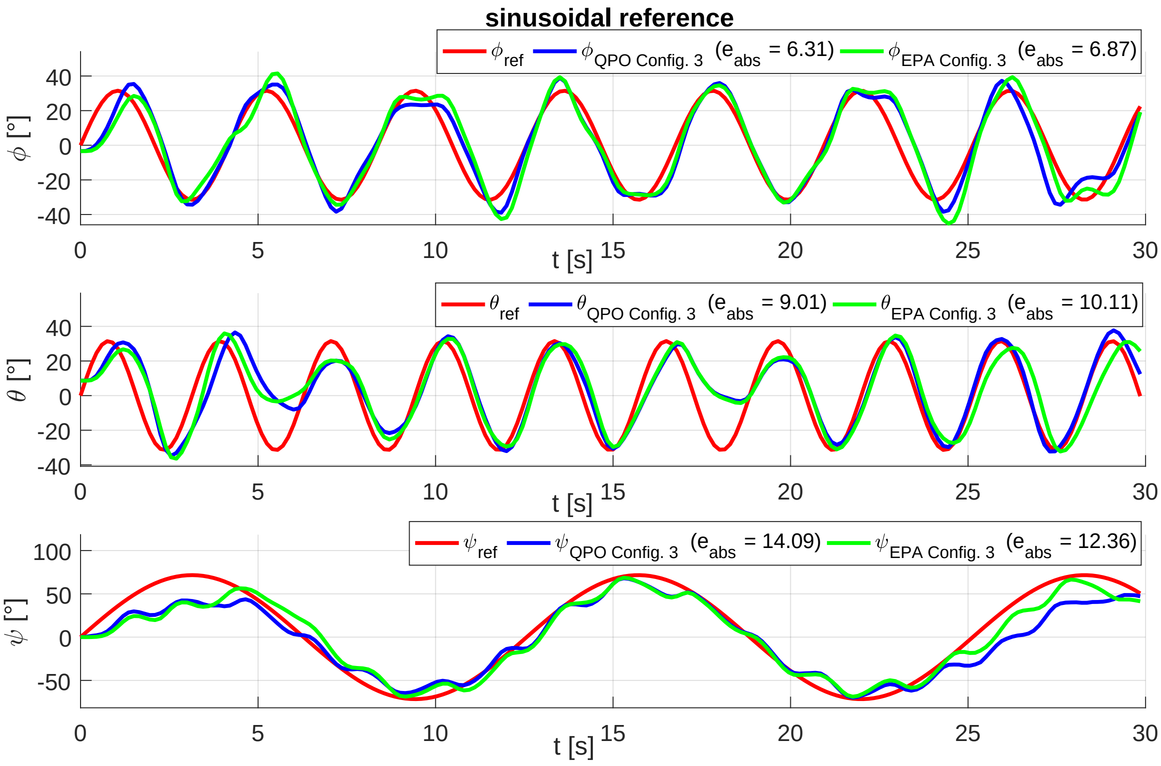

Figure 8.45.: These results of MPC for sinusoidal reference signals with configuration 3 (150 ms sample time) are obtained from the real experimental setup.

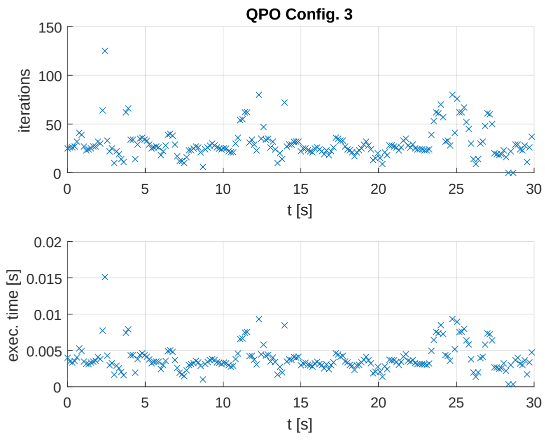

Figure 8.46.: Iteration number and execution times of EPA corresponding to Figure 8.45.

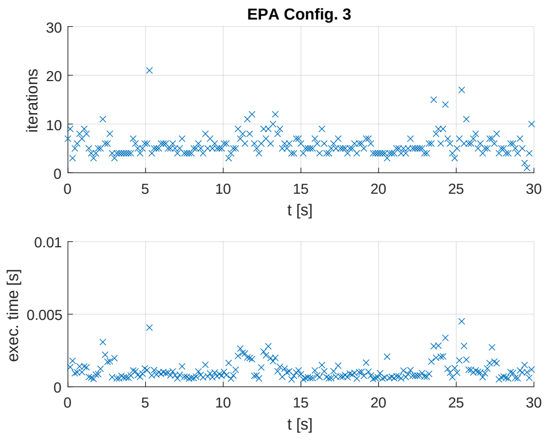

Figure 8.47.: Iteration number and execution times of EPA corresponding to Figure 8.45.

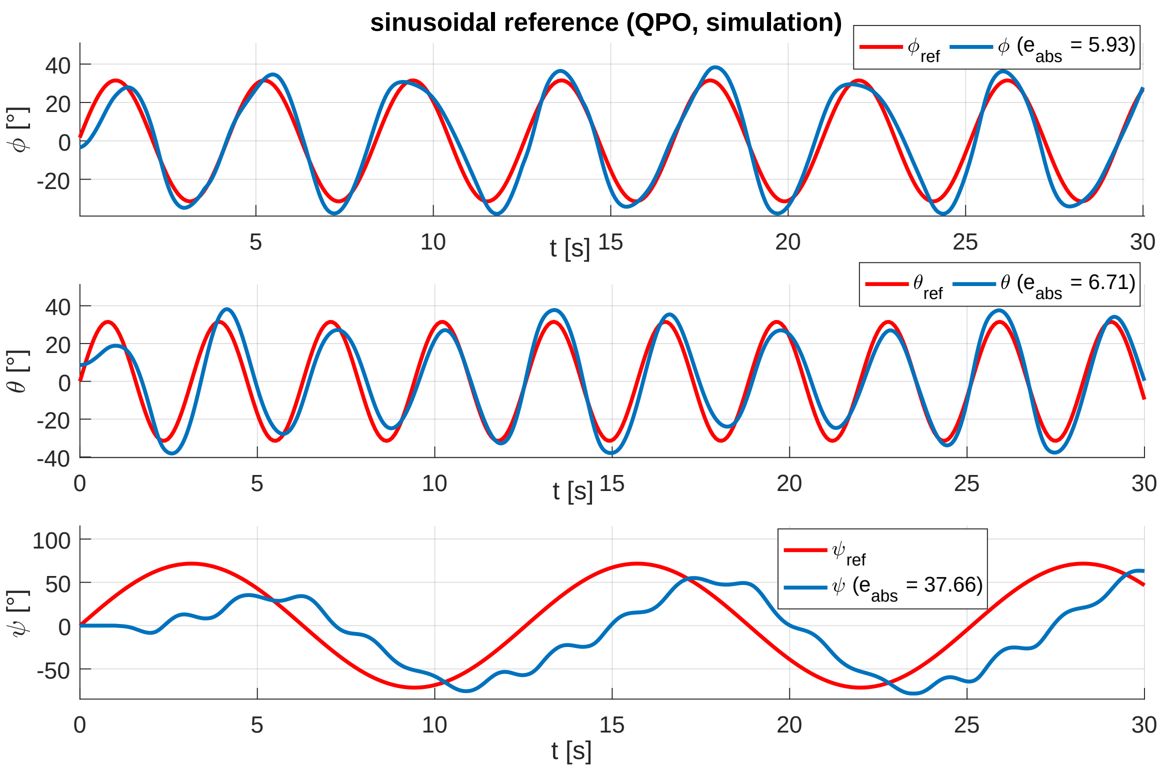

Figure 8.48.: Simulation results of MPC via QPO for sinusoidal reference signals with $n_p = n_c = n_i = 15$ (prediction time: $1.2\,s$). Especially tracking of ψ is worse than in case of the original configuration (see Figure 8.39).

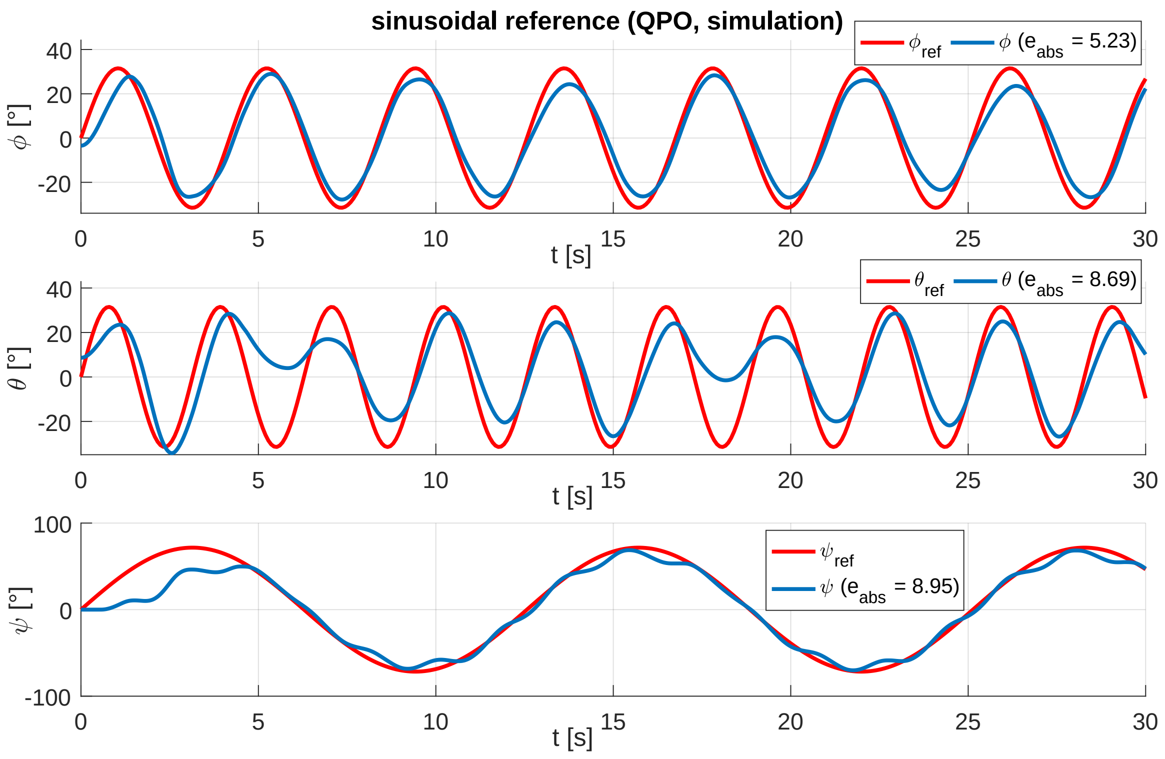

Figure 8.49.: Simulation results of MPC via QPO for sinusoidal reference signals with $n_p = n_c = n_i = 50$ (prediction time: $4\,s$). Tracking of ψ works here better than in case of the original configuration (see Figure 8.39).

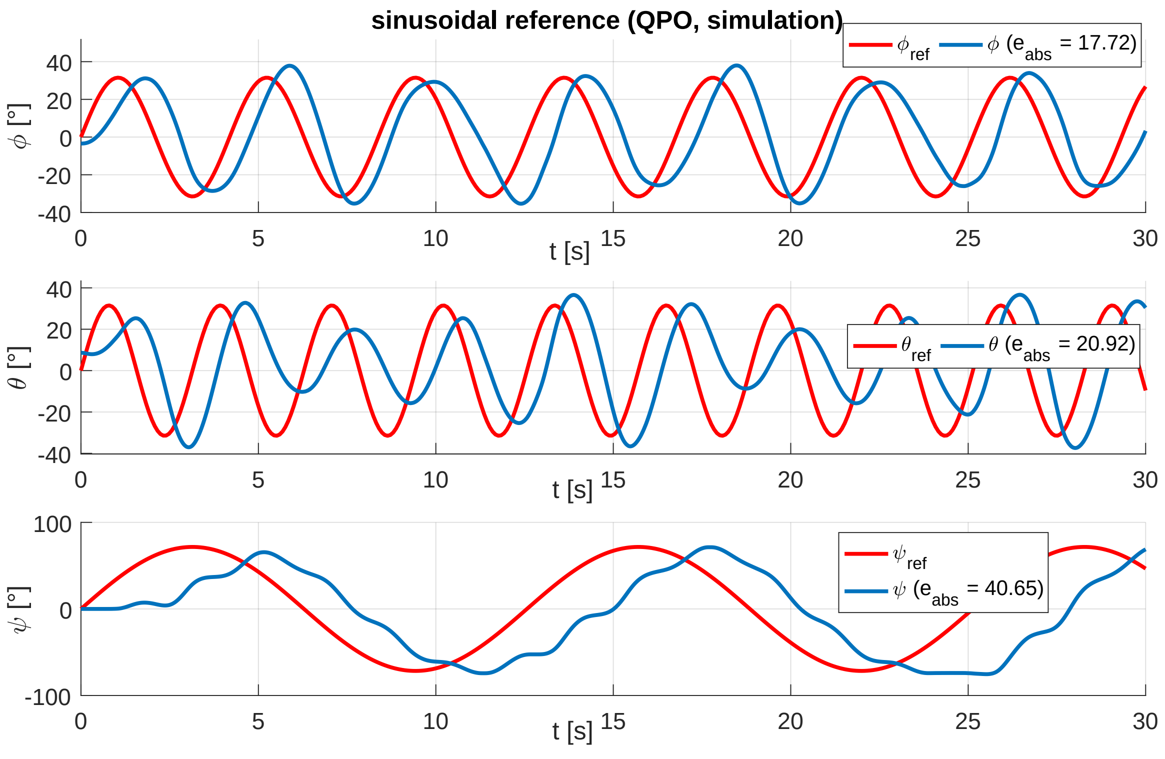

Figure 8.50.: Simulation results of MPC via QPO assuming unpredictable reference signals (compare with Figure 8.39).

8.6. Conclusion

In this chapter three different strategies for attitude control of a quadrotor have been developed and tested on a laboratory setup. Finally, these methods are evaluated with respect to four properties listed in Table 8.10. The first criterion ”performance” describes how well the

Table 8.10.: Comparison of the three approaches for reference angle tracking of the considered laboratory setup.

	CA-based	LQR	MPC
performance	high	low	high
parameter tuning effort	low / medium	low	high
complexity of implementation	medium	very low	medium
computational effort	low	very low	high

reference angle tracking task is solved. Both the majority of CA-based approaches and MPC achieve good results in this field. On the other hand LQR can only be used for small deflections from the equilibrium angles which is the reason why the corresponding results in Section 8.4.2 exhibit reference signals with smaller amplitude. MPC, whose results are comparable to those from CA, takes advantage of knowing the reference signals for the entire control horizon. Although, it can also be used if only the current values of the reference angles are available but then the outcome deteriorates dramatically.

The number of parameters of the CA-algorithms is rather low. It ranges from zero for DA and NINV to seven for WLA (typically only diagonal weighting matrices are used). Additionally, the three Super-Twisting controllers have altogether nine parameters which is the reason why the effort for tuning the CA-based method is labeled between low and medium. LQR has ten parameters if weighting matrices are restricted to diagonal ones. The MPC’s number of parameters related to diagonal weighting matrices is $n_p p + 2n_c m$. Moreover, the three horizons, four output constraints and γ have to be considered which add up to 393 parameters for configuration 1. If all weighting matrix entries are kept constant over the horizons this number reduces to 19 parameters.

In terms of implementation effort LQR is clearly the simplest algorithm. The realization of the Super-Twisting controller is also quite uncomplicated whereas for CA the situation heavily depends on the algorithm. If the solver is not included in the complexity considerations then WLA and EPA are straightforward to implement. NINV, RPINV, and ERPINV methods are moderately complex. In case of DA it depends on the chosen implementation. If formulated as linear program with a given solver it is trivial whereas the direct implementation based on [4] is cumbersome. The successive model approximation and the construction of the optimization problem make MPC moderately complex.

Due to the optimization problem dimensions MPC requires the greatest computational power of all. LQR basically involves two matrix multiplications per sample making it the least-demanding method. The computational complexities of the CA-based approaches differ very much, but all of them can be executed with the same sampling frequency (1 kHz) as the LQR.

The combination of high performance, low computational load, and moderate tuning effort makes CA the method of choice in this particular application.

9. Résumé

The main focus of this thesis lies on the development of computationally efficient algorithms for control allocation of input-redundant systems. The first presented algorithm (NINV) is an alternative way of computing a generalized inverse for the underdetermined linear systems of equations which describes the relationship between virtual and real controls. By considering the actuator constraints it leads to correct allocations for a greater portion of attainable virtual controls than it is typically the case for the weighted pseudoinverse method. With constant linear control effectivity matrices the online part of NINV is just a matrix multiplication. For varying matrices (e.g. due to online recalculations in case of nonlinear systems or actuator faults) the computationally most expensive part can be done offline.

The second algorithm (ERPINV) is an extension of the well-known RPINV algorithm. In situations with a large number of active actuator saturations the solution gained from the original RPINV will in most instances not reach the desired virtual control. Whereas the RPINV result cannot be influenced any more, ERPINV provides the possibility to specify the importance of the virtual control components in these cases. The allocation error of highly prioritized elements is preferably reduced.

The third developed algorithm (EPA) efficiently solves constrained QPs based on the penalty function approach. Though not restricted to CA applications (see Section 8.5 where it is utilized as MPC solver), they benefit from its characteristics such as hot-start ability and problem dimension reduction. The latter is realized by transforming the problem into null-space of the equality constraints. If applied to CA two strategies to handle infeasible virtual controls based on projection and gradient projection are implemented. It is also shown under which circumstances the two infeasibility handling methods are equivalent. Extensive comparisons with established QP solvers for CA as well as general purpose solvers demonstrate its competitiveness.

Finally, established and proposed CA methods are applied to practical applications. In comparison with the alternatives LQR and MPC the CA-based approaches show very good results with relatively low computational demands. Their easy and intuitive tuning process should also be emphasized. On the other hand, one has to admit that the CA performance in a closed-loop setup is naturally linked to the chosen high-level controller. The traditional rating of CA algorithms with respect to their accuracy in matching the desired virtual controls does not always conform to the closed-loop performance of the overall control system. Thus, choosing the theoretically "best" CA-method does not automatically lead to better results. This especially applies to situations with multiple active actuator constraints and is recognizable in Section 8.3.7 where ERPINV achieves lower errors than the methods based on numerical optimization. In the end an evaluation or ranking of CA-methods in closed-loop operation can only be valid for a particular application.

Appendix

A. Some Facts from Linear Algebra

A.1. Nullspace

Consider two arbitrary matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times o}$. The definition of the nullspace or kernel of $\boldsymbol{A}$ reads as [24]

$$\mathcal{N}_\mathrm{r}(\boldsymbol{A}) = \left\{ \boldsymbol{z} \in \mathbb{R}^n \middle| \boldsymbol{A}\boldsymbol{z} = \boldsymbol{0} \right\}. \tag{A.1}$$

The nullspace of the product of two matrices can be written as

$$\mathcal{N}_\mathrm{r}(\boldsymbol{A}\boldsymbol{B}) = \left\{ \boldsymbol{z} \in \mathbb{R}^o \middle| \, [\boldsymbol{z} \in \mathcal{N}_\mathrm{r}(\boldsymbol{B})] \vee [\boldsymbol{B}\boldsymbol{z} \in \mathcal{N}_\mathrm{r}(\boldsymbol{A})] \right\}. \tag{A.2}$$

A.2. Rank of matrix product

Given $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times o}$ their ranks satisfy (see [24])

$$\mathrm{rank}(\boldsymbol{A}) + \mathrm{rank}(\boldsymbol{B}) - n \leq \mathrm{rank}(\boldsymbol{A}\boldsymbol{B}) \leq \min\left[\mathrm{rank}(\boldsymbol{A}), \mathrm{rank}(\boldsymbol{B})\right]. \tag{A.3}$$

A.3. Rank of matrix difference

Assume two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same size then one can conclude according to Theorem 17 in [67] that

$$\begin{gathered} \mathrm{rank}\left([\boldsymbol{A}\ \boldsymbol{B}]\right) = \mathrm{rank}(\boldsymbol{A}) = \mathrm{rank}\left(\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{B}\end{bmatrix}\right) \text{ and } \boldsymbol{B}\boldsymbol{A}^\dagger\boldsymbol{B} = \boldsymbol{B} \\ \Updownarrow \\ \mathrm{rank}(\boldsymbol{A} - \boldsymbol{B}) = \mathrm{rank}(\boldsymbol{A}) - \mathrm{rank}(\boldsymbol{B}). \end{gathered} \tag{A.4}$$

B. Closed-loop Dynamics in Clutch's Locked Mode

Assumption B.0.1. The clutch is in locked mode. As a result it is not possible to directly modify T_c any more and (7.3) changes to

$$\begin{bmatrix}\dot{\omega}_1\\ \dot{\omega}_2\end{bmatrix} = \begin{bmatrix}-\frac{k_1}{J_1} & 0\\ 0 & -\frac{k_2}{J_2}\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\end{bmatrix} + \begin{bmatrix}\frac{1}{J_1} & 0\\ 0 & \frac{1}{J_2}\end{bmatrix}\begin{bmatrix}v_1\\ v_2\end{bmatrix} + \begin{bmatrix}-\frac{T_c}{J_1}\\ \frac{T_c}{J_2}\end{bmatrix} \tag{B.1}$$

Assumption B.0.2. $\forall t : |\overline{T}_c| \leq T_{c,max} \Rightarrow T_c \equiv \overline{T}_c$.

Assumption B.0.3. Both reference signals for the linear state controller (7.4) are identical, i.e. $r_1 \equiv r_2 \equiv r$.

Assumption B.0.4. The linear state controller for (7.12) reads as

$$v = -(k_{11} + k_{22})\omega - (k_1 + k_2 + k_{11} + k_{22})r. \tag{B.2}$$

Proposition B.0.1. *Under the assumptions B.0.1 - B.0.4 the closed-loop dynamics of (B.1) and controller (7.4) is the same as of (7.12) and controller (B.2).*

Proof. The closed-loop dynamics of the reduced first order system (7.12) together with (B.2) reads as

$$\dot{\omega} = -\frac{k_1 + k_2 + k_{11} + k_{22}}{J_1 + J_2}(\omega - r). \tag{B.3}$$

Similarly inserting (7.4) into (B.1) and considering $\omega_1 \equiv \omega_2$ yields

$$\begin{bmatrix}\dot{\omega}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}-\frac{k_1+k_{11}}{J_1} & 0\\ 0 & -\frac{k_2+k_{22}}{J_2}\end{bmatrix}\begin{bmatrix}\omega - r\\ \omega - r\end{bmatrix} + \begin{bmatrix}-\frac{T_c}{J_1}\\ \frac{T_c}{J_2}\end{bmatrix} \tag{B.4}$$

The explicit expressions of the elements of $\boldsymbol{v}$ are

$$v_1 = -k_{11}\omega + (k_1 + k_{11})r \tag{B.5a}$$

$$v_2 = -k_{22}\omega + (k_2 + k_{22})r. \tag{B.5b}$$

Please note that in locked mode there is a one-to-one mapping from virtual controls to the torques of ICE and EM, i.e. $v_1 = T_1$ and $v_2 = T_2$. Therefore, (B.5) and (1.9) provide the transmitted clutch torque (w.l.o.g. supposed that $T_{load} = 0$)

$$\overline{T}_c = \frac{J_2(-k_{11}\omega + k_1 r + k_1 1r) - J_1(k_{22}\omega + k_2 r + k_{22}r) - k_1\omega J_2 + k_2\omega J_1}{J_1 + J_2}. \tag{B.6}$$

Due to Assumption B.0.2 one can substitute (B.6) into both equations of (B.4) and obtains (B.3) in both cases. ■

C. Modeling and Control of an Electromagnetic Clutch Actuator

In case of ICE start-up the clutch control is of major importance for its success. This chapter deals with modeling, identification, and control of an electromagnetic clutch actuator. Its highly nonlinear dynamics require a model-based approach for controller development. Two methods relying on feedback linearization and flatness-based control are proposed. The latter one is implemented on an electronic control unit (ECU) and tested on an actuator test bed where it shows good results. Most of the content of this chapter has already been published in the author's works [52] and [53] and is cited literally.

C.1. Actuator modeling

The clutch under investigation is normally open, which means that it is closed by activating the electromagnet. Figure C.1 illustrates the working principle of the actuator. The electromagnet

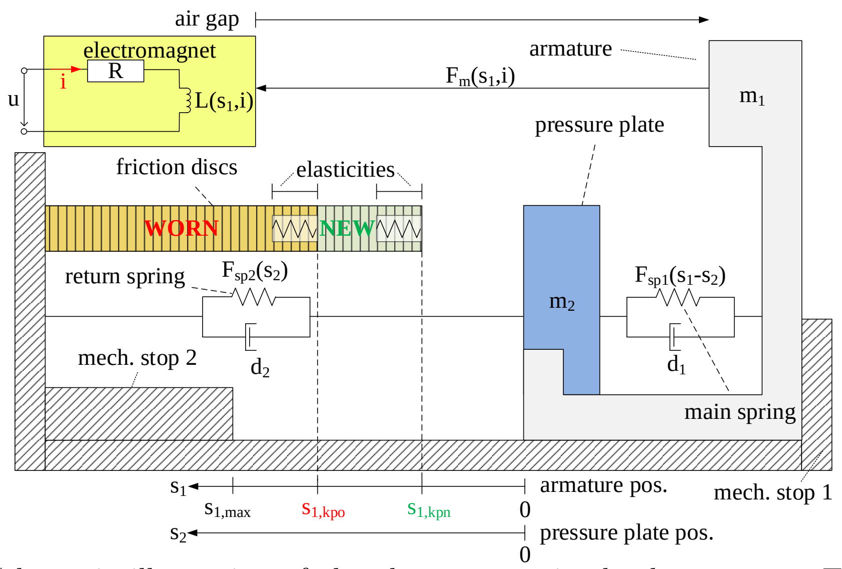

Figure C.1.: Schematic illustration of the electromagnetic clutch actuator. The kiss point position $s_{1,kpo}$ or $s_{1,kpn}$ depends on the degree of wear. (© 2014 IEEE, [52]).

generates a magnetic force F_m which acts on the lumped mass m_1. This so-called armature can move between two mechanical stops. The return spring opens the clutch if the actuator is deactivated. On the armature a pressure plate is mounted, which enables torque transmission by compressing the friction discs. The width of the friction discs depends on their degree of wear. The actuator is used in a dry clutch where this change in width is substantially higher than in wet clutches. The axial force acting on the pressure plate comes from the main spring. It is preloaded with several kilonewtons and enables the armature to travel to mechanical stop 2 regardless of the width of the friction discs. This fact is important, because it reduces the amount of power needed, to keep the clutch completely closed. At the so-called kiss point, contact between the friction discs is established and torque is transmitted. As the axial force is increased at this point, the elasticities of the mechanical components come into play, see Section C.1.2 [52].

C.1.1. Electrical subsystem

The dynamics of the electromagnetic subsystem are described by the voltage equation

$$u = Ri + \dot{\Psi}(s_1, i), \tag{C.1}$$

where s_1 is the actual armature position, R is the coil's resistance, $\Psi(s_1, i)$ is the total magnetic flux of all coil windings, i is the current, and u the input voltage of the actuator [68]. For sufficiently large air gaps the nonlinear material behavior like saturation effects can be neglected which leads to a position-dependent inductance ([69] and [70]). The considered actuator is designed to generate high axial forces with relatively low currents. Therefore the air gap between the electromagnet and the armature is very small in order to take advantage of the resulting gain in magnetic force. As a consequence there is a nonlinear relation between current and total magnetic flux. This can be modeled by introducing a current-dependent inductance $L(s_1, i)$ ([52], [69], [70]), i.e.

$$\Psi(s_1, i) = L(s_1, i)i. \tag{C.2}$$

A common approach in literature (e.g. in [38], [71], and [72]) to describe the inductance or the magnetic flux and also the magnetic force $F_m(s_1, i)$ is based on simplifications of the geometric shape and of the magnetic circuit. In the present work quasi-static finite element (FEM) simulations were used to determine these relationships. The resulting characteristic curves for magnetic force and inductance are shown in Figures C.2 and C.3 respectively. Using (C.1) and (C.2), the current dynamics of the electromagnet are given by [52]

$$u = Ri + \frac{\partial L(s_1, i)}{\partial s_1}\dot{s}_1 i + \left[\frac{\partial L(s_1, i)}{\partial i} i + L(s_1, i)\right]\frac{di}{dt}. \tag{C.3}$$

The partial derivatives of the inductance are computed by numerical methods using the FEM results. All the inductance related quantities are directly used in (C.3) by means of look-up tables. Magnetic hysteresis is not included in this model due to its minor influence on the system behavior. The input voltage of the coil $|u| \leq u_{max}$ is symmetrically bounded around the origin where $u_{max} > 0$ is a known positive constant [52].

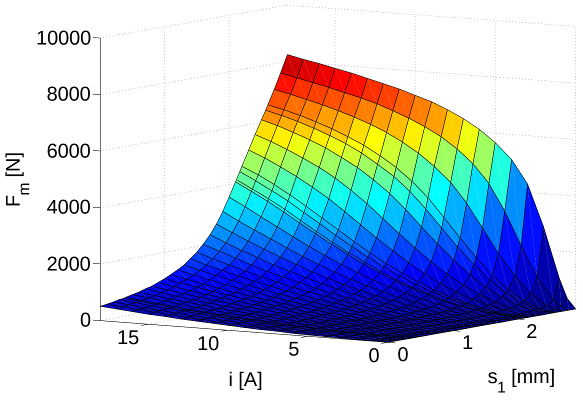

Figure C.2.: Magnetic force as a function of current and armature position. (© 2014 IEEE, [52]).

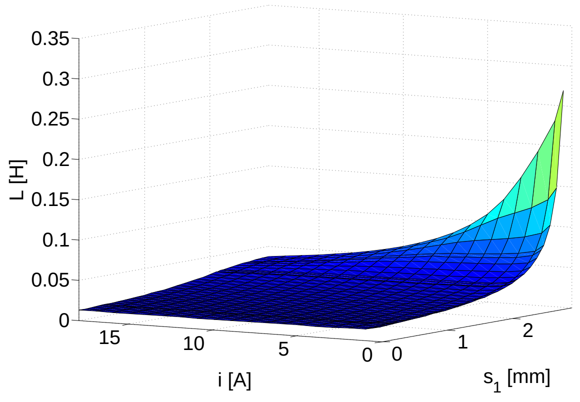

Figure C.3.: The inductance as a function of current and armature position. (© 2014 IEEE, [52]).

C.1.2. Mechanical subsystem

Neglecting all the kinematic constraints, the mechanical subsystem can be regarded as a two-mass spring-damper system. Figure C.4 illustrates the interaction of all involved elements. The equation describing the armature motion reads as [52]

$$m_1\ddot{s}_1 + F_{sp1}(s_1 - s_2) + d_1\left[\dot{s}_1 - \dot{s}_2\right] = F_m(s_1, i), \tag{C.4}$$

where m_1 is the armature mass, s_2 the pressure plate position, $F_{sp1}(s_1 - s_2)$ the main spring force, and d_1 the damping coefficient. Similarly the dynamics of the pressure plate are governed by equation [52]

$$\begin{aligned} m_2\ddot{s}_2 + F_{sp2}(s_2) + d_2\dot{s}_2 + F_{el}(s_2) &= \\ F_{sp1}(s_1 - s_2) &+ d_1\left[\dot{s}_1 - \dot{s}_2\right] \end{aligned} \tag{C.5}$$

with its mass m_2, the return spring force $F_{sp2}(s_2)$, the corresponding damping coefficient d_2, and the spring force modeling the system's elasticities $F_{el}(s_2)$. Two mechanical stops limiting the armature movement within the range $s_1 \in [0, s_{1,max}]$ are implemented as integrator bounds. Case 1 in Figure C.4 shows that the preloaded main spring $F_{sp1}(s_1 - s_2)$ pushes the pressure plate onto the armature. The return spring force $F_{sp2}(s_2)$ also acts on the pressure plate but it is several orders of magnitude smaller than the main spring's preload force $F_{sp1,pl}$ (see Figure C.5). Thus, while the pressure plate does not touch the friction discs, its position is equal to the armature position [52]. Figure C.4 also depicts what happens once the contact is established (case 2). As the magnetic force $F_m(s_1, i)$ is increased at this position, all the involved mechanical elements such as the armature, the pressure plate, the friction discs and the springs are slightly deformed. This effect comes from the material's elasticities and is modeled as a very stiff linear spring acting against the displacement of m_2. The region of this elastic movement is below a quarter millimeter wide. The positions s_1 and s_2 start to diverge as soon as the sum of forces on the left hand side of the pressure plate exceeds that on the right hand side. So formally it is necessary to model the motion of the two masses with separate differential equations. However, due to the steeply rising force $F_{el}(s_2)$ (see Figure C.5), the small damping coefficient, and the relatively moderate armature speed while crossing this area, the influence of the damping related force can be neglected for the determination of the instant of time, where $s_1 \neq s_2$. Instead one can compute a position $s_{2,max}$ where the preload force of the main spring is exceeded and it starts to compress. Hence, as long as the pressure plate does not reach that position (cases 1 and 2) s_1 and s_2 as well as their time derivatives are identical [52]:

$$s_1 \equiv s_2 \quad \rightarrow \quad \dot{s}_1 \equiv \dot{s}_2 \quad \text{and} \quad \ddot{s}_1 \equiv \ddot{s}_2 \tag{C.6}$$

Substituting the results from relations (C.6) into (C.4) and (C.5) yields

$$m_1\ddot{s}_1 + F_{sp1}(0) = F_m(s_1, i) \tag{C.7}$$

and

$$m_2\ddot{s}_1 + F_{sp2}(s_1) + d_2\dot{s}_1 + F_{el}(s_1) = F_{sp1}(0). \tag{C.8}$$

Equations (C.7) and (C.8) can be combined to obtain a single equation describing the motion of the coupled masses m_1 and m_2

$$\left[m_1 + m_2\right]\ddot{s}_1 + F_{sp2}(s_1) + d_2\dot{s}_1 + F_{el}(s_1) = F_m(s_1, i) \tag{C.9}$$

for both cases 1 and 2 shown in Figure C.4. The spring characteristics of the main spring is rather flat once the preload force is exceeded at $s_2 = s_{2,max}$. No more deformations occur

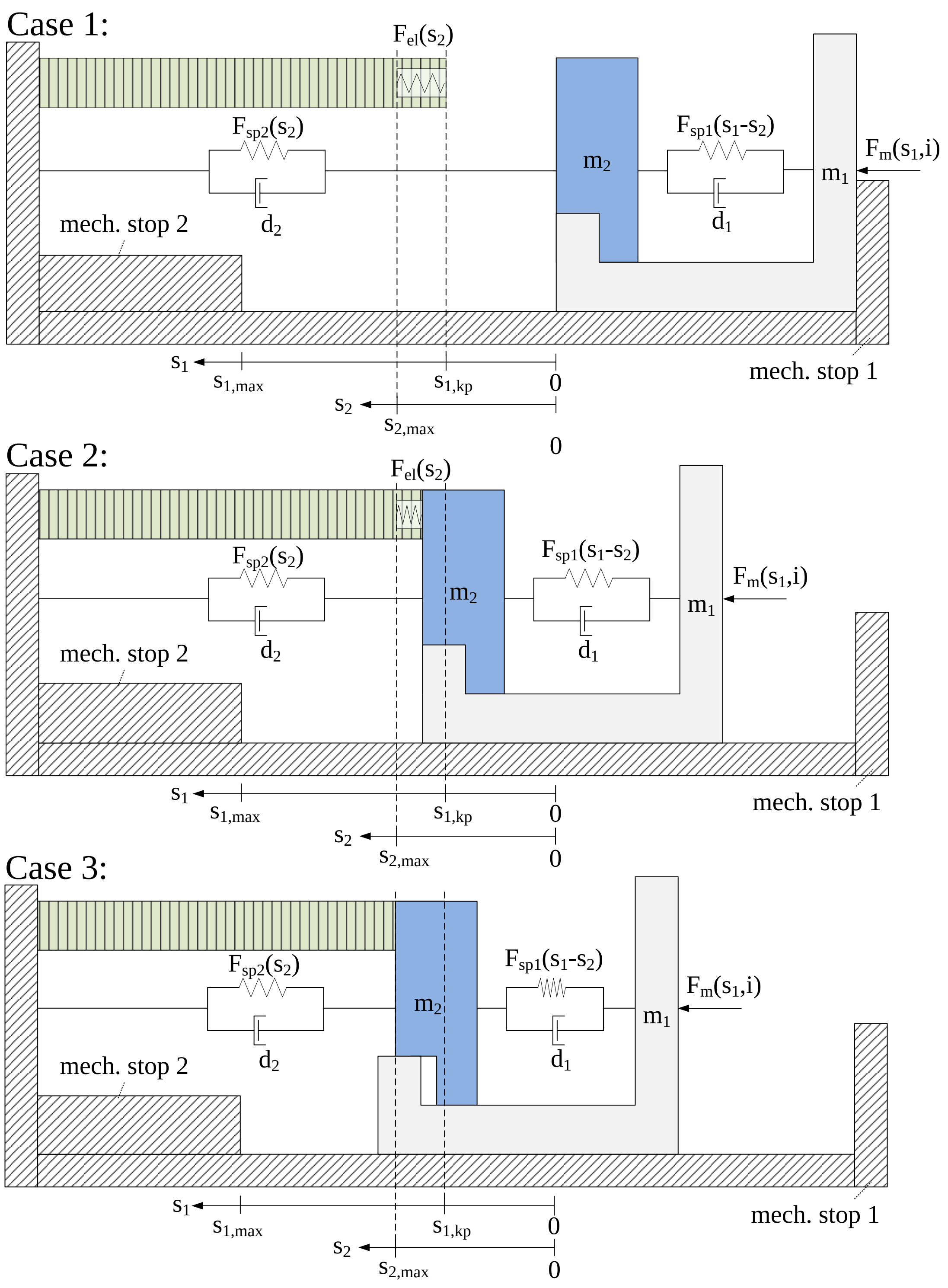

Figure C.4.: Movement of armature and pressure plate. The clutch's kiss point is denoted as $s_{1,kp}$. (© 2014 IEEE, [52]).

at this point and the pressure plate position remains constant, which yields $s_2 \equiv s_{2,max} \rightarrow \dot{s}_2 \equiv 0$ and $\ddot{s}_2 \equiv 0$. As a result the original equation of motion (C.4) changes in case 3 of Figure C.4 to [52]

$$m_1\ddot{s}_1 + F_{sp1}(s_1 - s_{2,max}) + d_1\dot{s}_1 = F_m(s_1, i) \tag{C.10}$$

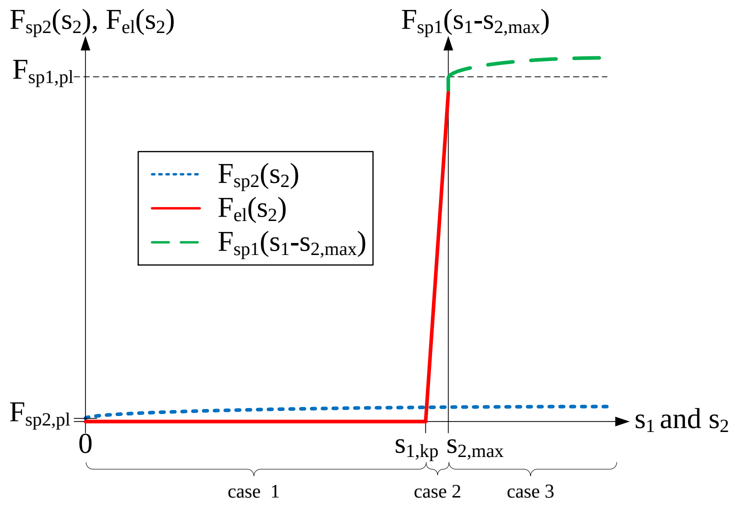

Figure C.5.: Principal sketch of the different spring forces occurring in the system. (© 2014 IEEE, [52]).

and moreover (C.5) simplifies to

$$F_{sp2}(s_{2,max}) + F_{el}(s_{2,max}) = F_{sp1}(s_1 - s_{2,max}) + d_1\dot{s}_1. \tag{C.11}$$

The armature itself will move towards mechanical stop 2 as the magnetic force rises (see case 3 in Figure C.4).

C.1.3. Overall system description

Based on the results of the previous Sections C.1.1 and C.1.2, a system of differential equations describing the actuator's behavior can be formulated. A state vector is introduced as

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T := \begin{bmatrix} s_1 & \dot{s}_1 & i \end{bmatrix}^T \tag{C.12}$$

consisting of armature position, speed, and coil current. Using (C.3), (C.9), (C.10), and (C.12) the state space representation of the system reads as [52]

$$\dot{x}_1 = x_2 \tag{C.13a}$$

$$\dot{x}_2 = \frac{1}{m(x_1)}\left[-F_{sp}(x_1) - d(x_1)x_2 + F_m(x_1, x_3)\right] \tag{C.13b}$$

$$\dot{x}_3 = \frac{1}{\tilde{L}(x_1, x_3)}\left[-\frac{\partial L(x_1, x_3)}{\partial x_1}x_2x_3 - Rx_3 + u\right]. \tag{C.13c}$$

The position-dependent mass $m(x_1)$ in (C.13b) represents the distinction between cases 1 and 2 on the one hand and case 3 on the other hand (see Figure C.4):

$$m(x_1) = \begin{cases} m_1 + m_2 & \text{if } x_1 < s_{2,max} \\ m_1 & \text{else} \end{cases} \tag{C.14}$$

In the same way, the actual spring force $F_{sp}(x_1)$ is chosen according to the armature position [52]

$$F_{sp}(x_1) = \begin{cases} F_{sp2}(x_1) + F_{el}(x_1) & \text{if } x_1 < s_{2,max} \\ F_{sp1}(x_1 - s_{2,max}) & \text{else} \end{cases} \tag{C.15}$$

as well as the damping coefficient

$$d(x_1) = \begin{cases} d_2 & \text{if } x_1 < s_{2,max} \\ d_1 & \text{else} \end{cases} . \tag{C.16}$$

Equation (C.13c) contains $\tilde{L}(x_1, x_3)$, which is a short notation for

$$\tilde{L}(x_1, x_3) = L(x_1, x_3) + \frac{\partial L(x_1, x_3)}{\partial x_3} x_3. \tag{C.17}$$

Finally, the computation of the axial force $F_{ax}(\mathbf{x})$, which allows torque transmission of the clutch, is presented. Before reaching the kiss point (case 1 in Figure C.4) there is no axial force. Beyond this position the sum of forces causing the elastic deformation acts on the friction discs, i.e. [52]

$$F_{ax}(\mathbf{x}) = \begin{cases} 0 & \text{if } x_1 < s_{1,kp} \\ F_{el}(x_1) & \text{if } s_{1,kp} \leq x_1 < s_{2,max} \\ F_{ax,3}(x_1) & \text{else} \end{cases} \tag{C.18}$$

with

$$F_{ax,3}(\mathbf{x}) = F_{sp1}(x_1 - s_{2,max}) + d_1 x_2 - F_{sp2}(x_1). \tag{C.19}$$

In case 2 of Figure C.4, it is equal to $F_{el}(x_1)$ and in case 3 it can be derived from (C.11) which leads to (C.19).

C.2. Controller design

The actuator control system is designed to work in three different operating modes. A major goal is to ensure smooth armature motion to and from the kiss point without unintended collisions with the friction discs. Therefore, two model-based armature position controllers are designed and compared in simulations. The flatness-based approach is additionally implemented and tested on an actuator testbed. During the slipping phase of the clutch, accurate axial force regulation has to be achieved. At the first step a PI controller is chosen to accomplish this task. When the clutch is completely closed the armature should be kept at mechanical stop 2 to minimize the required electrical power. This is done by a PI current controller [52].

C.2.1. Feedback linearization-based position control

The plant model characterized by (C.13) belongs to the class of affine-input systems whose generalized structure is given by

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x})u \tag{C.20a}$$

$$y = c(\mathbf{x}). \tag{C.20b}$$

For the purpose of position control, the system output is chosen as $y = x_1$. In order to achieve Input-Output Linearization the time derivatives of the output y are computed until the input u explicitly appears [73]. In the present case, this straight-forward process yields $\dddot{y} = L_a^3 c + L_b(L_a^2 c)u$, where $L_a c$ is the *Lie derivative*[1] of $\mathbf{a}(\mathbf{x})$ with respect to $c(\mathbf{x})$. That

[1]The Lie derivative is recursively defined as: $L_g f = \frac{\partial f}{\partial \mathbf{x}} g(\mathbf{x})$, $L_g^2 f = \frac{\partial (L_g f)}{\partial \mathbf{x}} g(\mathbf{x})$, and so on.

equation shows that the relative degree is $\delta = 3$, which is equal to the system order and implies that there are no internal dynamics. In the present case the objective of feedback linearization is to induce a triple-integrator relationship $\dddot{y} = v$, where v is the virtual input. With this in mind, a nonlinear compensation term [52]

$$u(\mathbf{x}) = \frac{1}{L_b(L_a^2 c)}\left(v - L_a^3 c\right) \tag{C.21}$$

is needed. Evaluating (C.21) for the state space model (C.13) leads to [52]

$$\begin{aligned}\tilde{u}(\mathbf{x}) =& \frac{\tilde{m}\tilde{L}(x_1,x_3)}{\frac{\partial F_m(x_1,x_3)}{\partial x_3}}\left[v - \frac{1}{\tilde{m}}\left(-\frac{\partial F_{sp}(x_1)}{\partial x_1}x_2 + \frac{d(x_1)^2 x_2}{\tilde{m}} + \right.\right.\\ & \left.\left.\frac{d(x_1)F_u(x_1,x_3)}{\tilde{m}} + \frac{\partial F_m(x_1,x_3)}{\partial x_1}x_2\right)\right] + \frac{\partial L}{\partial x_1}x_2 x_3 + R x_3.\end{aligned} \tag{C.22}$$

with $\tilde{m} = m(x_1)$. In contrast to the approach found in [73], expression (C.22) contains a distinction of cases given by [52]

$$F_u(x_1,x_3) = \begin{cases} 0 & \text{if}\,[F_m(x_1,x_3) < F_{sp2}(x_1)] \wedge [x_1 < \epsilon_1] \\ F_{sp}(x_1) - F_m(x_1,x_3) & \text{else} \end{cases} \tag{C.23}$$

where $\epsilon_1 > 0$ is a small positive constant. The else-branch contains an expression coming from the evaluation of (C.21) for model (C.13). The necessity for choosing $F_u(x_1,x_3) = 0$ in the first branch results from the facts that the limits of the armature movement (0 and $s_{1,max}$) are not contained in the state equations (C.13) and that the return spring is preloaded. This preload force would result in a negative input voltage for a zero-reference signal in the vicinity of mechanical stop 1 where $x_1 = 0$ [52]. After linearization two variants of feedback control are implemented. In both approaches the virtual control signal v is computed by linear state controllers. In variant 1 the closed-loop dynamics of the plant, the nonlinear compensator, and the linear controller can be denoted in controllability canonical form

$$\begin{bmatrix}\dot{\hat{x}}_1\\ \dot{\hat{x}}_2\\ \dot{\hat{x}}_3\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -\alpha_0 & -\alpha_1 & -\alpha_2\end{bmatrix}\begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\end{bmatrix} + \begin{bmatrix}0\\ 0\\ V\end{bmatrix} r \tag{C.24}$$

where α_0, α_1, α_2, and V are design parameters of the controller, r is the reference position signal, and the auxiliary state vector $\hat{\boldsymbol{x}} = [\hat{x}_1 \quad \hat{x}_2 \quad \hat{x}_3]^T = [x_1 \quad \dot{x}_1 \quad \ddot{x}_1]^T$. Altogether the virtual control signal v from the state controller reads as $v = Vr - \sum_{k=0}^{2}\alpha_k x_1^{(k)}$ where $x_1^{(k)}$ denotes the k-th time derivative of x_1. The parameters are chosen to ensure asymptotically stable dynamics and a vanishing steady-state error for reference steps. Variant 2 reduces the influence of parameter uncertainties and constant disturbances by introducing integral action into the state controller as illustrated in Figure C.6. The overall dynamics of the closed-loop system shown in Figure C.6 reads as

$$\begin{bmatrix}\dot{\hat{x}}_1\\ \dot{\hat{x}}_2\\ \dot{\hat{x}}_3\\ \dot{\tilde{x}}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ -k_1 & -k_2 & -k_3 & \tilde{k}\\ -1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \tilde{x}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix} r \tag{C.25}$$

with parameters k_1, k_2, k_3, and $\tilde{k}$ constituting the eigenvalues of the system matrix.

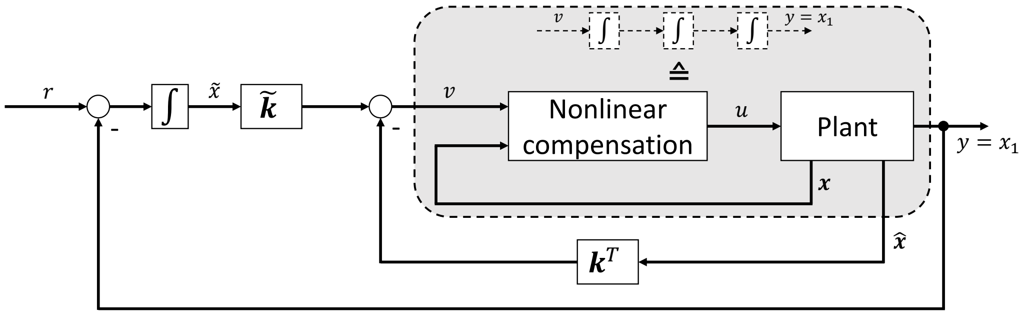

Figure C.6.: Structure of the closed-loop system with feedback linearization (nonlinear compensation) and integrating state controller. The controller's design parameters are $\boldsymbol{k} = [k_1 \;\; k_2 \;\; k_3]^T$ and $\tilde{k}$.

A prerequisite for the feedback linearization-based controller to work is a measurable or observable state vector. The subsequently presented results in Section C.3 were achieved under the assumption that $\mathbf{x}$ is measurable. The armature acceleration $\dot{x}_2$ which is needed in the linear control law was computed by the means of difference quotient. Similarly to the computation of the inductance's partial derivatives, the values for $\frac{\partial F_m(x_1,x_3)}{\partial x_1}$ and $\frac{\partial F_m(x_1,x_3)}{\partial x_3}$ were obtained by numerical differentiation of the FEM results and incorporated in the control law by means of look-up tables [52].

It should also be noted that $\tilde{u}(\mathbf{x})$ in (C.22) is not the actual input voltage of the actuator, which is instead written as

$$u(\mathbf{x}) = \begin{cases} 0 & \text{if } (x_3 < -\epsilon_2) \vee (r = 0) \\ \tilde{u}(\mathbf{x}) & \text{else} \end{cases} \tag{C.26}$$

with $\epsilon_2 > 0$ being a small positive constant. The first term in the condition of (C.26) prevents the superimposed linear controller from causing a negative current to speed up clutch opening. Negative currents generate magnetic forces acting in the same direction as those coming from positive currents [70] and so they are counter-productive for driving the armature back to $x_1 = 0$. It should be emphasized that (C.26) is not absolutely necessary for the compensator to work, it is rather an improvement of the control signal quality in this special case. Without expression (C.26), the controller would switch between positive and negative voltages with high frequency, generating a small amplitude current oscillation around zero [52].

As the electromagnet's behavior is the same for positive and negative currents (neglecting hysteresis) the characteristic curves in Figures C.2 and C.3 are mirrored along the s_1-axis to get the corresponding values for $x_3 < 0$. Therefore the memory space required for the look-up tables can be reduced by 50 %. At the same time (C.13b) and (C.13c) change to [52]

$$\dot{x}_2 = \frac{1}{m(x_1)} \left[-F_{sp}(x_1) - d(x_1)x_2 + F_m(x_1, |x_3|)\right] \tag{C.27a}$$

$$\dot{x}_3 = \frac{1}{\hat{L}(x_1, |x_3|)} \left[-\frac{\partial L(x_1, |x_3|)}{\partial x_1} x_2 x_3 - R x_3 + u\right] \tag{C.27b}$$

with $\hat{L}(x_1, |x_3|)$ being

$$\hat{L}(x_1, |x_3|) = L(x_1, |x_3|) + \frac{\partial L(x_1, |x_3|)}{\partial |x_3|} |x_3|. \tag{C.28}$$

Noting that $|x_3|$ is not differentiable at $x_3 = 0$ the resulting nonlinear compensation reads as

$$\hat{u}(\mathbf{x}) = \left\{ \frac{\tilde{m}\hat{L}(x_1, |x_3|)}{\frac{\partial F_m(x_1,|x_3|)}{\partial x_3}} \left[v - \frac{1}{\tilde{m}} \left(-\frac{\partial F_{sp}(x_1)}{\partial x_1} x_2 + \frac{d_2^2 x_2}{\tilde{m}} + \frac{d_2 F_u(x_1, |x_3|)}{\tilde{m}} + \frac{\partial F_m(x_1, |x_3|)}{\partial x_1} x_2 \right) \right] + \frac{\partial L}{\partial x_1} x_2 |x_3| + R|x_3| \right\} sign(x_3) \tag{C.29}$$

for $x_3 \neq 0$. Equation (C.29) shows that the control input alternates its sign with the current and so a high frequency switching is induced into the control signal for negative virtual inputs.

Expression (C.26) explicitly sets $u = 0$ in case of a zero reference. Otherwise the controller would compensate the return spring's preload force.

C.2.2. Flatness-based position control

Choosing the system output as the armature position $y = x_1$, the relative degree of (C.13) is $\delta = 3$, which is equal to the system order. Hence, y is called a *flat output* ([74]), which implies that state vector and input can be expressed as [53]

$$\boldsymbol{x} = \Phi\left(y, \dot{y}, \ldots, y^{(n-1)}\right) \tag{C.30}$$

$$u = \Psi\left(y, \dot{y}, \ldots, y^{(n)}\right). \tag{C.31}$$

The proposed controller consists of a model-based feedforward and a feedback part as illustrated in Figure C.7. A robustness analysis of this kind of control structure can be found in [75]. Due to *differential flatness* (see [76]) of the system regarding $y = x_1$, the feedforward control

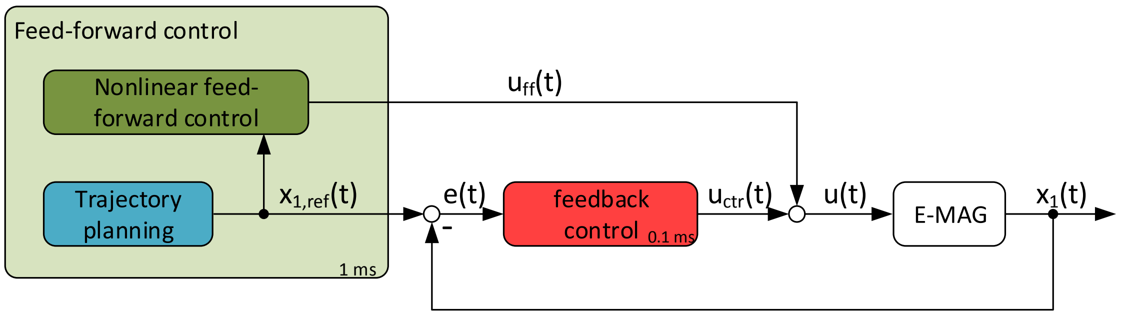

Figure C.7.: Structure of the flatness-based position controller. (© 2015 Elsevier, [53]).

can be written as inversion of (C.13). For a given sufficiently smooth reference trajectory $x_{1,ref}(t)$, one can compute armature speed $x_{2,ref}(t)$, acceleration $\dot{x}_{2,ref}(t)$, and the necessary magnetic force[2] from [53]

$$F_{m,ref} = (m_1 + m_2)\,\dot{x}_{2,ref} + F_{sp}(x_{1,ref}) + d_2 x_{2,ref}. \tag{C.32}$$

[2]The time parameter t is omitted from now on for better readability.

The magnetic force characteristic curve, which has been identified in Section C.5.1, must now be inverted for the determination of the required current

$$x_{3,ref} = F_m^{-1}\left(x_{1,ref}, F_{m,ref}\right). \tag{C.33}$$

Since the determination of $x_{3,ref}$ from the polynomial representation of F_m is not possible, it is computed from the related look-up table. The slope of the magnetic force in the vicinity of $x_{1,ref}$ reads as

$$\frac{\Delta F_m(x_3)}{\Delta x} = \frac{F_m\left(x_H, x_3\right) - F_m\left(x_L, x_3\right)}{x_H - x_L} \tag{C.34}$$

for all currents x_3 in the look-up table. x_H and x_L are the neighboring position entries of $x_{1,ref}$. Now the resulting force at $x_{1,ref}$ can be calculated from [53]

$$\begin{aligned} F\left(x_{1,ref}, x_3\right) = {} & F_m\left(x_L, x_3\right) + \\ & \frac{\Delta F_m(x_3)}{\Delta x}\left(x_{1,ref} - x_L\right) \end{aligned} \tag{C.35}$$

for every current x_3. After determining the neighboring forces (F_H and F_L) and currents ($x_{3,H}$ and $x_{3,L}$) from (C.35) the slope of F_m regarding current is given by

$$\frac{\Delta F_m}{\Delta x_3} = \frac{F_H - F_L}{x_{3,H} - x_{3,L}}. \tag{C.36}$$

Finally, the sought current $x_{3,ref}$ is obtained from

$$x_{3,ref} = x_{3,L} + \frac{F_{m,ref} - F_L}{\frac{\Delta F_m}{\Delta x_3}}. \tag{C.37}$$

This calculation sequence is carried out for several position-force combinations in order to construct the look-up table for $F_m^{-1}\left(x_{1,ref}, F_{m,ref}\right)$. After computing the desired current's time derivative $\dot{x}_{3,ref}$, the feedforward control reads as [53]

$$\begin{aligned} u_{ff} = {} & \left[L\left(x_{1,ref}, x_{3,ref}\right) + \frac{\partial L\left(x_{1,ref}, x_{3,ref}\right)}{\partial x_3} x_{3,ref}\right]\dot{x}_{3,ref} \\ & + \frac{\partial L\left(x_{1,ref}, x_{3,ref}\right)}{\partial x_1} x_{2,ref} x_{3,ref} + R x_{3,ref}. \end{aligned} \tag{C.38}$$

Remark C.2.1. One can summarize (C.32), (C.33), and (C.38) and express $x_{2,ref}$, $x_{3,ref}$, $\dot{x}_{2,ref}$, and $\dot{x}_{3,ref}$ by means of the flat output y to get to a notation that is consistent with (C.31) [53].

Remark C.2.2. All time derivatives in (C.32) and (C.38) are computed by means of difference quotient. As all signals in these equations are reference signals this does not cause any problems concerning noise [53].

The return spring is preloaded, which means that $F_{sp}(0) = F_{pl} > 0$. Hence, (C.32) and (C.33) yield a discontinuous magnetic force and current if the controller is activated. However, the current cannot change infinitely fast from zero to $x_{3,ref}$ and so even for a sufficiently smooth reference signal, trajectory following would not be achievable. Therefore, the preload force is filtered by means of a second order low pass filter in the feedforward control, whose cutoff frequency is $\omega_c = 300\,\frac{rad}{s}$ [53].

Trajectory planning generates a three times continuously differentiable reference position signal. Denoting the initial position as $z_0 := x_{1,ref}(0)$ and the target position at time T as $z_T := x_{1,ref}(T)$ it reads as

$$x_{1,ref}(t) = z_0 + (z_T - z_0) \sum_{i=n+1}^{2n+1} a_i \left(\frac{t}{T}\right)^i, \tag{C.39}$$

with $t \in [0, T]$, $n = 3$ being the system order, and $[a_4\ a_5\ a_6\ a_7]^T = [35\ \text{-}84\ 70\ \text{-}20]^T$.

The feedback part of the controller is basically a PD controller with minor modifications to increase robustness in case of parameter variations. If F_m and F_{sp} are not known exactly, a steady state deviation regarding the desired end position x_r will occur. In order to reduce this position error $e = r - y$ the controller is augmented by a saturation function [53]

$$u_{sat}(e) = \begin{cases} \frac{\overline{u}_{sat}}{b} e & \text{if } |e| < b \\ sign(e)\overline{u}_{sat} & \text{else} \end{cases} \tag{C.40}$$

whose amplitude $\overline{u}_{sat}$ and linear region b is chosen as a compromise between oscillation around the desired position and the ability to overcome model uncertainties. While the feedback controller uses a sample time of $0.1\,ms$, trajectory planning and feedforward control only need to run every millisecond. All computations are done using 32-bit fixed-point arithmetic. As a consequence, the computational load is reduced significantly which is important with respect to an implementation on a series car ECU [53].

C.3. Simulation results

Plant and controllers described in the previous sections were implemented and tested in Matlab/Simulink by means of the variable step solver *ode45*. The controllers ran at a sampling rate of 10 kHz. The input voltage was limited to $u_{max} = 10$ V. In the first place, the performance of the model based approaches is compared to that of a PID controller whose parameters were tuned in simulation. In fact it is slightly modified to prevent it from generating negative currents. The parameter sets for all controllers were kept constant during all experiments [52]. The main tasks for the position controllers are:

- Smoothly drive the armature from the initial position (mechanical stop 1 in Figure C.4) into the vicinity of the kiss point.
- Keep the armature at the kiss point position in order to enable a fast clutch engagement.
- Retrieve it to the original position.

The time spans for those tasks were defined as $100\,ms$, $80\,ms$, and $100\,ms$ respectively (the hold time is chosen so small just to improve the visibility of the transients).

The behavior of the actuator strongly depends on the air gap between electromagnet and armature. The smaller the air gap (or equivalently the greater the armature position) the greater the impact of the nonlinear magnetic characteristics on the movement. In a real clutch the degree of wear of the friction discs determines the kiss point position and therefore the air gap. Hence, the following position control simulations are carried out for three different reference positions corresponding to moderate, low, and high wear. The first presented result in Figure C.8 comes from a PID position controller which was specifically tuned for this reference

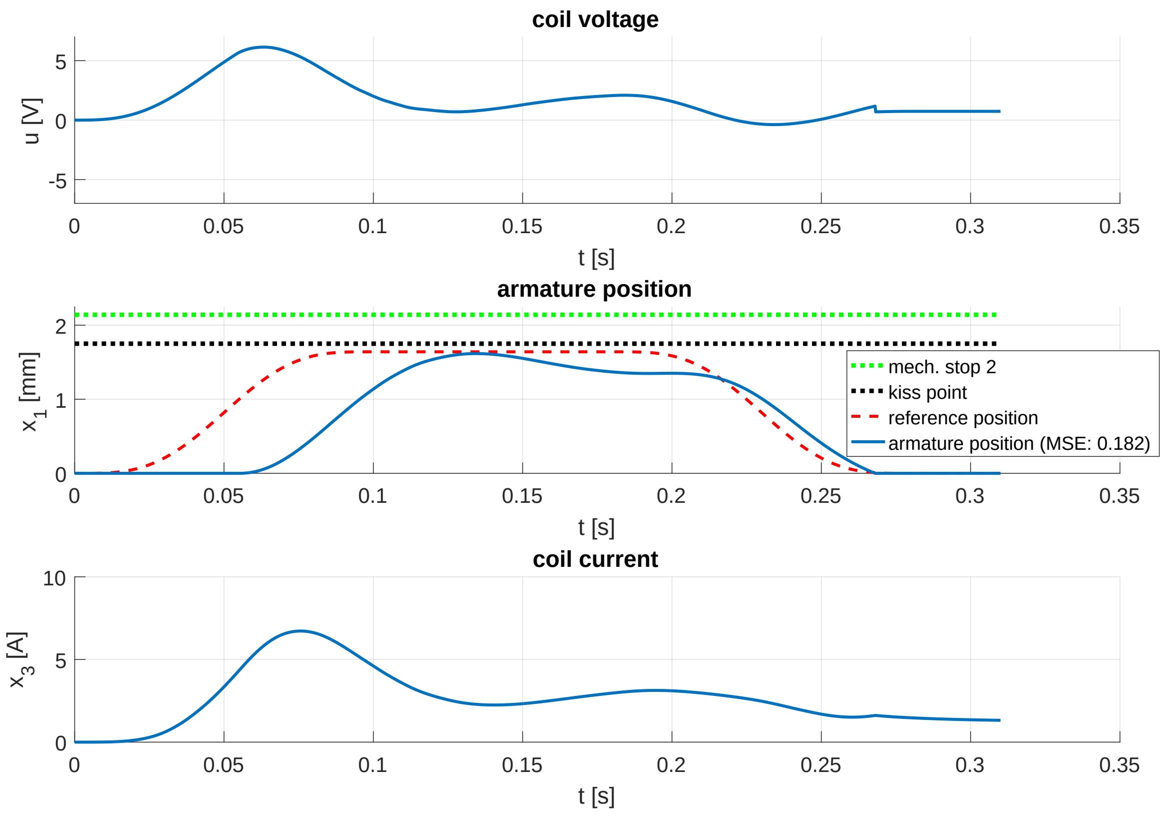

Figure C.8.: Result of the PID armature position controller for moderate clutch wear. This is the nominal case for which the controller parameters were tuned.

signal. After an initial lag of about 50 ms it almost reaches the goal position but is not able to hold it. For longer hold times the armature would slowly approach the goal position again. In case of low or high friction disc wear the kiss point is further away or closer to mechanical stop 2 respectively. Figures C.9 and C.10 show the related simulation results. Moving the armature to a smaller position than in the nominal case (moderate wear) results in a greater deviation from the desired value as it can be seen in Figure C.9. The PID controller could achieve similar results as in Figure C.8 if a new set of parameters were used. The position error resulting from the nominal parametrization is not acceptable because it would cause an hard impact on the kiss point if the control mode is switched to force / torque control. When positions which are closer to the electromagnet are desired the PID controller cannot decelerate the armature early enough to prevent it from hitting the kiss point (Figure C.10). Figures C.11 - C.13 show the results from the flatness-based position controller. Almost perfect trajectory tracking is accomplished regardless of the reference position. This behavior is not surprising since the feedforward control is computed from the same model with exactly the same parameters as the plant. The influence of parameter uncertainties is investigated later on. Comparing the minor deviations in coil current during the first 75 ms of Figures C.11 - C.13 that lead to end position differences of almost 20 % gives an impression on the sensitivity of the actuation system. Finally, the results of the controllers based on feedback linearization are depicted in Figures C.14 - C.16. Both of the solve the position tracking problem satisfactorily. One can recognize that the response of the controller with integral part is slightly slower than in the other case.

In practice the behavior of the developed actuator model will not perfectly match that

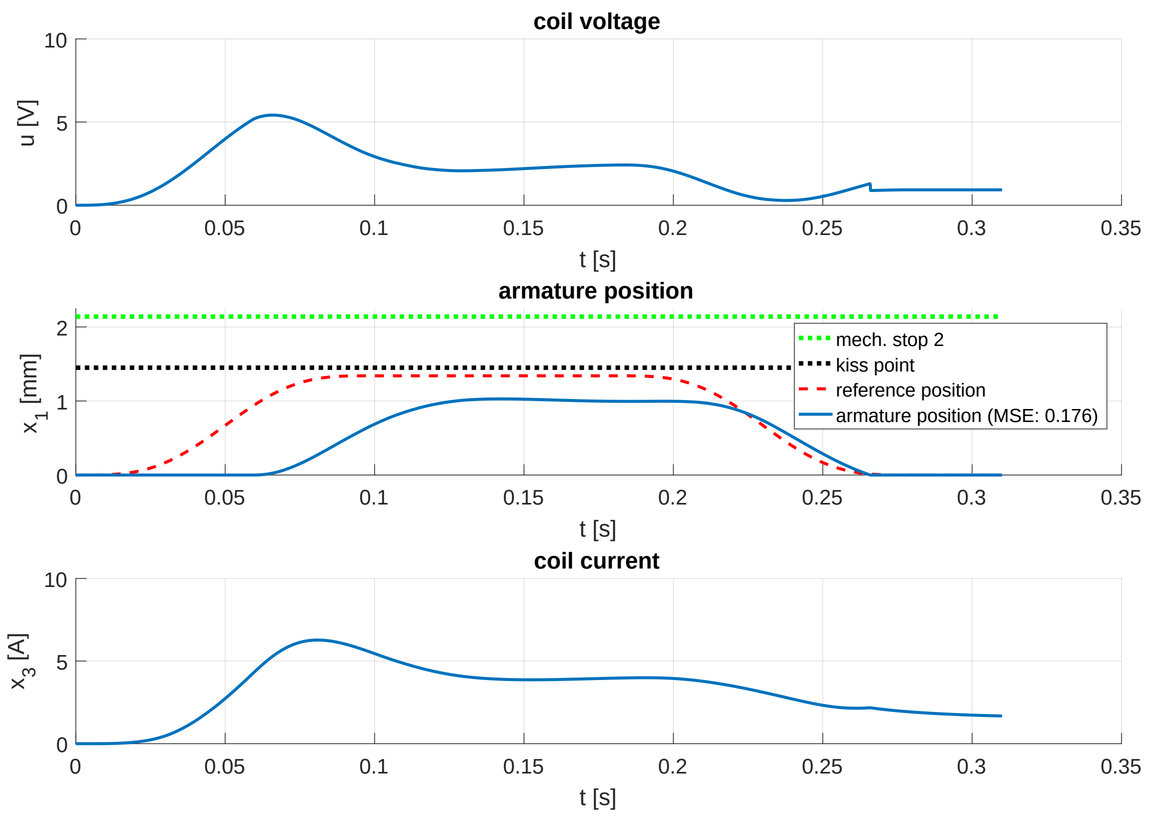

Figure C.9.: Result of the PID armature position controller for low clutch wear. A significant deviation between reference and actual position can be observed which prevents switching to force / torque control mode.

of the real-world actuator and the model parameters are not known exactly. Thus, it has to be investigated how such uncertainties influence the performance of the control strategies. Exemplary, the influence of an error in the magnetic force characteristic is determined. To this end, a scaling factor which modifies the magnetic force in the plant by -25 %, -10 %, +10 %, and +25 % is used. Starting with the PID controller, Figure C.17 shows the results. Although this parameter variation obviously changes the resulting peak position there is no hard impact on the kiss point. For the flatness-based position controller the situation is different as it can be seen in Figure C.18. In case of a 25 % positive magnetic force error at the plant the armature collides with the friction discs at the kiss point. The reason for this is that for the flatness-based controller the majority of the commanded voltage comes from the feedforward control law. The gain of the feedback controller is rather low and so it cannot compensate the model error. In a real application the feedback parameters must be chosen as a compromise between model error compensation and measurement noise tolerance. Furthermore, such a big error on the entire magnetic force characteristic, which is the defining property of the actuator, is rather unlikely. Figures C.19 - C.20 show the results for the position controllers based on feedback linearization. Both of them do not induce a collision at the kiss point. In case of the ordinary state controller a steady state error is recognizable. In contrast, the position course of the integrating state controller remains almost unaffected.

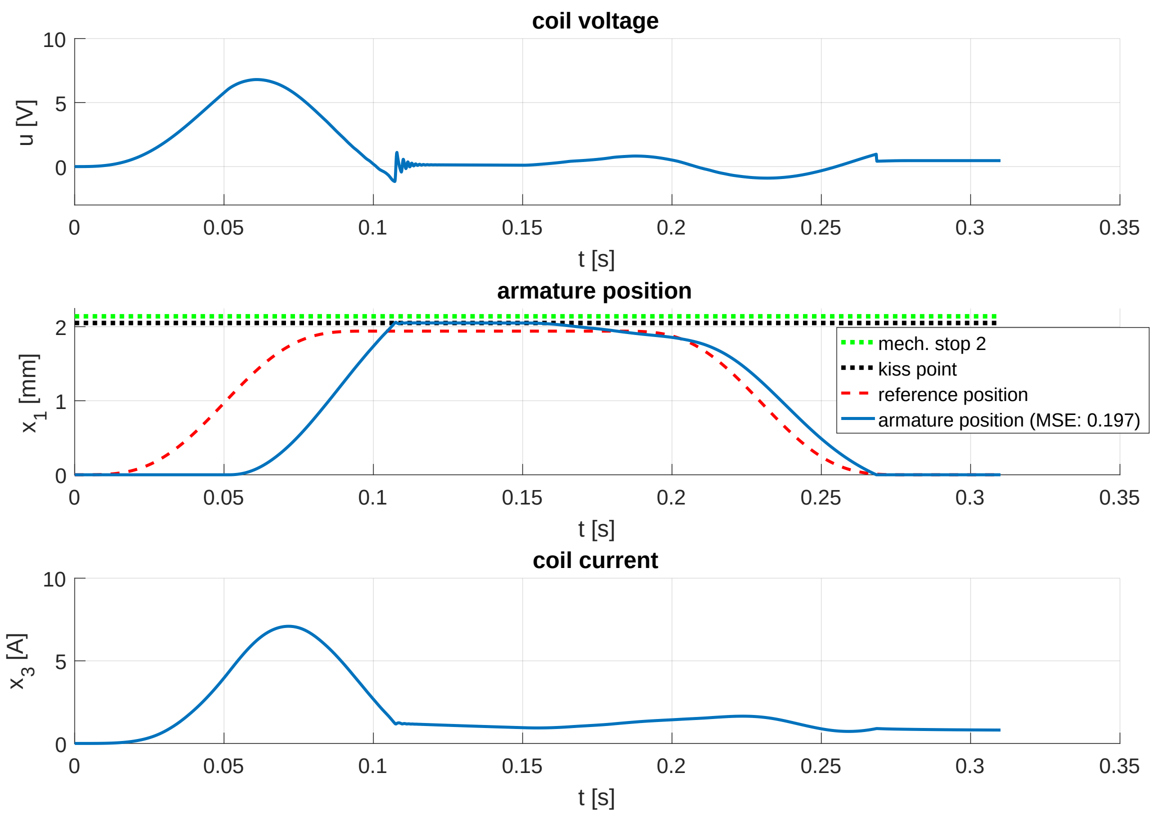

Figure C.10.: Result of the PID armature position controller for high clutch wear. The position overshoot leads to a hard impact of the armature on the kiss point.

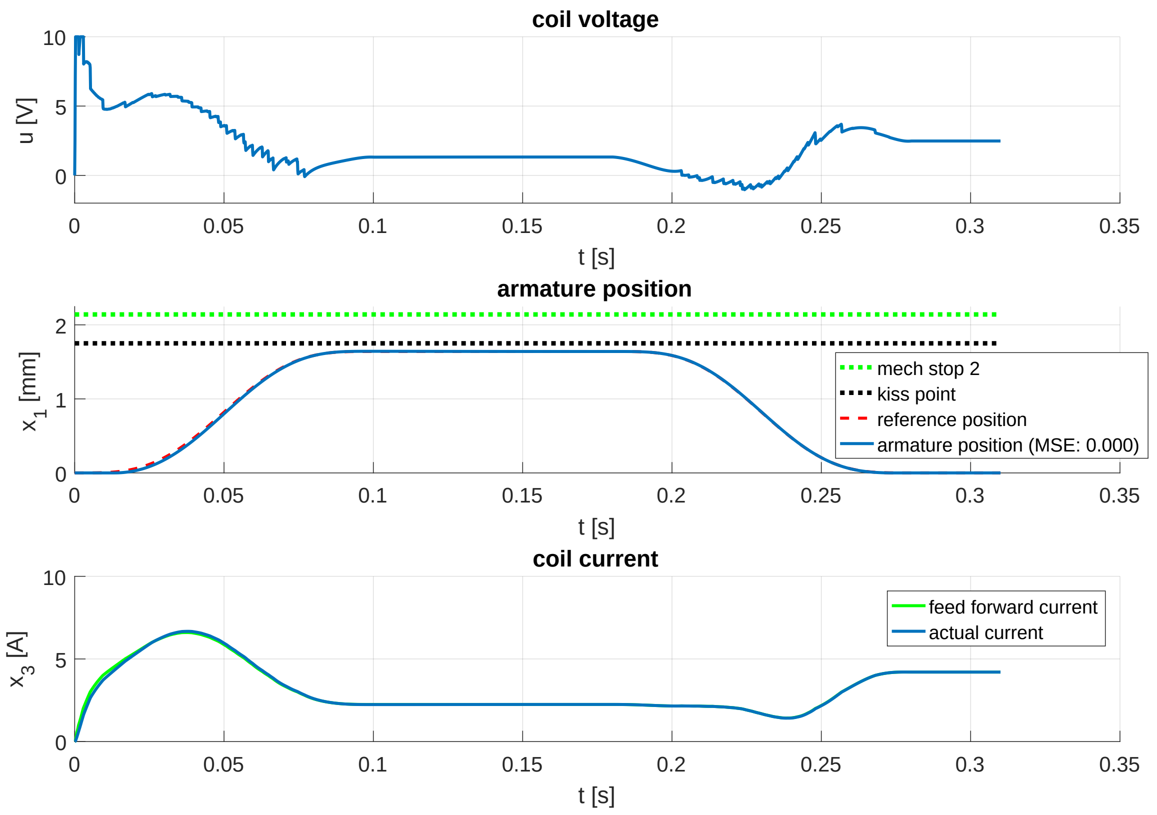

Figure C.11.: Result of the flatness-based armature position controller for moderate clutch wear. This is the nominal case for which the controller parameters were tuned.

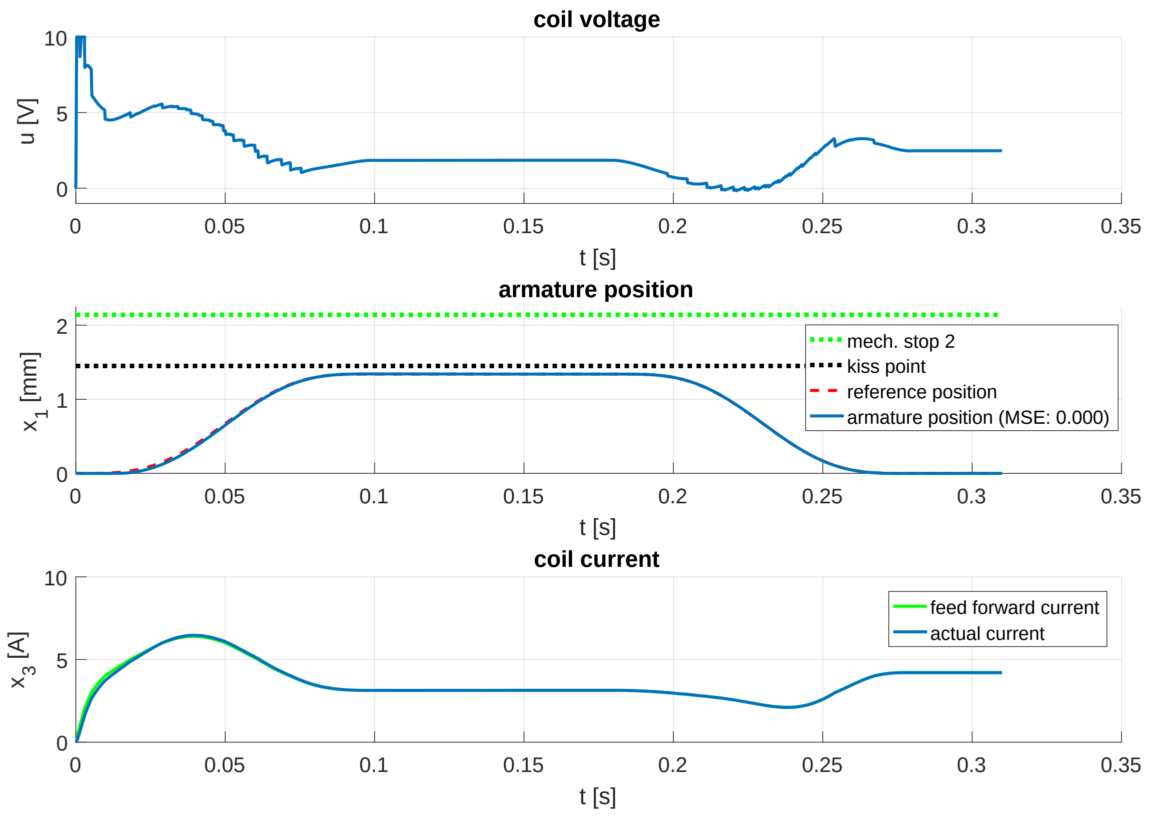

Figure C.12.: Result of the flatness-based armature position controller for low clutch wear.

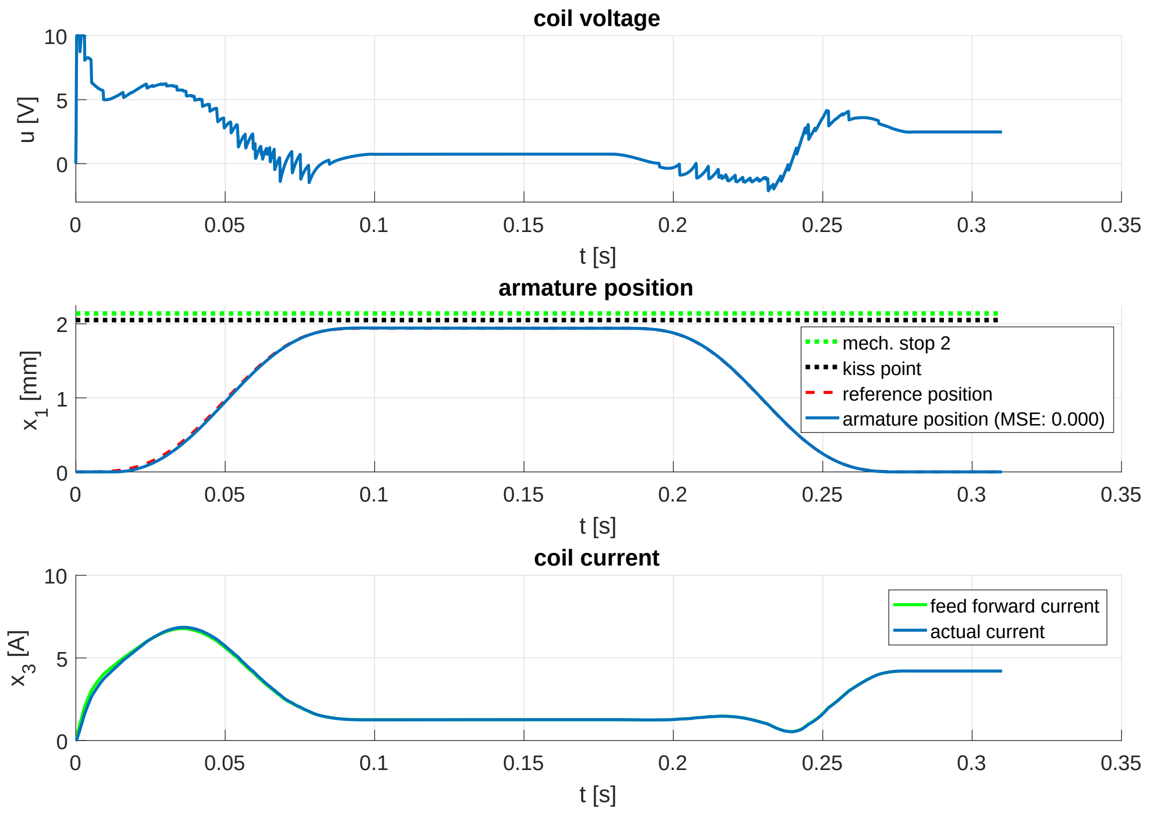

Figure C.13.: Result of the flatness-based armature position controller for high clutch wear.

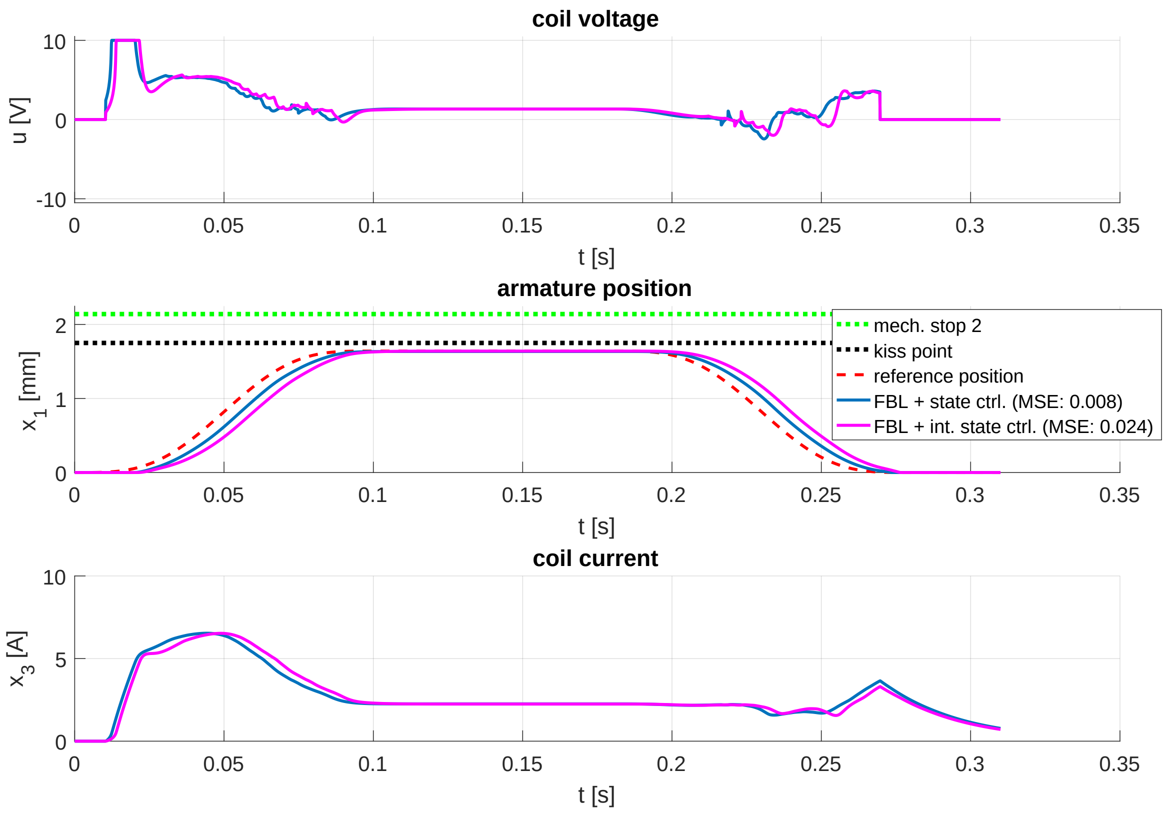

Figure C.14.: Results of the feedback linearization-based armature position controllers for moderate clutch wear. This is the nominal case for which the controller parameters were tuned.

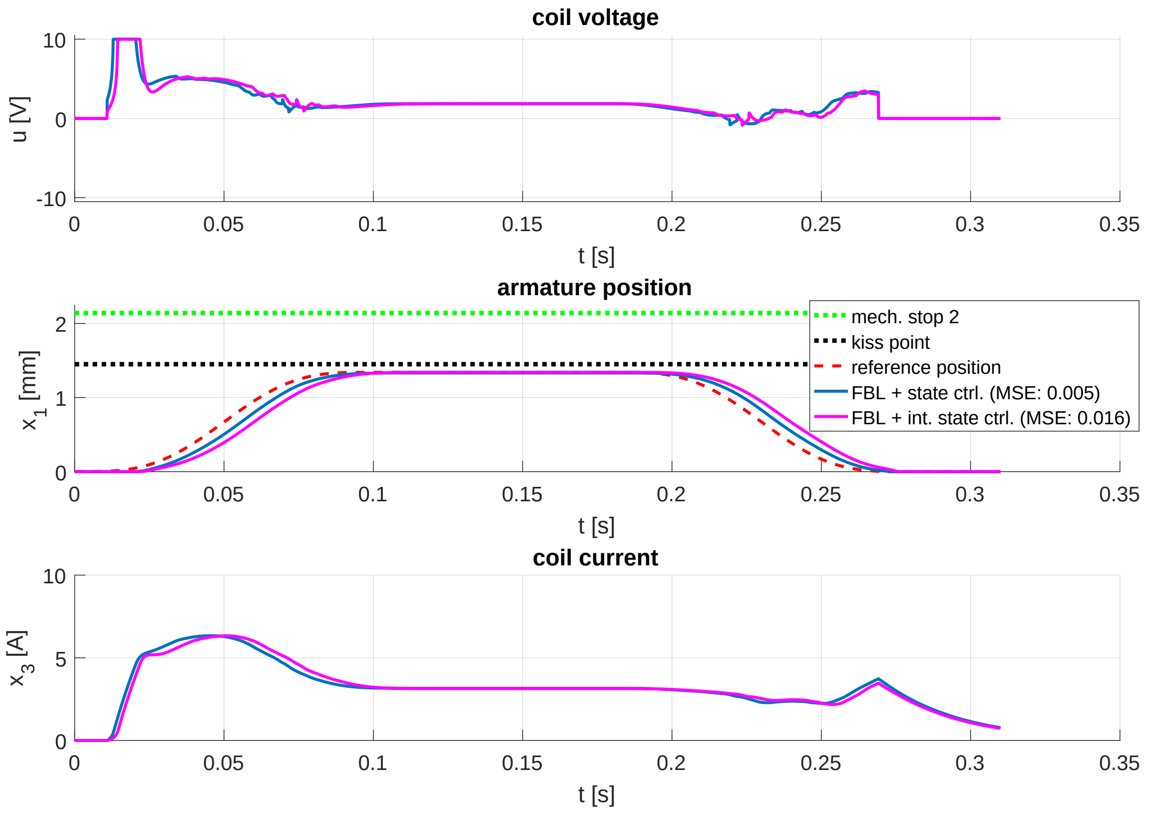

Figure C.15.: Results of the feedback linearization-based armature position controllers for low clutch wear.

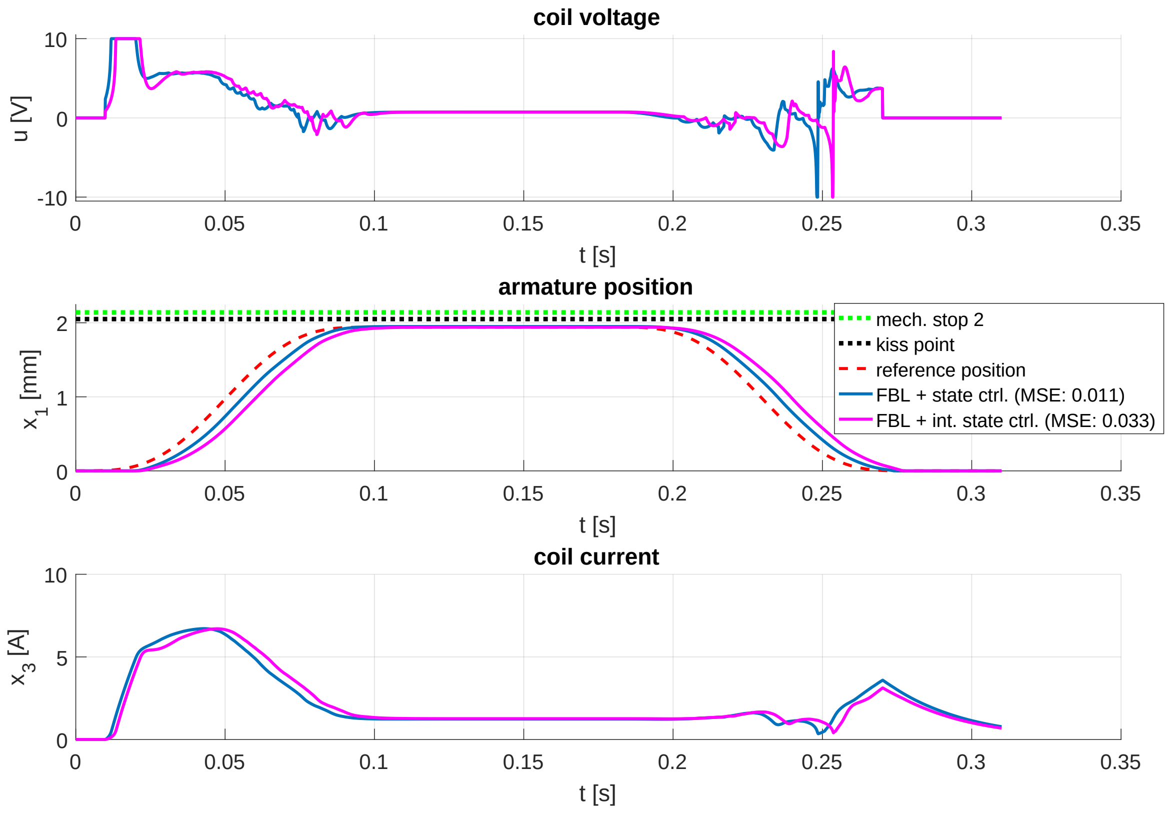

Figure C.16.: Results of the feedback linearization-based armature position controllers for high clutch wear.

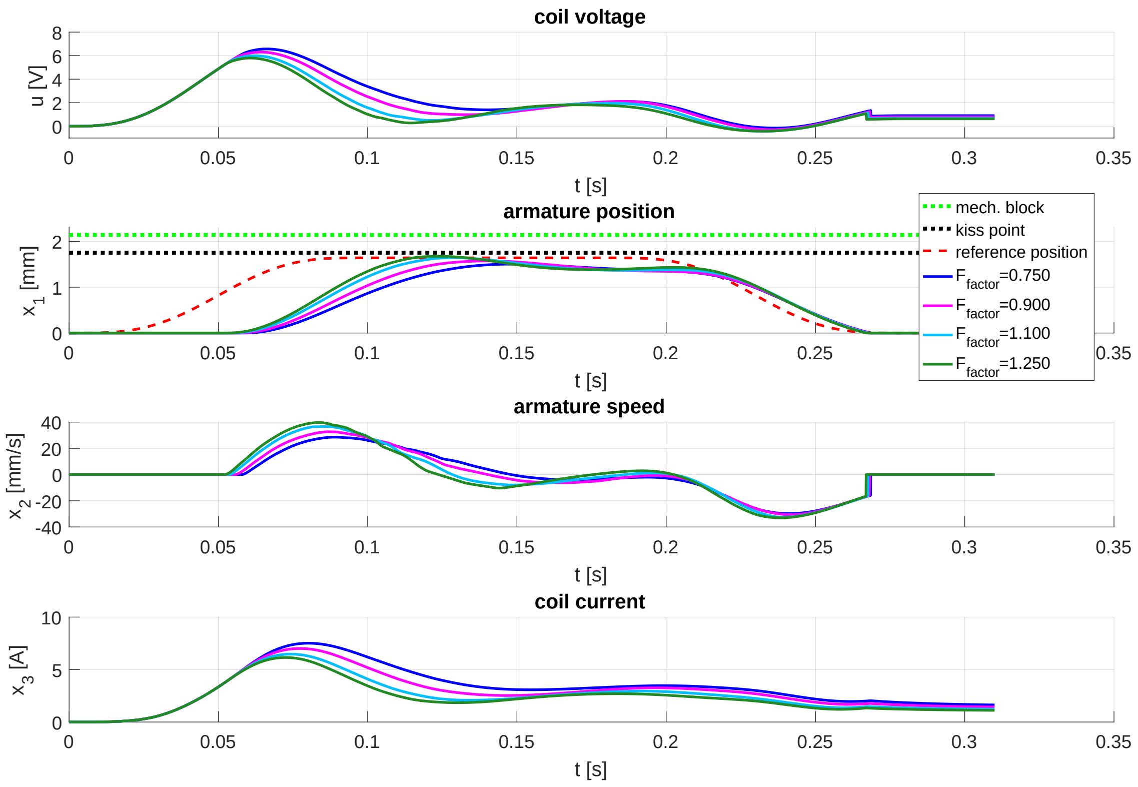

Figure C.17.: Influence of magnetic force uncertainties on the PID position controller.

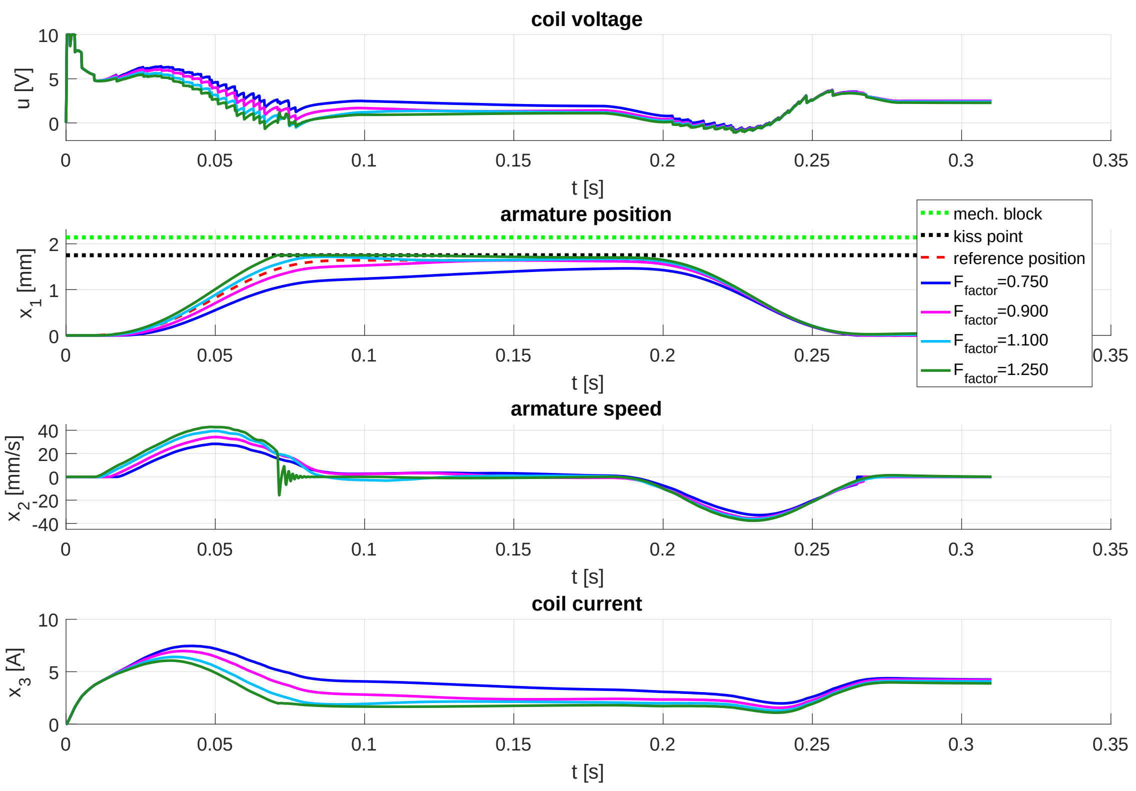

Figure C.18.: Influence of magnetic force uncertainties on the flatness-based position controller. A 25 % higher than nominal magnetic force leads to a collision at the kiss point.

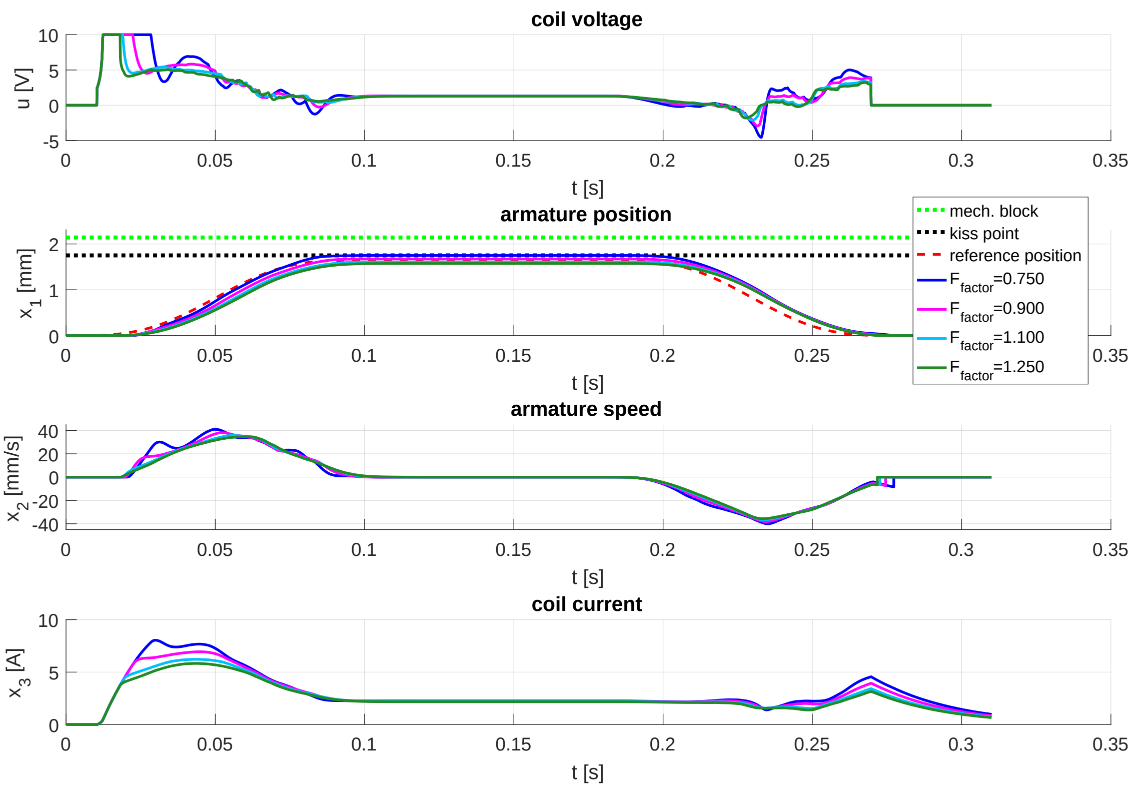

Figure C.19.: Influence of magnetic force uncertainties on the state position controller with feedback linearization.

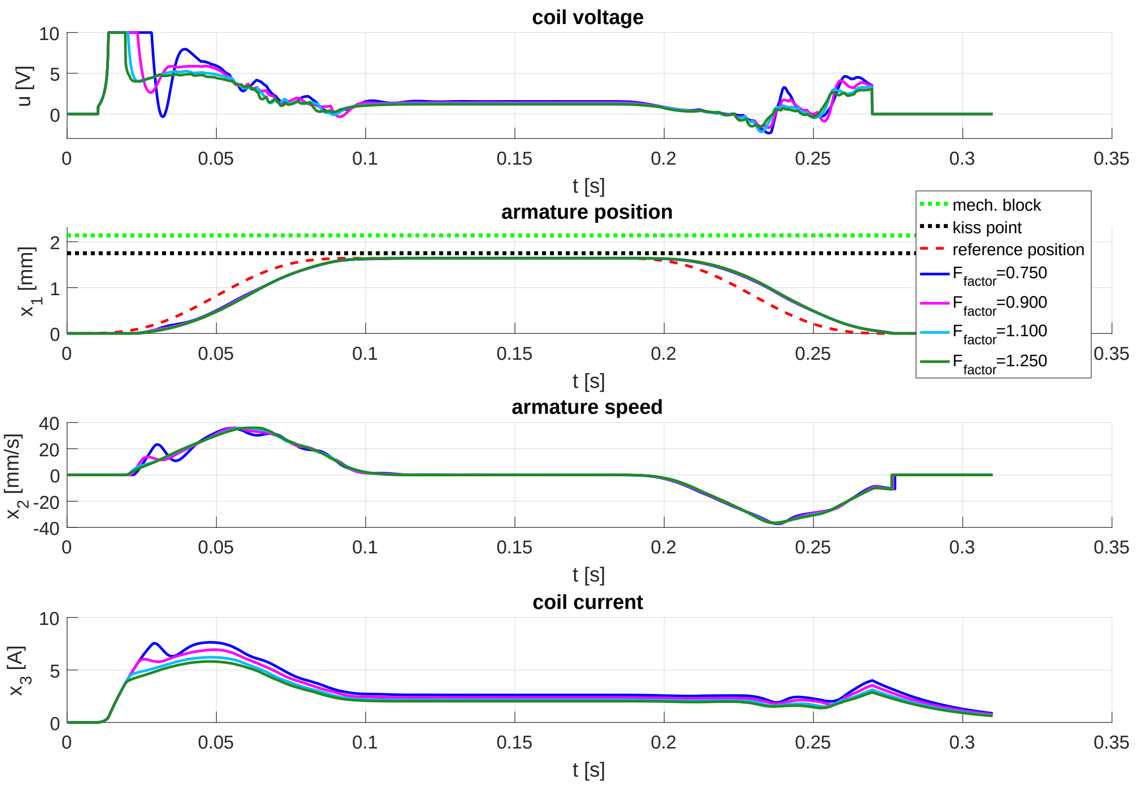

Figure C.20.: Influence of magnetic force uncertainties on the integrating state position controller with feedback linearization. There is almost no impact of the magnetic force error on the armature position.

C.4. Actuator test bed

The results shown in Section C.6 come from experiments on a prototype version of the actuator. It is equipped with comprehensive sensors which are not only used for control purposes but also for identification of the plant parameters. The armature position is measured by two eddy current sensors because the armature's rotation requires a contact-free measurement principle. The reason for using two sensors is the possibility to detect eventual tilting of the armature. During position control the average of both signals is used. Current is measured by hall-effect-based sensors integrated in the H-bridge that drives the coil. The electromagnet is not directly mounted on the clutch's housing but on a force sensor which works with strain gauge elements. This allows the measurement of the magnetic force. Furthermore, the coil temperature is determined by thermocouples inside the electromagnet. The ECU of the actuator is based on a *TMS320F2808* processor from Texas Instruments. The coil is controlled by means of pulse-width modulation (PWM) and all measurements are synchronized with the PWM-frequency of the H-bridge $f_{PWM} = 10\,kHz$. All communication and measurement data acquisition are done over Controller Area Network (CAN) with a sample time of $T_s = 1\,ms$ [53].

C.5. Parameter identification

A prerequisite for doing model-based control design is the determination of the parameters of (C.13). The values for m_1 and m_2 can be calculated from construction data. Most of the remaining parameters and characteristic curves were estimated individually by means of specific experiments conducted on the actuator test bed. The reason for not choosing a global approach to estimate all parameters in (C.13) at once is the high computational complexity. As it is shown in this section the characteristic curves require high-order polynomials for their description, which would lead to an optimization problem with more than 200 variables. Furthermore additional complexity would emerge due to the constraints that ensure physical consistency of the estimated characteristics [53].

C.5.1. Magnetic force and electric resistance

Due to the complex geometry of the magnetic circuit and the occurrence of saturation of the material physically inspired modeling of the magnetic force $F_m(x_1, x_3)$ is not considered. Instead surface fitting with a 2D-polynomial is used to approximate the measurement data. This polynomial is used to create a look-up table for the position and current range where the actuator is operating. $F_m(x_1, x_3)$ is directly measured by the force sensor. Usually, the armature would start to move as soon as the magnetic force exceeds the preload force of the return spring and therefore it would not be possible to measure the force for high currents at low positions and vice versa. For this purpose, the test bed allows the removal of the two springs and blocking of the armature motion at arbitrary positions. During one experiment the PWM duty cycle of the coil is increased in 2.5% steps from zero to 95% and reduced to zero afterwards. The measurement points which were used for identification are indicated by red markers in Figure C.21 and represent those instants of time where current and magnetic force reach their stationary values. Due to the materials elasticities the measured armature position slightly rises as the magnetic force is increased[3] despite its fixation. This experiment is repeated for 20 different positions between zero and maximum, which results in 1540 measurement points

[3]These elasticities differ from those described in Section C.1 due to the removal of the two springs.

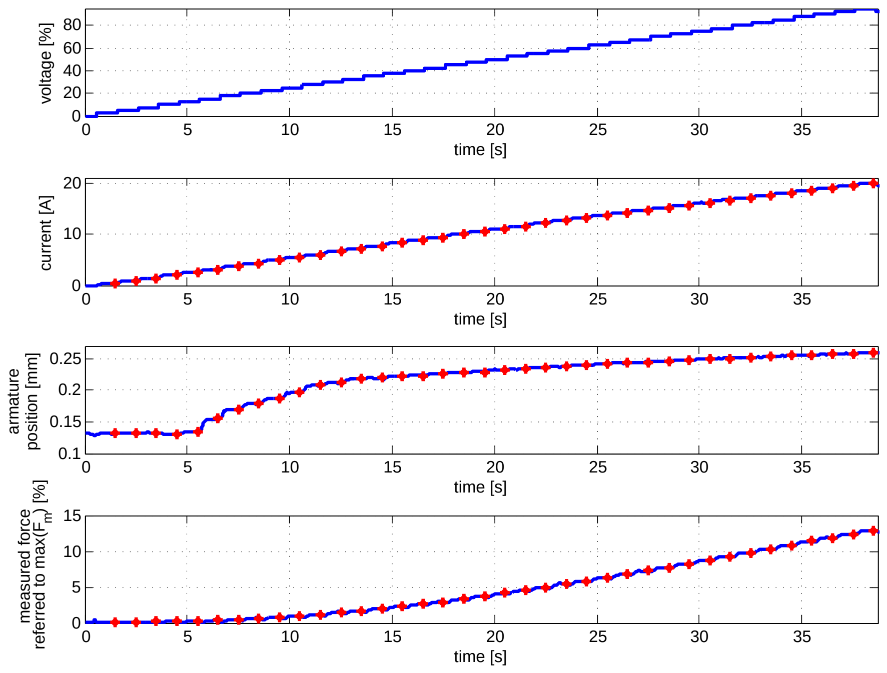

Figure C.21.: Half of one of the experiments for magnetic force determination. Red markers highlight stationary values. (© 2015 Elsevier, [53]).

altogether. Figure C.22 shows all results and the approximation of the data, that is used for look-up table generation. The ninth-degree 2D-polynomial fits the measurement data in the least squares sense, the mean squared error is 1.33% of $max(F_m)$ [53].

The actuator's electric resistance is calculated for all the stationary points that were used to determine F_m according to

$$R = \frac{u_{sup} \cdot d}{i}, \tag{C.41}$$

where u_{sup} is the measured supply voltage of the h-bridge, d is the PWM duty cycle, and i is the measured current. During the experiment the temperature of the windings increased up to about 42 °C and so the temperature dependency of the resistance could be determined. A linear approximation was done in order to get the resistance as a function of temperature. At 25 °C coil temperature the resistance is $R = 0.51\,\Omega$ [53].

C.5.2. Inductance

The identification of the inductance is based on the measurement data from the same experiments as described in Section C.5.1. But rather than using the stationary values, the transient of the current is considered. The approach works similar to the identification of a constant inductance and is based on the fact that the small voltage steps in the experiments result in small changes in current. The identification procedure consists of two steps that are carried out for each of the n_{steps} voltage steps in every experiment [53].

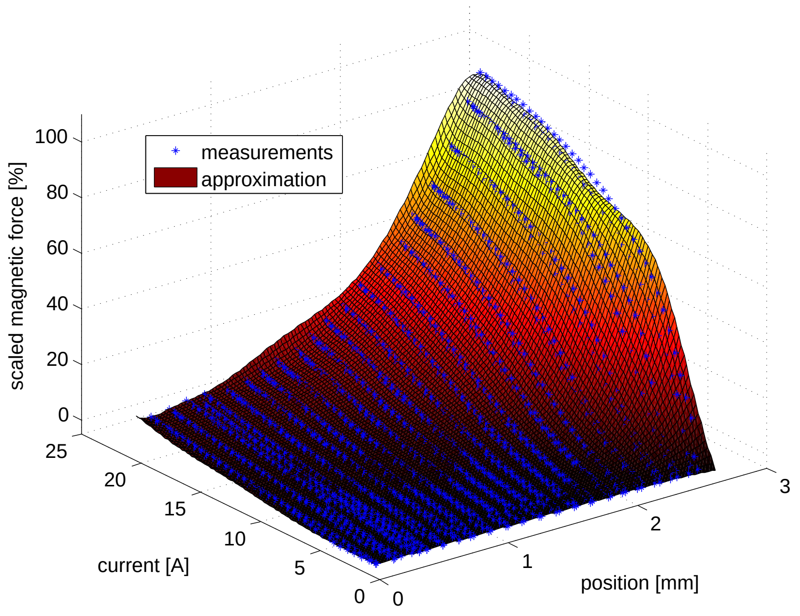

Figure C.22.: Magnetic force measurements for all experiments are shown in blue. The surface is an approximation of measurement data. (© 2015 Elsevier, [53]).

C.5.2.1. Step 1

The first step deals with identifying the principal dynamic behavior of one voltage step. Defining the instant of time where the voltage step is applied as t_{step}, one can introduce voltage and current deviations as $\Delta u(t) = u(t) - u(t_{step})$ and $\Delta x_3(t) = x_3(t) - x_3(t_{step})$ respectively. Assuming a constant armature position (the armature is fixed and the elasticities related motion is negligible during one voltage step), it follows that $x_2 = 0$ and the current dynamics (C.13c) simplify to [53]

$$\dot{\Delta x_3} = -\frac{\hat{R}}{\hat{L}}\Delta x_3 + \frac{1}{\hat{L}}\Delta u \tag{C.42}$$

where $\hat{R}$ is the identified resistance and $\hat{L}$ includes all inductance related parameters. The resistance is identified for each voltage step separately in order to be consistent in steady state. After that, (C.42) is used to compute the step response $\Delta x_{3,resp}(t)$ when $\Delta u(t)$ is applied and the error $e_{x_3}(t) = \Delta x_{3,meas}(t) - \Delta x_{3,resp}(t)$. By minimizing the sum of squared errors, one can find the optimal value [53]

$$\hat{L}^* = \arg\min_{\hat{L}} \left[\sum_k e_{x_3}^2(t_{step} + kT_s)\right]. \tag{C.43}$$

Evaluating (C.43) for every voltage step yields

$$\overline{\mathbf{L}} = \begin{bmatrix}\hat{L}_1^* & \hat{L}_2^* & \dots & \hat{L}_{n_{steps}}^*\end{bmatrix}^T. \tag{C.44}$$

C.5.2.2. Step 2

Once $\overline{\mathbf{L}}$ is determined, the influence of the inductance's partial derivative with respect to current can be examined, see (C.17). The identified value $\hat{L}^*$ corresponding to the i-th voltage step can be expressed as [53]

$$\begin{aligned}\hat{L}_i^* &= L\left(x_1, \Delta x_{3,i}(t) + x_{3,0,i}\right) \\ &\quad + \frac{\partial L\left(x_1, \Delta x_{3,i}(t) + x_{3,0,i}\right)}{\partial x_3}\left(\Delta x_{3,i}(t) + x_{3,0,i}\right)\end{aligned} \tag{C.45}$$

where $x_{3,0,i} = x_3(t_{step,i})$ is the initial current before the voltage starts to change. Due to the small increase in current, the approximation

$$\Delta x_{3,i}(t) + x_{3,0,i} \approx \underbrace{\frac{x_{3,0,i} + x_{3,0,i+1}}{2}}_{=:\hat{x}_{3,i}} \tag{C.46}$$

is reasonable. Assuming a constant inductance and partial derivative during one voltage step, (C.45) can be rewritten as

$$\hat{L}_i^* = L_{1,i} + L_{2,i}\hat{x}_{3,i} \tag{C.47}$$

with $L_{1,i} = L\left(x_1, \hat{x}_{3,i}\right)$ and $L_{2,i} = \frac{\partial L(x_1, \hat{x}_{3,i})}{\partial x_3}$. Now the estimation of inductance and its partial derivative can be formulated as minimization problem [53]

$$\min_{\mathbf{L_1},\mathbf{L_2}} \left[\left(\overline{\mathbf{L}} - \mathbf{L_1} - \hat{\mathbf{X}}_\mathbf{3}\mathbf{L_2}\right)^T \left(\overline{\mathbf{L}} - \mathbf{L_1} - \hat{\mathbf{X}}_\mathbf{3}\mathbf{L_2}\right)\right] \tag{C.48}$$

for the entire experiment with

$$\begin{aligned}\mathbf{L_1} &= \left[L_{1,1}\ L_{1,2}\ \dots\ L_{1,n_{steps}}\right]^T \\ \mathbf{L_2} &= \left[L_{2,1}\ L_{2,2}\ \dots\ L_{2,n_{steps}}\right]^T \\ \hat{\mathbf{X}}_\mathbf{3} &= \begin{bmatrix} \hat{x}_{3,1} & 0 & \dots & 0 \\ 0 & \hat{x}_{3,2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \hat{x}_{3,n_{steps}} \end{bmatrix}.\end{aligned}$$

During the optimization the following physically motivated constraints (see [68] and [70]) are considered:

- The inductance is always positive. During the first voltage steps the current is rather small and so the influence of $L_{2,i}$ can be neglected. Hence, the average of the first values in $\overline{\mathbf{L}}$ gives a good estimate on the maximum value $L_{1,max}$ [53].

$$0 < L_{1,i} \leq L_{1,max} \quad \text{with} \quad i = 1, ..., n_{steps} \tag{C.49}$$

- At a constant position the inductance decreases as the current increases and vice versa [53].

$$\begin{aligned}L_{1,i} - L_{1,i-1} &\begin{cases} \leq 0 & \text{if } \hat{x}_{3,i} > \hat{x}_{3,i-1} \\ > 0 & \text{else} \end{cases} \\ \text{with} &\quad i = 2, ..., n_{steps}\end{aligned} \tag{C.50}$$

- The partial derivative of the inductance with respect to the current is always negative [53].

$$L_{2,i} < 0 \quad \text{with} \quad i = 1, ..., n_{steps} \tag{C.51}$$

- As the current increases, the same happens with $\frac{\partial L}{\partial x_3}$ (it gets closer to zero) and vice versa [53].

$$L_{2,i} - L_{2,i-1} \begin{cases} > 0 & \text{if } \hat{x}_{3,i} > \hat{x}_{3,i-1} \\ \leq 0 & \text{else} \end{cases} \qquad \text{(C.52)}$$
$$\text{with} \quad i = 2, ..., n_{steps}$$

The optimization problem is solved numerically by means of Matlab using *fmincon(...)* with an interior-point algorithm. After repeating steps 1 and 2 for each experiment (different fixed armature positions x_1), the two data sets for $L(x_1, x_3)$ and $\frac{\partial L(x_1,x_3)}{\partial x_3}$ are approximated by fifth-degree 2D-polynomials. Those are used for generating equally spaced data points which can be stored in look-up tables. Figure C.23 shows the results for the inductance. The mean squared

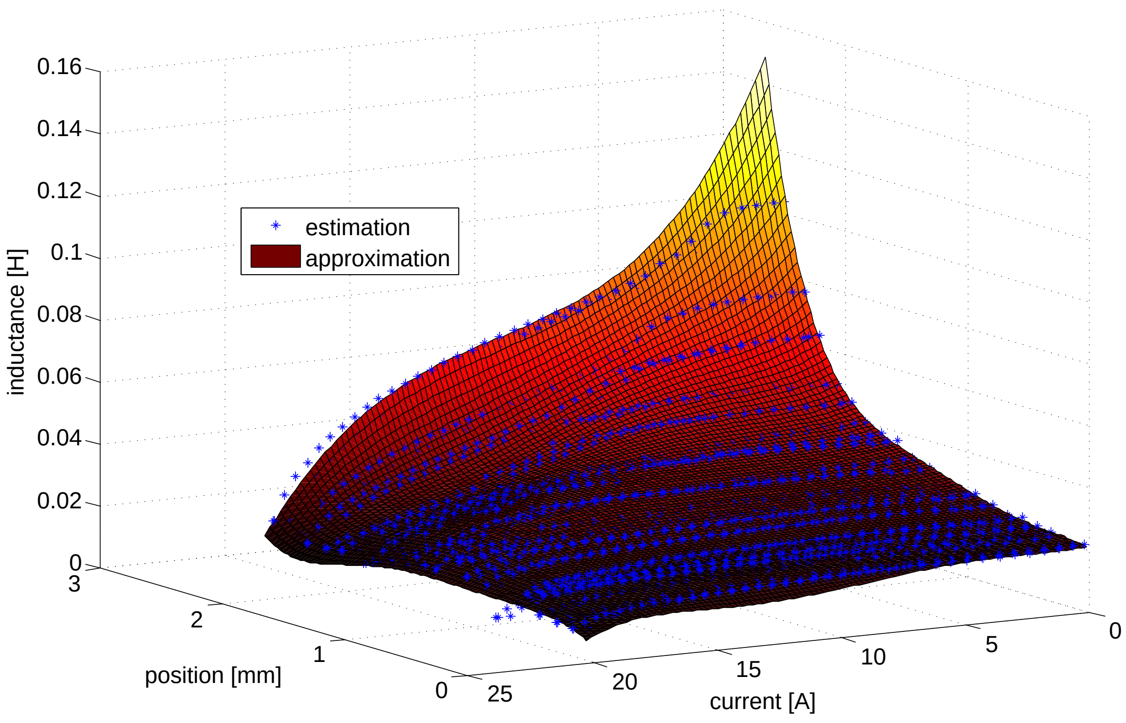

Figure C.23.: Estimated inductance for all experiments is shown in blue. The surface is a polynomial approximation of the estimated data. (© 2015 Elsevier, [53]).

error between the estimated inductance and the approximation is about $4.5 \cdot 10^{-6}$ H. An error with respect to the real inductance cannot be presented because it is not measurable directly. The remaining partial derivative with respect to position $\frac{\partial L(x_1,x_3)}{\partial x_1}$ is computed analytically from the polynomial that fits the inductance [53].

C.5.3. Spring force and elasticities

The determination of these forces is carried out indirectly by measuring the magnetic force that is necessary to keep the armature at a certain position. A PI current controller is used for moving the armature within the elasticities' region ($s_{1,kp}$ and $s_{2,max}$ in Figure C.4) and gathering the force data. Small changes in current (and magnetic force) lead to rather small changes in position because of the steeply rising spring characteristics in this region. The opposite is true in the region where only the return spring acts on the armature. If the magnetic force exceeds a certain value, the armature travels the complete distance until it reaches the kiss point. Therefore, current control is not useful for determination of the return spring force and a proportional position controller[4] was used instead. The measured position-to-force relationship is approximated by one polynomial for each region as shown in Figure C.24. The resulting functions are used to build a look-up table that combines return spring and elasticities related forces [53].

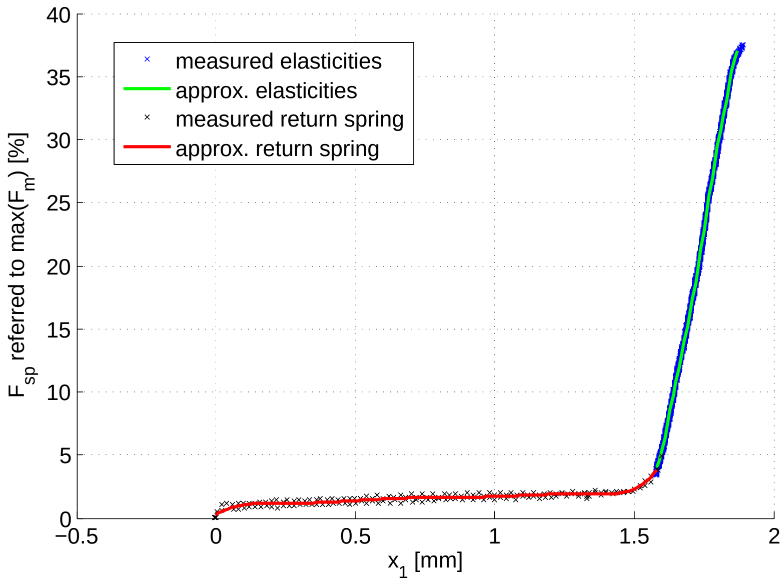

Figure C.24.: Measurements for determining the spring forces and elasticities. The polynomial approximations of the data are shown in red and green. (© 2015 Elsevier, [53]).

C.5.4. Damping coefficient

After the identification of the other parameters is completed, their values are fed into a simulation model of the plant. A voltage test signal, that causes the armature to move, is applied on the real actuator and the resulting position data are recorded. Now the damping coefficient is varied in the simulation model in order to minimize the sum of squared errors between the recorded and the simulated position signals [53].

C.6. Actuator test bed results

The first presented result compares the flatness-based controller's performance with a PID controller. The air gap between armature and electromagnet is initially adjusted to 2.66 mm resulting in a kiss point position of around 1.75 mm. The goal is to drive the armature to $x_1 = 1.5\,mm$ within $0.3\,s$ and to hold that position (see Figure C.25). That movement is much slower than required for the clutch's normal operation, but the PID controller does not work at all for faster reference signals. Compared to the PID controller, the armature movement is much smoother when using the flatness-based controller. In case of faster reference signals, the visible oscillations in the PID controller's result would worsen. Figure C.26 shows the composition of the voltage from feedforward and feedback part for the same experiment.

[4]This controller works only if the reference position changes very slowly. Thus it is not applicable for solving the control task of closing the clutch.

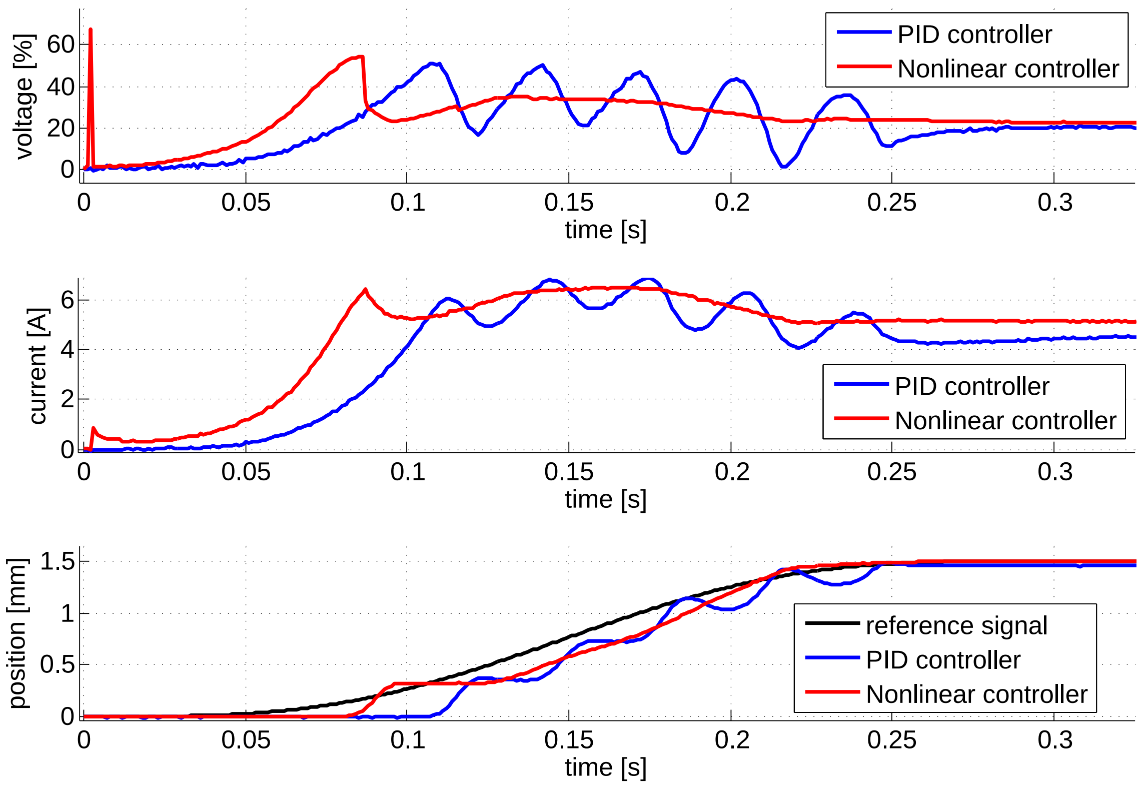

Figure C.25.: Comparison of flatness-based and PID controller. The PID controller already shows oscillations which would get worse or even unstable for faster reference signals. (© 2015 Elsevier, [53]).

Most of the control signal comes from the feedforward control which is an indication that the identified model is a proper approximation of the real actuator behavior [53]. Figure C.27 shows that the flatness-based position controller can reach the kiss point position within 0.15 s. This is not possible with the PID controller. One can clearly see the influence of the stick-slip effect on the armature movement. This phenomenon is not explicitly considered in the feedforward control. The next result in Figure C.28 shows the armature behavior under similar conditions as in the clutch's normal operating mode. In this experiment the air gap of the actuator test bed is reduced to about 2.34 mm. At first the kiss point position $1.33\,mm$ is reached in $0.15\,s$ and after that the armature approaches $1.55\,mm$ in the steeper region of the spring characteristic within $0.1\,s$, which would be necessary if the position controller is used for setting the transmitted torque of the clutch. The main reason for the jerks in armature movement while reaching the kiss point is once again the unmodeled friction (stick-slip effect).

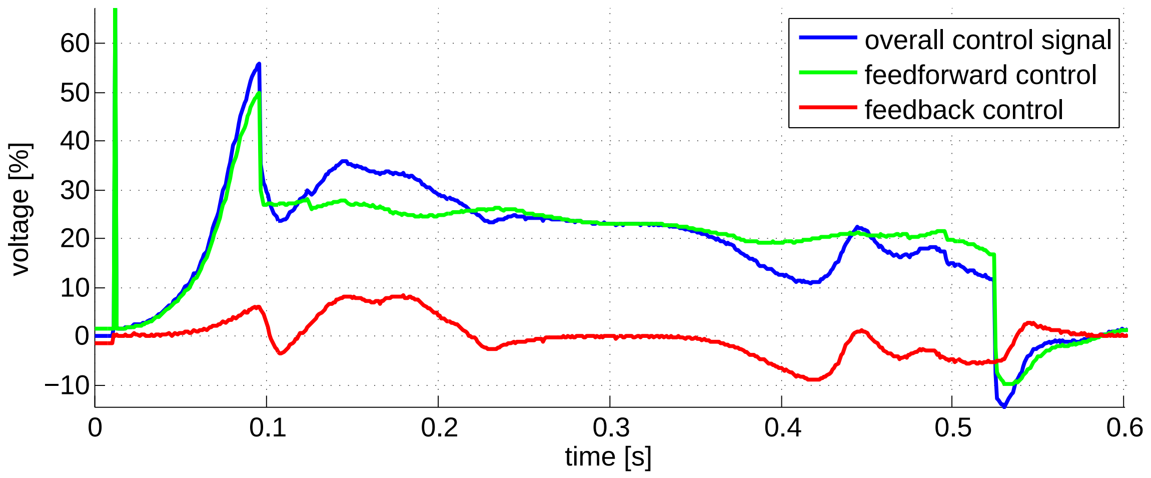

Figure C.26.: Segmentation of the controller output in feedforward and feedback part. The initial peak comes from the preload force filter. (© 2015 Elsevier, [53]).

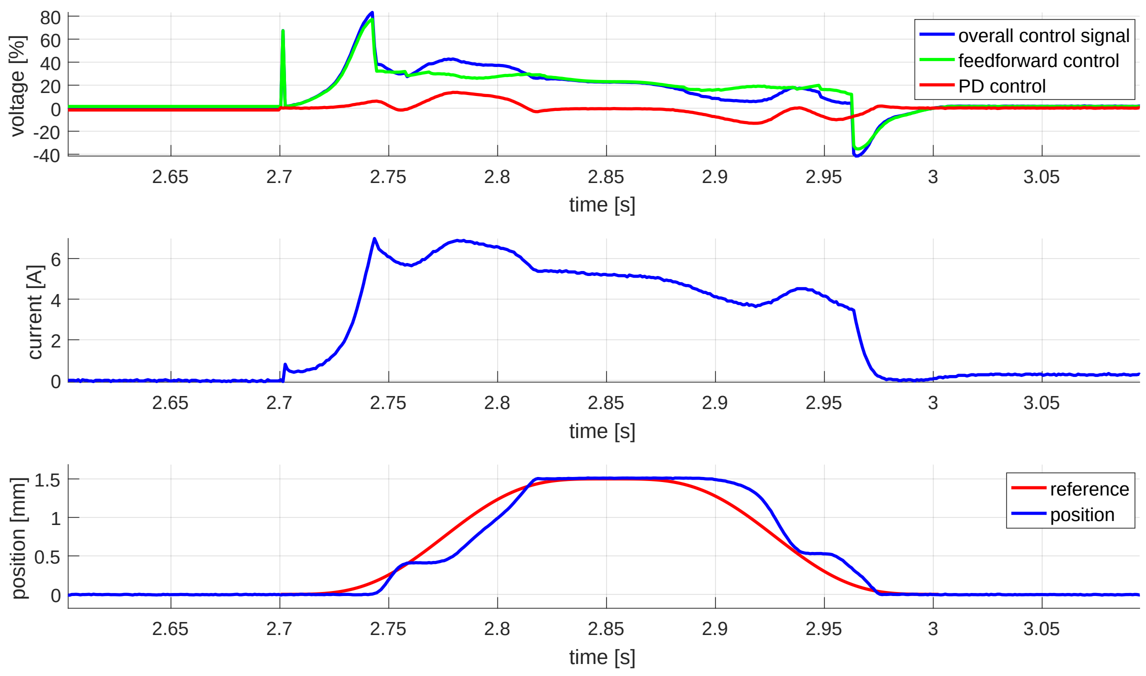

Figure C.27.: Approaching the kiss point in 0.15 s by means of the flatness-based position controller.

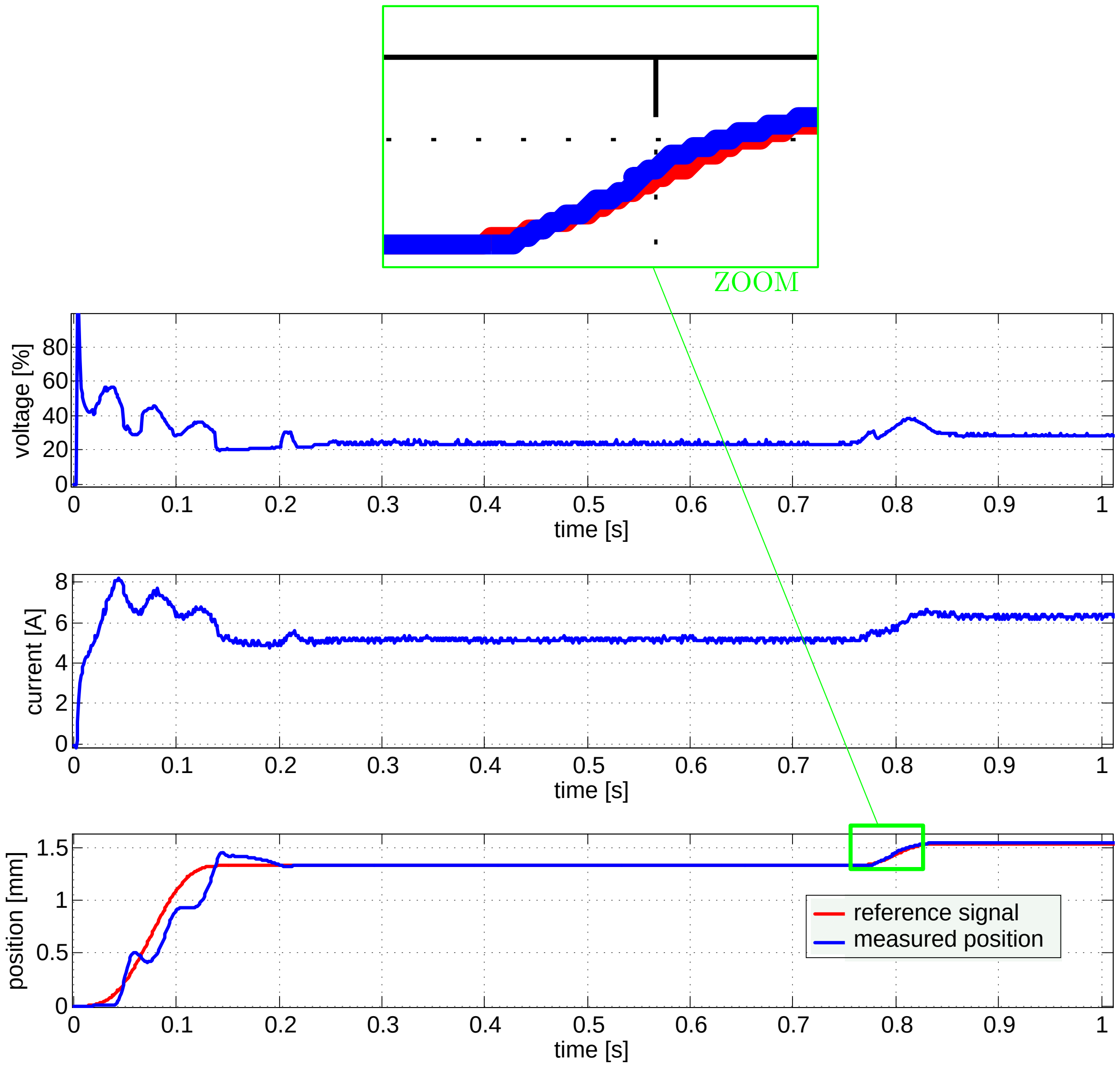

Figure C.28.: The flatness-based position controller consecutively drives the armature to two positions: The kiss point and a position within the torque setting range. (© 2015 Elsevier, [53]).

D. Detailed Hover Model

D.1. State-dependent input matrix

$\boldsymbol{B_u(x)}$ is given by (D.1) and because $|x_2| \leq \pi/4$ is guaranteed it is always finite. Note that this is the input matrix for the accelerations of the Euler angles and *not* for the angular accelerations in body frame (in this case it would be constant). This is the reason why tangent and secant occur in (D.1) just as they do in $\dot{\boldsymbol{q}} = \boldsymbol{J_b}^{-1}\boldsymbol{\omega_b}$ [58] [60].

$$\boldsymbol{B_u}(\boldsymbol{x}) = \begin{bmatrix}\boldsymbol{B_{u,12}} & \boldsymbol{B_{u,34}}\end{bmatrix} \tag{D.1a}$$

$$\boldsymbol{B_{u,12}} = \begin{bmatrix} \left(\frac{k_\psi \mathrm{co}_{x_1}}{J_z} + \frac{k_\theta \mathrm{si}_{x_1}}{J_y}\right)\mathrm{ta}_{x_2} & \frac{k_\phi}{J_x} - \frac{k_\psi \mathrm{co}_{x_1}\mathrm{ta}_{x_2}}{J_z} \\ \frac{k_\theta \mathrm{co}_{x_1}}{J_y} - \frac{k_\psi \mathrm{si}_{x_1}}{J_z} & \frac{k_\psi \mathrm{si}_{x_1}}{J_z} \\ \mathrm{se}_{x_2}\left(\frac{k_\psi \mathrm{co}_{x_1}}{J_z} + \frac{k_\theta \mathrm{si}_{x_1}}{J_y}\right) & -\frac{k_\psi \mathrm{co}_{x_1}\mathrm{se}_{x_2}}{J_z} \end{bmatrix} \tag{D.1b}$$

$$\boldsymbol{B_{u,34}} = \begin{bmatrix} \left(\frac{k_\psi \mathrm{co}_{x_1}}{J_z} - \frac{k_\theta \mathrm{si}_{x_1}}{J_y}\right)\mathrm{ta}_{x_2} & -\frac{k_\phi}{J_x} - \frac{k_\psi \mathrm{co}_{x_1}\mathrm{ta}_{x_2}}{J_z} \\ -\frac{k_\theta \mathrm{co}_{x_1}}{J_y} - \frac{k_\psi \mathrm{si}_{x_1}}{J_z} & \frac{k_\psi \mathrm{si}_{x_1}}{J_z} \\ \mathrm{se}_{x_2}\left(\frac{k_\psi \mathrm{co}_{x_1}}{J_z} - \frac{k_\theta \mathrm{si}_{x_1}}{J_y}\right) & -\frac{k_\psi \mathrm{co}_{x_1}\mathrm{se}_{x_2}}{J_z} \end{bmatrix} \tag{D.1c}$$

Furthermore, please note that the right nullspace of $\boldsymbol{B_u(x)}$ is

$$\forall \boldsymbol{x}: \ \mathcal{N}_\mathrm{r}(\boldsymbol{B_u}(\boldsymbol{x})) = \mathrm{span}\left(\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}^T\right), \tag{D.2}$$

i.e. it does not depend on the state of the hover.

D.2. Linearized model for LQR

The result of the Taylor series expansion (8.56) is computed in Mathematica [65] and evaluated for the equilibrium $\boldsymbol{x_e} = [\phi_e \ \ \theta_e \ \ 0 \ \ 0 \ \ 0 \ \ 0]^T$ resulting in

$$\boldsymbol{A} \approx \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1.3422 & 0.0158 & 0 & -0.3070 & 0.0018 & -30.2280 \\ 0.0154 & -2.7240 & 0 & 0.0035 & -0.3064 & 20.6489 \\ -0.1502 & 0.1050 & 0 & -0.0344 & 0.0118 & -201.419 \end{bmatrix} \tag{D.3a}$$

$$\boldsymbol{B_u} \approx \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.0057u_0 & 0.2966u_0 & 0.0101u_0 & -0.3124u_0 \\ 0.2428u_0 & -0.0031u_0 & -0.2366u_0 & -0.0031u_0 \\ 0.0381u_0 & -0.0527u_0 & 0.0672u_0 & -0.0527u_0 \end{bmatrix} \tag{D.3b}$$

$$\boldsymbol{C} = \begin{bmatrix}\boldsymbol{I_3} & \boldsymbol{0}\end{bmatrix}. \tag{D.3c}$$

D.3. Full nonlinear model

The state equations describing the hover's dynamics have been derived by means of the Lagrange formalism read as follows:

$$\dot{\boldsymbol{x}} = \begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\\ \dot{x}_5\\ \dot{x}_6\end{bmatrix} = \boldsymbol{f_1}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix}x_4\\ x_5\\ x_6\\ f_4(\boldsymbol{x},\boldsymbol{u})\\ f_5(\boldsymbol{x},\boldsymbol{u})\\ f_6(\boldsymbol{x},\boldsymbol{u})\end{bmatrix} \tag{D.4}$$

$$\begin{aligned}
f_4(\boldsymbol{x},\boldsymbol{u}) = -\frac{1}{4J_xJ_yJ_z}\mathrm{se}_{x_2}^2 \Big(&-2\mathrm{si}_{x_2}J_x\left(-J_y - J_z + \mathrm{co}_{2x_1}(J_z - J_y)\right)\left(\frac{1}{2}\mathrm{si}_{x_2}\left(\mathrm{si}_{2x_1}(J_z - J_y)x_5^2 + 2k_\phi\left(u_2^2 - u_4^2\right)\right) - \right.\\
&\mathrm{co}_{x_2}\Big(\mathrm{co}_{x_2}\mathrm{si}_{2x_1}(J_z - J_y)x_4x_6 + x_5\left(\left(J_x + \mathrm{co}_{2x_1}(J_z - J_y)\right)x_4 + \mathrm{si}_{x_2}\left(-2J_x + J_y + J_z + \mathrm{co}_{2x_1}(J_z - J_y)\right)x_6\right) + \mathrm{si}_{x_1}k_\theta\left(u_1^2 - u_3^2\right) + \mathrm{co}_{x_1}k_\psi\left(u_1^2 - u_2^2 + u_3^2 - u_4^2\right)\Big)\\
&\left.+x_6\left(e^{-|x_6|d_{\psi,0}}d_{\psi,1} + d_{\psi,2}\right)\Big)\right) + 4\left(\frac{1}{2}\left(J_y + \mathrm{co}_{2x_1}(J_y - J_z) + J_z\right)\left(\left(J_z\mathrm{co}_{x_1}^2 + \mathrm{si}_{x_1}^2J_y\right)\mathrm{co}_{x_2}^2 + \mathrm{si}_{x_2}^2J_x\right) - \mathrm{co}_{x_1}^2\mathrm{co}_{x_2}^2\mathrm{si}_{x_1}^2(J_y - J_z)^2\right)\cdot\\
&\left(\mathrm{co}_{x_1}\mathrm{si}_{x_1}(J_y - J_z)x_5^2 - k_\phi\left(u_2^2 - u_4^2\right) + e^{-|x_4|d_{\phi,0}}x_4d_{\phi,1} - \mathrm{co}_{x_2}\left(x_6\left(\left(J_x + \mathrm{co}_{2x_1}(J_y - J_z)\right)x_5 + \mathrm{co}_{x_1}\mathrm{co}_{x_2}\mathrm{si}_{x_1}(J_y - J_z)x_6\right) - \mathrm{se}_{\phi_e}\mathrm{si}_{x_1-\phi_e}m_{g,z}\right)\right) +\\
&\mathrm{co}_{x_1}\mathrm{co}_{x_2}\mathrm{si}_{x_1}\mathrm{si}_{x_2}J_x(J_z - J_y)\Big(-\mathrm{si}_{2x_2}\left(2J_x - J_y + \mathrm{co}_{2x_1}(J_y - J_z) - J_z\right)x_6^2 + 4\mathrm{co}_{x_2}\left(J_x + \mathrm{co}_{2x_1}(J_y - J_z)\right)x_4x_6 - 4\mathrm{co}_{x_1}k_\theta\left(u_1^2 - u_3^2\right) + 4\mathrm{si}_{x_1}k_\psi\left(u_1^2 - u_2^2 + u_3^2 - u_4^2\right) -\\
&4x_5\left(\mathrm{si}_{2x_1}(J_y - J_z)x_4 - e^{-|x_5|d_{\theta,0}}d_{\theta,1}\right) + 4m_{g,z}\left(\mathrm{si}_{x_2}\left(\mathrm{co}_{x_1} + \mathrm{si}_{x_1}\mathrm{ta}_{\phi_e}\right) - \mathrm{co}_{x_2}\mathrm{se}_{\phi_e}\mathrm{ta}_{\theta_e}\right)\Big)\Big)
\end{aligned} \tag{D.5}$$

$$\begin{aligned}
f_5(\boldsymbol{x},\boldsymbol{u}) = -\frac{1}{4J_yJ_z}\mathrm{se}_{x_2}\Big(&4\mathrm{co}_{x_1}\mathrm{si}_{x_1}(J_z - J_y)\left(\frac{1}{2}\mathrm{si}_{x_2}\left(\mathrm{si}_{2x_1}(J_z - J_y)x_5^2 + 2k_\phi\left(u_2^2 - u_4^2\right)\right) - \right.\\
&\mathrm{co}_{x_2}\Big(\mathrm{co}_{x_2}\mathrm{si}_{2x_1}(J_z - J_y)x_4x_6 + x_5\left(\left(J_x + \mathrm{co}_{2x_1}(J_z - J_y)\right)x_4 + \mathrm{si}_{x_2}\left(-2J_x + J_y + J_z + \mathrm{co}_{2x_1}(J_z - J_y)\right)x_6\right) + \mathrm{si}_{x_1}k_\theta\left(u_1^2 - u_3^2\right) + \mathrm{co}_{x_1}k_\psi\left(u_1^2 - u_2^2 + u_3^2 - u_4^2\right)\Big) +\\
&\left.x_6\left(e^{-|x_6|d_{\psi,0}}d_{\psi,1} + d_{\psi,2}\right)\Big)\right) + 4\mathrm{co}_{x_1}\mathrm{si}_{x_1}\mathrm{si}_{x_2}(J_z - J_y)\Big(\mathrm{co}_{x_1}\mathrm{si}_{x_1}(J_y - J_z)x_5^2 - k_\phi\left(u_2^2 - u_4^2\right) + e^{-|x_4|d_{\phi,0}}x_4d_{\phi,1} -\\
&\mathrm{co}_{x_2}\left(x_6\left(\left(J_x + \mathrm{co}_{2x_1}(J_y - J_z)\right)x_5 + \mathrm{co}_{x_1}\mathrm{co}_{x_2}\mathrm{si}_{x_1}(J_y - J_z)x_6\right) - \mathrm{se}_{\phi_e}\mathrm{si}_{x_1-\phi_e}m_{g,z}\right)\Big) + \mathrm{co}_{x_2}\left(J_z\mathrm{co}_{x_1}^2 + \mathrm{si}_{x_1}^2J_y\right)\Big(-\mathrm{si}_{2x_2}\left(2J_x - J_y + \mathrm{co}_{2x_1}(J_y - J_z) - J_z\right)x_6^2 +\\
&4\mathrm{co}_{x_2}\left(J_x + \mathrm{co}_{2x_1}(J_y - J_z)\right)x_4x_6 - 4\mathrm{co}_{x_1}k_\theta\left(u_1^2 - u_3^2\right) + 4\mathrm{si}_{x_1}k_\psi\left(u_1^2 - u_2^2 + u_3^2 - u_4^2\right) -\\
&4x_5\left(\mathrm{si}_{2x_1}(J_y - J_z)x_4 - e^{-|x_5|d_{\theta,0}}d_{\theta,1}\right) + 4m_{g,z}\left(\mathrm{si}_{x_2}\left(\mathrm{co}_{x_1} + \mathrm{si}_{x_1}\mathrm{ta}_{\phi_e}\right) - \mathrm{co}_{x_2}\mathrm{se}_{\phi_e}\mathrm{ta}_{\theta_e}\right)\Big)\Big)
\end{aligned} \tag{D.6}$$

$$\begin{aligned}
f_6(\boldsymbol{x},\boldsymbol{u}) = -\frac{1}{4J_yJ_z}\mathrm{se}_{x_2}^2\Big(&2\left(J_y + \mathrm{co}_{2x_1}(J_y - J_z) + J_z\right)\left(\frac{1}{2}\mathrm{si}_{x_2}\left(\mathrm{si}_{2x_1}(J_z - J_y)x_5^2 + 2k_\phi\left(u_2^2 - u_4^2\right)\right) - \right.\\
&\mathrm{co}_{x_2}\Big(\mathrm{co}_{x_2}\mathrm{si}_{2x_1}(J_z - J_y)x_4x_6 + x_5\left(\left(J_x + \mathrm{co}_{2x_1}(J_z - J_y)\right)x_4 + \mathrm{si}_{x_2}\left(-2J_x + J_y + J_z + \mathrm{co}_{2x_1}(J_z - J_y)\right)x_6\right) + \mathrm{si}_{x_1}k_\theta\left(u_1^2 - u_3^2\right) + \mathrm{co}_{x_1}k_\psi\left(u_1^2 - u_2^2 + u_3^2 - u_4^2\right)\Big) +\\
&\left.x_6\left(e^{-|x_6|d_{\psi,0}}d_{\psi,1} + d_{\psi,2}\right)\Big)\right) - 2\mathrm{si}_{x_2}\left(-J_y - J_z + \mathrm{co}_{2x_1}(J_z - J_y)\right)\Big(\mathrm{co}_{x_1}\mathrm{si}_{x_1}(J_y - J_z)x_5^2 - k_\phi\left(u_2^2 - u_4^2\right) + e^{-|x_4|d_{\phi,0}}x_4d_{\phi,1} -\\
&\mathrm{co}_{x_2}\left(x_6\left(\left(J_x + \mathrm{co}_{2x_1}(J_y - J_z)\right)x_5 + \mathrm{co}_{x_1}\mathrm{co}_{x_2}\mathrm{si}_{x_1}(J_y - J_z)x_6\right) - \mathrm{se}_{\phi_e}\mathrm{si}_{x_1-\phi_e}m_{g,z}\right)\Big) + \mathrm{co}_{x_1}\mathrm{co}_{x_2}\mathrm{si}_{x_1}(J_z - J_y)\Big(-\mathrm{si}_{2x_2}\left(2J_x - J_y + \mathrm{co}_{2x_1}(J_y - J_z) - J_z\right)x_6^2 +\\
&4\mathrm{co}_{x_2}\left(J_x + \mathrm{co}_{2x_1}(J_y - J_z)\right)x_4x_6 - 4\mathrm{co}_{x_1}k_\theta\left(u_1^2 - u_3^2\right) + 4\mathrm{si}_{x_1}k_\psi\left(u_1^2 - u_2^2 + u_3^2 - u_4^2\right) -\\
&4x_5\left(\mathrm{si}_{2x_1}(J_y - J_z)x_4 - e^{-|x_5|d_{\theta,0}}d_{\theta,1}\right) + 4m_{g,z}\left(\mathrm{si}_{x_2}\left(\mathrm{co}_{x_1} + \mathrm{si}_{x_1}\mathrm{ta}_{\phi_e}\right) - \mathrm{co}_{x_2}\mathrm{se}_{\phi_e}\mathrm{ta}_{\theta_e}\right)\Big)\Big)
\end{aligned} \tag{D.7}$$

Bibliography

[1] T. A. Johansen and T. I. Fossen, “Control allocation - a survey”, *Automatica*, 49, no., pp. 1087–1103, 5 May 2013.

[2] L. Zaccarian, “Dynamic allocation for input redundant control systems”, *Automatica*, 45, no., pp. 1431–1438, 6 Mar. 2009.

[3] W. Durham, “Constrained control allocation”, *Journal of Guidance, Control, and Dynamics*, 17, no., pp. 717–725, 2 Jul. 1993.

[4] K. A. Bordignon, “Constrained control allocation for systems with redundant control effectors”, PhD thesis, Virginia Polytechnic Institute and State University, 1996.

[5] J. B. Davidson, F. J. Lallman, and T. Bundick, “Integrated reconfigurable control allocation”, in *Proc. of 2001 AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal, Canada, Aug. 2001, pp. 1–11.

[6] O. Härkegard, “Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation”, in *Proc. of the 41st IEEE Conference on Decision and Control*, Las Vegas, USA, Dec. 2002, pp. 1295–1300.

[7] ——, “Backstepping and control allocation with applications to flight control”, PhD thesis, Department of Electrical Engineering Linköoping University, 2003.

[8] M. W. Oppenheimer, D. B. Doman, and M. A. Bolender, “Control allocation for overactuated systems”, in *Proc. of 2006 14th Mediterranean Conference on Control and Automation (MED’06)*, Ancona, Italy, Jun. 2006, pp. 1–6.

[9] H. Alwi and C. Edwards, “Sliding mode ftc with on–line control allocation”, in *Proc. of 49th IEEE Conference on Decision and Control*, San Diego, USA, Dec. 2006, pp. 5579–5584.

[10] T. A. Johansen, T. P. Fuglseth, P. Tondel, and T. I. Fossen, “Optimal constrained control allocation in marine surface vessels with rudders”, *Control Engineering Practice*, 16, no., pp. 457–464, 2008.

[11] T. I. Fossen, T. A. Johansen, and T. Perez, “A survey of control allocation methods for underwater vehicles”, in *Underwater Vehicles*, InTech, 2009.

[12] R. Skjetne and Ø. Kjerstad, “Recursive nullspace-based control allocation with strict prioritization for marine craft”, in *Proc. of 9th IFAC Conference on Control Applications in Marine Systems*, Osaka, Japan, Sep. 2013, pp. 49–54.

[13] S. Cordiner, S. Galeani, F. Mecocci, V. Mulone, G. Perantoni, and L. Zaccarian, “Dynamic input allocation of torque references for parallel hev”, in *Proc. of 49th IEEE Conference on Decision and Control*, Atlanta, USA, Dec. 2010, pp. 4920–4925.

[14] Y. Chen and J. Wang, “Fast and global optimal energy-efficient control allocation with applications to over-actuated electric ground vehicles”, *IEEE Transactions on Control Systems Technology*, 20, no., pp. 1202–1211, 5 Sep. 2012.

[15] J. Brembeck and P. Ritzer, "Energy optimal control of an over actuated robotic electric vehicle using enhanced control allocation approaches", in *Proc. of 2012 Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, Jun. 2012, pp. 322–327.

[16] J. Krüger, A. Pruckner, and C. Knobel, "Control allocation for road vehicles - a system-independent approach for integrated vehicle dynamics control", in *Proc. of 2010 Aachen Colloquium*, Aachen, Germany, Oct. 2010, p. 13.

[17] A. Sinigaglia, K. Tagesson, P. Falcone, and B. Jacobson, "Coordination of motion actuators in heavy vehicles using model predictive control allocation", in *Proc. of 2016 IEEE Intelligent Vehicles Symposium*, Gothenburg, Sweden, Jun. 2016, pp. 590–596.

[18] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*, 1st. Dover Publications, 2007.

[19] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control – Theory and Design*, 5th. Nob Hill Publishing LLC, 2015.

[20] P. Pisu and G. Rizzoni, "A comparative study of supervisory control strategies for hybrid electric vehicles", *IEEE Transactions on Control Systems Technology*, 15, no., pp. 506–518, May 2007.

[21] G. Rizzoni and S. Onori, "Energy management of hybrid electric vehicles: 15 years of development at the ohio state university", *Oil and Gas Science and Technology*, 70, no., pp. 41–54, Jan. 2015.

[22] A. Serrarens, M. Dassen, and M. Steinbuch, "Simulation and control of an automotive dry clutch", in *Proc. of 2004 IEEE American Control Conference*, Boston, USA, Jun. 2004, pp. 4078–4083.

[23] D. B. Doman and M. W. Oppenheimer, "Improving control allocation accuracy for nonlinear aircraft dynamics", in *Proc. of 2002 AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, USA, Aug. 2002, pp. 1–9.

[24] R. A. Horn and C. R. Johnson, *Matrix Analysis*, First. New York: Cambridge University Press, 2006.

[25] M. Kirchengast, M. Steinberger, and M. Horn, "A new method to compute generalized inverses for control allocation", in *Proc. of the 55th IEEE Conference on Decision and Control*, Las Vegas, USA, Dec. 2016, pp. 5328–5334.

[26] T. A. Johansen, "Optimizing nonlinear control allocation", in *Proc. of 43th IEEE Conference on Decision and Control*, Atlantis, Paradise Island, Bahamas, Dec. 2004, pp. 3435–3440.

[27] A. Ben-Israel and T. N. Greville, *Generalized Inverses: Theory and Applications*, 2nd, ser. CMS Series in Mathematics. Springer, 2003.

[28] W. Ford, *Numerical Linear Algebra with Application*, First. Elsevier Academic Press, 2015.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*, Seventh. New York: Cambridge University Press, 2009.

[30] M. Kirchengast, M. Steinberger, and M. Horn, "Input matrix factorizations for constrained control allocation", *IEEE Transactions on Automatic Control*, 63, no., pp. 1163–1170, 4 Apr. 2018, early access: 2017-08-14.

[31] M. Bodson, “Evaluation of optimization methods for control allocation”, *AIAA Journal of Guidance, Control, and Dynamics*, 25, no., pp. 703–711, 4 Jul. 2002.

[32] J. Nocedal and S. J. Wright, *Numerical Optimization*, Second. Springer, 2006.

[33] M. Bodson and S. A. Frost, “Control allocation with load balancing”, in *Proc. of 2009 AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, Aug. 2009, p. 13.

[34] J. J. Burken, P. Lu, Z. Wu, and C. Bahm, “Two reconfigurable fligh-control design methods: Robust servomechanism and control allocation”, *AIAA Journal of Guidance, Control, and Dynamics*, 24, no., pp. 482–493, 3 Jun. 2001.

[35] J. A. M. Petersen and M. Bodson, “Constrained quadratic programming techniques for control allocation”, *IEEE Transactions on Control Systems Technology*, 14, no., pp. 91–98, 1 Jan. 2006.

[36] D. A. Pierre, *Optimization Theory with Applications*, 1st (reprint). Dover Publications, 1969/1986.

[37] J. M. Maciejowski, *Predictive Control with Constraints*, 1st. Pearson Education Limited, 2002.

[38] S. D. Cairano, A. Bemporad, I. Kolmanovsky, and D. Hrovat, “Model predictive control of magnetic automotive actuators”, in *Proc. IEEE 2007 American Control Conference*, New York City, USA, Jul. 2007, pp. 5082–5087.

[39] O. Härkegard and S. T. Glad, “Resolving actuator redundancy - optimal control vs. control allocation”, *Automatica*, 41, no., pp. 137–144, 1 Jan. 2005.

[40] I. C. F. Ipsen, *Numerical Matrix Analysis*, 1st. Society for Industrial and Applied Mathematics, 2009.

[41] D. Sommerville, *An introduction to the geometry of n dimensions*, 1st. Methuen and Co. Ltd., 1929.

[42] E. L. Allgower and P. H. Schmidt, “Computing volumes of polyhedra”, *Mathematics of Computation*, 46, no., pp. 171–174, 173 Jan. 1986.

[43] M. Henk, J. Richter-Gebert, and G. M. Ziegler, “Basic properties of convex polytopes”, in *Handbook of Discrete and Computational Geometry*, J. E. Goodman and J. O’Rourke, Eds., 2nd, Chapman and Hall/CRC, 2004.

[44] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, “The quickhull algorithm for convex hulls”, *ACM Transactions on Mathematical Software*, 22, no., pp. 469–483, 4 Dec. 1996.

[45] A. L. Peressini, F. E. Sullivan, and J. J. J. Uhl, *The Mathematics of Nonlinear Programming*, First. Springer, 1988.

[46] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*, Third, ser. Wiley-Interscience Series in Discrete Mathematics and Optimization. New Jersey: Wiley, 2008.

[47] M. Ben-Daya and K. S. Al-Sultan, “A new penalty function algorithm for convex quadratic programming”, *European Journal of Operational Research*, 101, no., pp. 155–163, 1 Aug. 1997.

[48] Intel Corporation, *Intel® Math Kernel Library (Intel® MKL) 2018*, Sep. 2017. [Online]. Available: `https://software.intel.com/en-us/mkl-developer-reference-c`.

[49] O. Härkegard, *Quadratic programming control allocation toolbox for matlab (QCAT)*, Aug. 2004. [Online]. Available: `http://research.harkegard.se/qcat/intro.html`.

[50] Gurobi Optimization, Inc., *Gurobi optimizer reference manual*, 2016. [Online]. Available: `http://www.gurobi.com`.

[51] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, “qpOASES: A parametric active-set algorithm for quadratic programming”, *Mathematical Programming Computation*, 6, no., pp. 327–363, 2014.

[52] M. Kirchengast, M. Steinberger, and M. Horn, “Modeling and feedback linearization based control of an electromagnetic clutch actuator”, in *Proc. of the 2014 IEEE International Conference on Control Applications (CCA) Part of 2014 IEEE Multi-conference on Systems and Control*, Antibes, France, Oct. 2014, pp. 59–64.

[53] M. Kirchengast, M. Steinberger, S. Laimgruber, D. Prix, and M. Horn, “Identification and position control of an electromagnetic clutch actuator”, in *Proc. of the 1st IFAC Conference on Modelling, Identification and Control of Nonlinear Systems*, Saint Petersburg, Russia, Jun. 2015, pp. 59–64.

[54] S. Gupte, P. I. T. Mohandas, and J. M. Conrad, “A survey of quadrotor unmanned aerial vehicles”, in *Proc. of 2012 IEEE Southeastcon*, Orlando, FL, USA, Mar. 2012, pp. 1–6.

[55] A. Tayebi and S. McGilvray, “Attitude stabilization of a vtol quadrotor aircraft”, *IEEE Transactions on Control Systems Technology*, 14, no., pp. 562–571, 3 May 2006.

[56] S. Bouabdallah and R. Siegwart, “Backstepping and sliding-mode techniques applied to an indoor micro quadrotor”, in *Proc. of 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, Barcelona, Spain, Apr. 2005, pp. 2247–2252.

[57] F. Hoffmann, N. Goddemeier, and T. Bertram, “Attitude estimation and control of a quadrocopter”, in *Proc. of 2010 IEEE International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 2010, pp. 1072–1077.

[58] L. Derafa, L. Fridman, A. Benallegue, and A. Ouldali, “Super twisting control algorithm for the four rotors helicopter attitude tracking problem”, in *Proc. of 11th International Workshop on Variable Structure Systems*, Mexico City, Mexico, Jun. 2010, pp. 62–67.

[59] G. P. Falconi and F. Holzapfel, “Adaptive fault tolerant control allocation for a hexacopter system”, in *Proc. of 2016 American Control Conference*, Boston, MA, USA, Jul. 2016, pp. 6760–6766.

[60] Q.-L. Zhou, Y. Zhan, C.-A. Rabbath, and D. Theilliol, “Design of feedback linearization control and reconfigurable control allocation with application to a quadrotor uav”, in *Proc. of 2010 Conference on Control and Fault-Tolerant Systems*, Nice, France, Oct. 2010, pp. 371–376.

[61] Quanser Inc., 2018. [Online]. Available: `https://www.quanser.com/`.

[62] L. G. Carrillo, A. D. Lopez, R. Lozano, and C. Pegard, “Modeling the quad-rotor minirotorcraft”, in *Quad Rotorcraft Control*, 1st, Springer, London, 2013.

[63] L. Derafa, T. Madani, and A. Benallegue, “Dynamic modelling and experimental identification of four rotors helicopter parameters”, in *Proc. of 2006 IEEE International Conference on Industrial Technology*, Mumbai, India, Dec. 2006, pp. 1834–1839.

[64] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st. Cambridge University Press, 2017.

[65] Wolfram Research, 2018. [Online]. Available: `https://www.wolfram.com/`.

[66] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems*, 3rd. Addison Wesley Longman, 1998.

[67] G. Marsaglia and G. P. H. Styan, “Equalities and inequalities for ranks of matrices”, *Linear and Multilinear Algebra*, 2, no., pp. 269–292, 1974. DOI: `10.1080/03081087408817070`.

[68] N. Cheung, K. Lim, and M. Rahman, “Modelling a linear and limited travel solenoid”, in *International Conference on Industrial Electronics, Control and Instrumentation (IECON’93)*, vol. 3, Maui, Hawaii, Nov. 1993, pp. 1567–1572.

[69] K. Lim, N. Cheung, and M. Rahman, “Proportional control of a solenoid actuator”, in *20th International Conference on Industrial Electronics, Control and Instrumentation (IECON’94)*, vol. 3, Bologna, Italy, Sep. 1994, pp. 2045–2050.

[70] E. Kallenbach, R. Eick, P. Quendt, T. Stroehla, K. Feindt, M. Kallenbach, and O. Radler, *Elektromagnete: Grundlagen, Berechnung, Entwurf und Anwendung*, Fourth. Wiesbaden, Germany: Vieweg+Teubner, 2012.

[71] T. Schwarzgruber, H. Trogmann, T. Passenbrunner, S. Fizek, and P. Dolovai, “Nonlinear control of an electro-magnetic actuator under highly dynamic disturbances”, in *2012 IEEE International Conference on Control Applications (CCA)*, Dubrovnik, Croatia, Oct. 2012, pp. 974–979.

[72] A. Bemporad, S. D. Cairano, I. Kolmanovsky, and D. Hrovat, “Hybrid modeling and control of a multibody magnetic actuator for automotive applications”, in *46th IEEE Conference on Decision and Control*, New Orleans, USA, Dec. 2007, pp. 5270–5275.

[73] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. New Jersey, USA: Prentice Hall, 1991.

[74] V. Hagenmeyer and E. Delaleau, “Exact feedforward linearization based on differential flatness: The siso case”, in *Nonlinear and Adaptive Control*, A. Zinober and D. Owens, Eds., vol. 281, Springer, 2003, pp. 161–170.

[75] ——, “Robustness analysis of exact feedforward linearization based on differential flatness”, *Automatica*, 39, no., pp. 1941–1946, 2003.

[76] M. Fliess, J. Levine, P. Martin, and P. Rouchon, “Flatness and defect of nonlinear systems: Introductory theory and examples”, *International Journal of Control*, 61, no., pp. 1327–1361, 1995.