Masterarbeit

Control Allocation for over-actuated Road Vehicles

Astrid Rupp

Institut für Regelungs- und Automatisierungstechnik
Technische Universität Graz
Vorstand: Univ.-Prof. Dipl.-Ing. Dr. techn. Nicolaos Dourdoumas

Betreuer: Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Martin Horn

Graz, im September 2013

Kurzfassung

Diese Arbeit behandelt die Reduktion von Heben und Nicken eines Straßenfahrzeuges während eines Bremsvorganges. Das Heben, Nicken und Bremsen kann mit Hilfe von mechanischen Bremsen, Elektromotoren und semi-aktiven Dämpfern beeinflusst werden. Ein vereinfachtes mathematisches Modell zur Beschreibung der Fahrzeugdynamik auf ebener Fahrbahn wird herangezogen, Kurven und Steigungen werden nicht berücksichtigt. Da nur die Längsbewegung des Fahrzeuges untersucht wird, betrachtet man das dynamische Verhalten an der Vorder- und Hinterachse. Durch diese Aufteilung erhält man 6 mögliche Stellgrößen für die Aktoren, wobei nur 3 dynamische Größen beeinflusst werden - es handelt sich um ein überaktuiertes System. Daher entsteht ein unterbestimmtes Gleichungssystem, welches mit Hilfe der sogenannten "Control Allocation" als Optimierungsproblem gelöst wird. Einige Methoden der Control Allocation werden verglichen, wobei Active Set Algorithmen und Modifikationen genauer betrachtet und anhand von Simulationen mit MATLAB/Simulink evaluiert werden.

Abstract

This thesis deals with the control of lift and pitch of a car while braking on a flat surface. A simplified model of a car is used since cornering is not taken into account. Lift, pitch and braking forces are influenced by mechanical brakes, electric motors and semi-active suspension on the front and on the rear tires. Since there are 6 actuating signals to achieve the desired 3 virtual forces the system is over-actuated. Finding the numerical solution of this underdetermined set of equations is called "control allocation" and goal of this thesis. Active set methods and modifications are discussed and simulated with MATLAB/Simulink.

Contents

List of Figures

Statutory Declaration

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 marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Acknowledgement

First of all, I would like to thank my advisor Professor Martin Horn for the opportunity to work on this topic. The meetings with him increased my motivation significantly and he always had words of encouragement at the ready.
Many thanks go to thank Daniel Lindvai-Soos for supervising my work. He always lent a helping hand when I did not see the forest for the trees. I appreciate his help, motivation and humor.
Special thanks to my brother Karl Rupp, who encouraged me to my studies over and over again. He always provides me with good advice - and good jokes. Proofreading is only a tiny part of all the help he has given me obligingly. I am proud to call such an outstanding person my brother.
A hearty thanks to Matthias Leitner for proofreading the thesis and all the understanding of working late hours.
Finally, I want to thank my family and all of my friends who are always there for me. Thanks for making my limited spare-time unforgettable.

Thesis Outline

This thesis starts with a mathematical model of a road vehicle in Chapter 1. The controller design for the lift and pitch motions is presented. Since the system is over-actuated, control allocation is used to generate the forces required by the controller.
Chapter 2 deals with the control allocation in general. Different methods for a linear, static problem are discussed. Active set methods are known to work well for automotive problems and are therefore described in detail. Small modifications of these methods including dynamic control allocation are discussed briefly.
Chapter 3 consist of the application of the most suitable control allocation algorithms to the mathematical model. The Simulink-Model, constraints of the actuators and settings relevant for the simulation are outlined. Application-oriented modifications on the algorithms are discussed.
The results of the simulations are outlined in Chapter 4.
Finally an outlook on further ideas for future work and a conclusion is given in Chapter 5.

Chapter 1

Motivation of the Work

In the last decades research on vehicle dynamics control (VDC) has been conducted extensively. Many approaches regarding vehicle safety and maneuverability have been developed. Active safety systems play an important role in road vehicles nowadays, e.g. ABS and ESP are used widely in modern road vehicles. Methods to prevent rollover or to stabilize the yaw motion of the vehicle using control allocation have been proposed, e.g. in [1], [25], and [27]. Improved passenger comfort and reduced driver workload are to a greater or lesser extent consequences of these techniques. The goal of this thesis is to improve the passenger comfort during braking by suppressing lift and pitch oscillations of the car. It is important that braking is not delayed, the longitudinal movement of the car must not be changed in order to guarantee safety. The following section shows the lift-pitch-model of the car that is based on [15]. The resulting system equations and the controller are outlined in this chapter.

1.1 Linearized Model of the Car

A linearized mathematical model of a car has been presented in [15] using the following assumptions:

- Only longitudinal movement is considered. Lateral movement, i.e. cornering and turning, is objective of further work and not included in this thesis.
- Since only longitudinal movement is of interest, the tires are grouped into front and rear tires. There is no differentiation into left and right tires.
- The vehicle moves on a flat track without inclination.
- Simplified driving resistances are used, aerodynamics and lift are neglected.
- Isotropic tire characteristics with no run-in behaviour are considered.
- Unsprung mass dynamics are neglected.

These simplifications yield three variables that shall be controlled to improve the comfort: lift z, pitch θ and the velocity $\dot{x}$. Figure 1.1 shows how these variables are defined for the longitudinal model and how they affect the car. The simulation parameters shown in this figure are defined

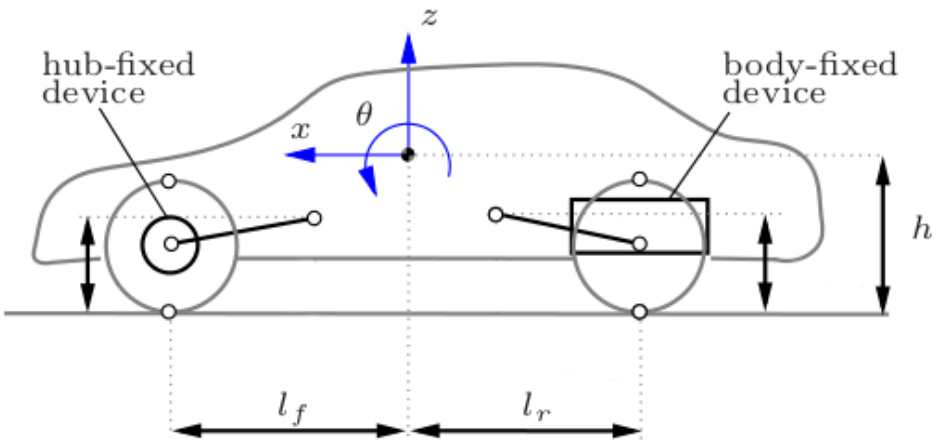


Figure 1.1: Lift-pitch-model of the car as shown in [15]

in [15] and shown in Table 1.1. For the lift-pitch-model the following states are introduced:

$$\mathbf{x} = \begin{bmatrix} z \\ \dot{z} \\ \theta \\ \dot{\theta} \\ \dot{x} \end{bmatrix}, \tag{1.1}$$

where z and θ are the lift and pitch variables, $\dot{z}$ and $\dot{\theta}$ the corresponding velocities and $\dot{x}$ is the velocity of the car. The virtual forces, i.e. the forces and torques that shall be controlled, are defined as

$$\mathbf{v} = \begin{bmatrix} F_\mathrm{z} \\ T_\mathrm{y} \\ F_\mathrm{x} \end{bmatrix}, \tag{1.2}$$

where F_z stands for the lift, T_y the pitch and F_x the velocity of the vehicle. The output of the system is given by

$$\mathbf{y} = \begin{bmatrix} z \\ \theta \\ \dot{x} \end{bmatrix}. \tag{1.3}$$

The system equations of the model linearized around an operating point defined by $\mathbf{x}_0$,$\mathbf{v}_0$ can be written as

$$\begin{aligned} \Delta\dot{\mathbf{x}} &= \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{v}, \\ \Delta\mathbf{y} &= \mathbf{C}\Delta\mathbf{x}, \end{aligned} \tag{1.4}$$

where $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0$, $\Delta\mathbf{v} = \mathbf{v} - \mathbf{v}_0$. The initial conditions $\mathbf{x}_0$ are given as

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 80/3.6 \end{bmatrix}, \tag{1.5}$$

which corresponds to a vehicle driving at a constant speed of 80 km/h without any lift or pitch. This vector is also chosen as the point of interest for linearization, i.e. $\mathbf{x}_0 = \mathbf{x}_s$. The matrices in 1.4 derived in [15] are defined as follows:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{2(c_f+c_r)}{-m_A} & \frac{2(d_f+d_r)}{-m_A} & \frac{2(c_r l_r - c_f l_f)}{-m_A} & \frac{2(d_r l_r - d_f l_f)}{-m_A} & \frac{-b_z \tilde{d}}{m_A m^*} \\ 0 & 0 & 0 & 1 & 0 \\ \frac{2(c_r l_r - c_f l_f)}{-J_y} & \frac{2(d_r l_r - d_f l_f)}{-J_y} & \frac{2(c_f l_f^2 + c_r l_r^2)}{-J_y} & \frac{2(d_r l_r^2 + d_f l_f^2)}{-J_y} & \frac{-b_y \tilde{d}}{J_y m^*} \\ 0 & 0 & 0 & 0 & \frac{-\tilde{d}}{m^*} \end{bmatrix}, \tag{1.6}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{m_A} & 0 & \frac{b_z}{m_A m^*} \\ 0 & 0 & 0 \\ 0 & \frac{1}{J_y} & \frac{b_y}{J_y m^*} \\ 0 & 0 & \frac{1}{m^*} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.7}$$

where the individual parameters are listed in Table 1.1. The remaining parameters are abbreviations defined as

$$m^* = m + 4\frac{J_w}{r^2}$$

and

$$b_y = 2\frac{J_w}{r^2}(\tan(\epsilon_{2,f}) - \tan(\epsilon_{2,r})),$$
$$b_z = -2\frac{J_w}{r^2}(\tan(\epsilon_{2,f})l_f + \tan(\epsilon_{2,r}l_r + 2r - 2h)),$$

where the so-called support angles ϵ_i are defined as shown in Figure 1.2.

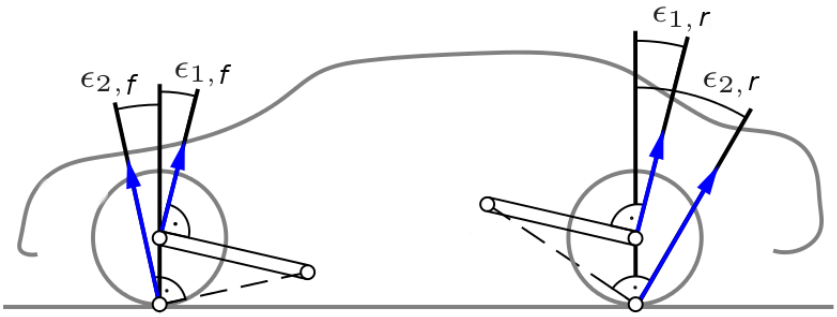


Figure 1.2: Model of the car with support angles taken from [15].

So far only the virtual forces have been used in the description of the model. These virtual forces can be generated by 6 actuators $\mathbf{u}$ by the following relation:

$$\mathbf{v} = \mathbf{H}\mathbf{u}, \tag{1.8}$$

where $\mathbf{H}$ is a 3×6 matrix. The actuators are described in more detail in Chapter 3 while the important role of the matrix $\mathbf{H}$ is outlined in Chapter 2. The matrix $\mathbf{H}$ for this model is given by

$$\mathbf{H} = \begin{bmatrix} -\tan(\epsilon_{1,f}) & \tan(\epsilon_{1,r}) & -\tan(\epsilon_{2,f}) & \tan(\epsilon_{2,r}) & 1 & 1 \\ (\tan(\epsilon_{1,f}) \cdot l_f - h) & (\tan(\epsilon_{1,r}) \cdot l_r - h) & (\tan(\epsilon_{2,f}) \cdot l_r - h) & (\tan(\epsilon_{2,r}) \cdot l_r - h) & -l_f & l_r \\ 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}. \tag{1.9}$$

Name	Abbrevation	Unit	Value
Vehicle Mass	m	kg	1725
Body Mass	m_A	kg	$0.9 \cdot m$
Body Inertia	J_y	kgm^2	2646
Wheel Inertia	J_w	kgm^2	1
Dist. COG-FA	l_f	m	1.3
Dist. COG-RA	l_r	m	1.46
Height of COG	h	m	501
Spring stiff.	c_f	N/m	24350
Spring stiff.	c_r	N/m	40900
Damper coeff.	d_f	Ns/m	1317.5
Damper coeff.	d_r	Ns/m	1445
Hub support angle	$\epsilon_{1,f}$	deg	4
Hub support angle	$\epsilon_{1,r}$	deg	22
Body support angle	$\epsilon_{2,f}$	deg	1
Body support angle	$\epsilon_{2,r}$	deg	5.5
Tire radius	$r_f = r_r = r$	m	0.3
Aerodynamic resistances	$\tilde{d}$	Ns/m	29.1464

Table 1.1: Simulation parameters of the simulated vehicle.

1.2 Controller Design

The design of the controller is not an objective of this thesis and therefore an existing controller is presented. For the controller the sky-hook control design based on [10] or [9] is used. The sky-hook technique is a common method when using semi-active suspension as is the case for this vehicle (see Chapter 3). As shown in [9], the sky-hook law can be written as

$$F_{sky,z} = -d_{SH}^{z}\dot{z} \tag{1.10}$$

and

$$T_{sky,\theta} = -d_{SH}^{\theta}\dot{\theta}. \tag{1.11}$$

Now, the virtual forces

$$\mathbf{v} = \begin{bmatrix} F_{\mathrm{z}} \\ T_{\mathrm{y}} \\ F_{\mathrm{x}} \end{bmatrix}$$

shall be controlled for input state $\mathbf{x}$. The modal skyhook control works similar to a state-feedback-controller:

$$\mathbf{v} = -\mathbf{K}\mathbf{x} + \mathbf{H}\mathbf{u}_{\mathrm{des}}$$

with the design matrix $\mathbf{K}$. For the linear system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{v}, \tag{1.12}$$

the controller

$$\mathbf{v} = -\mathbf{K}\mathbf{x} + \mathbf{H}\mathbf{u}_{\mathrm{des}} \tag{1.13}$$

yields the following closed-loop structure:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}\mathbf{x} + \mathbf{B}\mathbf{H}\mathbf{u}_{\mathrm{des}} = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x} + \mathbf{B}\mathbf{H}\mathbf{u}_{\mathrm{des}}. \tag{1.14}$$

For the lift and pitch forces the corresponding column in this matrix $\mathbf{K}$ can be chosen to achieve specific dynamics. This desired dominant behaviour of the dynamics of lift and pitch are chosen to match the transfer function

$$G_i^*(s) = \frac{y_i(s)}{v_i(s)} = \frac{k_i}{s^2 + 2\zeta_i\omega_{0,i}s + \omega_{0,i}^2}, \tag{1.15}$$

which represents a system with a dominant polepair with the angular frequency $\omega_{0,i}$ and the damping ratio ζ_i. Since the braking force $\mathbf{F}_x$ must not be delayed or influenced, the last column of the controller matrix $\mathbf{K}$ is set to zeros.
The system structure is in Luenberger control canonical form for MIMO-systems (cf. [16]) and therefore the matrix $\mathbf{K}$ can be determined by comparison of coefficients. Choosing $\zeta_{\mathrm{lift}} = 0.5$ and $\zeta_{\mathrm{pitch}} = 0.5$ the resulting controller reads

$$\mathbf{K} = \begin{bmatrix} 0 & d_{SH}^{z} & 0 & \tilde{d}_1 & 0 \\ 0 & \tilde{d}_2 & 0 & d_{SH}^{\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{1.16}$$

with the sky-hook parameters

$$d_{SH}^{z} = 8708.8, \quad d_{SH}^{\theta} = 15447,$$

and the parameters

$$\tilde{d}_1 = \tilde{d}_2 = -793.9.$$

This is the controller used in the remainder of the thesis.

Chapter 2

Control Allocation

Control allocation is used for over-actuated systems in order to solve an underdetermined set of equations. This underdetermined set of equations describes the relation of the control input and the different actuators. In this chapter control allocation in general is discussed and some methods are presented. Active set methods are used to solve quadratic programs and are outlined below.

2.1 Introduction

This section is based on [8] and describes the basics of control allocation. Control allocation separates the actuator selection from the controller task such that the controller only takes care of the virtual control. One benefit of the separation is that the actuator constraints can be taken into account, actuator changes or failures can also be handled. Another advantage is that the problem can be optimized for the specific application independently. Figure 2.1, as presented in [8], shows the overall system with the separation of controller and control allocation.

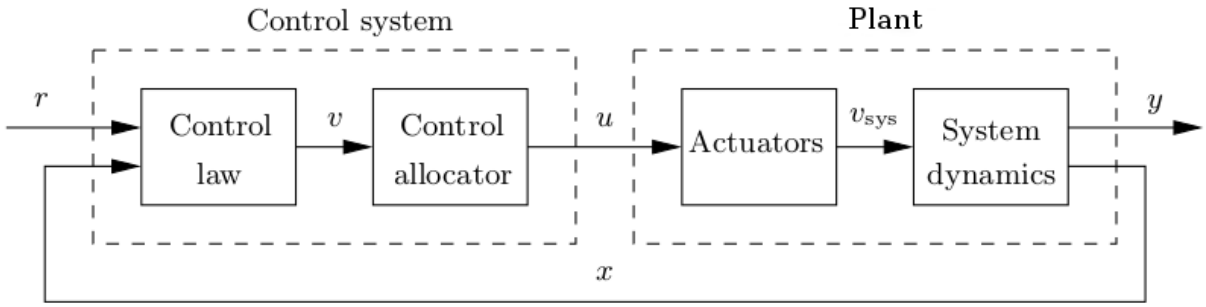


Figure 2.1: Control Allocation from [8].

The virtual control input that is calculated by the controller is denoted by $\mathbf{v}$. $\mathbf{u}$ is the control input for the actuators that is calculated by the control allocation algorithm. The general mathematical description for a system considered for control allocation reads as follows:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{v}), \tag{2.1}$$

$$\mathbf{v} = \mathbf{h}(\mathbf{x}, \mathbf{u}). \tag{2.2}$$

In the linear case this can be written as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{v}, \tag{2.3}$$
$$\mathbf{v} = \mathbf{H}\mathbf{u}, \tag{2.4}$$

with $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{v} \in \mathbb{R}^k$ and $k < m$. The fact that there are more actuators than virtual control variables is a necessary condition for the system to be over-actuated and for control allocation to be applied.
For the position and rate constraints of the actuators the following equations must hold:

$$\mathbf{u}_{\min} \leq \mathbf{u}(t) \leq \mathbf{u}_{\max}, \tag{2.5}$$
$$\boldsymbol{\rho}_{\min} \leq \dot{\mathbf{u}}(t) \leq \boldsymbol{\rho}_{\max}. \tag{2.6}$$

The inequalities apply componentwise. Rate constraints can be rewritten as position constraints with

$$\underline{\mathbf{u}}(t) = \max\{\mathbf{u}_{\min}, \mathbf{u}(t-T) + T \cdot \boldsymbol{\rho}_{\min}\}, \tag{2.7}$$
$$\overline{\mathbf{u}}(t) = \min\{\mathbf{u}_{\max}, \mathbf{u}(t-T) + T \cdot \boldsymbol{\rho}_{\max}\}, \tag{2.8}$$

where T is the sampling time. This results in

$$\underline{\mathbf{u}}(t) \leq \mathbf{u}(t) \leq \overline{\mathbf{u}}(t). \tag{2.9}$$

The constraints correspond to a convex hyperbox. Dropping the time dependencies, the goal of control allocation in the linear case is then to solve

$$\mathbf{v} = \mathbf{H}\mathbf{u}, \tag{2.10}$$
$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}, \tag{2.11}$$

where

$$\begin{aligned} &\mathbf{x} \in \mathbb{R}^n, \\ &\mathbf{u} \in \mathbb{R}^m, \\ &\mathbf{v} \in \mathbb{R}^k, \quad k < m. \end{aligned} \tag{2.12}$$

This is known as the standard constrained linear control allocation problem. $\mathbf{H}$ is often called *control effectiveness matrix* and is of rank k.
There are 3 possible results:

- an infinite number of solutions
- only one unique solution
- no solution

In either case the control allocation should return the "best" solution considering the constraints. For an infinite number of solutions an additional objective function may be used to pick only one specific solution. In the algorithms below, an objective function is used that accounts for the desired control input $\mathbf{u}_{\text{des}}$. If no solution considering the constraints is possible, a method for minimizing the error between $\mathbf{v}$ and $\mathbf{Hu}$ is necessary to attain the "best possible" solution. In the literature several methods for solving the control allocation problem have been proposed. The control allocation problem in general with different approaches on static linear control allocation is presented in [22]. In early research, control allocation has mostly been studied for

over-actuated aircrafts, e.g. in [3], [4]. Model predictive control allocation has been shown to be useful in many applications. Yu Luo *et al.* have presented several approaches on this field, e.g. [17], [18], and [19]. Härkegard proposed an active set algorithm in [8], which most of this thesis is based on, and showed simulations and results on aircrafts. In [23] a comparison of the active set algorithm with the interior point method is presented. Schofield extends the active set algorithm in [24] and concentrates on automotive examples. In [25] rollover prevention is presented. Johansen *et al.* also work on nonlinear control allocation in different applications, e.g. automotive examples in [13], [11], [26] and examples on ships in [7]. The control allocation method strongly depends on the application. Aircrafts, marine vessels and road vehicles represent the main fields of research on these methods and yield different approaches.
In the following section some of the above mentioned methods for control allocation are discussed in general.

2.2 Comparison of static linear Control Allocation Methods

For static, linear control allocation the following methods are outlined below:

- Weighted Pseudo-Inverse
- Direct CA
- Linear Programming
- Quadratic Programming
  - Active Set Methods
    - ▶ Sequential Least Squares
    - ▶ Weighted Least Squares
  - Fixed-Point Method
  - Primal-Dual Interior-Point Method

These methods have been used in MATLAB/Simulink with the model of the car given in Chapter 1. Approaches for control allocation such as Daisy Chaining and Explicit Ganging as discussed in [22] are not desirable for this automotive example and therefore neglected.

The **Weighted Pseudo-Inverse**, as discussed in [22], represents the first idea for the solution of the control allocation task. The control allocation problem is to find

$$\mathbf{u} = \underset{\mathbf{u}}{\operatorname{argmin}}\ J = \frac{1}{2}\mathbf{u}^T\mathbf{W}\mathbf{u} \tag{2.13}$$

$$\text{subject to } \mathbf{v} = \mathbf{H}\mathbf{u}, \tag{2.14}$$

where $\mathbf{W} \in \mathbb{R}^m$ is a matrix penalizing the errors for the individual actuators and the only design parameter. The solution is found by

$$\mathbf{u} = \mathbf{W}^{-1}\mathbf{H}^T(\mathbf{H}\mathbf{W}^{-1}\mathbf{H}^T)^{-1}\mathbf{v}. \tag{2.15}$$

If the constraints do not become active during the entire simulation, i.e. the unique solution lies within the feasible region, the results are accurate. If there is no unique solution to the problem, the results may become inaccurate since the constraints may be violated. There is no guarantee

that the commanded control input will not exceed the constraints. Since the constraints may be violated, this method is no proper choice for this application on road vehicles.

The **Direct Control Allocation**, as presented by Durham and discussed in [22], aims at finding a real number α and a vector $\mathbf{u}_1$ such that

$$\mathbf{H}\mathbf{u}_1 = \alpha\mathbf{v}, \tag{2.16}$$

$$\underline{\mathbf{u}}(t) \leq \mathbf{u}(t) \leq \overline{\mathbf{u}}(t). \tag{2.17}$$

If $\alpha > 1$, let $\mathbf{u} = \frac{1}{\alpha}\mathbf{u}_1$, otherwise $\mathbf{u} = \mathbf{u}_1$. This method thereby scales the control inputs if the solution $\mathbf{u}^*$ to $\mathbf{v} = \mathbf{H}\mathbf{u}$ does not lie within the feasible region. This means that the vector $\mathbf{u}$ has the same direction as the unconstrained solution $\mathbf{u}^*$, but smaller magnitude. There are no additional design variables to be selected. For the application in this thesis, at least a desired control input variable should be taken into account. Maintaining the direction of the vector is not important for this application, primarily the error $\mathbf{v} - \mathbf{H}\mathbf{u}$ should be minimized. Therefore, this method is not useful for the problem stated in Chapter 1.

Linear programming, as described in [22], aims at minimizing the error

$$J_1 = \|(\mathbf{v} - \mathbf{H}\mathbf{u})\|_1. \tag{2.18}$$

The linear program uses slack variables $\mathbf{u}_{\text{slack}}$ and can be posed as follows:

$$\min_{\mathbf{u}} J_1 = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{u}_{\text{slack}} \end{bmatrix} \tag{2.19}$$

$$\text{subject to } \begin{bmatrix} \mathbf{u}_{\text{slack}} \\ -\mathbf{u} \\ \mathbf{u} \\ -\mathbf{H}\mathbf{u} + \mathbf{u}_{\text{slack}} \\ \mathbf{H}\mathbf{u} + \mathbf{u}_{\text{slack}} \end{bmatrix} \geq \begin{bmatrix} \mathbf{0} \\ -\overline{\mathbf{u}} \\ \underline{\mathbf{u}} \\ -\mathbf{v} \\ \mathbf{v} \end{bmatrix}. \tag{2.20}$$

If $J_1 = 0$ then the virtual control $\mathbf{v}$ is feasible and there exists a set of control inputs $\mathbf{u}_{\boldsymbol{\Omega}}$ that solve the equation $\mathbf{v} = \mathbf{H}\mathbf{u}$. A secondary objective may be stated which takes the desired control input for the actuators $\mathbf{u}_{\text{des}}$ into account:

$$J_2 = \|(\mathbf{u} - \mathbf{u}_{\text{des}})\|_1. \tag{2.21}$$

The new linear program can then be written as

$$\min_{\mathbf{u} \in \mathbf{u}_\Omega} J_2 = \mathbf{w}_{\text{u}}^T \mathbf{u}_{\text{slack}} \tag{2.22}$$

$$\text{subject to } \begin{bmatrix} \mathbf{u}_{\text{slack}} \\ -\mathbf{u} \\ \mathbf{u} \\ -\mathbf{u} + \mathbf{u}_{\text{slack}} \\ \mathbf{u} + \mathbf{u}_{\text{slack}} \end{bmatrix} \geq \begin{bmatrix} \mathbf{0} \\ -\overline{\mathbf{u}} \\ \underline{\mathbf{u}} \\ -\mathbf{u}_{\text{des}} \\ \mathbf{u}_{\text{des}} \end{bmatrix}, \tag{2.23}$$

where $\mathbf{w}_{\text{u}}^T$ is a vector penalizing the error $\mathbf{u} - \mathbf{u}_{\text{des}}$. Although a linear program can be solved faster than a quadratic one, the linear programming approach is nevertheless undesirable: As

stated in [8], it utilizes only as few actuators as possible instead of using all to satisfy the virtual control demand.

In literature **Quadratic Programming** is often regarded as the best method for automotive applications of the type considered in this thesis. As stated in [8], quadratic programming utilizes all actuators and divides them uniformly which is more desirable than the linear programming approach.
There are several ways to solve the quadratic program. The first objective is to solve

$$\mathbf{u_\Omega} = \underset{\mathbf{u}}{\operatorname{argmin}}\ J = \|\mathbf{W}_\mathrm{v}(\mathbf{v} - \mathbf{H}\mathbf{u})\|_2. \tag{2.24}$$

Of this set $\mathbf{u_\Omega}$ the "best" solution can be found by the second objective function

$$\mathbf{u} = \underset{\mathbf{u_\Omega}}{\operatorname{argmin}}\ J = \|\mathbf{W}_\mathrm{u}(\mathbf{u} - \mathbf{u}_\mathrm{des})\|_2. \tag{2.25}$$

Active Set Methods are similar to simplex methods only for quadratic programs. These methods, as presented in [8], have become very popular and are widely used. The idea of the algorithm is to partition the control inputs $\mathbf{u}$ into an active set with saturated and a free set with unsaturated control inputs. With the notation

$$\mathbf{A} = \begin{bmatrix} \sqrt{\gamma}\mathbf{W}_\mathrm{v}\mathbf{H} \\ \mathbf{W}_\mathrm{u} \end{bmatrix}, \tag{2.26}$$

$$\mathbf{b} = \begin{bmatrix} \sqrt{\gamma}\mathbf{W}_\mathrm{v}\mathbf{v} \\ \mathbf{W}_\mathrm{u}\mathbf{u}_\mathrm{des} \end{bmatrix}, \tag{2.27}$$

a residual $\mathbf{d}$ is computed by

$$\mathbf{d} = \mathbf{b} - \mathbf{A}\mathbf{u}. \tag{2.28}$$

A suboptimal solution is calculated only by the use of the free variables:

$$\mathbf{u}_\mathrm{f} = \mathbf{A}_\mathrm{f}^{+}\mathbf{d}, \tag{2.29}$$

where the subscript f denotes the columns corresponding to the free variables and $\mathbf{A}_\mathrm{f}^{+}$ is the pseudo-inverse of $\mathbf{A}_\mathrm{f}$. If the solution is infeasible, a step size α is calculated and the most bounding control input is set active. If the solution is feasible the Lagrangian multipliers are calculated:

$$\boldsymbol{\lambda}_\mathrm{a} = \mathbf{A}_\mathrm{a}^{T}\mathbf{d}, \tag{2.30}$$

where a denotes the indizes of the active variables. The optimum is reached if all Lagrangian multipliers $\boldsymbol{\lambda}_\mathrm{a}$ are non-negative. Otherwise the control input corresponding to the most negative $\lambda_{\mathrm{a},i}$ is removed from the active set. The first tests show the best results when using active set methods, especially the weighted least squares method is the fastest algorithm with the best performance. Active set methods are therefore used for further simulations and are described in more detail later in this chapter.

The *fixed-point method* is also discussed in [8] and is similar to a gradient-search method. It solves the quadratic program

$$\min_{\mathbf{u}} J = \|\mathbf{W}_{\mathrm{v}}(\mathbf{H}\mathbf{u} - \mathbf{v})\|_2^2 + \gamma\|\mathbf{W}_{\mathrm{u}}(\mathbf{u} - \mathbf{u}_{\mathrm{des}})\|_2^2 \tag{2.31}$$

as follows:

$$\mathbf{u}^k = \mathrm{sat}\left[(1-\epsilon)\omega\mathbf{H}^T\mathbf{W}_{\mathrm{v}}^T\mathbf{W}_{\mathrm{v}}\mathbf{v} - (\omega\mathbf{Q} - \mathbf{I})\mathbf{u}^{k-1}\right], \tag{2.32}$$

with

$$\epsilon = \frac{1}{\gamma + 1}, \tag{2.33}$$

$$\mathbf{Q} = (1-\epsilon)\mathbf{H}^T\mathbf{W}_{\mathrm{v}}^T\mathbf{W}_{\mathrm{v}}\mathbf{H} + \epsilon\mathbf{W}_{\mathrm{u}}^T\mathbf{W}_{\mathrm{u}}, \tag{2.34}$$

where $\omega = \|Q\|_F^{-1}$ decides the step length and $\|Q\|_F$ denotes the Frobenius Norm of $\mathbf{Q}$. This method considers only problems where the desired control input for the actuators $\mathbf{u}_{\mathrm{des}}$ equals zero which is not useful for the given problem set. The simulations do not yield satisfying results.

The *primal-dual interior-point method* is discussed in [23]. For optimization the problem is converted to a quadratic program

$$\begin{aligned} \min_{\mathbf{u}} J = \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{f}^T\mathbf{x} \\ \text{subject to} \quad \mathbf{x} + \mathbf{w} \leq \mathbf{x}_{\max} \\ \mathbf{x} \geq \mathbf{0} \\ \mathbf{w} \geq \mathbf{0} \end{aligned} \tag{2.35}$$

where

$$\begin{aligned} \mathbf{x} = \mathbf{u} - \mathbf{u}_{\min}, \\ \mathbf{x}_{\max} = \mathbf{u}_{\max} - \mathbf{u}_{\min}, \end{aligned} \tag{2.36}$$

and $\mathbf{w}$ is a slack variable. The virtual control $\mathbf{v}$, the constraints $\mathbf{u}_{\min}$, $\mathbf{u}_{\max}$ and the control effectiveness matrix $\mathbf{H}$ are given. The matrix $\mathbf{G}$ is defined by the control allocation problem. The matrizes $\mathbf{H}$, $\mathbf{W}_{\mathrm{v}}$, $\mathbf{W}_{\mathrm{u}}$ and the vectors $\mathbf{v}$, $\mathbf{u}_{\mathrm{des}}$ are used to compute $\mathbf{G}$. An additional update parameter ρ and the stopping tolerance ϵ must be chosen before starting the optimization. This method is known to be a very good approach for a large number of actuators. For $m > 10$ this method is proven to show better performance than the active set methods since the number of iterations is smaller, but the computational effort per iteration is high in comparison. Since the given problem set uses only 6 actuators, the active set method is the better choice for further studies.

In Table 2.1 the comparison regarding consumption of time and performance of the mentioned methods is shown. The results are average values calculated over 100 simulations.

Method	Time in s	Performance
Weighted PI	1.2121	inaccurate
Direct CA	36.9668	no design parameters
SLS	4.1114	very good
WLS	2.0852	very good
FixedPoint	2.5000	$\mathbf{u}_{\text{des}}$ inaccurate
InteriorPoint	7.5124	very good
LP	50.4691	good

Table 2.1: Comparison of different control allocation methods regarding time consumption and performance.

The results correspond to those in [8]. It is shown that the active set methods "SLS" and "WLS" implemented by Härkegard yield the best results in a minimum of time. Therefore these methods are dealt with in the next section.

2.3 Active Set Methods

The active set methods have become quite popular and show good results in the first tests. Therefore, these methods are used for further tests and simulations. For the application it is important to analyze the algorithm in order to guarantee safety for the driver of the vehicle at all times while comfort is being improved. The methods are presented in [8] and are discussed in this section.
An active set algorithm can solve the bounded and equality constrained least squares problem

$$\min_{\mathbf{u}} \|\mathbf{A}\mathbf{u} - \mathbf{b}\|_2 \tag{2.37}$$

$$\mathbf{v} = \mathbf{H}\mathbf{u} \tag{2.38}$$

$$\mathbf{C}\mathbf{u} \geq \mathbf{U}, \tag{2.39}$$

efficiently by making use of the special structure with

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_m \\ -\mathbf{I}_m \end{bmatrix}, \tag{2.40}$$

where $\mathbf{I}_m$ denotes the identity matrix of size $m \times m$ and

$$\mathbf{U} = \begin{bmatrix} \underline{\mathbf{u}} \\ -\overline{\mathbf{u}} \end{bmatrix}. \tag{2.41}$$

The method is called "active set" because the algorithm divides the problem into a set of control inputs in saturation, which is called the active set $\mathcal{A}$, and a set of free variables, i.e. the variables that are not saturated. The active constraints are regarded as equality constraints, the free constraints correspond to the inequality constraints which are disregarded. Thus, the free variables are handled as unbounded variables during one optimization step. Optimality is checked by the Karush-Kuhn-Tucker conditions. The main steps of the active set method in pseudo-code are shown in Algorithm 1.

Algorithm 1 Active Set

Given a feasible starting point $\mathbf{u}^{(0)}$ and the working set $\mathcal{W}_0$
for $i = 1$ to maximum number of iterations **do**
solve

$$\min \|\mathbf{A}(\mathbf{u}^{(i)} + \mathbf{p}) - \mathbf{b}\|$$
$$p_i = 0, \quad i \in \mathcal{W}$$

where

$$\mathbf{A} = \begin{bmatrix} \sqrt{\gamma}\mathbf{W}_{\mathrm{v}}\mathbf{H} \\ \mathbf{W}_{\mathrm{u}} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \sqrt{\gamma}\mathbf{W}_{\mathrm{v}}\mathbf{v} \\ \mathbf{W}_{\mathrm{u}}\mathbf{u}_{\mathrm{des}} \end{bmatrix}.$$

if $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \mathbf{p}$ is feasible **then**
compute Lagrangian multipliers $\boldsymbol{\lambda}$
if $\boldsymbol{\lambda} > \mathbf{0}$ **then**
Optimum found, return
else
Remove constraint with most negative λ_i
end if
else
calculate $\alpha \in [0, 1]$ such that $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \alpha\mathbf{p}$ is feasible
add most binding constraint to working set
end if
end for

The Lagrangian multipliers are determined by

$$\mathbf{A}^T(\mathbf{A}\mathbf{u} - \mathbf{b}) = \begin{bmatrix} \mathbf{H}^T & \mathbf{C}_0^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\lambda} \end{bmatrix}, \tag{2.42}$$

where $\mathbf{C}_0$ contains the rows of $\mathbf{C}$ corresponding to the active constraints. It is important to notice that at each suboptimal iteration either one free constraint becomes active or one active constraint is set free. Active set methods are very efficient if a good estimate of the working set $\mathcal{W}$ is available and the number of changes in the working set is rather small. One benefit of the active set method is that each iteration yields a lower value of the objective function. The iterates $\mathbf{u}^{(k)}$ are always feasible, the constraints are not violated if the algorithm terminates at a suboptimal solution. Since the problem is divided into a smaller subproblem, i.e. the equality constrained least squares problem, the calculations during one iteration are computationally cheap. Active set methods only pass the working set and the previous solution to the next sampling instant. The parameters $\mathbf{W}_{\mathrm{v}}$, $\mathbf{W}_{\mathrm{u}}$, $\mathbf{u}_{\mathrm{des}}$ can be chosen properly in every time step, even the matrix $\mathbf{H}$, the virtual control $\mathbf{v}$ and the constraints $\underline{\mathbf{u}}$, $\overline{\mathbf{u}}$ may be changed for every optimization problem each time step. This allows for time-varying control effectiveness matrix $\mathbf{H}(t)$ without further limitations.

There are two different methods for solving the problem with active set algorithms:

- Sequential least-squares control allocation

$$\mathbf{u_\Omega} = \underset{\mathbf{u}}{\operatorname{argmin}}\ J = \|\mathbf{W}_\mathrm{v}(\mathbf{v} - \mathbf{H}\mathbf{u})\|_2, \tag{2.43}$$

$$\mathbf{u} = \underset{\mathbf{u_\Omega}}{\operatorname{argmin}}\ J = \|\mathbf{W}_\mathrm{u}(\mathbf{u} - \mathbf{u}_\mathrm{des})\|_2, \tag{2.44}$$

  with

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}.$$

- Weighted least-squares control allocation

$$\mathbf{u} = \underset{\mathbf{u}}{\operatorname{argmin}}\ J = \|\mathbf{W}_\mathrm{u}(\mathbf{u} - \mathbf{u}_\mathrm{des})\|_2^2 + \gamma\|\mathbf{W}_\mathrm{v}(\mathbf{v} - \mathbf{H}\mathbf{u})\|_2^2 \tag{2.45}$$

  with

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}.$$

These methods are applied using Algorithm 1. With γ chosen very high (e.g. 10^6) the weighted least squares problem yields the same results as the sequential least squares but in fewer iterations.
The following example demonstrates the sequential least squares method:

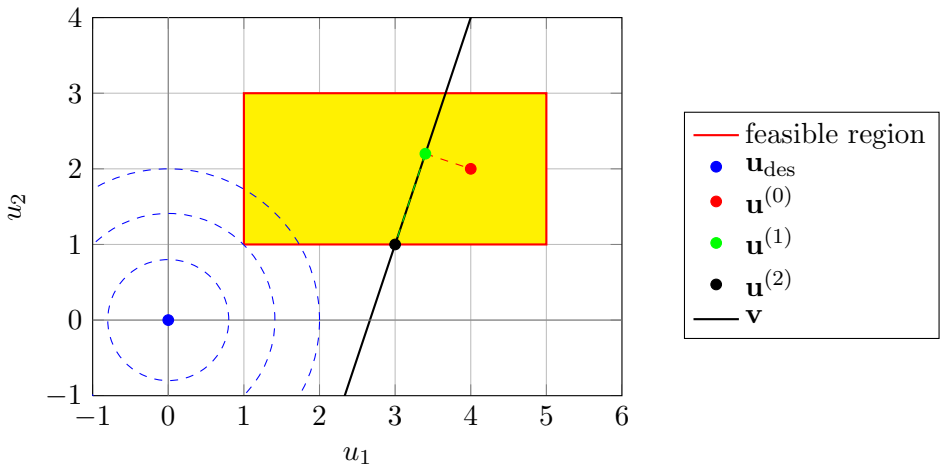


Figure 2.2: Example for the sequential least squares method.

The black line represents the virtual control $\mathbf{v} = \mathbf{H}\mathbf{u}$ that must be reached in the first step (2.43). When starting at $\mathbf{u}^{(0)}$ the sequential least squares method calculates the next iterate $\mathbf{u}^{(1)}$ that lies on the black line since the virtual control $\mathbf{v}$ is feasible. The first objective function is zero, the set of control inputs $\mathbf{u_\Omega}$ that is used for the second objective function consists of the control inputs on the black line inside the feasible region. The blue dashed curves show different levels of values for the second objective function $J = \|\mathbf{W}_\mathrm{u}(\mathbf{u} - \mathbf{u}_\mathrm{des})\|_2$. In the second step (2.44), this second objective function is minimized using the set of control inputs $\mathbf{u_\Omega}$. The algorithm tries to reach $\mathbf{u}_\mathrm{des}$ while sticking on the line $\mathbf{v} = \mathbf{H}\mathbf{u}$. This results in the iterate $\mathbf{u}^{(2)}$

with the constraint $\mathbf{u}_1$ becoming active. In the last iteration, the Lagrangian multiplier of the active constraints, i.e. λ_1, is found to be positive and the algorithm terminates with the solution $\mathbf{u}_{\text{opt}} = \mathbf{u}^{(2)}$ and the working set $\mathcal{W} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

One drawback of the active set method is the computational complexity in worst case, which is exponential as has been proved for simplex methods by Klee and Minty in [14]. The average case, however, is much lower than exponential. For safety reasons it is nevertheless essential to guarantee an upper limit of iterations.
In [8], Härkegard presents further details on the solution of the two active set methods. He also provided the MATLAB-code (called QCAT) for the computation which is available at `http://research.harkegard.se/qcat/` .
A number of MATLAB-routines for solving quadratic programs are available. When using `quadprog`, setting the option "algorithm" to "active-set" yields the desired algorithm. The drawback of this routine is the slow computation, therefore it is to the best advantage to find a faster way to compute the solution. For the simulations presented in Chapter 3 the MATLAB-code provided by Härkegard is used since it is much faster than the MATLAB-routines. The computation is presented in detail in his thesis which makes it easier to analyze the behaviour of the algorithm in the case of failures.

2.3.1 Hotstart

One benefit of using active set methods is the use of hotstart (or warm start). Hotstart benefits of the use of the solution of the previous time step as starting point of the current time step. It can also be used for simplex-methods and yields fewer iterations if the bounding constraints do not change substantially. As Schofield stated in [24], there may be trouble with hotstart when the constraints are changing since the previous solution may no longer be feasible. The algorithm needs a feasible starting point so the hotstart variables $\mathbf{u}_{k-1}$, $\mathcal{W}_{k-1}$ must be checked before starting the iterations. Härkegard requires that the previous solution is always feasible and updates the starting point in the provided code according to the working set:

Algorithm 2 Hotstart by Härkegard

for $i = 1$ to m **do**
  if $\mathcal{W}_{0,i} == -1$ **then**
    set $\mathbf{u}_i^{(0)} = \mathbf{u}_{\min}$
  end if
  if $\mathcal{W}_{0,i} == +1$ **then**
    set $\mathbf{u}_i^{(0)} = \mathbf{u}_{\max}$
  end if
end for

Schofield suggests that the control input is set to its nearest boundary of the feasible region:

Algorithm 3 Hotstart by Schofield

```
for i = 1 to m do
  if u_i ≤ u_min,i then
    set u_i = u_min,i
    set W_i = −1
  end if
  if u_i ≥ u_max,i then
    set u_i = u_max,i
    set W_i = 1
  end if
end for
```

As mentioned in [24], hotstart may also be detrimental to convergence as demonstrated by the following example:

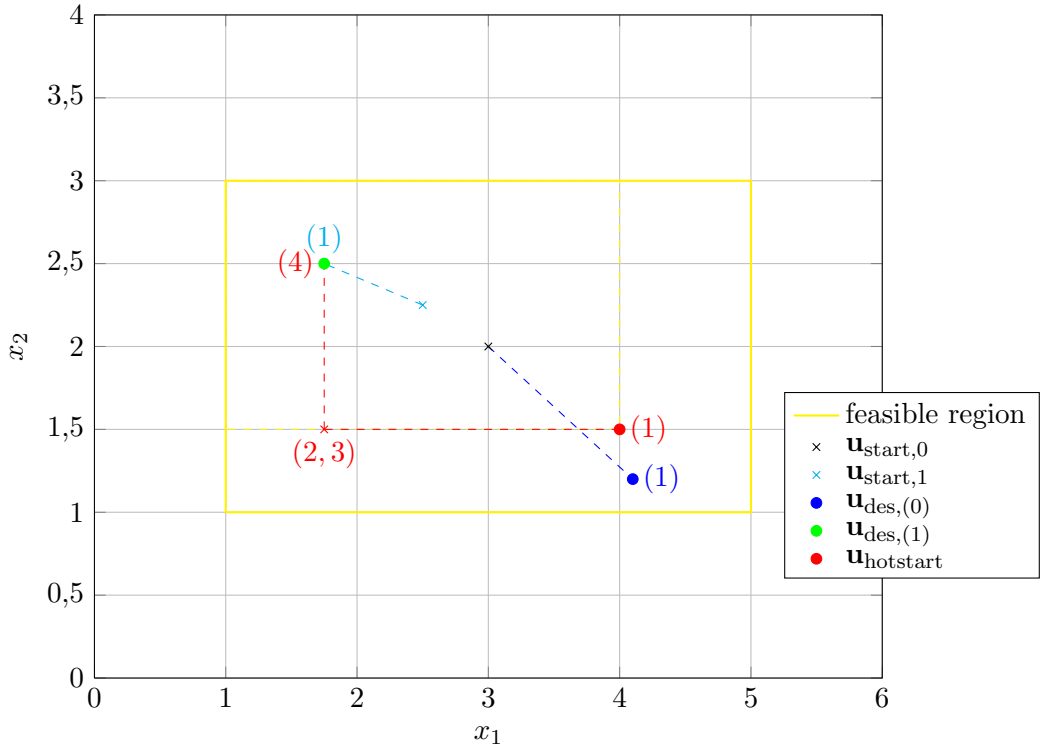


Figure 2.3: Example of a bad choice for hotstart.

In Figure 2.3 the constraints in the first time step are given by the yellow solid lines. The initial point is set to the middle of the feasible region since there is no other initial point available. The optimum is found in one iteration since it is in the feasible region and no variables have to be saturated or freed. In the second time step, if hotstart is used, the initial point is chosen to be the optimum from the previous time step, which is the blue dot $\mathbf{u}_{\text{des},0}$. At this time step the constraints are changed to the dashed yellow line. The new initial point when using hotstart obviously lies outside of the feasible region and must be set to the boundaries before continuing. In doing so, the two variables become saturated. To reach the optimum, both variables must be freed which takes two additional iterations. Iterating in the direction of the new optimum also takes two iterations which yields a total of four iterations. If no hotstart is used, the initial

point is again set to the middle of the *new* feasible region and only takes one iteration to find the optimum. This example shows that using hotstart with changing constraints may even yield a higher number of iterations.

2.4 Modified Active Set

A modified active set algorithm for solving bound-constrained problems is presented in [24]. The major drawback of active set methods, i.e. the computational complexity in worst case, is mitigated by a small change when adding constraints to the working set. The algorithm allows for more than one constraint per iteration to become active. Only constraints that satisfy the KKT-conditions become active, although others might be at their boundary as well. The modified active set is outlined in pseudo-code in Algorithm 4. Only the second part of the algorithm is slightly different from Algorithm 1.

Algorithm 4 Modified Active Set

set $\mathbf{u}^{(0)} = (\mathbf{u}_{\max} - \mathbf{u}_{\min})/2$, $\mathcal{W}_0 = \mathbf{0}$
for $i = 1$ to maximum number of iterations **do**
  solve

$$\begin{aligned} &\min \|\mathbf{A}(\mathbf{u}^{(i)} + \mathbf{p}) - \mathbf{b}\| \\ &p_i = 0, \quad i \in \mathcal{W} \end{aligned}$$

  where

$$\mathbf{A} = \begin{bmatrix} \sqrt{\gamma}\mathbf{W}_{\mathrm{v}}\mathbf{H} \\ \mathbf{W}_{\mathrm{u}} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \sqrt{\gamma}\mathbf{W}_{\mathrm{v}}\mathbf{v} \\ \mathbf{W}_{\mathrm{u}}\mathbf{u}_{\mathrm{des}} \end{bmatrix}.$$

  if $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \mathbf{p}$ is feasible **then**
    compute Lagrangian multipliers $\boldsymbol{\lambda}$
    if $\boldsymbol{\lambda} > \mathbf{0}$ **then**
      Optimum found, return
    else
      Remove constraint with negative λ_i
    end if
  else
    calculate $\boldsymbol{\Gamma}$ such that $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \boldsymbol{\Gamma}\mathbf{p}$ is feasible
    compute Lagrangian multipliers $\boldsymbol{\kappa}$
    add constraints with positive κ_i to working set
  end if
end for

In [24] a proof is presented that this algorithm is guaranteed to terminate in $2m - 1$ iterations.

2.5 Dynamic Control Allocation

The dynamic control allocation is an extension of the methods presented above. It allows for a distribution of control inputs in the frequency domain such that some actuators are used for low-frequency tasks while others work in the high-frequency domain. This distribution is implemented by considering previous samples of the control inputs:

$$\mathbf{u}(t) = \mathbf{f}(\mathbf{v}(t), \mathbf{v}(t-T), \mathbf{u}(t-T), \mathbf{v}(t-2T), \mathbf{u}(t-2T), ...), \tag{2.46}$$

with the sampling interval T. To penalize the change in control inputs, a new weighting matrix W_2 is introduced and the new optimization problem reads as follows:

$$\begin{aligned}\min_{\mathbf{u}} J =& \|\mathbf{W}_1(\mathbf{u}(t) - \mathbf{u}_{\text{des}}(t))\| + \|\mathbf{W}_2(\mathbf{u}(t) - \mathbf{u}(t-T))\|_2^2 \\ &+ \gamma\|\mathbf{W}_{\text{v}}(\mathbf{v}(t) - \mathbf{H}\mathbf{u}(t))\|_2^2.\end{aligned}$$

The higher the weighting factor $W_{2,(i,i)}$ the slower is the change of control input $\mathbf{u}_i$ and therefore the corresponding actuator only takes care of low-frequency tasks. Härkegard showed in [8] that this optimization problem can be seen as a filter when no saturations occur. Assuming that $\mathbf{W}_1$ and $\mathbf{W}_2$ are symmetric and $\mathbf{W}$ in 2.48 is nonsingular, the filter can be written as:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{F}\mathbf{x}_k + \mathbf{F}(\mathbf{E}\mathbf{u}_{\text{des}} + \mathbf{G}\mathbf{v}_k) \\ \mathbf{u}_k &= \mathbf{x}_k + \mathbf{E}\mathbf{u}_{\text{des}} + \mathbf{G}\mathbf{v}_k\end{aligned} \tag{2.47}$$

with

$$\begin{aligned}\mathbf{W} &= \sqrt{\mathbf{W}_1^2 + \mathbf{W}_2^2} \\ \mathbf{G} &= \mathbf{W}^{-1}(\mathbf{H}\mathbf{W}^{-1})^+ \\ \mathbf{F} &= (\mathbf{I} - \mathbf{G}\mathbf{H})\mathbf{W}^{-2}\mathbf{W}_1^2 \\ \mathbf{E} &= (\mathbf{I} - \mathbf{G}\mathbf{H})\mathbf{W}^{-2}\mathbf{W}_2^2\end{aligned} \tag{2.48}$$

The real eigenvalues of the filter are located between 0 and 1 if $\mathbf{W}_1$ is nonsingular. The proof can be found in [8]. The following example shows the effect of dynamic control allocation. Assuming that the dynamics of the actuators can be described by first-order lag elements with the time constant t_i, the transfer function of the actuator can be written as

$$G(s) = \frac{1}{1 + t_i \cdot s}.$$

Each actuator can be modelled separately. For this example the first actuator is assumed to be the "slowest" one that cannot respond to fast changes in the control input and is therefore used for low-frequency tasks. The time constants of the three actuators are chosen as follows:

$$\begin{aligned}t_1 &= 0.1, \\ t_2 &= 0, \\ t_3 &= 0.\end{aligned}$$

The second and third actuator are arbitrarily fast and can be used over the whole bandwidth. Given the equation

$$v = \mathbf{h}^T\mathbf{u} = \begin{bmatrix}2 & 1 & 1\end{bmatrix}\mathbf{u}, \tag{2.49}$$

$$v \in \mathbb{R}, \quad \mathbf{u} \in \mathbb{R}^3, \tag{2.50}$$

and the weights

$$\mathbf{W}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{W}_2 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

the resulting frequency behaviour can be seen in Figure 2.4. The weighting factor $W_{2,(1,1)}$ has been chosen such that the cut-off frequency equals the time constant. The first actuator is used for low-frequency tasks.

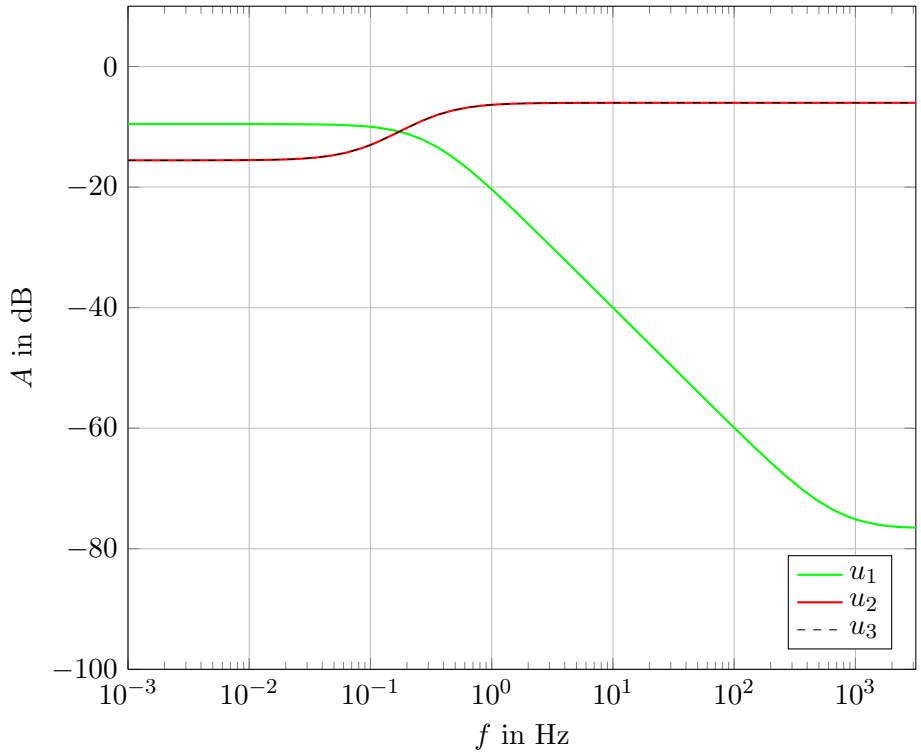


Figure 2.4: Example of the filtering properties of DCA.

The desired virtual control for this example is shown in Figure 2.5. It is a simple step function from 0 to 1 at $t = 1s$.

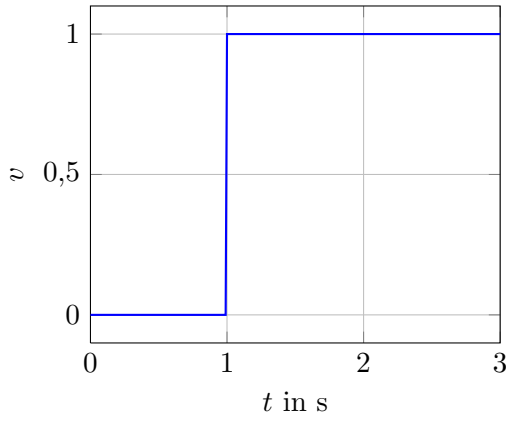
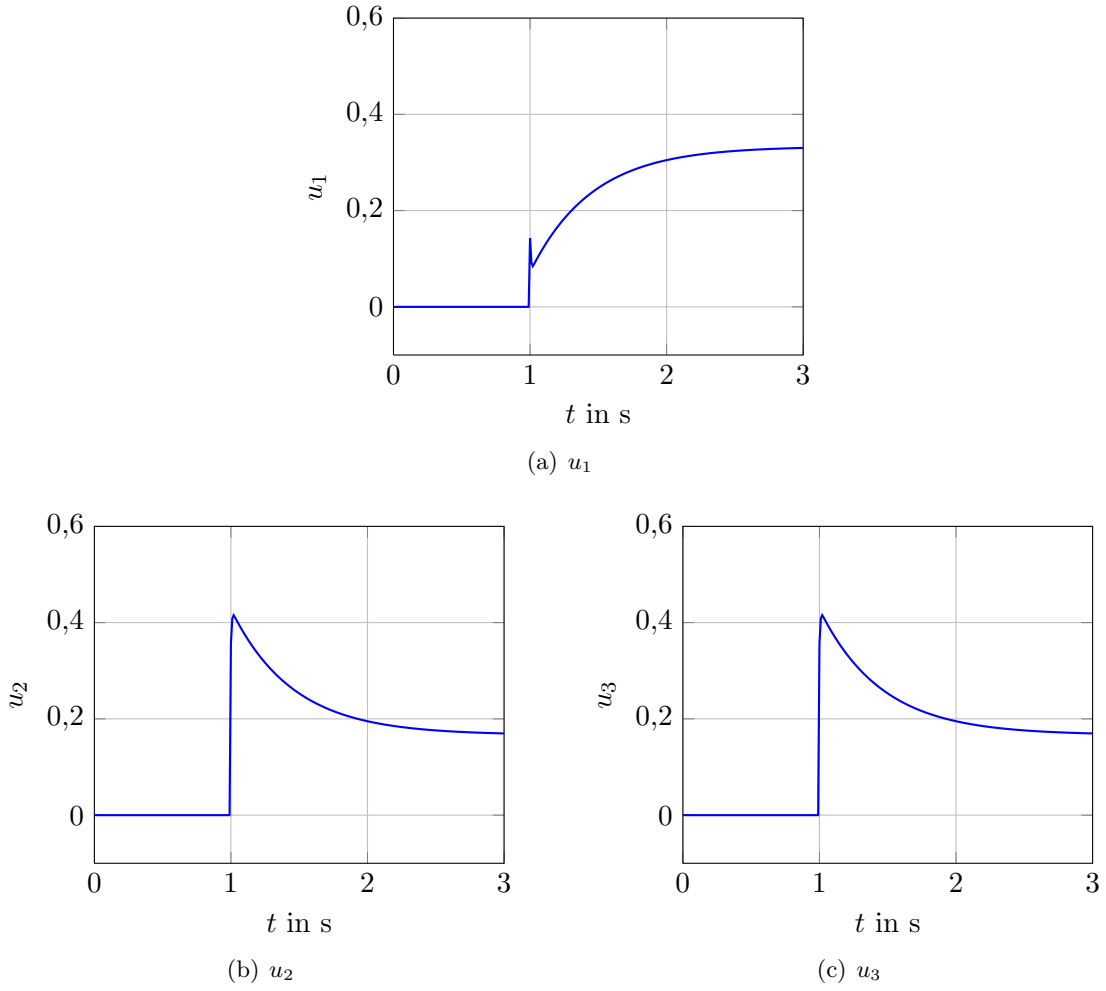


Figure 2.5: Virtual control of the DCA example.



(a) u_1

(b) u_2

(c) u_3

Figure 2.6: Control inputs of the DCA example.

The resulting control inputs for the actuators can be seen in Figure 4.10. The second and third actuators are used for the high-frequency part, i.e. the transients of v, while the first actuator slowly changes its control input until it reaches the steady state value. It is clear that the first actuator has a higher steady state value since it has the highest influence on v in $\mathbf{h}^T = \begin{bmatrix} 2 & 1 & 1 \end{bmatrix}$, and the desired control input $\mathbf{u}_{\text{des}}$ is $\mathbf{0}$ with equal weightings in $\mathbf{W}_1$.
The real eigenvalues of the resulting filter are located between 0 and 1 - the filter is stable:

$$\boldsymbol{\lambda} = \begin{bmatrix} 0.9714 \\ 0 \\ 0.5 \end{bmatrix}.$$

It is important to note that this dynamic approach only works this way as long as no saturations occur. As soon as one actuator saturates, the penalizing term $\|\mathbf{W}_2(\mathbf{u}(t) - \mathbf{u}(t-T))\|_2^2$ becomes rather irrelevant since the main task is to ensure $\mathbf{v} = \mathbf{Hu}$. This dynamic control allocation approach does not guarantee that the actuator dynamics are considered, it offers only the possibility to distribute the actuators over the frequency as long as the constraints hold.

Chapter 3

Application of Active Set Algorithms to a Road Vehicle

This chapter presents control allocation algorithms applied to a road vehicle as outlined in Chapter 1. The first section describes the simulation and necessary design parameters for control allocation, the second part of this chapter is devoted to the implemented active set algorithm and its different modifications.

3.1 Simulation

For the simulation, the vehicle starts at a velocity of 80 km/h, while the remaining states are zero, i.e.

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 80/3.6 \end{bmatrix}. \tag{3.1}$$

After one second ($t = 1$ s) the driver hits the brake pedal and deceleration starts with $0.4 \cdot m \cdot g$. m is the mass of the vehicle, g the acceleration due to gravity. In the simulation there are two systems: a passive one that is uncontrolled and only the desired control inputs are passed on to the actuators, and an active one where virtual controls are controlled as described in (1.12). Control allocation is used to determine the control inputs for the actuators. The velocities of the systems must be equal, lift and pitch of the active system should be improved compared to the passive one. In Figure 3.1, a simplified model in MATLAB/Simulink is presented. The step function with the deceleration gain on the left represents the driver hitting the brake pedal. The following gain `bkv` calculates the desired control inputs $\mathbf{u}_{\text{des}}$. The active system is in the upper and the passive system in the lower part. The active system requires state feedback for the controller and uses the desired control input, previous solutions and the current state for optimization, while the passive system passes the values for the control inputs directly to the actuators.

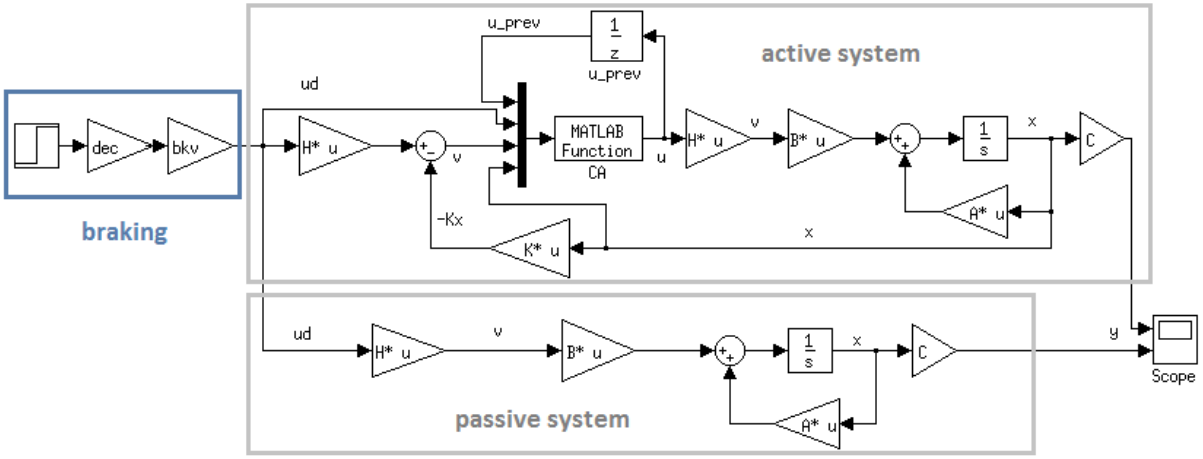


Figure 3.1: Simulink model with active and passive systems.

3.2 Definition of Control Allocation Variables

For the solution of the weighted least squares optimization problem

$$\min_{\mathbf{u}} J = \|\mathbf{W}_{\mathrm{u}}(\mathbf{u} - \mathbf{u}_{\mathrm{des}})\|_2^2 + \gamma\|\mathbf{W}_{\mathrm{v}}(\mathbf{v} - \mathbf{H}\mathbf{u})\|_2^2 \tag{3.2}$$

subject to the constraints

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}},$$

several variables must be defined. The controller calculates the values of the virtual control $\mathbf{v}$ which consist of the following factors:

$$\mathbf{v} = \begin{bmatrix} F_{\mathrm{z}} \\ T_{\mathrm{y}} \\ F_{\mathrm{x}} \end{bmatrix},$$

where F_{z} represents lift, T_{y} pitch and F_{x} the velocity of the car. The control inputs of the actuators are given by

$$\mathbf{u} = \begin{bmatrix} T_{\mathrm{H,f}}/r \\ T_{\mathrm{H,r}}/r \\ T_{\mathrm{B,f}}/r \\ T_{\mathrm{B,r}}/r \\ F_{\mathrm{a,f}} \\ F_{\mathrm{a,r}} \end{bmatrix}.$$

The torques are split up into those produced by hub-fixed devices T_{H} and others produced by body-fixed devices T_{B}. Additionally, there is a differentiation between front and rear torques, indicated by the subscripts f and r. The semi-active suspension forces are abbreviated by $\mathbf{F}_{\mathrm{a}}$, once again divided into front and rear forces. The control effectiveness matrix $\mathbf{H}$ is given by

$$\mathbf{H} = \begin{bmatrix} -\tan(\epsilon_{1,f}) & \tan(\epsilon_{1,r}) & -\tan(\epsilon_{2,f}) & \tan(\epsilon_{2,r}) & 1 & 1 \\ (\tan(\epsilon_{1,f}) \cdot l_f - h) & (\tan(\epsilon_{1,r}) \cdot l_r - h) & (\tan(\epsilon_{2,f}) \cdot l_r - h) & (\tan(\epsilon_{2,r}) \cdot l_r - h) & -l_f & l_r \\ 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

The parameters of this matrix are specified in Chapter 1 on page 11. The vectors $\mathbf{v}$, $\mathbf{u}$ and the matrix $\mathbf{H}$ are defined by the problem set. The additional design parameters are $\mathbf{W}_\mathrm{u}$, $\mathbf{W}_\mathrm{v}$, γ and $\mathbf{u}_\mathrm{des}$. These design parameters can be chosen individually. The weighting matrix $\mathbf{W}_\mathrm{u}$ penalizes the error between the control input $\mathbf{u}$ and the desired value $\mathbf{u}_\mathrm{des}$. Similarly, the matrix $\mathbf{W}_\mathrm{v}$ penalizes the error between the achieved virtual control $\mathbf{Hu}$ and the desired virtual control $\mathbf{v}$. These two matrices are typically diagonal matrices, weighting each divergence separately. Thus, the matrices are symmetric and nonsingular (i.e. no diagonal element is zero). The factor γ is a large number in order to guarantee that the virtual control is achieved primarily. These two design matrices are mainly chosen to be identity matrices with appropriate dimensions:

$$\mathbf{W}_\mathrm{v} = \mathbf{I}_3, \tag{3.3}$$

$$\mathbf{W}_\mathrm{u} = \mathbf{I}_6, \tag{3.4}$$

and the factor γ is 10^6. In most applications it is desirable to minimize the total control effort, hence the desired value $\mathbf{u}_\mathrm{des}$ is often chosen to be zero for all control inputs. The desired control inputs for this application are discussed in more detail in the next section.

3.2.1 Desired Control Input

The desired control input is the partition of braking forces. u_des is given by the following distribution of the actuators

$$\mathbf{u}_\mathrm{des} = \begin{bmatrix} 0.66 \cdot T_\mathrm{mech} \\ 0.34 \cdot T_\mathrm{mech} \\ 0.66 \cdot T_\mathrm{el} \\ 0.34 \cdot T_\mathrm{el} \\ 0 \\ 0 \end{bmatrix}, \tag{3.5}$$

where T_mech is the torque for the mechanical brakes and T_el the torque for the electric motors. During braking the vehicle mass is mainly on the front tires of the vehicle. Consequently, the distribution of the torques is chosen such that 66% of the braking force must be provided by the front axle, and 34% by the rear axle. This proportion might change slightly depending on the vehicle and application (cf. [20]). The simulations, however, use these percentages. The desired deceleration for the vehicle in the simulation is given by $0.4 \cdot m \cdot g$ with the body mass m and the gravity g. The hybrid part T_el is supposed to transfer 33% of the braking force, the mechanical part 67%. Altogether these partly user-defined distributions yield the desired control input $\mathbf{u}_\mathrm{des}$. In summary, the desired control input calculates as

$$\mathbf{u}_\mathrm{des} = 0.4 \cdot mg \cdot \begin{bmatrix} b \cdot t_\mathrm{mech} \\ (1-b) \cdot t_\mathrm{mech} \\ b \cdot (1 - t_\mathrm{mech}) \\ (1-b) \cdot (1 - t_\mathrm{mech}) \\ 0 \\ 0 \end{bmatrix}, \tag{3.6}$$

where $b = 0.66$ is the brake balance and t_mech is the proportion of the mechanical torque with $t_\mathrm{mech} = 0.67$ as mentioned above. The desired values are passed on directly to the passive system and should be achieved by the active system in steady state. This ensures that the braking behaviour of the active system equals the passive one. The semi-active suspensions play the major part in damping lift and pitch oscillations and are not considered in the passive system. In the active system, however, it is clear that these forces cannot be exactly equal to the desired values, otherwise the virtual control cannot be achieved and lift and pitch would not be improved.

3.3 Constraints

For control allocation the constraints play a major role. As already mentioned, one of the benefits of separating the control tasks into controller and control allocation is that constraints can be considered and failures can be detected. It is important to choose the constraints appropriately and check for degeneracies, otherwise the algorithm may not terminate. For this application, the constraints are assumed to be simply box-constrained. There are no linear dependencies, so degenerate solutions are excluded. For the simulation of the model in Chapter 1 the following configuration of actuators is considered:

- Brakes: hub-fixed,
- Motors: body-fixed,
- semi-active suspensions.

This configuration is rather common, e.g. used in the Mercedes-Benz SLS AMG E-Cell, see Figure 3.2.

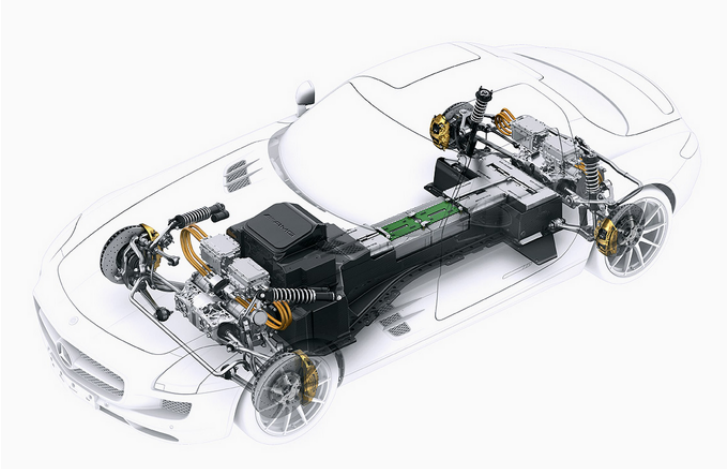


Figure 3.2: Mercedes-Benz SLS AMG E-Cell taken from Hambrecht, Lukas: "Mercedes lässt tief blicken" In: autobild.de on 12 March 2013, available at: `http://www.autobild.de/artikel/mercedes-sls-amg-e-cell-technik-2895646.html` (Accessed: 12 September 2013).

Each tire has one mechanical brake, one electric motor and one semi-active suspension. Since the tires are grouped into front and rear tires there are only 6 actuators. One should keep in mind that in reality these actuator groups can be again divided into right and front tires. The actuators have different constraints which partly depend on the states of the model. These dependencies do not complicate the control allocation since in every time step all states are known and a new optimization problem is considered. Time and state dependencies, e.g. $\mathbf{H} = \mathbf{H}(\mathbf{x}, t)$, can be considered without further changes in the implementation. The hub-fixed mechanical brakes can only apply negative torque:

$$-T_{\mathrm{H,max}} < T_{\mathrm{H}} < 0,$$

where the maximum torque is given as $T_{\mathrm{H,max}} = 2400$ Nm. The electric motors may apply positive and negative torques. The maximum torque depends on the power and velocity of the motor as shown in Figure 3.3.

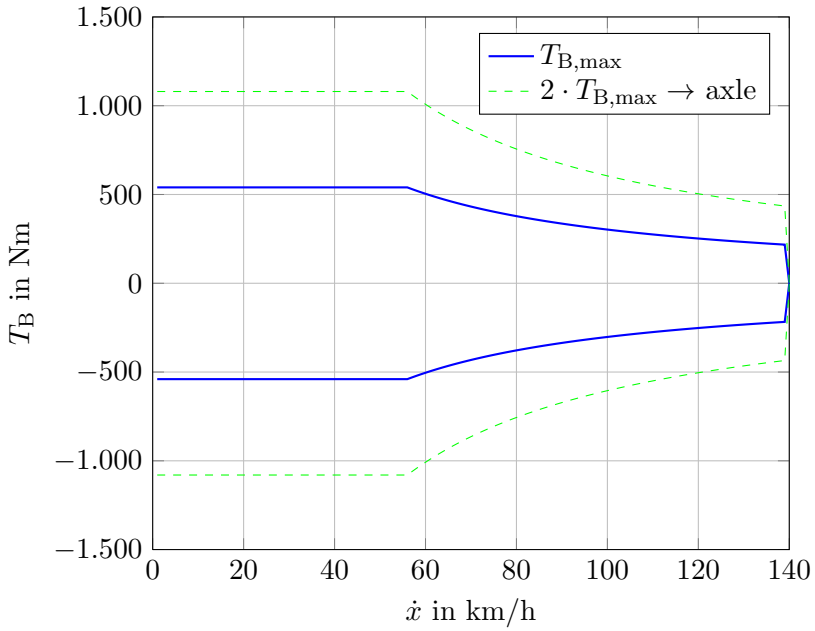


Figure 3.3: Constraints for the motors T_{B}.

In this example of application the power is $P = 28$ kW with a maximum attainable force $F_{\mathrm{max,abs}} = 600$ Nm. The maximum force per velocity calculates as

$$F_{\mathrm{max,vel}} = \frac{P}{v}, \tag{3.7}$$

$$F_{\mathrm{max}} = \min(F_{\mathrm{max,abs}}, F_{\mathrm{max,vel}}), \tag{3.8}$$

where v is the velocity of the vehicle. The constraints for semi-active suspensions are displayed in Figure 3.4. Semi-active suspensions can only counteract to the lowering vehicle. The forces depend on the velocity at the suspension $\dot{z}_a$. Downward movement implies negative velocity and therefore positive force since compressed dampers may be counteracted. It is not possible to enforce the downward movement by applying negative force since **semi**-active suspensions are used.

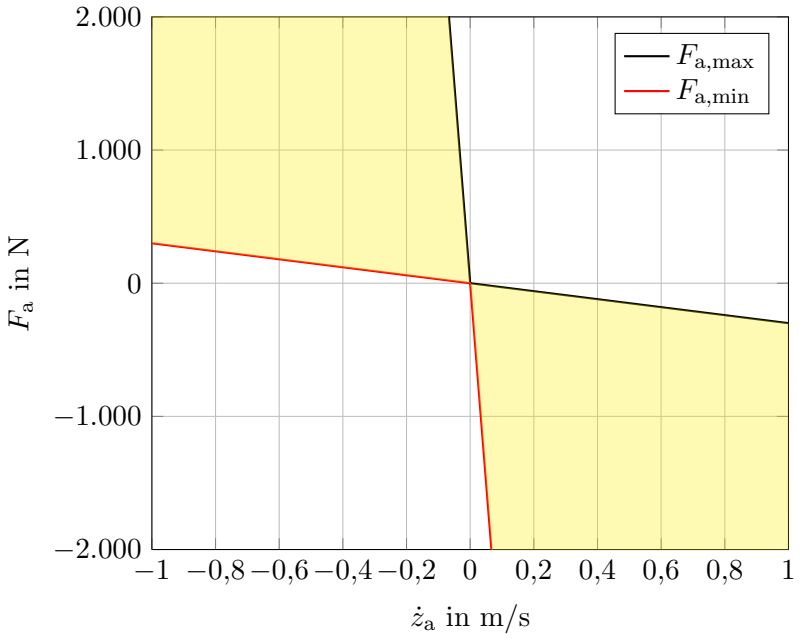


Figure 3.4: Constraints for the semi-active suspension F_{a}.

Once the design parameters and constraints are defined, the control allocation algorithms can be implemented and simulated on the vehicle. The following sections describe shortly how these algorithms are implemented on the model in Chapter 1.

3.4 Active Set Method

Active set methods achieve good results in automotive field and are consequently used for the given problem set. The design parameters are $\mathbf{W}_{\mathrm{v}}$, $\mathbf{W}_{\mathrm{u}}$, $\mathbf{u}_{\mathrm{des}}$ and γ. The usage of Härkegard's MATLAB-Code is straightforward and results can easily be achieved. The sequential least squares method (abbreviated by SLS) generates the exact solution to the optimization problem. With the weighting factor γ chosen high enough the weighted least squares method (WLS) shows equal results. The plots of the results are therefore combined and shown in one graph in the next chapter.

3.4.1 Hotstart

Hotstart seems at first to be a really good extension of the active set algorithm. Even when the maximum number of iterations is limited to one or two, the algorithm still achieves the goal of damping pitch and lift while the deceleration is not delayed. These good results are shown in the next chapter. Nevertheless hotstart can be a problem when constraints change a lot. This is the case with semi-active suspensions. The constraints of control input for the rear semi-active suspension $F_{\mathrm{a,r}}$ change a lot with the movement of the car. If the car does not move up or down anymore, the minimum and maximum value for the forces are almost the same. In the algorithm this control input causes trouble since it occupies a lot of iterations when saturating or freeing the optimization variable frequently.

3.4.2 Modified Active Set Method

Simulations show that the modified algorithm as described in Algorithm 4 for the given problem set solves the optimization in a maximum of three iterations per time step. Termination is guaranteed after 11 iterations, even when using fast-changing constraints like the semi-active suspensions. If the constraints are limited even further, the algorithm needs more iterations since more variables need to be freed in the optimization. If the maximum number of iterations is limited to less than three, the algorithm does not yield satisfying results. Since the results of the modified active set are identical to the ones obtained from SLS and WLS, additional plots are omitted for the sake of brevity.

3.5 Dynamic Control Allocation

The brakes are considered as "slow" actuators for the simulation with time constants chosen as

$$\begin{aligned} t_H &= 0.03, \\ t_B &= 0, \\ t_A &= 0. \end{aligned}$$

The transfer function is a first-order lag element:

$$G(s) = \frac{1}{1 + t_i \cdot s}.$$

The weighting matrices for the simulation are chosen as follows:

$$\mathbf{W}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{W}_2 = \begin{bmatrix} w_\mathrm{H} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_\mathrm{H} & 0 & 0 & 0 & 0 \\ 0 & 0 & w_\mathrm{B} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_\mathrm{B} & 0 & 0 \\ 0 & 0 & 0 & 0 & w_\mathrm{a} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_\mathrm{a} \end{bmatrix},$$

with

$$\begin{aligned} w_\mathrm{H} &= 10^{\frac{\log(j)}{2} + 1.5 + \frac{l}{2}}, \\ w_\mathrm{B} &= 1, \quad w_\mathrm{a} = 1. \end{aligned}$$

The calculation of the weighting factor w_H yields cut-off-frequencies according to $t_H = j \cdot 10^l$. For this example $j = 3$ and $l = -2$ applies, thus $w_\mathrm{H} = 5.4772$. The eigenvalues of the resulting filter are located between 0 and 1, i.e. the filter is stable:

$$\lambda = \begin{bmatrix} 0 \\ 0.9653 \\ 0.9380 \\ 0.5 \\ 0 \\ 0 \end{bmatrix}.$$

For the simulation the constraints of the actuators are neglected. Using the weighting matrices above the results show the expected behaviour, i.e. the motors are used for high frequencies and the brakes for lower frequencies. If the constraints are included, the weighting matrix $\mathbf{W}_2$ is only

part of the optimization problem. The major task is still to achieve the virtual control $\mathbf{v} = \mathbf{Hu}$, hence it is possible that weighting of the matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ is neglected. It is important to note that this dynamic control allocation offers a way to divide the actuators in high- and low-frequency parts if and only if the virtual control $\mathbf{v}$ is achieved. It is not guaranteed that the slow actuator actually reacts slowly in the simulation as long as $\mathbf{v} = \mathbf{Hu}$ does not hold.

3.6 Second Validation Maneuver

One benefit of separating the control task into controller and control allocation is that it is easier to handle actuator failures. Suppose the electric motors are applying negative torques and thereby charge the batteries of the motor. If the batteries are fully charged and the electric motors can no longer absorb energy, these actuators can no longer account for braking. Another example is a motor failure. In the simulation the constraints of the motors are both set to zero, i.e. the motors no longer work at all. To see the effects on lift and pitch, this is done at $t = 1.4$ s which is 0.4 s after hitting the brake pedal. The simulation shows very good behaviour, the brakes immediately take the part of the motors. There is no undesired change in lift and pitch movement, the virtual control can be achieved without difficulty. This result of course is expected since the optimization problem is solved every time step.

3.7 Comparison to other Quadratic Programming Algorithms

Several algorithms exist for solving quadratic programs. MATLAB offers the routine `quadprog` with several options, e.g. "active-set" can be chosen in order to solve the program. Older versions of MATLAB use the routine `qpdantz` which is implemented in `C` with an interface for MATLAB. This routine is rather fast and is widely used, hence it is compared to the active set algorithm by Härkegard.

3.7.1 Dantzig-Wolfe-Algorithm

The Dantzig-Wolfe algorithm is similar to the simplex method. It is used for quadratic programming and is described in detail in [5]. The MATLAB routine `qpdantz` is the implementation of this algorithm. For usage in the simulation, the optimization problem must be reformulated:

$$min \frac{1}{2}\mathbf{u}^T\mathbf{G}\mathbf{u} + \mathbf{f}^T\mathbf{u} \tag{3.9}$$

subject to

$$\tilde{\mathbf{C}}\mathbf{u} \leq \tilde{\mathbf{U}}, \quad \mathbf{u} \geq \underline{\mathbf{u}}, \tag{3.10}$$

where

$$\mathbf{A} = \begin{bmatrix} \sqrt{\gamma}\mathbf{W}_{\mathrm{v}}\mathbf{H} \\ \mathbf{W}_{\mathrm{u}} \end{bmatrix}, \tag{3.11}$$

$$\mathbf{b} = \begin{bmatrix} -\sqrt{\gamma}\mathbf{W}_{\mathrm{v}}\mathbf{v} \\ \mathbf{W}_{\mathrm{u}}\mathbf{u}_{\mathrm{des}} \end{bmatrix}, \tag{3.12}$$

$$\mathbf{f} = \mathbf{b}^T\mathbf{A}. \tag{3.13}$$

The Hessian matrix $\mathbf{G}$ must be positive definite:

$$\tilde{\mathbf{A}} = \mathbf{A}^T\mathbf{A} \tag{3.14}$$

$$\mathbf{G} = \frac{1}{2}(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T). \tag{3.15}$$

The constraints for the minima can be passed immediately while the constraints for the maxima must be defined via $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{U}}$:

$$\tilde{\mathbf{C}} = \mathbf{I}_m, \quad \tilde{\mathbf{U}} = \overline{\mathbf{u}}. \tag{3.16}$$

The call `u = qpdantz(G,f,`$\tilde{\mathbf{C}}$`,`$\tilde{\mathbf{U}}$`,`$\underline{\mathbf{u}}$`,`*iter_max*`)` returns the solution for $\mathbf{u}$. The results are satisfying as long as the maximum number of iterations is sufficiently high.

3.7.2 qpOases

Several algorithms for solving constrained least squares problems are already available as code, e.g. qpOases (see `http://www.kuleuven.be/optec/software/qpOASES`). qpOases is written in C but offers an interface for MATLAB and has been tested on the given model. Although C-code is very fast, the MATLAB-routine consumes more time than the active set methods by Härkegard. The MATLAB-call is similar to the Dantzig-Wolfe-Algorithm:
`qpOASES(G,f,`$\underline{\mathbf{u}}$`,`$\overline{\mathbf{u}}$`,`$\mathbf{u_0}$`,`*qp_opts*`)` where $\mathbf{G}$ and $\mathbf{f}$ are the same matrizes as defined above and $\mathbf{u}_0$ is the starting point for the current time step. The maximum number of iterations can be defined in *qp_opts*. The results are satisfying but the computation time is higher than the active set methods.

3.8 Two-Phase Algorithm

The velocity of the active system must be equal to the velocity of the passive system. It is necessary to guarantee that the braking force F_x is always achieved exactly. The first approach is to use the current algorithms with a large weighting factor for the third virtual control:

$$\mathbf{W}_\mathrm{v} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10^6 \end{bmatrix}, \tag{3.17}$$

which ensures that the error for F_x is minimized primarily. If the constraints are more stringent and the iterations are limited strictly, the algorithm may be forced to stop at a suboptimal solution. Suboptimal solutions do not guarantee that the virtual force F_x is achieved, hence this approach is not satisfying. For this reason another algorithm has been implemented that calculates the control inputs only for the correct deceleration in the first step. If there are enough resources left to continue the computation, the second step is entered and the other two virtual forces are considered. This modification shall be called the two-phase algorithm and is outlined in algorithm 5. In the first phase the problem is reduced to ensure that the virtual braking force F_x is always achieved, even if the algorithm terminates at a suboptimum. At first only the brakes and the electric motors are considered for control allocation:

$$v^* = F_\mathrm{x}, \tag{3.18}$$

$$\mathbf{u}^* = \begin{bmatrix} T_\mathrm{H,f} \\ T_\mathrm{H,r} \\ T_\mathrm{B,f} \\ T_\mathrm{B,r} \end{bmatrix}. \tag{3.19}$$

The problem reduces to only one virtual control and four control inputs for the actuators. The optimization problem for this application can be solved optimally in two iterations. Note that these two iterations take less computation time than two iterations with the standard approach since the optimization problem becomes significantly smaller. During these two iterations the algorithm must not be interrupted, otherwise it is not guaranteed that the optimal solution for $\mathbf{u}^*$ is returned. If the power and maximum torque of the electric motors is chosen such that none of the control inputs becomes saturated in the first iteration, there is no need for a second iteration in the first phase. Accordingly, if the desired value for the control inputs $\mathbf{u}_{\text{des}}$ is attainable at all times, phase 1 only takes one iteration. In phase 2 the algorithm starts with the calculated control input $\mathbf{u}^*$ and working set $\mathcal{W}$ from phase 1 and 0 for the remaining variables for semi-active suspensions F_{a} and $\mathcal{W}_{F_{\text{a}}}$. In each iteration the results improve until the optimum is reached as in the methods above. An interruption in phase 2 does not violate the deceleration F_{x}. The reason why the original algorithm takes more iterations are the changing constraints of the semi-active suspensions. Since these changes may keep the algorithm busy, it may happen that the remaining control inputs are disregarded. The algorithm can only free one active constraint per iteration, so very often the semi-active suspension constraints are set free or active. In the modified algorithm with two phases, the important control inputs for deceleration are calculated in the first step, then the "problematic" control inputs F_{a} are considered. Therefore it is guaranteed that the deceleration is achieved, regardless of the optimality of the solution. An outline of this modified algorithm in pseudo-code can be seen in Algorithm 5.

Algorithm 5 2-Phase Active Set

—— PHASE I ——

for $i = 1$ to 2 **do**

solve

$$\begin{aligned} &\min \|\mathbf{A}(\mathbf{u}^{(i)} + \mathbf{p}) - \mathbf{b}\| \\ &p_i = 0, \quad i \in \mathcal{W} \end{aligned}$$

$\mathbf{u} \in \mathbb{R}^4$, $v \in \mathbb{R}$

where

$$\mathbf{A} = \begin{bmatrix} \sqrt{\gamma} W_{\mathrm{v}}^* \mathbf{h}^T \\ \mathbf{W}_{\mathrm{u}}^* \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \sqrt{\gamma} W_{\mathrm{v}}^* v^* \\ \mathbf{W}_{\mathrm{u}}^* \mathbf{u}_{\mathrm{des}}^* \end{bmatrix},$$

if $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \mathbf{p}$ is feasible **then**

compute Lagrangian multipliers $\boldsymbol{\lambda}$

if $\boldsymbol{\lambda} > 0$ **then**

Optimum found, go to PHASE II

else

Remove constraint with negative λ_i

end if

else

calculate $\boldsymbol{\Gamma}$ such that $\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \boldsymbol{\Gamma}\mathbf{p}$ is feasible

compute Lagrangian multipliers $\boldsymbol{\kappa}$

add constraints with positive κ_i to working set

end if

end for

—— PHASE II ——

for $i = 2$ to maximum number of iterations **do**

solve

$$\begin{aligned} &\min \|\mathbf{A}(\mathbf{u}^{(i)} + \mathbf{p}) - \mathbf{b}\| \\ &p_i = 0, \quad i \in \mathcal{W} \end{aligned}$$

$\mathbf{u} \in \mathbb{R}^6$, $\mathbf{v} \in \mathbb{R}^3$

where

$$\mathbf{A} = \begin{bmatrix} \sqrt{\gamma} \mathbf{W}_{\mathrm{v}} \mathbf{H} \\ \mathbf{W}_{\mathrm{u}} \end{bmatrix}, \quad b = \begin{bmatrix} \sqrt{\gamma} \mathbf{W}_{\mathrm{v}} \mathbf{v} \\ \mathbf{W}_{\mathrm{u}} \mathbf{u}_{\mathrm{des}} \end{bmatrix}.$$

continue as in Algorithm 4

end for

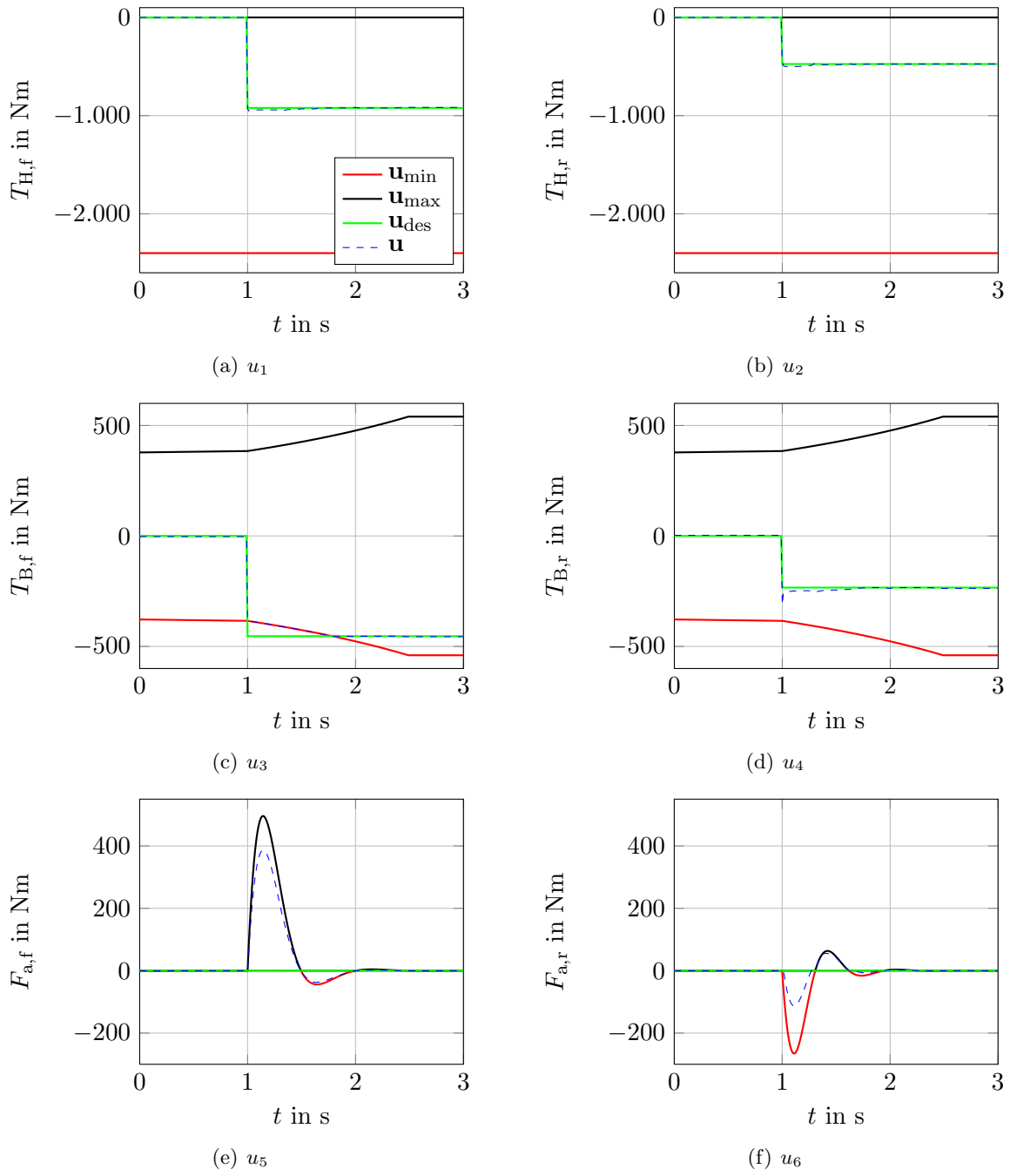
Chapter 4

Results

This chapter presents the results of the different methods in the simulations explained in Chapter 3.

4.1 Sequential and Weighted Least Squares Algorithms

Figures 4.1 show that the sequential least squares (SLS) algorithm yields the same results as the weighted least squares (WLS) algorithm. The only difference of the algorithms is the required number of iterations: SLS takes approximately 2.4 iterations on average, while WLS only requires 1.05 iterations. The weighting matrices are chosen as $W_v = \mathbf{I}_3$, $W_u = \mathbf{I}_6$. The maximum number of iterations is $iter\_max = 100$. The algorithm terminates at all times with the optimal solution. Figure 4.1 shows the control inputs of the actuators. The constraints are illustrated by the black and red lines. The desired control input $\mathbf{u}_{\text{des}}$ is represented by the green line. This green line is at the same time the control input of the passive system. The solution of the optimization algorithms is $\mathbf{u}$, which is represented by the blue dotted line. The power and maximum torque of the electric motors are limited such that the desired value at $t = 1$ s can not be achieved by the front motor. This choice is on purpose since it shows that the constraints are not violated even though the desired value has not been reached. It is clear that the brakes and the rear motors have to apply more torque in order to balance this error in virtual control. As soon as the desired values can be reached, the control inputs set for these values. The constraints of the semi-active suspensions vary strongly due to the fact that they depend on the pitch movement of the vehicle. Nevertheless, the control inputs never exceed their constraints. In Figure 4.2 the virtual control is shown. The desired values can at all times be achieved. While the pitch and lift factors are calculated by the controller the deceleration in F_{x} is the same as for the passive system. At the step $t = 1$ s the constant deceleration starts. The resulting states are presented in Figure 4.3. Lift z and pitch θ of the active system show much better behaviour, the oscillations of these states are damped. Figure 4.4 shows the most important state of the vehicle: the velocity. It is the same for both the passive and the active system. These results show that while maintaining the deceleration lift and pitch can be improved significantly by the WLS and SLS algorithms.

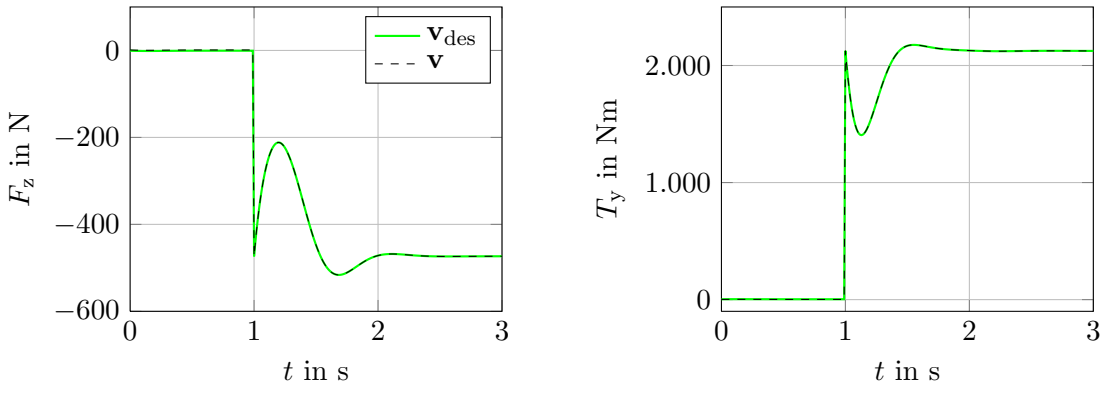


(a) u_1 (b) u_2

(c) u_3 (d) u_4

(e) u_5 (f) u_6

Figure 4.1: Control inputs of actuators using SLS or WLS.

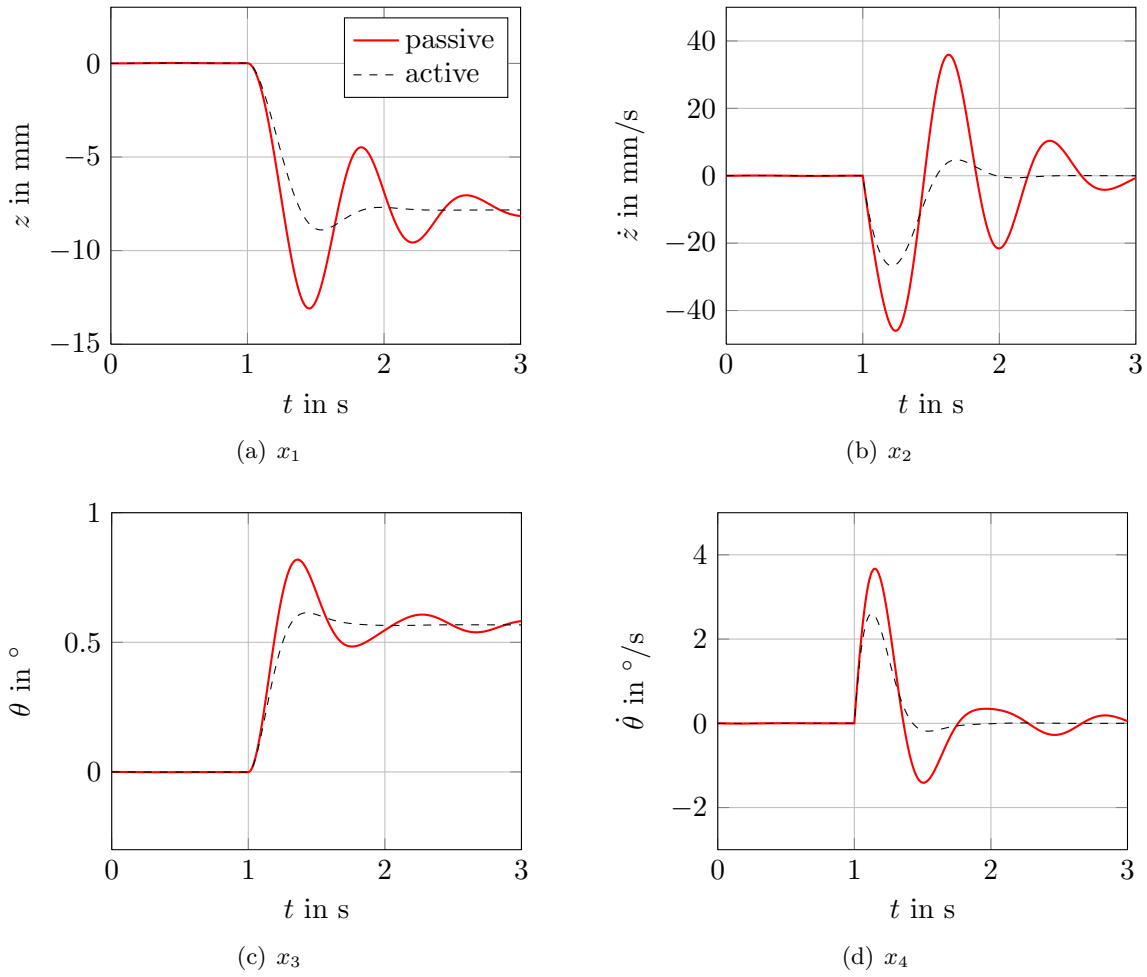


(a) v_1

(b) v_2

(c) v_3

Figure 4.2: Virtual control using SLS or WLS.



(a) x_1 (b) x_2

(c) x_3 (d) x_4

Figure 4.3: Resulting states using SLS or WLS.

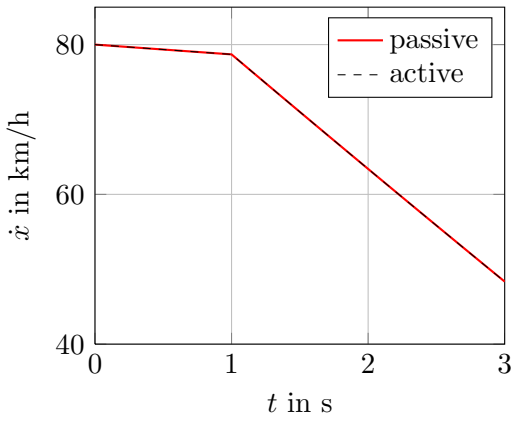


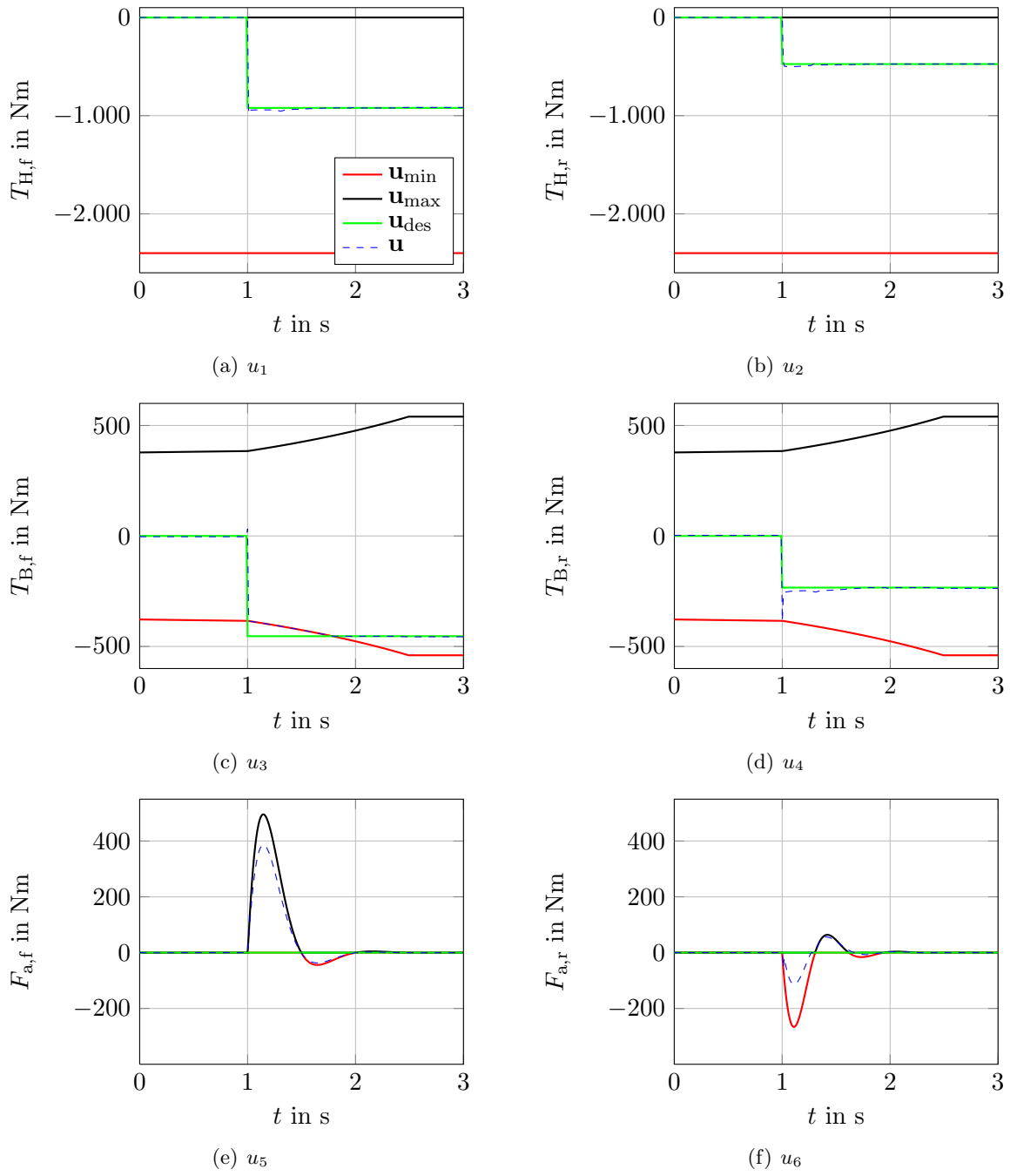
Figure 4.4: Velocity using SLS or WLS.

4.1.1 Hotstart

Hotstart yields good results even when the number of iterations is limited to one. In this case, the optimum might not be found exactly at all times. The suboptimum is only suboptimal regarding the second objective function $\mathbf{W}_{\mathrm{u}}(\mathbf{u}-\mathbf{u}_{\mathrm{des}})$. Therefore, the suboptimum is very often sufficient if only applied for a few time steps. At the sudden change of the desired variables, i.e. at the onset of braking, only the suboptimum is reached. It takes approximately 10 time steps ($10 \cdot 0.001s$) to find the optimum of the entire optimization problem. Figures 4.5 - 4.8 show the results when using hotstart. The weighting matrices are simply $W_v = \mathbf{I}_3$, $W_u = \mathbf{I}_6$. In order to demonstrate that the algortihm still yields adequate results, the maximum number of iterations is limited to one. The control input for the actuators $\mathbf{u}$ may differ slightly from the results without limitation for a few time steps, but the virtual control can be achieved sufficiently and the states show satisfying behaviour.

4.1.2 Modified Active Set

As shown in Chapter 3, the modified algorithm yields the same results as the WLS or SLS methods, but only takes at most $2m-1$ iterations. For the problem at hand this means that for the 6 actuators the maximum number of iterations $iter\_max$ is guaranteed to be $2 \cdot 6 - 1 = 11$ iterations. The design parameters are again chosen as $\mathbf{W}_{\mathrm{v}} = \mathbf{I}_3$, $\mathbf{W}_{\mathrm{u}} = \mathbf{I}_6$. The algorithm shows exactly the same results as the other methods. The modified algorithm needs on average more iterations than the WLS algorithm, but the maximum number of used iterations for the modified algorithm is less than the maximum of the WLS method.

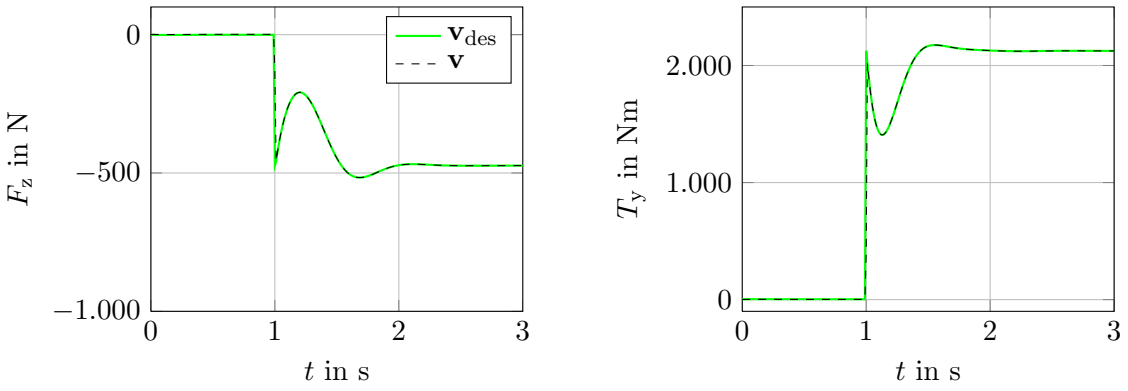


(a) u_1 (b) u_2

(c) u_3 (d) u_4

(e) u_5 (f) u_6

Figure 4.5: Control inputs of actuators using hotstart.

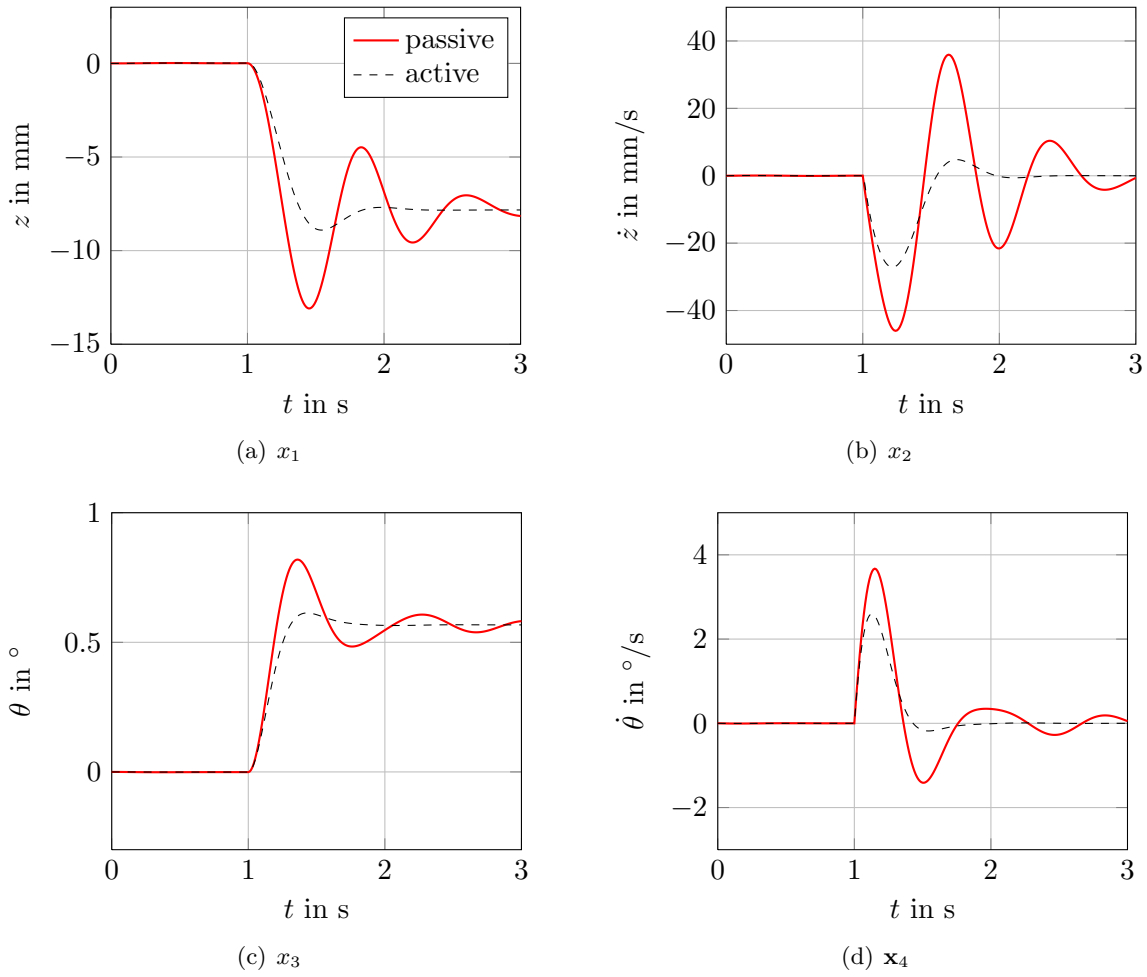


(a) v_1

(b) v_2

(c) v_3

Figure 4.6: Virtual control using hotstart.



(a) x_1 (b) x_2

(c) x_3 (d) $\mathbf{x}_4$

Figure 4.7: States using hotstart.

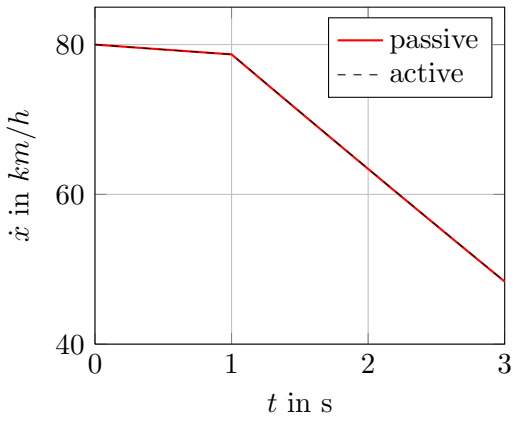


Figure 4.8: Velocity using hotstart.

4.2 Dynamic Control Allocation

As discussed in the previous chapter, the dynamic control allocation (DCA) allows for partitioning of frequency bandwitdh among the actuators. The weighting matrices for this example are chosen as follows: $\mathbf{W}_{\mathrm{v}} = \mathbf{I}_3$, $\mathbf{W}_1 = \mathbf{I}_6$, and

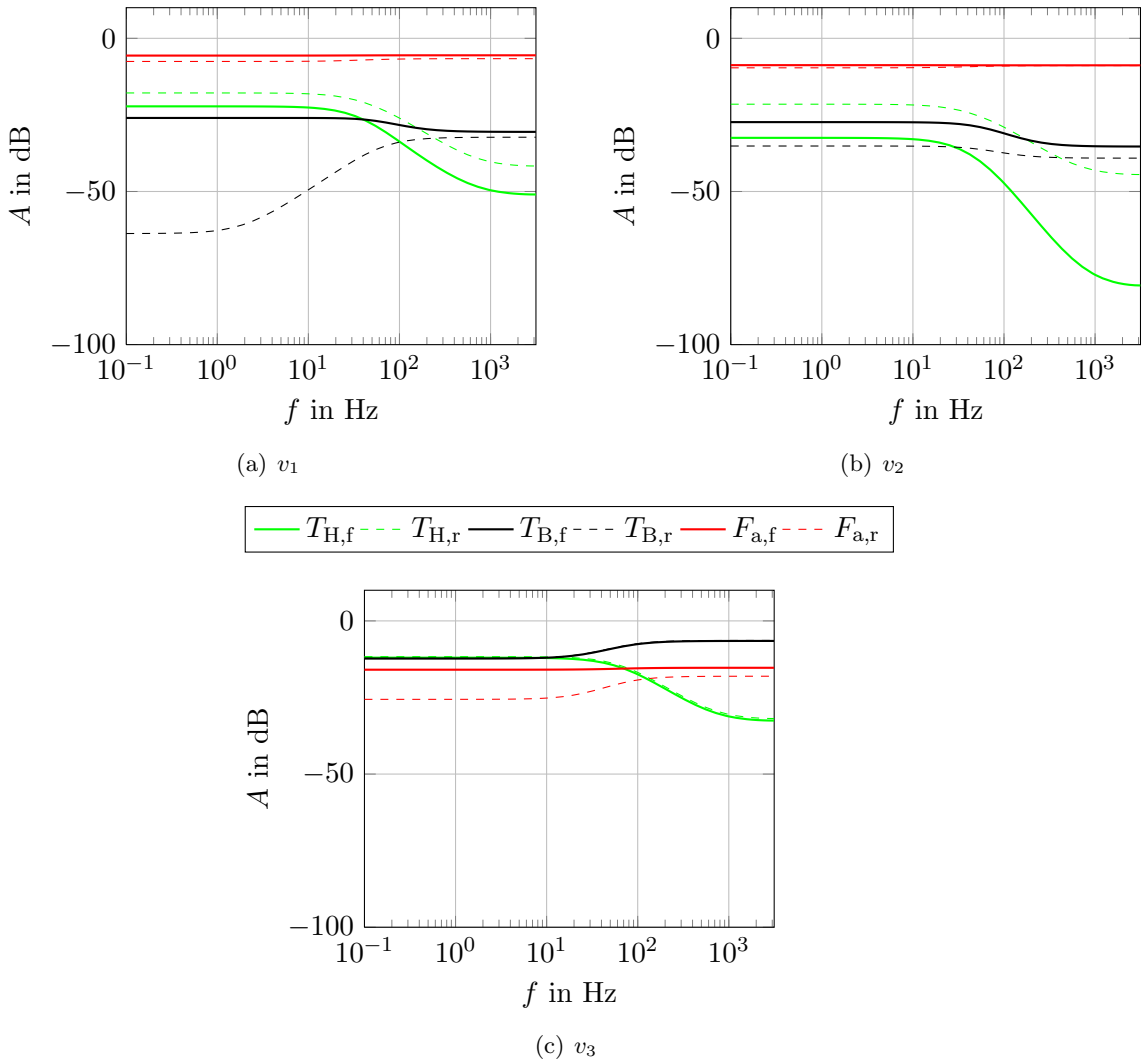
$$\mathbf{W}_2 = \begin{bmatrix} w_{\mathrm{H}} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{\mathrm{H}} & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{\mathrm{B}} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{\mathrm{B}} & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{\mathrm{a}} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{\mathrm{a}} \end{bmatrix},$$

with

$$\begin{aligned} w_{\mathrm{H}} &= 10^{\frac{\log(3)}{2}+1.5+\frac{-2}{2}} = 5.4772, \\ w_{\mathrm{B}} &= 1, \\ w_{\mathrm{a}} &= 1. \end{aligned}$$

These choices for the design parameters yield the frequency partition shown in Figure 4.9. The time constant of the brakes is $t_{\mathrm{H}} = 0.03$, the corresponding cut-off frequency is

$$f_{\mathrm{H}} = \frac{1}{t_{\mathrm{H}}} = \frac{1}{0.03} = 33.33.$$

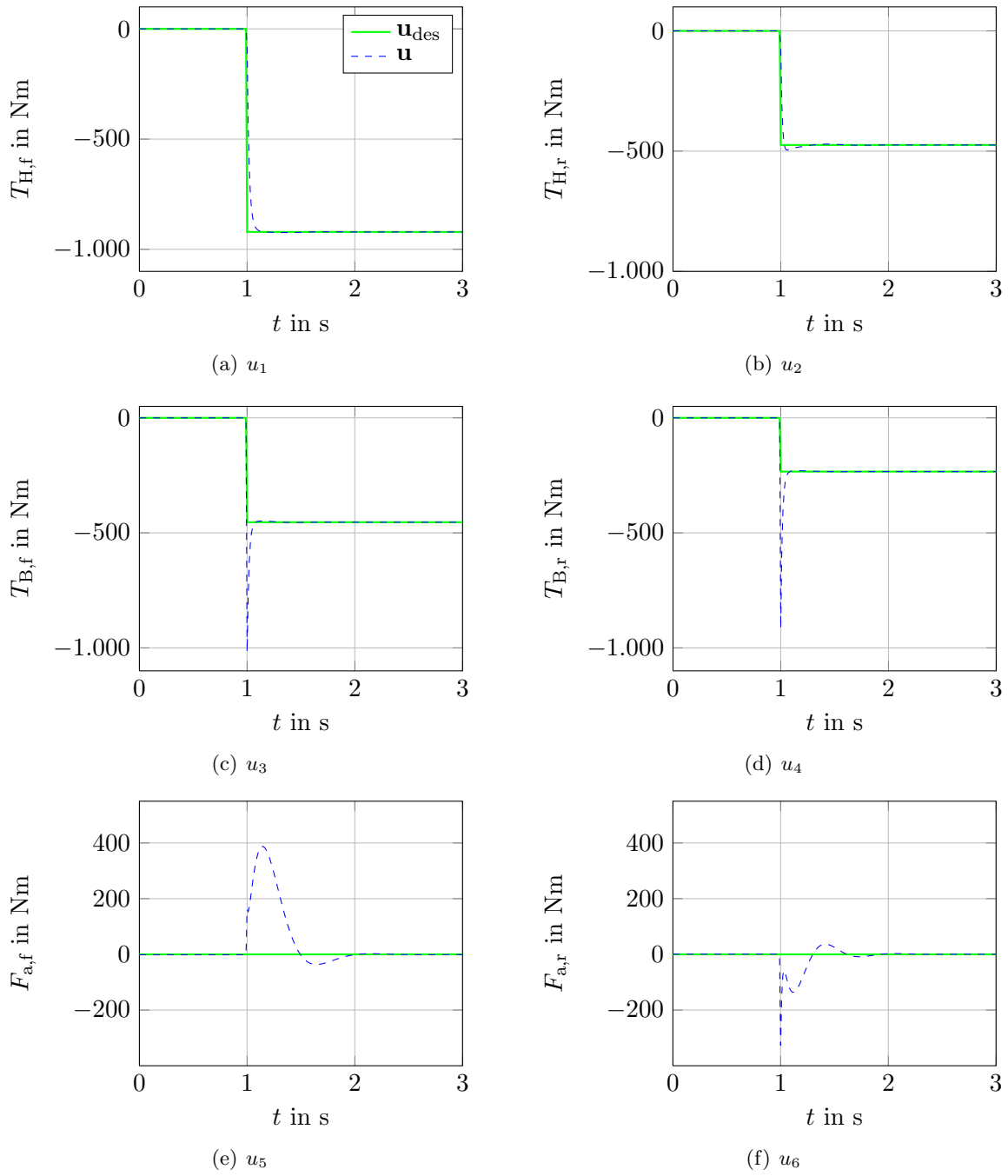


(a) v_1 (b) v_2

(c) v_3

Figure 4.9: Filtering behaviour of the Dynamic Control Allocation.

Without constraints the resulting control inputs in Figure 4.10 show the prioritization of the "fast" motors and suspensions when braking starts: Since the brakes cannot react fast enough when deceleration starts, other control inputs must take over their part. The motors are considered to be high-frequency components and apply quite high torques when the step occurs. The semi-active suspensions also show different behaviour since constraints are neglected.

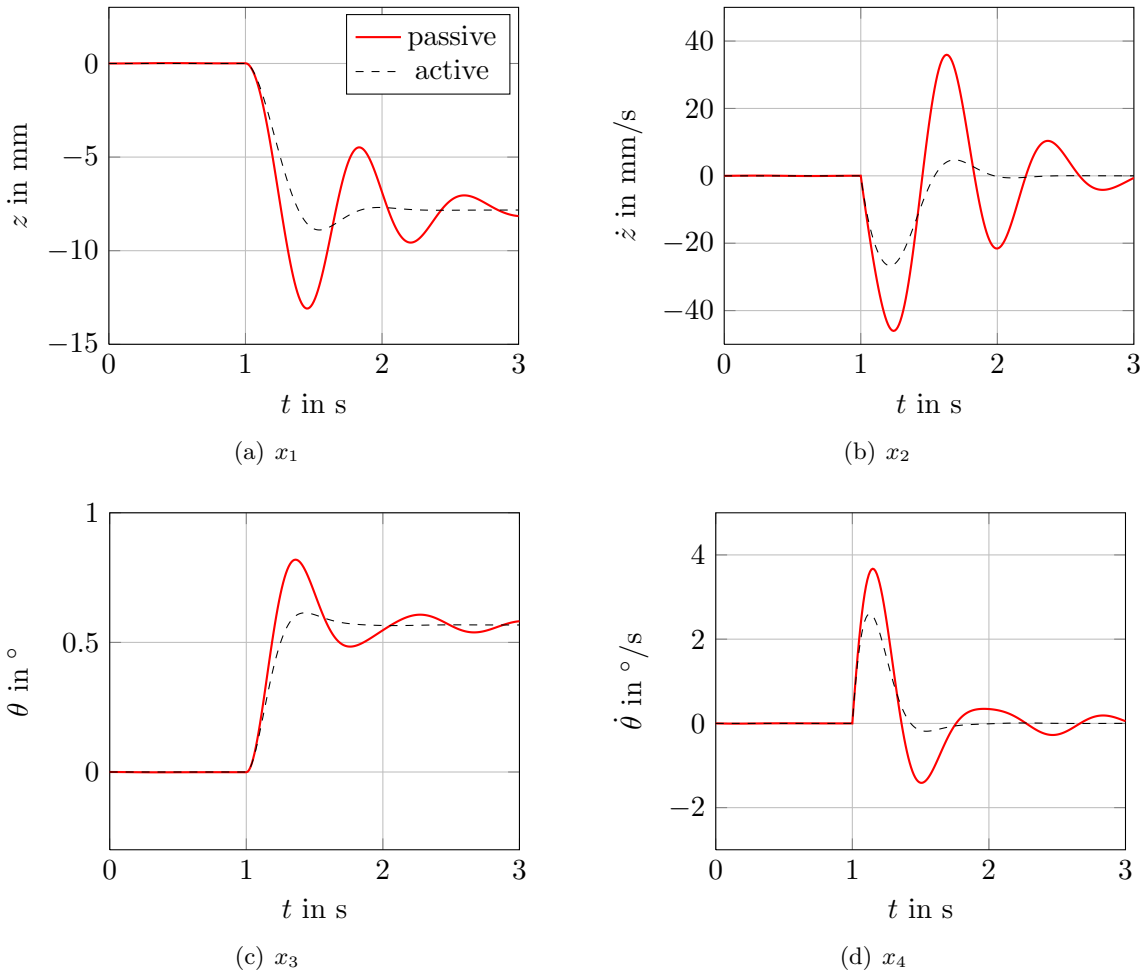


(a) u_1 (b) u_2

(c) u_3 (d) u_4

(e) u_5 (f) u_6

Figure 4.10: Control inputs of actuators using DCA (no constraints).



(a) x_1 (b) x_2

(c) x_3 (d) x_4

Figure 4.11: States using DCA.

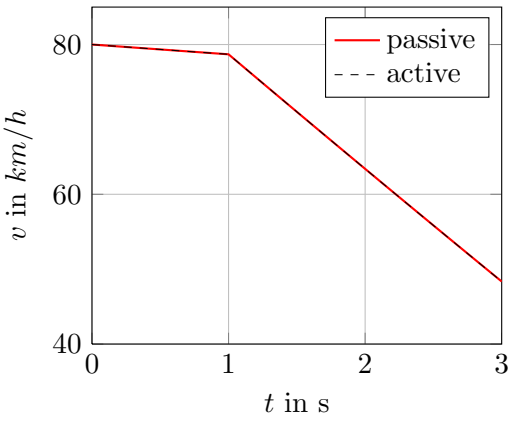
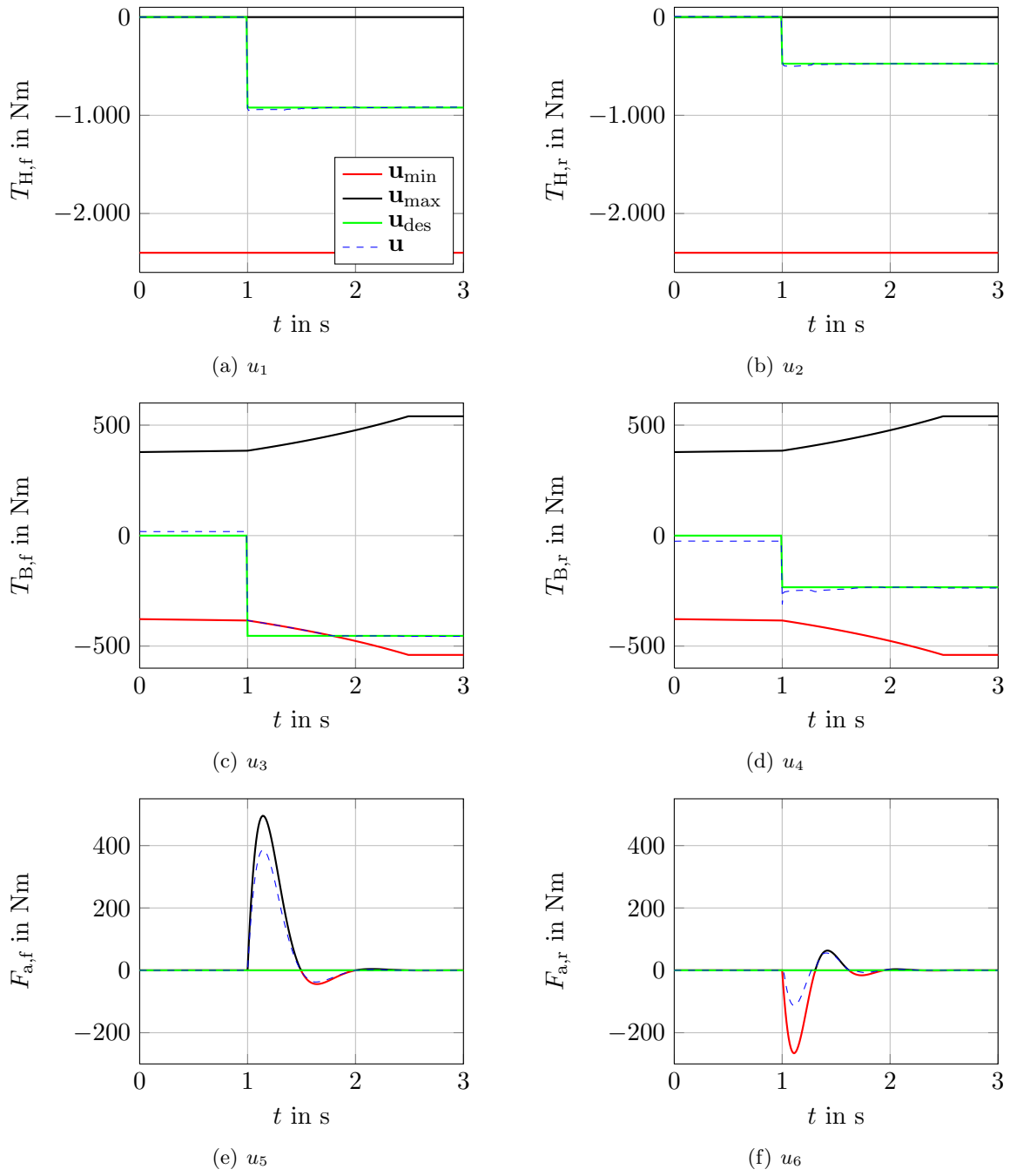


Figure 4.12: Velocity using DCA.

4.3 Dantzig-Wolfe Algorithm

The Dantzig-Wolfe algorithm has also been simulated with the same choice of parameters, that is $\mathbf{W}_\mathrm{v} = \mathbf{I}_3$, $\mathbf{W}_\mathrm{u} = \mathbf{I}_6$. Reducing the maximum number of iterations to three, the algorithm starts having problems while the WLS method still operates as desired. If the maximum number of iterations is large enough, this algorithm yields the same results as the active set methods. Methods similar to the simplex algorithm usually allow for the use of hotstart, while the MATLAB-routine `qpdantz` does not offer the possibility for hotstart. In this simulation the hotstart has not been implemented separately. The control inputs vary from the desired values from the beginning which is very undesired. Nevertheless the virtual control and states can be achieved sufficiently. Figure 4.13 shows the control inputs for the actuators which differ from the optimal solution especially before the deceleration starts.



(a) u_1 (b) u_2

(c) u_3 (d) u_4

(e) u_5 (f) u_6

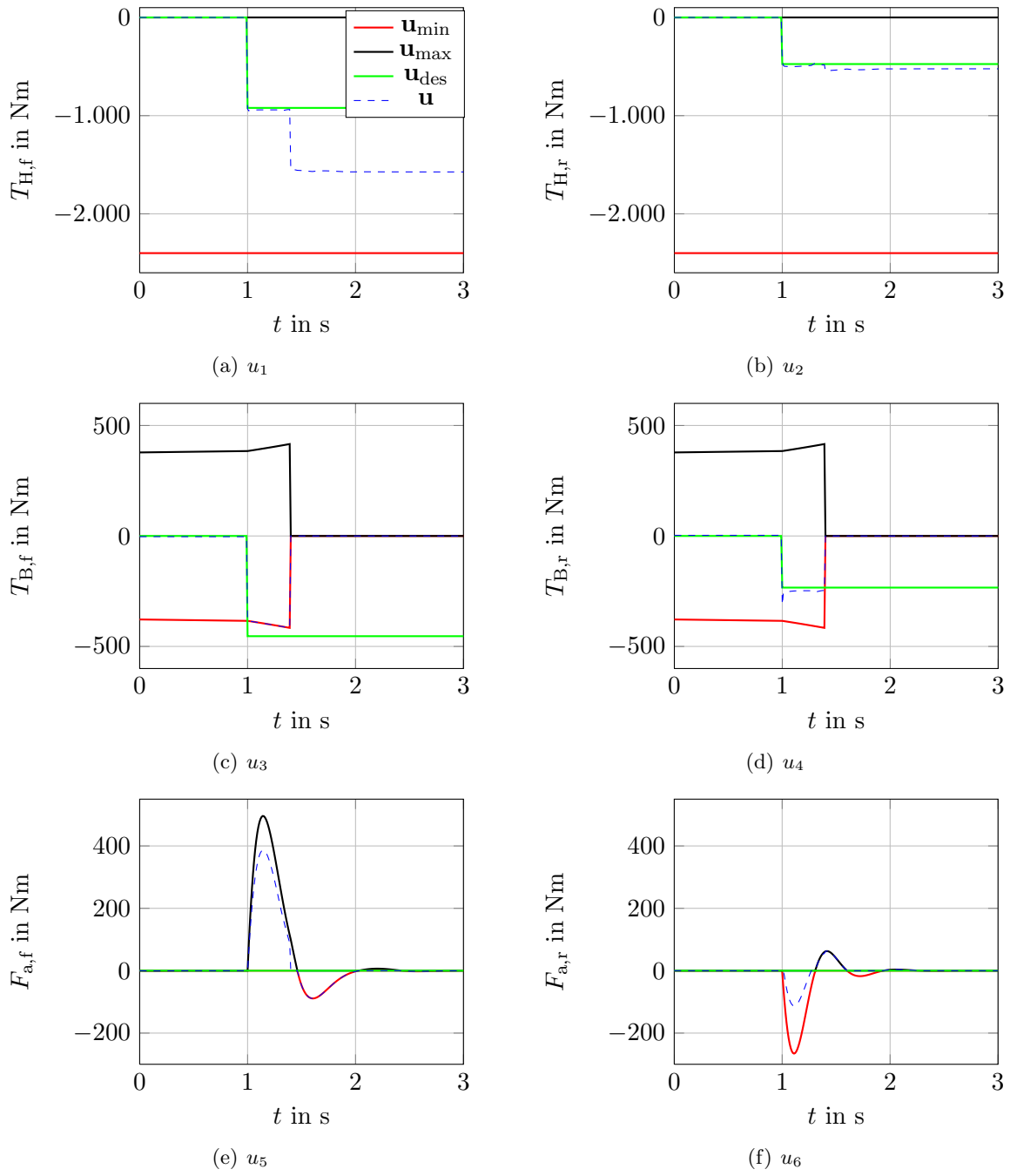
Figure 4.13: Control inputs of actuators using `qpdantz`.

4.4 Second Validation Maneuver

If the motors are defective or can no longer apply negative torque because the battery is fully charged, the control inputs must be distributed accordingly. In the simulation the constraints of the motors are simply set to zero at $t = 1.4s$. The parameters for the motors are

$$T_{\mathrm{B}} = \begin{cases} 600 \text{ Nm}, & t < 1.4s \\ 0 \text{ Nm}, & t \geq 1.4s \end{cases}$$

The weighting matrices are again $\mathbf{W}_{\mathrm{v}} = \mathbf{I}_3$, $\mathbf{W}_{\mathrm{u}} = \mathbf{I}_6$ and the maximum number of iterations is set to 50. Figure 4.14 shows the behaviour of the system for this test case.
The virtual control can not be achieved correctly as shown in Figure 4.15
Since the virtual control does not reach the desired values, the states differ from the optimal solution, too. In Figure 4.16 and 4.17 the resulting states are presented. The results are compared to the WLS method. Lift and pitch cannot be improved as well as with the WLS method but are still much better than without control allocation. When motors are defective, the most important task is to guarantee the deceleration. The velocity for the active system is still the same, so these results are satisfactory.

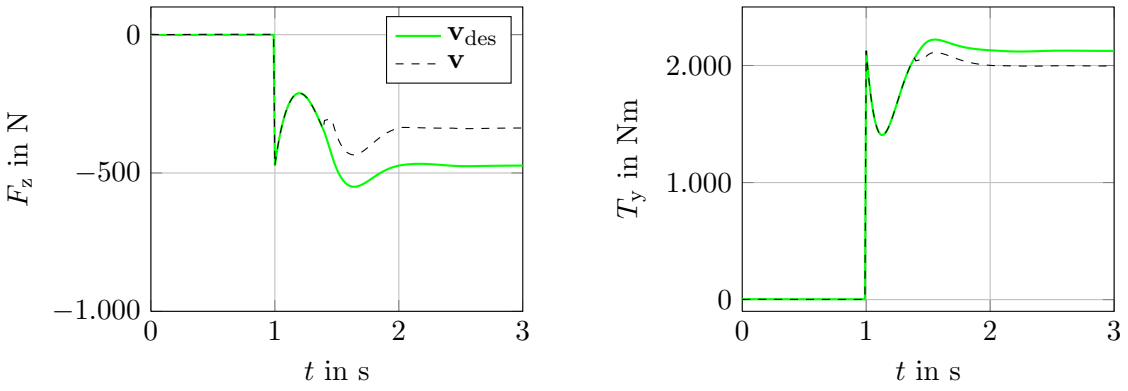


(a) u_1 (b) u_2

(c) u_3 (d) u_4

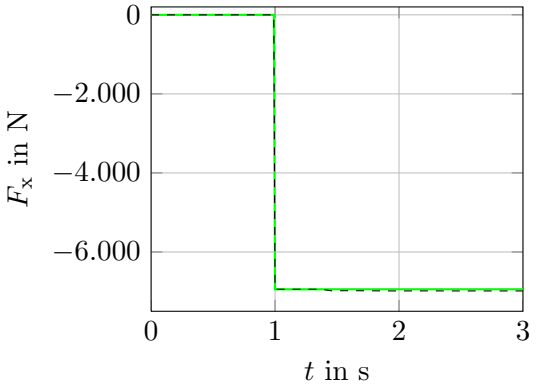
(e) u_5 (f) u_6

Figure 4.14: Control inputs of actuators when motors fail at $t = 1.4$ s.



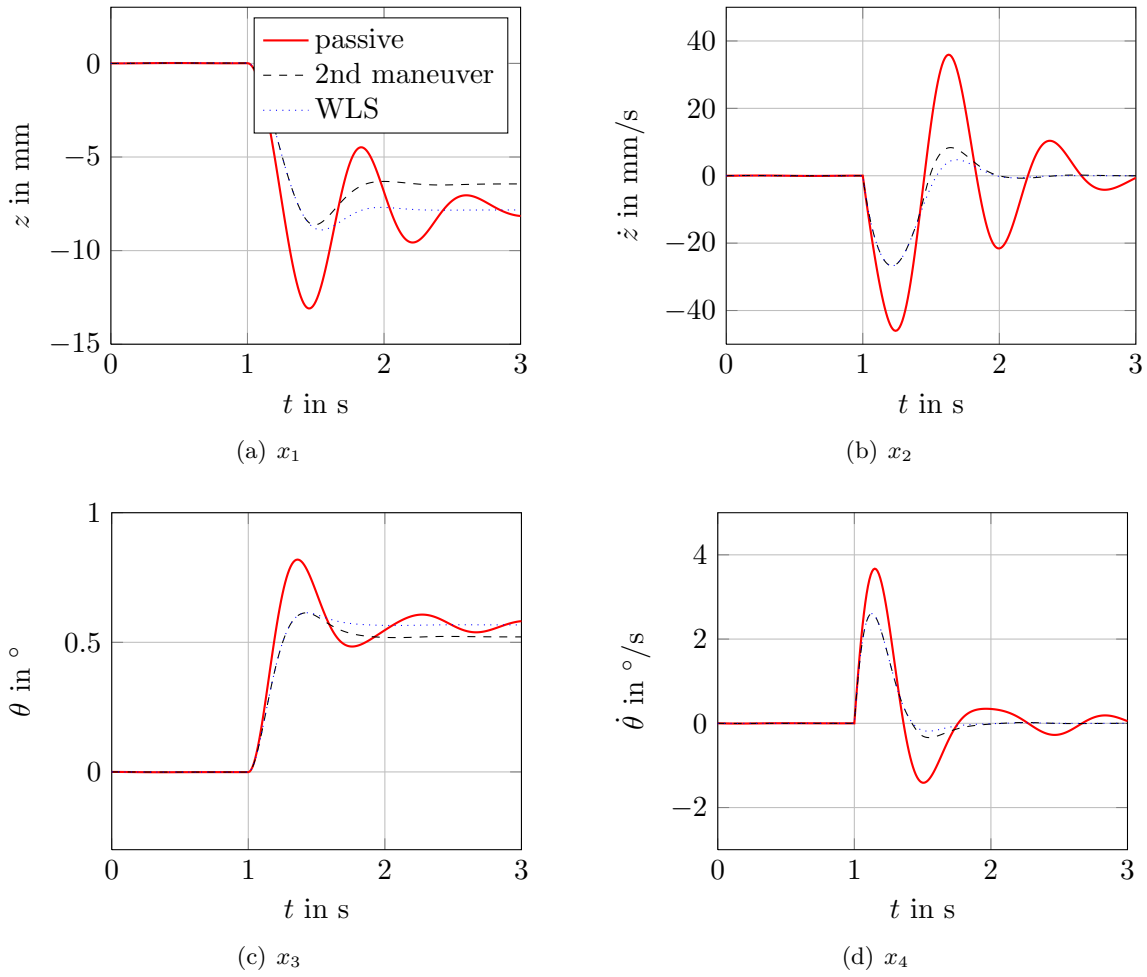
(a) v_1

(b) v_2



(c) v_3

Figure 4.15: Virtual control when motors fail at $t = 1.4$ s.



(a) x_1 (b) x_2

(c) x_3 (d) x_4

Figure 4.16: States when motors fail at $t = 1.4$ s.

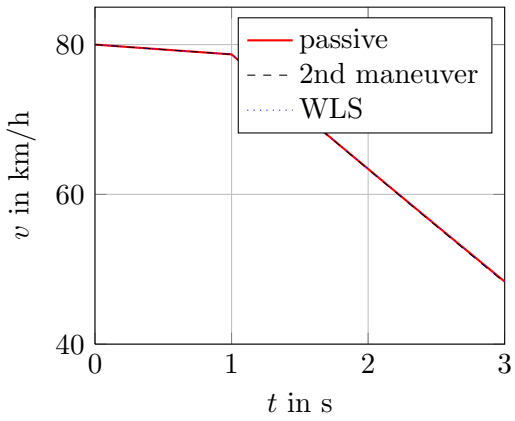


Figure 4.17: Velocity when motors fail at $t = 1.4$ s.

4.5 Prioritization of Brake Force

Since the braking is the most important virtual force and may decide over life and death, the braking distance must not be increased. First, WLS with hotstart has been tried with prioritization of the braking force F_{x} in the weighting matrix $\mathbf{W}_{\mathrm{v}}$. In the second approach, the two-phase algorithm presented in Chapter 3 is used. The control inputs have been limited even more in order to show that the prioritization is still working. The design parameters have been chosen as follows:

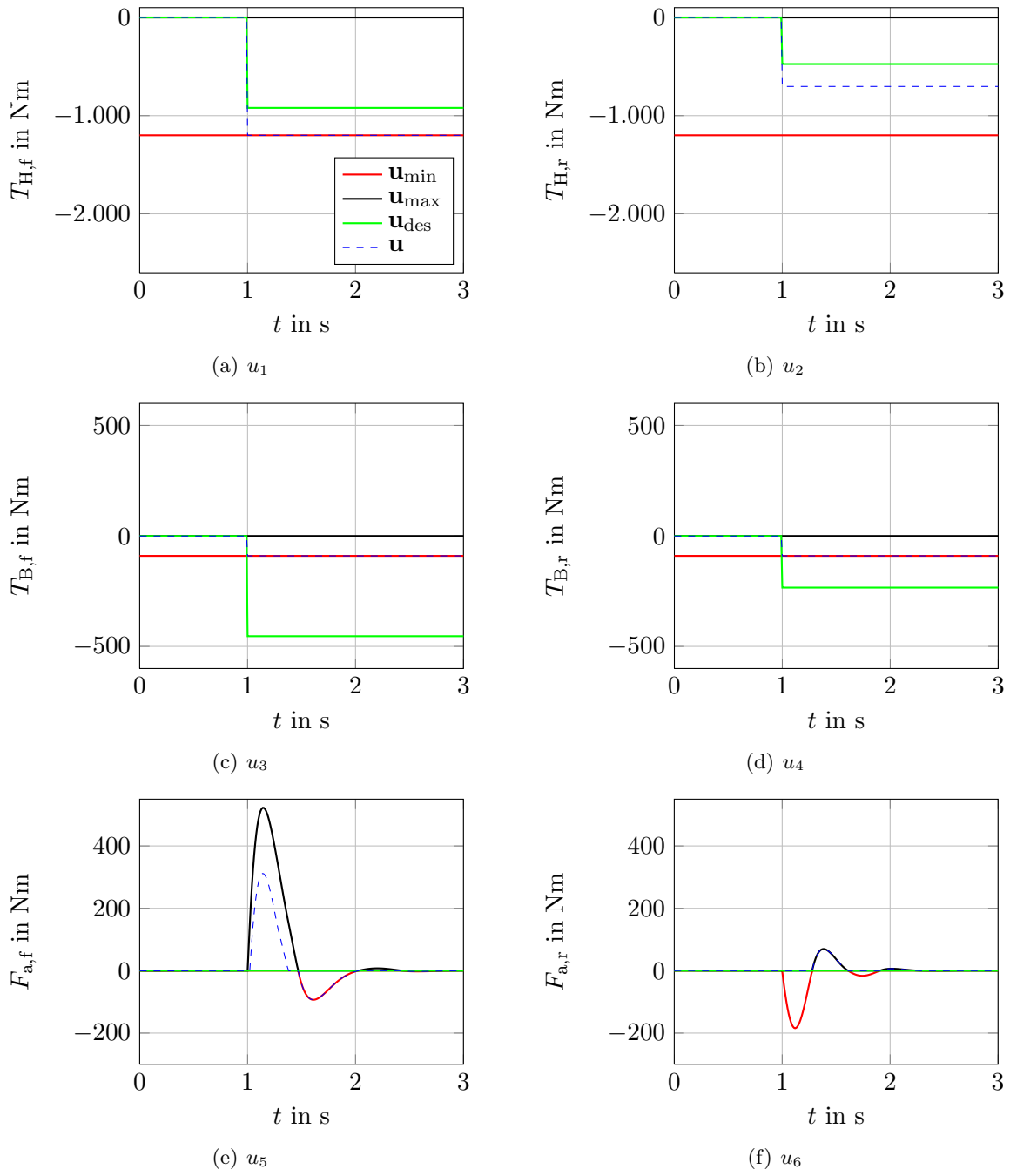
$$\mathbf{u}_{\min} = \begin{bmatrix} -4000 & -4000 & -300 & -300 & F_{\mathrm{a,f,min}} & F_{\mathrm{a,r,min}} \end{bmatrix}^T,$$
$$\mathbf{u}_{\max} = \begin{bmatrix} 0 & 0 & 0 & 0 & F_{\mathrm{a,f,max}} & F_{\mathrm{a,r,max}} \end{bmatrix}^T,$$

$$\mathbf{W}_{\mathrm{v}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1000 \end{bmatrix}, \quad \mathbf{W}_{\mathrm{u}} = \mathbf{I}_6.$$

The maximum number of iterations is set to 2.

4.5.1 WLS with Hotstart

The prioritization of the third virtual control using WLS with hotstart yields good results. Figure 4.18 shows the control inputs for the actuators. Although the constraints are more binding, the algorithm still works satisfactorily.
Figure 4.19 shows that F_{x} can be achieved while the remaining virtual control variables are not fulfilled. Again, the states in Figure 4.20 and 4.21 differ from the desired behaviour (labeled by "WLS") but still are improved in comparison to the passive system.
The prioritization of the third virtual control using WLS with hotstart works well. The problem that arises is the usage of hotstart with the constraints of semi-active suspension. As already discussed, hotstart may or may not yield good results depending on the choice of constraints. Therefore the second approach via a two-phase algorithm is implemented and tested.

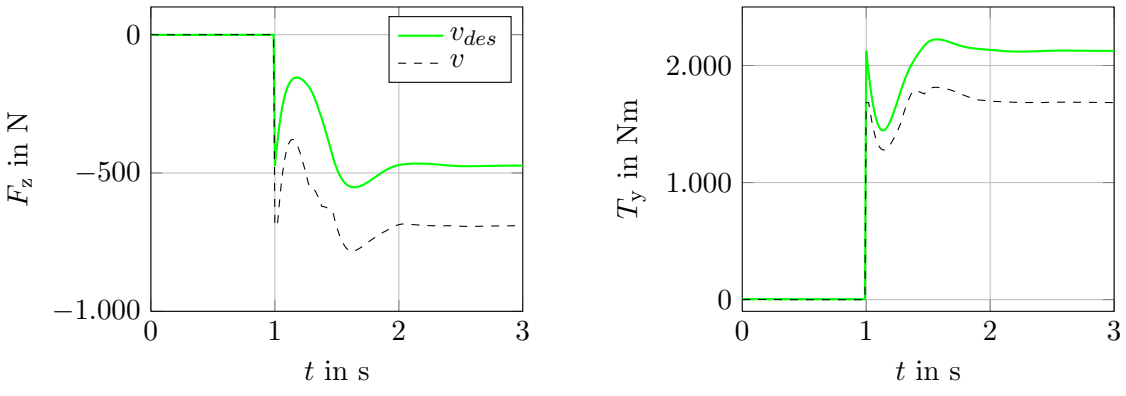


(a) u_1 (b) u_2

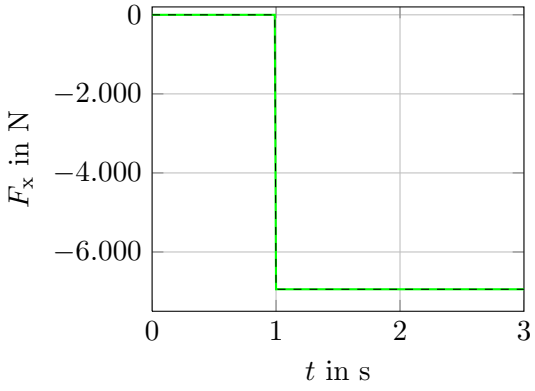
(c) u_3 (d) u_4

(e) u_5 (f) u_6

Figure 4.18: Control inputs of actuators with prioritization of F_{x} using WLS with hotstart.

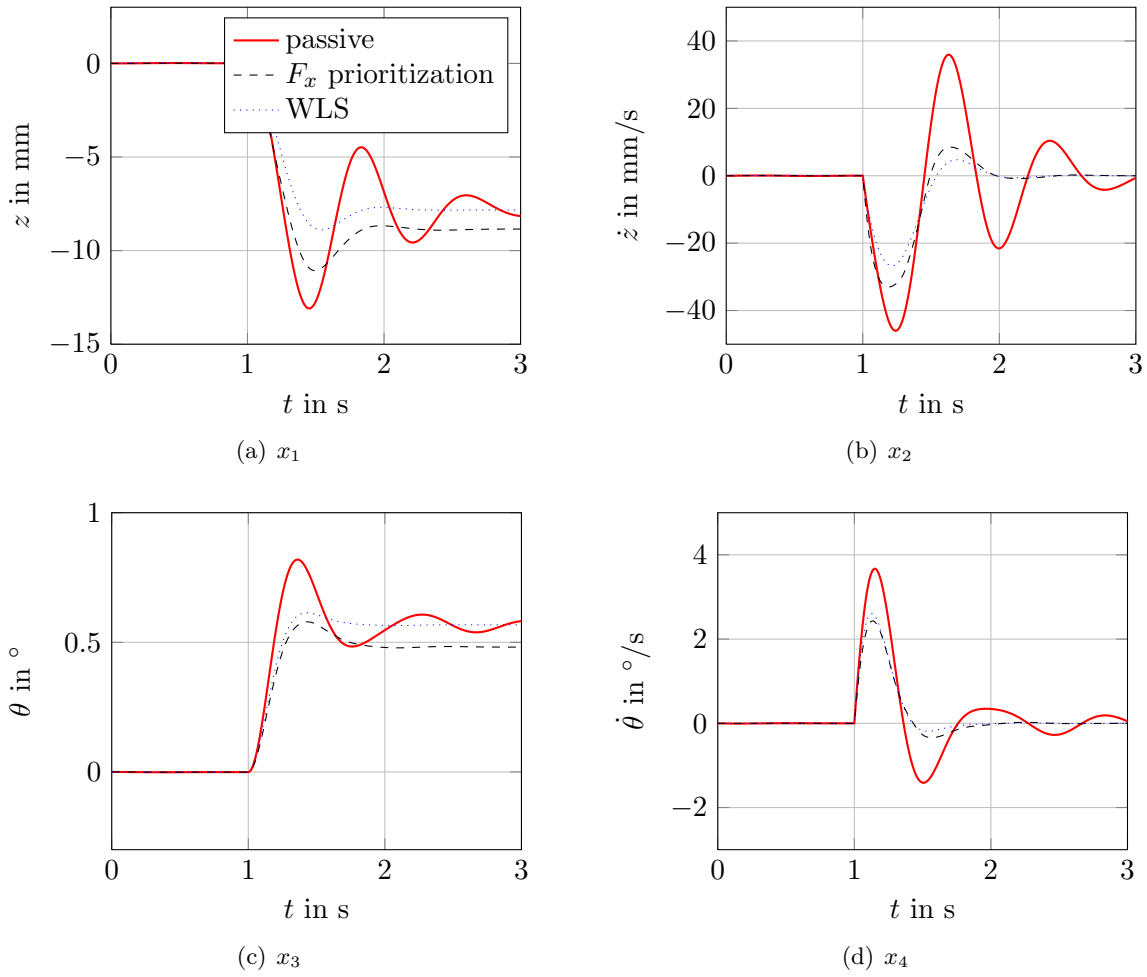


(a) v_1 (b) v_2



(c) v_3

Figure 4.19: Virtual control with prioritization of F_x using WLS with hotstart.



(a) x_1 (b) x_2

(c) x_3 (d) x_4

Figure 4.20: States with prioritization of F_x using WLS with hotstart.

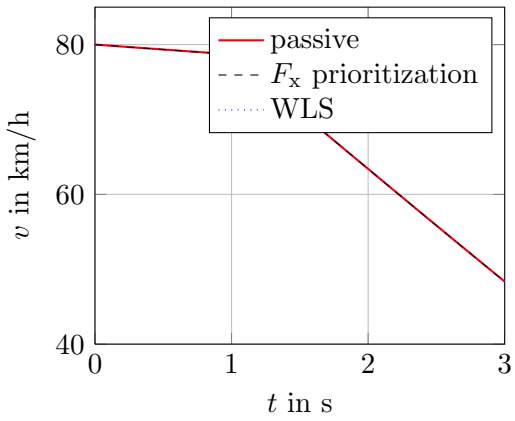


Figure 4.21: Velocity with prioritization of F_x using WLS with hotstart.

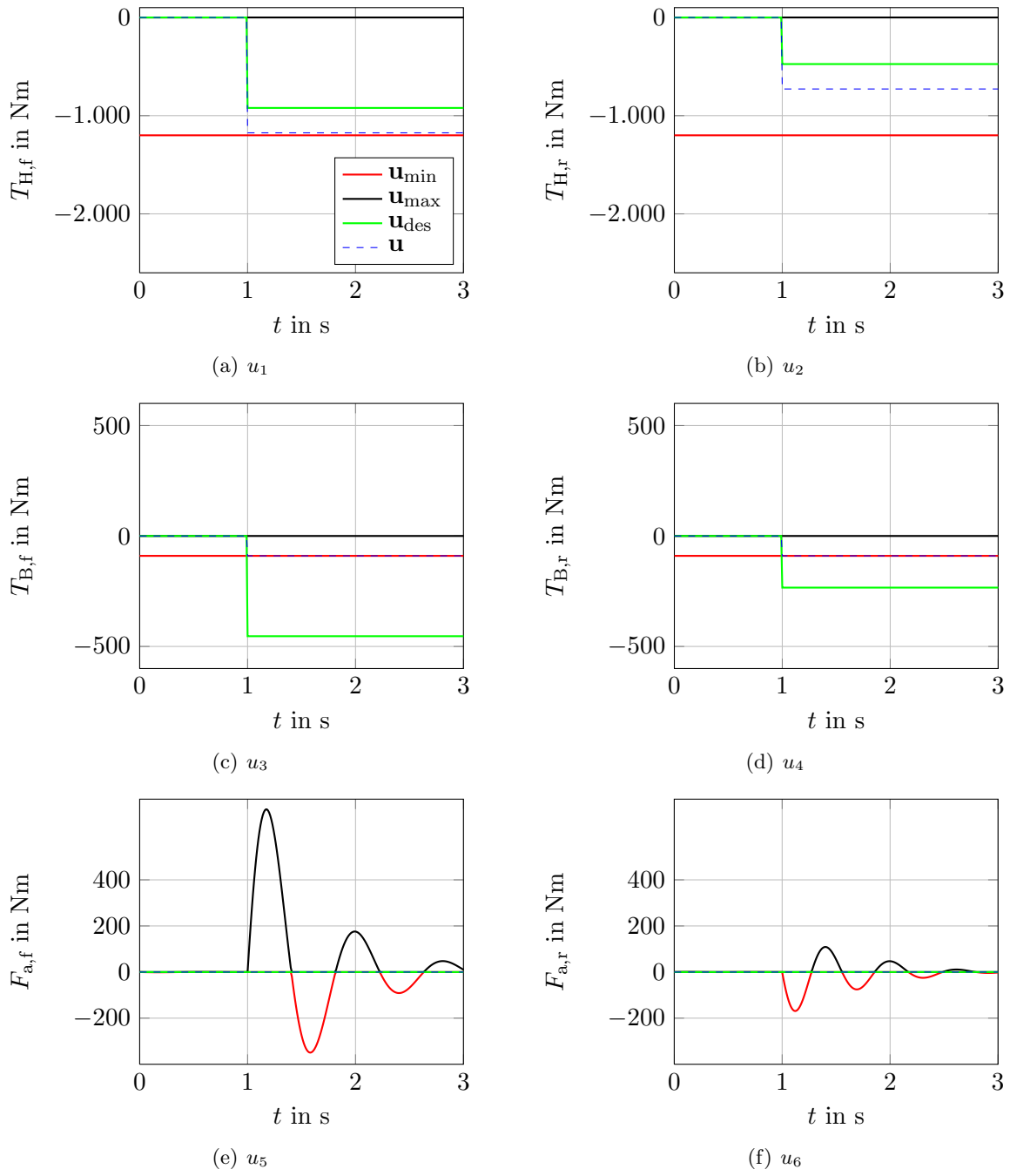
4.5.2 Two-Phase Active Set

For comparison the constraints and the weighting matrices are chosen as above:

$$\mathbf{u}_{\min} = \begin{bmatrix} -4000 & -4000 & -300 & -300 & F_{\mathrm{a,f,min}} & F_{\mathrm{a,r,min}} \end{bmatrix}^T,$$
$$\mathbf{u}_{\max} = \begin{bmatrix} 0 & 0 & 0 & 0 & F_{\mathrm{a,f,max}} & F_{\mathrm{a,r,max}} \end{bmatrix}^T,$$

$$\mathbf{W}_{\mathrm{v}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1000 \end{bmatrix}, \quad \mathbf{W}_{\mathrm{u}} = \mathbf{I}_6.$$

In this example the algorithm stops after phase 1. For this application phase 1 takes two iterations so the control inputs shown in Figure 4.22 are the resulting optimal control inputs of phase 1. Note that although phase 1 takes two iterations, the iterations themselves are solved faster than the original optimization problem. The semi-active suspension is not considered in phase 1 and the corresponding control inputs are set to zero. The optimization without the strongly changing constraints of the semi-active suspension is a quite simple problem and can be solved easily. The two neglected virtual controls of course do not achieve their desired values while F_{x} is achieved perfectly as shown in Figure 4.23.
Lift and pitch of the vehicle is almost the same as for the passive system since semi-active suspensions are not used. The reason for the variance is that the passive system uses the desired control inputs for the actuators as usual, constraints are not considered. The active system considers the binding constraints for the motors and the static, desired values cannot be achieved. Therefore the states - of course excluding the velocity - cannot be the same in this example.

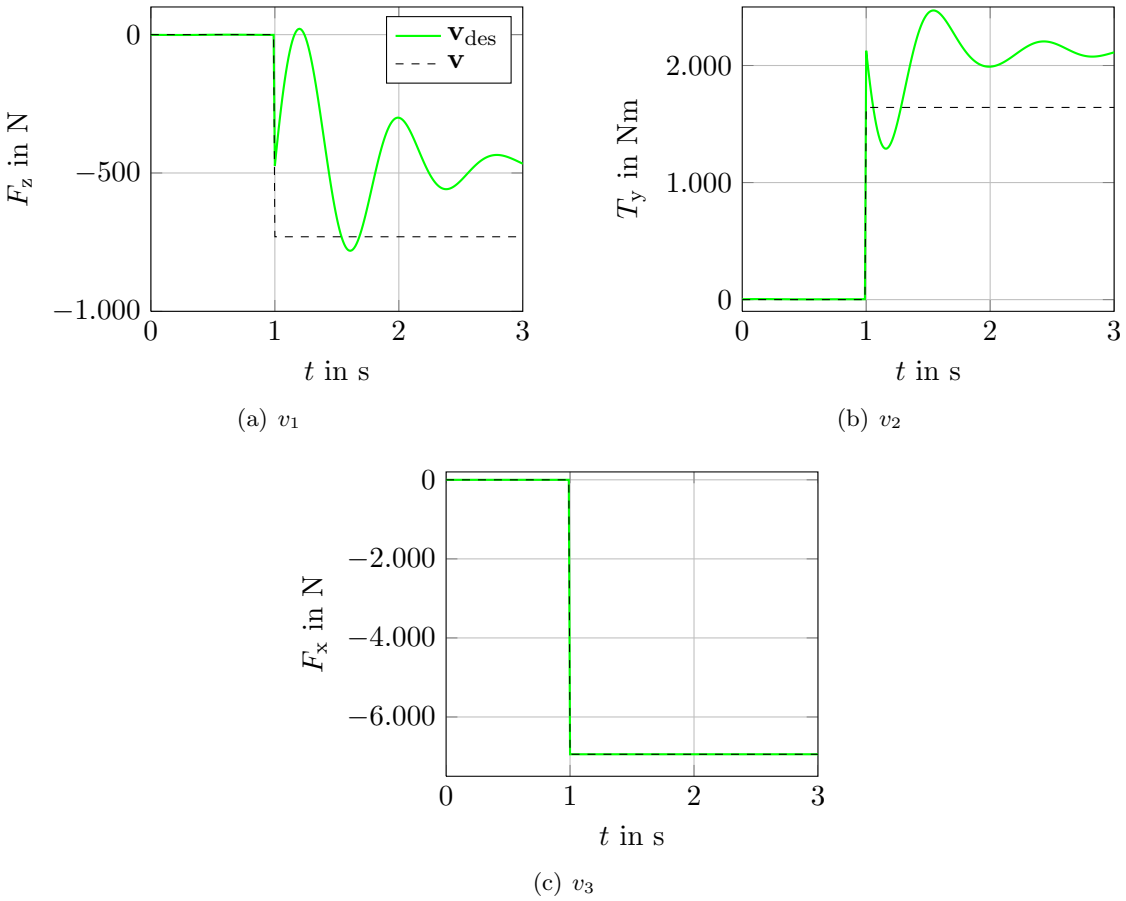


(a) u_1 (b) u_2

(c) u_3 (d) u_4

(e) u_5 (f) u_6

Figure 4.22: Control inputs of actuators with prioritization of F_x using the 2-Phase-Algorithm.

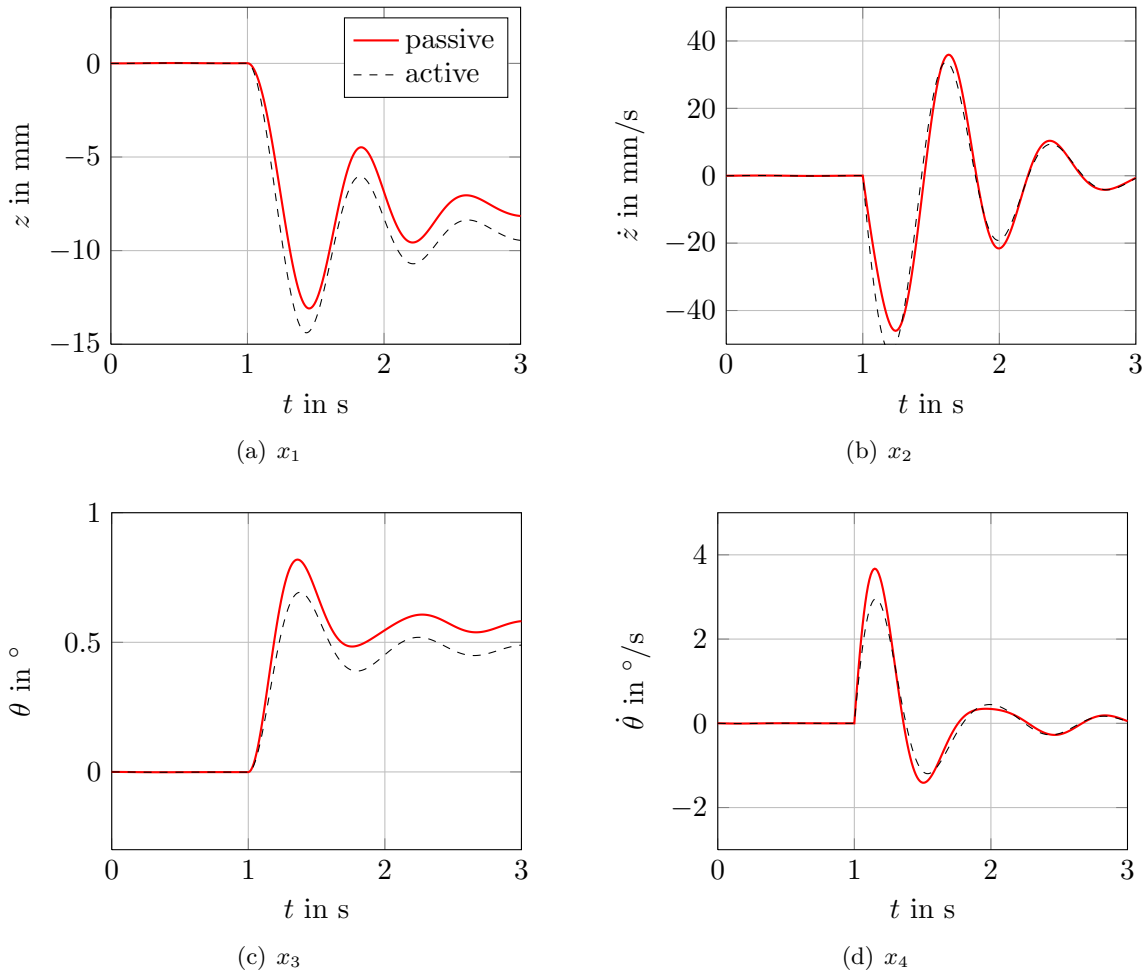


(a) v_1

(b) v_2

(c) v_3

Figure 4.23: Virtual control with prioritization of F_x using the 2-Phase-Algorithm.



(a) x_1 (b) x_2

(c) x_3 (d) x_4

Figure 4.24: States with prioritization of F_x using the 2-Phase-Algorithm.

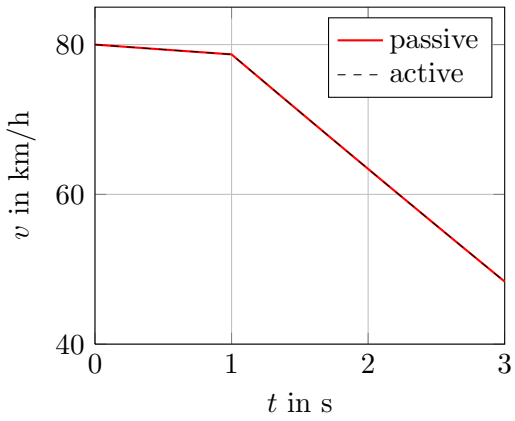


Figure 4.25: Velocity with prioritization of F_x using the 2-Phase-Algorithm.

Chapter 5

Outlook and Conclusion

So far, the application of the modified active set methods show very satisfactory results. In practice, however, additional problems arise and the already discussed methods may not be sufficient. This chapter outlines some of these problems and ends with a conclusion of this thesis.

5.1 Outlook

The methods that have been presented so far only handle the following constraints:

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}. \tag{5.1}$$

Position and rate constraints can be considered, but the algorithms presented by Härkegard and Schofield do not take linear dependencies of constraints into account. In practice, additional constraints arise that must be considered.

5.1.1 Linear Dependencies

One example for a linear dependency is the maximum braking force that can be achieved, i.e.

$$F_{\mathrm{x,max}} \leq \mu F_{\mathrm{z}}, \tag{5.2}$$

where F_{x} is the braking force in x-direction, i.e.

$$F_{\mathrm{x,i}} = T_{\mathrm{H,i}} + T_{\mathrm{B,i}}, \tag{5.3}$$

with $i \in \{f, r\}$ for the front or rear wheels. μ is the tire-surface friction coefficient and F_{z} is the force corresponding to the wheel load. Equation (5.2) states that the maximum braking force $F_{\mathrm{x,max}}$ cannot take arbitrarily large numbers, but is limited by the slip of a tire on the road and the wheel load, see [20]. For the active system the wheel load force F_{z} at time step k is calculated by

$$F_{\mathrm{z,i}} = F_{\mathrm{static,i}} + F_{\mathrm{sp,i}} + F_{\mathrm{damp,i}} + F_{\mathrm{a,i}}^{(k-1)} + F_{\mathrm{susp,i}}. \tag{5.4}$$

The static part $F_{\mathrm{static,i}}$ is defined as

$$F_{\mathrm{static,f}} = mg\frac{l_r}{l_r + l_f}, \tag{5.5}$$

$$F_{\mathrm{static,r}} = mg\frac{l_f}{l_r + l_f}. \tag{5.6}$$

The spring forces $F_{\text{sp,i}}$ and damping forces $F_{\text{damp,ki}}$ read

$$F_{\text{sp,i}} = 2(c_i z + c_i l_i \theta), \tag{5.7}$$

$$F_{\text{damp,i}} = 2(d_i \dot{z} + d_i l_i \dot{\theta}). \tag{5.8}$$

$F_{\text{a,i}}$ corresponds to the force on the semi-active suspension calculated by the optimization in the previous time step:

$$F_{\text{a,f}}^{(k-1)} = u_5^{(k-1)}, \tag{5.9}$$

$$F_{\text{a,r}}^{(k-1)} = u_6^{(k-1)}. \tag{5.10}$$

The additional suspension force $F_{\text{susp,i}}$ is calculated as follows:

$$F_{\text{susp,f}} = \tan(\epsilon_{1,f}) \cdot \frac{T_{H,f}^{(k-1)}}{r} + \tan(\epsilon_{2,f}) \cdot \frac{T_{B,f}^{(k-1)}}{r} \tag{5.11}$$

$$= \tan(\epsilon_{1,f}) \cdot u_1^{(k-1)} + \tan(\epsilon_{2,f}) \cdot u_3^{(k-1)}, \tag{5.12}$$

$$F_{\text{susp,r}} = \tan(\epsilon_{1,r}) \cdot \frac{T_{H,r}^{(k-1)}}{r} + \tan(\epsilon_{2,r}) \cdot \frac{T_{B,r}^{(k-1)}}{r} \tag{5.13}$$

$$= \tan(\epsilon_{1,r}) \cdot u_2^{(k-1)} + \tan(\epsilon_{2,r}) \cdot u_4^{(k-1)}. \tag{5.14}$$

The constraints can be written as

$$\mathbf{C}\mathbf{u} \leq \mathbf{U}, \tag{5.15}$$

with

$$\mathbf{C} = \begin{bmatrix} & & \mathbf{I}_6 & & & \\ & & -\mathbf{I}_6 & & & \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \mathbf{u}_{\min} \\ -\mathbf{u}_{\max} \\ -\mu F_{\text{z,f}} \\ -\mu F_{\text{z,r}} \\ -\mu F_{\text{z,f}} \\ -\mu F_{\text{z,r}} \end{bmatrix}. \tag{5.16}$$

With these additional constraints the algorithms have to be adapted. The working set has to be extended, a simplified calculation of the Lagrangian multipliers is no longer possible. The code presented by Härkegard can no longer be used directly, since linear constraints must be taken into account. Other methods for solving active set algorithms may be applied, but other problems may arise (e.g. inverting matrices). For solving this new problem, a modification of the problem in Chapter 3 has been made. The constraints can be rewritten as

$$0 = u_1 + u_3 - \mu F_{\text{z,f}}, \tag{5.17}$$

$$0 = u_2 + u_4 - \mu F_{\text{z,r}}. \tag{5.18}$$

New artificial control inputs are introduced:

$$\mathbf{u}^* = \begin{bmatrix} T_{\text{H,f}} \\ T_{\text{H,r}} \\ T_{\text{B,f}} \\ T_{\text{B,r}} \\ F_{\text{a,f}} \\ F_{\text{a,r}} \\ u_7 \\ u_8 \end{bmatrix}. \tag{5.19}$$

The linear dependencies now read

$$0 = u_1 + u_3 - u_7, \tag{5.20}$$
$$0 = u_2 + u_4 - u_8, \tag{5.21}$$

with the new constraints

$$-\mu F_{\mathrm{z,f}} \leq u_7 \leq \mu F_{\mathrm{z,f}}, \tag{5.22}$$
$$-\mu F_{\mathrm{z,r}} \leq u_8 \leq \mu F_{\mathrm{z,r}}. \tag{5.23}$$

The new virtual forces are

$$\mathbf{v}^* = \begin{bmatrix} F_{\mathrm{z}} \\ T_{\mathrm{y}} \\ F_{\mathrm{x}} \\ 0 \\ 0 \end{bmatrix}. \tag{5.24}$$

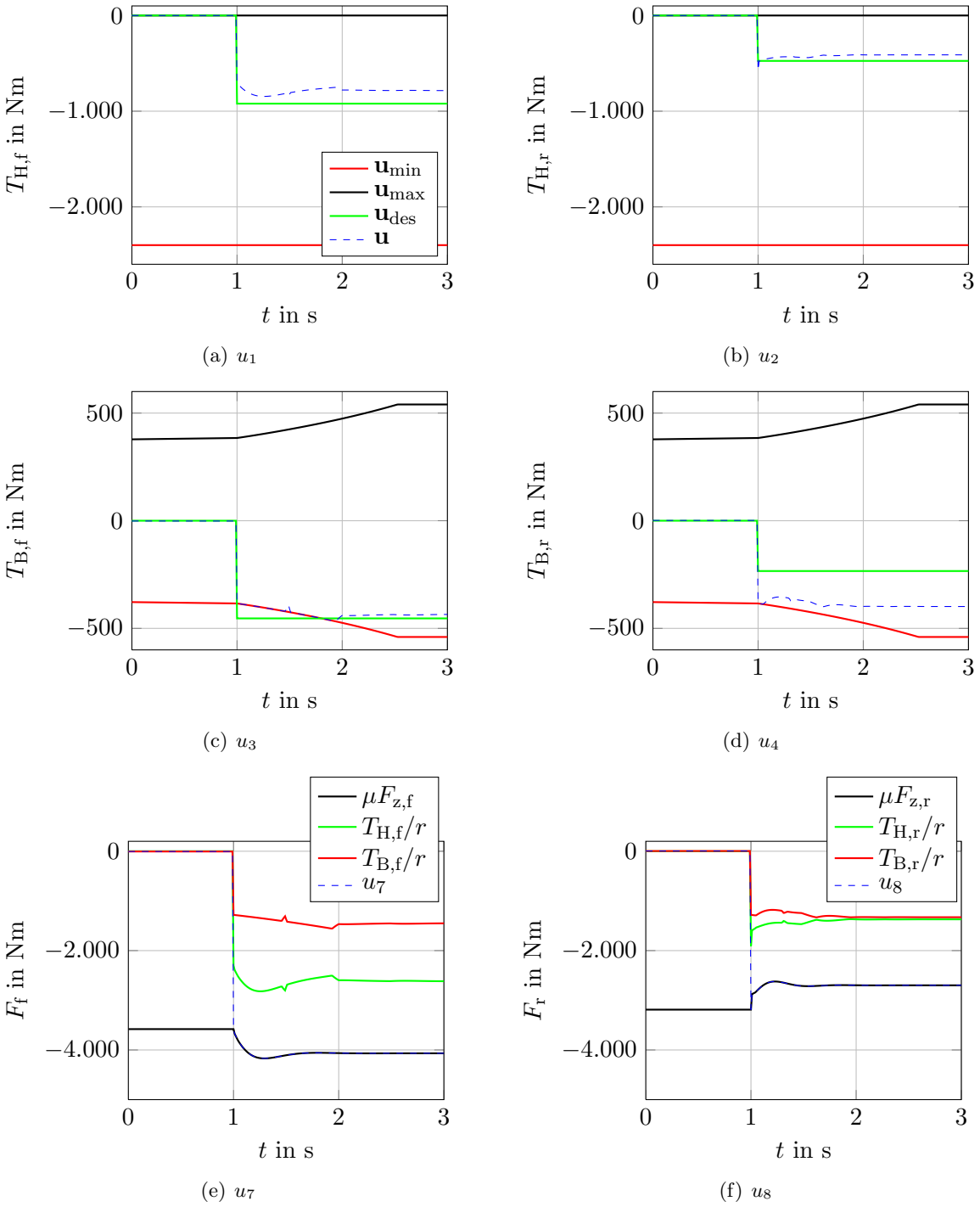
The control effectiveness matrix $\mathbf{H}$ changes to

$$\mathbf{H}^* = \begin{bmatrix} & & & \mathbf{H} & & & 0 & \\ & & & & & & & \\ 1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{5.25}$$

Other parameters such as the working set $\mathcal{W}$, the desired control input $\mathbf{u}_{\mathrm{des}}$, etc. have to be adopted accordingly. One drawback of this approach is that the additional constraints are defined by the control effectiveness matrix $\mathbf{H}$. Constraints must always hold in the active set methods. Virtual forces may not always be achieved and it is possible that the equations in (5.20) do not hold when using this modification. In practice there holds $F_x \leq \mu F_z$, and it is impossible that these additional constraints are violated. Therefore, the weighting of the artificial control inputs u_7, u_8 is chosen rather high. Nevertheless, this approach is not satisfying. The additional constraints may still be violated in the simulation while in practice this violation can never happen. Another problem arises when using Schofield's algorithm: the proof that the algorithm terminates in $2m - 1$ iterations does no longer hold. The algorithm uses all iterations possible (defined by *iter_max*) at some time steps. The algorithm was tested on the following settings:

$$\mathbf{W}_{\mathrm{v}} = \begin{bmatrix} \mathbf{I}_3 & & \\ & 10^3 & \\ & & 10^3 \end{bmatrix}, \quad \mathbf{W}_{\mathrm{u}} = \mathbf{I}_8.$$

The maximum number of iterations is set to 50. The road-tire friction factor was chosen as $\mu = 0.4$. The control inputs for the mechanical brakes and electric motors and the additional constraints are shown in Figure 5.1. The control inputs do not violate the constraints because of the high weighting factors for the artificial variables. In practice this problem must be considered accordingly. Other methods and algorithms may be chosen to solve the problem.



(a) u_1 (b) u_2

(c) u_3 (d) u_4

(e) u_7 (f) u_8

Figure 5.1: Control inputs of actuators with additional constraints.

5.1.2 Nonlinear Constraints

Adding linear constraints already shows that there are limitations to the application of the algorithms presented by Härkegard and Schofield. If the problem shall be extended to a vehicle with lateral movement, the constraints become nonlinear according to Kamm's circle (cf. [20]):

$$\sqrt{F_{\mathrm{x}}^2 + F_{\mathrm{y}}^2} \leq \mu F_{\mathrm{z}}. \tag{5.26}$$

This equation states that if the velocity in x-direction is too high, the vehicle cannot move according to the curve in x- AND y-direction. Using nonlinear constraints, the previously mentioned methods can no longer be used. A nonlinear optimization problem arises that reads

$$\begin{aligned} \min_{\mathbf{x}} \ & \mathbf{f}(\mathbf{x}) \\ \text{s. t.} \quad & \mathbf{g}(\mathbf{x}) = 0 \\ & \mathbf{h}(\mathbf{x}) \leq 0. \end{aligned} \tag{5.27}$$

As described in [12], there are several approaches to nonlinear control allocation. One approach of handling nonlinearities is sequential quadratic programming as presented in [21]. In sequential quadratic programming, the objective function is quadratically approximated locally around $\mathbf{x}^{(k)}$. A new quadratic program arises where the constraints are linearized, too. The resulting optimization problem reads

$$\begin{aligned} \min_{\mathbf{s}} & \frac{1}{2}\mathbf{s}^T\mathbf{G}^{(k)}\mathbf{s} + \nabla\mathbf{f}(\mathbf{x}^{(k)})^T\mathbf{s} \\ \text{s. t.} \ & \nabla g_i(\mathbf{x}^{(k)})^T\mathbf{s} + g_i(\mathbf{x}^{(k)}) = 0 \quad i = 1, ..., p \\ & \nabla h_i(\mathbf{x}^{(k)})^T\mathbf{s} + h_i(\mathbf{x}^{(k)}) \leq 0 \quad i = 1, ..., q \end{aligned} \tag{5.28}$$

with the search direction s. $\mathbf{G}^{(k)}$ is an approximation of the Hessian matrix of the objective function $\mathbf{f}(\mathbf{x})$ at $\mathbf{x}^{(\mathbf{k})}$. The matrix $\mathbf{G}^{(k)} > 0$ is symmetric and is typically updated after each iteration, using e.g. the BFGS formula, cf. [6]. The new iterate $\mathbf{x}^{(k+1)}$ is found by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^k\mathbf{s}^k, \tag{5.29}$$

where $\mathbf{s}^k$ is the solution of (5.28) at step k that can be solved by the common methods solving quadratic programs. α^k can be found by a line search using a penalty function, e.g. the L_1 exact penalty function

$$\Phi(\mathbf{x}, \sigma) = \mathbf{f}(\mathbf{x}) + \sigma\left(\sum_i^p |g_i(\mathbf{x})| + \sum_i^q \max\{0, h_i(\mathbf{x})\}\right), \tag{5.30}$$

or using a barrier function, e.g. the logarithmic barrier function

$$\Phi(\mathbf{x}, \tau) = \mathbf{f}(\mathbf{x}) - \tau\sum_i^p \ln(-h_i(\mathbf{x})), \tag{5.31}$$

which should be minimized. The algorithm continues until some termination criterium is fulfilled. The quadratic program is strictly convex and therefore has a solution if the Hessian matrix $\mathbb{H}^{(k)}$ is positive definite. Note that SQP methods only find local solutions to nonlinear programs, cf. [2]. If the linearized constraints are infeasible, there exists no solution at all. The choice of operating point plays an important role. If $\mathbf{x}^{(k)}$ is too remote from the optimum the standard algorithm may fail to converge. Clearly, the termination of the algorithm depends on many factors. Further details on sequential quadratic programming are discussed in [6].

5.1.3 Nonlinear Control Allocation

Nonlinear control allocation formulations are necessary for example if a time-varying linearization does not provide accurate approximations. The objective function as well as the constraints may be nonlinear. If the linear effector model

$$\mathbf{v} = \mathbf{H}(\mathbf{x}, t) \cdot \mathbf{u}, \tag{5.32}$$

is used and the control allocation problem is viewed as static, then $\mathbf{H}(\mathbf{x}, t)$ is clearly defined for each time step, so the same methods for linear control allocation already mentioned can be applied. If the control effectiveness matrix $\mathbf{H}(\mathbf{x}, t)$ may become rank-deficient, i.e. it has no longer full rank for example due to the use of sine or cosine functions, the problem becomes underactuated and the system may no longer be controllable, see [13]. In [13] a new optimization problem is defined, which makes the control allocation more robust and avoids near-singular configurations. For the problem at hand this case of singularity does not occurr, the angles cannot become so small that singularity avoidance is necessary.
If the effector model is nonlinear, i.e.

$$\mathbf{v} = \mathbf{h}(\mathbf{x}, t, \mathbf{u}), \tag{5.33}$$

there also exist several approaches for solving the problem, as summarized in [12].
Some other nonlinear methods are

- Piece-wise Linear Control Allocation (MILP)
- Lyapunov-Design
- Adaptive Dynamic Control Allocation (tire-road friction estimation)
- Direct Nonlinear Allocation.

It is impossible to provide a general approach to nonlinear control allocation, the choice strongly depends on the application.

5.2 Conclusion

Summarizing this work, control allocation algorithms have been discussed and implemented on a simplified model of a road vehicle. The lift and pitch motions of the vehicle can be improved significantly compared to the passive system while considering constraints of the actuators. Active set algorithms are widely used in automotive applications and have been confirmed to yield very robust and appropriate results. The following active set methods have been studied in more detail:

- Härkegard's algorithm
- Schofield's algorithm
- 2-Phase algorithm.

Härkegard's algorithm shows very adequate results. The drawback of this algorithm, which is the exponential computational effort in the worst case, can be eliminated by Schofield's modification, ensuring termination after at most $(2m-1)$ iterations. The braking distance of the passive system must not be increased and the corresponding virtual force F_x can be prioritized accordingly. However, it is desirable to guarantee that the braking force is achieved at all times. Therefore,

an algorithm consisting of two phases has been implemented. This algorithm considers lift and pitch forces only if the braking force is achieved properly. Additionally, linear dependencies have been studied. In order to avoid a violation of these constraints, further work on the computation of the active set methods is necessary.

Bibliography

[1] Matthäus B. Alberding and Tor A. Johansen. Nonlinear hierarchical control allocation for vehicle yaw stabilization and rollover prevention, 2008.

[2] Paul T. Boggs and Jon W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–51, 1 1995.

[3] Jovan D. Boskovic and Raman K. Mehra. Control allocation in overactuated aircraft under position and rate limiting. In *American Control Conference, 2002. Proceedings of the 2002*, volume 1, pages 791–796, 2002.

[4] David B. Doman and Anhtuan D. Ngo. Dynamic inversion-based adaptive/reconfigurable control of the x-33 on ascent. In *Aerospace Conference, 2001, IEEE Proceedings.*, volume 6, pages 2683–2697, 2001.

[5] Roger Fletcher. *Practical methods of optimization (2nd ed.)*. Wiley-Interscience, New York, USA, 1987.

[6] Roger Fletcher. The sequential quadratic programming method. In Gianni Di Pillo and Fabio Schoen, editors, *Nonlinear Optimization*, Lecture Notes in Mathematics, pages 165–214. Springer Berlin Heidelberg, 2010.

[7] Thor I. Fossen and Tor A. Johansen. A survey of control allocation methods for ships and underwater vehicles. In *14th Mediterranean Conference on Control and Automation*, pages 1–6, 2006.

[8] Ola Harkegard. *Backstepping and Control Allocation with Applications to Flight Control*. PhD thesis, Linkoping University, Department of Electrical Engineering, SE-581 83 Linkoping, Sweden, 2003.

[9] Norbert Hohenbichler and Klaus Six. Potenziale aktiver und semiaktiver skyhook-regelgesetze in der sekundärfederstufe von schienenfahrzeugen (potentials of active and semiactive skyhook control laws applied to the secondary suspensions of railway vehicles). *Automatisierungstechnik*, 54(3):130–138, 2006.

[10] Klaus Holzmann. *Modellierung und Regelung von elektrorheologischen Dämpfern*. PhD thesis, E376, TU Wien, 2011.

[11] Tor A. Johansen. Optimizing nonlinear control allocation. In *43rd IEEE Conference on Decision and Control*, volume 4, pages 3435–3440, 2004.

[12] Tor A. Johansen and Thor I. Fossen. Control allocation - a survey. *Automatica*, 49(5):1087–1103, 2013.

[13] Tor A. Johansen, Thor I. Fossen, and Svein P. Berge. Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming. *IEEE Transactions on Control Systems Technology*, 12(1):211–216, 2004.

[14] Victor Klee and George J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities*, volume 3, pages 159–175. Academic Press, New York, 1972.

[15] Daniel Lindvai-Soos. Interner Bericht: Lift-pitch-model. 2013.

[16] David G. Luenberger. Canonical forms for linear multivariable systems. *IEEE Transactions on Automatic Control*, 12(3):290–293, 1967.

[17] Yu Luo, Andrea Serrani, Stephen Yurkovich, David B. Doman, and Michael W. Oppenheimer. Model predictive dynamic control allocation with actuator dynamics. In *American Control Conference, 2004. Proceedings of the 2004*, volume 2, pages 1695–1700, 2004.

[18] Yu Luo, Andrea Serrani, Stephen Yurkovich, David B. Doman, and Michael W. Oppenheimer. Dynamic control allocation with asymptotic tracking of time-varying control input commands. In *American Control Conference, 2005. Proceedings of the 2005*, volume 3, pages 2098–2103, 2005.

[19] Yu Luo, Andrea Serrani, Stephen Yurkovich, David B. Doman, and Michael W. Oppenheimer. Model-predictive dynamic control allocation scheme for reentry vehicles. *Journal of Guidance, Control, and Dynamics*, 30(1):100–113, 2007.

[20] Manfred Mitschke and Henning Wallentowitz. *Dynamik der Kraftfahrzeuge*. Springer, 4., neubearb. aufl. edition, 2004.

[21] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[22] Michael W. Oppenheimer, D.B. Doman, and M.A. Bolender. Control Allocation for Overactuated Systems. *Mediterranean Conference on Control and Automation*, 0:1–6, 2006.

[23] John A.M Petersen and Marc Bodson. Constrained Quadratic Programming Techniques for Control Allocation. *IEEE Transactions on Control Systems Technology*, 14(1):91–98, 2006.

[24] Brad Schofield. On active set algorithms for solving bound-constrained least squares control allocation problems. In *Proceedings of the American Control Conference*, pages 2597–2602, July 2008.

[25] Brad Schofield and Tore Hagglund. Optimal Control Allocation on vehicle dynamics control for rollover mitigation. In *American Control Conference, 2008*, pages 3231–3236, 2008.

[26] Johannes Tjonnas and Tor A. Johansen. Optimizing adaptive control allocation with actuator dynamics. In *46th IEEE Conference on Decision and Control*, pages 3780–3785, 2007.

[27] Petter Tøndel and Tor A. Johansen. Control allocation for yaw stabilization in automotive vehicles using multiparametric nonlinear programming. In *American Control Conference, 2005. Proceedings of the 2005*, pages 453–458, 2005.