Institutionen för systemteknik
Department of Electrical Engineering

**Examensarbete**

# Optimal Vehicle Speed Control Using a Model Predictive Controller for an Overactuated Vehicle

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan vid Linköpings universitet
av

**Mathias Mattsson och Rasmus Mehler**

LiTH-ISY-EX--15/4841--SE

Linköping 2015


**Linköpings universitet**
**TEKNISKA HÖGSKOLAN**

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

# Optimal Vehicle Speed Control Using a Model Predictive Controller for an Overactuated Vehicle

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan vid Linköpings universitet
av

**Mathias Mattsson och Rasmus Mehler**

LiTH-ISY-EX--15/4841--SE

Handledare: **Dr. Mats Jonasson**
Volvo Cars
**Dr. Andreas Thomasson**
ISY, Linköpings universitet

Examinator: **Associate Professor Lars Eriksson**
ISY, Linköpings universitet

Linköping, 15 juni 2015

**Avdelning, Institution**
Division, Department

Division of Vehicular Systems
Department of Electrical Engineering
SE-581 83 Linköping

**Datum**
Date

2015-06-15

**Språk**
Language

□ Svenska/Swedish
☒ Engelska/English
□ _____

**Rapporttyp**
Report category

□ Licentiatavhandling
☒ Examensarbete
□ C-uppsats
□ D-uppsats
□ Övrig rapport
□ _____

**ISBN**
—

**ISRN**
LiTH-ISY-EX--15/4841--SE

**Serietitel och serienummer**
Title of series, numbering

**ISSN**
—

**URL för elektronisk version**
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-XXXXX

**Titel**
Title

Optimal fartreglerig med modelbaserad prediktionsreglering för en överaktuerad personbil

Optimal Vehicle Speed Control Using a Model Predictive Controller for an Overactuated Vehicle

**Författare**
Author

Mathias Mattsson och Rasmus Mehler

**Sammanfattning**
Abstract

To control the speed of an overactuated vehicle there may be many possible ways to use the actuators of the car achieving the same outcome. The actuators in an ordinary car is a combustion engine and a friction brake. In some cases it is trivial how to coordinate actuators for the optimal result, but in many cases it is not.

The goal with the thesis is to investigate if it is possible to achieve the same or improved performance with a more sophisticated control structure than today's, using a model predictive controller. A model predictive controller combines the possibility to predict the outcome through an open-loop controller with the stability of a closed loop controller and gives the optimal solution for a finite horizon optimization problem.

A simple model of the longitudinal dynamics of a car is developed and used in the model predictive controller framework. This is then used in simulations and in a real car. It is shown that it is possible to replace the current controller structure with a model predictive controller, but there are advantages and disadvantages with the new control structure.

**Nyckelord**
Keywords

MPC, Overactuated, Control Allocation, Vehicle Dynamics, Speed Control

# Abstract

To control the speed of an overactuated vehicle there may be many possible ways to use the actuators of the car achieving the same outcome. The actuators in an ordinary car is a combustion engine and a friction brake. In some cases it is trivial how to coordinate actuators for the optimal result, but in many cases it is not.

The goal with the thesis is to investigate if it is possible to achieve the same or improved performance with a more sophisticated control structure than today's, using a model predictive controller. A model predictive controller combines the possibility to predict the outcome through an open-loop controller with the stability of a closed loop controller and gives the optimal solution for a finite horizon optimization problem.

A simple model of the longitudinal dynamics of a car is developed and used in the model predictive controller framework. This is then used in simulations and in a real car. It is shown that it is possible to replace the current controller structure with a model predictive controller, but there are advantages and disadvantages with the new control structure.

# Acknowledgments

We would like to thank to our supervisor at Volvo Cars, Mats Jonasson for all the help he gave us during our thesis work. We would also like to thank the entire group at Volvo Cars for helpful advices, especially Mikael Thor who helped us with models and testing our work in a car.

Thanks goes also our supervisor at the university Andreas Thomasson for being helpful when we wrote this report. Finally we would like to thank our examiner Lars Eriksson at Linköping University.

*Linköping, June 2015*
*Mathias Mattsson and Rasmus Mehler*

# Contents

# Notation

### Parameters

| Notation | Description |
|---|---|
| $g$ | Gravity constant |
| $m$ | Mass of car |
| $\rho$ | Density of air |
| $C_d$ | Aerodynamic drag coefficient |
| $A_a$ | Cross sectional area of car |
| $\mu$ | Friction coefficient between tire and ground material |
| $C_{rr}$ | Rolling resistance coefficient |
| $\alpha$ | Slope of road |
| $s$ | Time derivative operator |
| $v$ | Longitudinal velocity of car |
| $F_{\text{eng}}$ | Longitudinal force from engine |
| $F_{\text{brake}}$ | Longitudinal force from brakes |
| $F_{\text{road load}}$ | Longitudinal force from road load |
| $F_{\text{air}}$ | Longitudinal force from air resistance |
| $F_{\text{roll}}$ | Longitudinal force from rolling resistance |
| $F_{\text{slope}}$ | Longitudinal force from slope |
| $F_{\text{drag}}$ | Longitudinal force from air resistance and rolling resistance |
| $F_{\text{pt,in}}$ | Force generated by the inertia of the powertrain |
| $x$ | State vector |
| $u$ | Control signal vector |
| $u_e$ | Control signal for engine |
| $u_b$ | Control signal for brake |
| $T_e$ | Time constant for engine model |
| $T_{b,up}$ | Time constant for brake model when building pressure |
| $T_{b,down}$ | Time constant for brake model when releasing pressure |
| $L_e$ | Time delay for engine model |
| $L_b$ | Time delay for brake model |
| $a_{\text{ref}}$ | Reference signal in acceleration |
| $T_s$ | Sampling time |
| $\eta_e$ | Time delay for the engine expresed in number of samples |
| $J_{\text{eng}}$ | Inertia in the engine |

### Parameters

| Notation | Description |
|---|---|
| $k_{\text{trans}}$ | Transmission ratio |
| $p$ | Pressure in the braking system |
| $\tau_b$ | Torque generated by the brakes |
| $r_b$ | Distance from the wheel center to the point where the braking pads are applied on the wheels |
| $r_w$ | Wheel radius |
| $A_b$ | Contact area between the braking pads and the wheels |
| $Q_{\text{ref}}$ | Cost for deviating from reference |
| $Q_{\text{cone}}$ | Linear increasing cost for deviating from reference |
| $k_{\text{cone}}$ | Number of sample before the linear increasing factor is applied |
| $Q_{\text{ref,total}}$ | Total cost for deviating from the reference |
| $u_{\text{e,min}}$ | Minimum force that the engine can generate |
| $u_{\text{e,max}}$ | Maximum force that the engine can generate |
| $u_{\text{b,min}}$ | Minimum force that the brakes can generate |
| $u_{\text{b,max}}$ | Maximum force that the brakes can generate |
| $j_{\text{lim}}$ | Jerk limit |
| $\epsilon_j$ | Slack variable on the jerk state |

### Abbreviations

| Abbreviation | Descrition |
|---|---|
| PID | Proportional, integral, differential (regulator) |
| MPC | Model Predictive Control |
| MPCA | Model Predictive Control Allocation |
| LP | Liner Programming |
| QP | Quadratic Programming |
| QCQP | Quadraticly Constrained Quadratic Programming |
| CC | Cruise Control |
| ACC | Adaptive Cruise Control |
| ASDM | Active Safety Domain Master |
| VDDM | Vehicle Dynamics Domain Master |
| ECM | Engine Control Module |
| BCM | Brake Control Module |
| ERAD | Electric Rear Axle Drive |
| PMP | Pontryagin Maximum Principle |
| DP | Dynamic Programming |
| HEV | Hybrid Electric Vehicle |
| ECMS | Equivalent Consumption Minimization Strategy |
| ICE | Internal Combustion Engine |
| RPM | Revolutions Per Minute |
| CIT | Cut In Target |
| CAN | Controller Area Network |

# 1

# Introduction

If a control system is designed with more control signals than states it is said to have redundancy [1]. Another expression for this is that the system is over-actuated. In Figure 1.1 this is illustrated by representing the number of control signals and states in each system with the dimensions of the systems. The thin system illustrates an overactuated system with more ways to control the system than states to control. The square and the fat system illustrates systems with equally or more states than control signals.

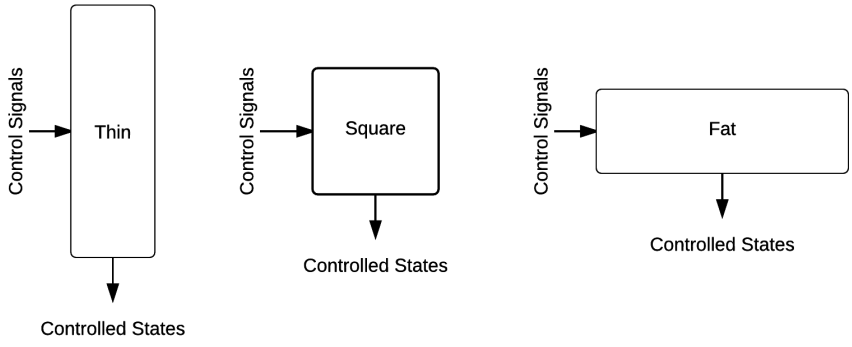


***Figure 1.1:*** *Comparisons between the concept of overactuated ,or thin, systems with non-overactuated systems, square and fat.*

An actuator controlling a system often has some limitations on what possible control signals it can deliver. This may vary depending on the current state of the system. In an overactuated system there can exist regions where the actuators ranges overlap each other while in other ranges they do not.

In Figure 1.2, an illustrating example of a system with two actuators that for some states the possible control signals overlaps. This means that there exists several, or even infinite, ways to combine the actuators to obtain identical effect on the system output. But even tough the effect is identical there can still be some combinations that is unwanted, for example if the actuators oppose each other.

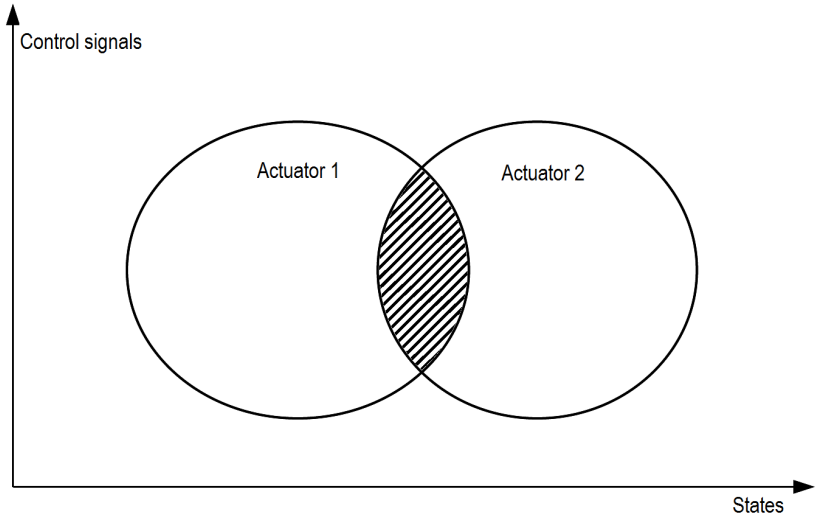


***Figure 1.2:*** *Illustration of how the actuators overlap in an overactuated system with two actuators.*

Some systems are designed to be overactuated to achieve stability and robustness by the additional actuators. One example out of many on overactuated systems is a multicopter. They often have several more actuators than it actually needs for control of their movement. In [2] it is shown that a multicopter can be controlled with one ore more lost actuators. This is because the system is designed to be overactuated. The performance and stability goes down, but the multicopter is still able to complete its purpose.

Another example of a system that often is overactuated is a car. To change the speed in the longitudinal direction an ordinary car is equipped with a combustion engine and a friction brake. The friction brake can be generate a large negative torque while the combustion engine can generate both negative and positive

torques. This makes the car overactuated since the negative torque can be generated by the two different actuators. In modern cars it is also common to have an electric machine since it affects the environment less than a combustion engine. This makes the car even more overactuated.

When driving there are several criterias that needs to be fulfilled. The speed that is desired by either the driver or some underlying system for speed control, like cruise control (CC) or adaptive cruise control (ACC), must be kept close to the desired level. At the same time the energy consumption must be kept as low as possible to minimize the cost for fuel and to minimize the effects on the environment because of emissions. Besides this the persons sitting in the vehicle should be experiencing a good comfort at the same time as the wear on the actuators should be minimized.

By using the fact that the vehicle is overactuated, these criterias can be accommodated when deciding how to control the vehicle. But this is a complex problem and this is why it is justified to believe that sophisticated methods are needed to obtain a near optimal solution.

## 1.1 Problem Formulation

Today the coordination part of control system in the Volvo cars for the longitudinal propulsion is mostly rule based for the different actuators and the control of the individual actuators is done with a PID-controller. The benefits with that solution is the simplicity and the robustness but the performance is not always optimal.

The goal with the thesis is to investigate if it is possible to achieve the same performance or improve the performance with a more sophisticated control structure, a model predictive controller (MPC). An MPC combines the possibility to predict the outcome through an open-loop controller with the stability of a closed-loop controller and gives the optimal solution for a finite horizon optimization problem.

The system needs to be modelled simple enough to be able to execute in real time while still reflect the dynamics to an acceptable level. The goal is to make the performance as good as possible without violating the constraints and limitations that are given for the system.

## 1.2 Previous Work

The basic theory of a model predictive controller is described in [3]. This is a course compendium for a course in industrial control engineering where MPC is described in a few chapters. There are a few extensions to the basic MPC design described in this book, for example how to handle integral action. Another publication that covers the MPC implementation and discussions about stability and robustness of the MPC is [4]. This article shows that stability can be obtained

by assigning the value function as a Lyapanov function, and therefor Lyapanov theory can be used to prove stability. Stability and Lyapanov functions is also covered by [5].

A usual extension of MPC that can be found in literature is the use of soft constraints or slack variables. This is done in [6]. The authors of this article introduces slack variables on the constraints they want to relax. They then penalizes the slack variable by assigning a cost to it instead of having an exact constraint. The article also shows that this solution will be the same as the exact constraint solution if the penalty on the slack variables is high enough.

The article [7] discusses the possibility to have a control horizon that is shorter than the prediction horizon. The benefit of this method is that the computational time is reduced since the number of variables is reduced. This is also covered in [8].

A comprehensive book in particular about predictive control for hybrid systems is [9]. It also covers the background theory of discrete optimization and convexity. It contains numerous useful theorems on convexity which is important to understand when solving optimization problems. It also describes the quadratic programming problem.

Many papers have been written about how to optimize the coordination of the actuators and find a global minimum using offline optimization methods. In [10], [11] and [12] dynamic programming (DP) and Pontryagin's maximum principle (PMP) algorithms are presented to illustrate the possible benefits with hybrid electrical vehicles (HEV).

[10] and [12] have also compared the offline solutions with equivalent consumption minimization strategy (ECMS) which is an instantaneous minimization method and the authors claims that it is possible to implement in real time.

An early MPC approach is used in [8] to control a electric vehicle with multiple energy storage units. The article also describe how zone control can be used in the MPC framework when it is desired to let a variable vary within a given interval. The performance of an MPC for a HEV is compared with both a DP and an ECMS approach in [13]. The conclusion is that the performance is good and there is several advantages such as it is potentially real-time implementable and rather easy to tune.

In [14] and [15] model predictive control allocation (MPCA) is described, an approach to coordinate the actuators for an overactuated system when a specific behavior is desired. The foucs in [14] is on how to do this for an system with different limitations and dynamics for the actuators.

## 1.3 Delimitations

The thesis is delimited to only discuss the use of a car with an internal combustion engine (ICE) and friction brake as actuators. A natural extension to this

would have been to use a HEV. This would have required to much effort in modelling that unfortunately there simply is not room for in this thesis. Another reason for this is that the test vehicle available is not a HEV.

The focus of the thesis will be on control. Advanced plant models are used in the evaluation of the controller, but these are not created by the thesis workers.

The controller is not designed for use when starting from standstill or stopping. The modelling close to zero speed is rather different from higher speeds. This will therefor be left out as a delimitation.

## 1.4 Thesis Outline

The thesis is divided into 6 chapters which contents are summarized here.

- **Chapter 1 - Introduction**
  The introduction chapter introduces the problem and states it in the problem formulation. A literature study on previous work is also presented.
- **Chapter 2 - Background**
  The background chapter describes the system on a basic level. It also describes a real life situation the controller is going to solve.
- **Chapter 3 - Theory**
  The theory chapter covers the theory that is used in the thesis. In some sections several methods to solve a problem is presented. This chapter does not describe what solution that is chosen in the implementation.
- **Chapter 4 - Theory**
  The method chapter describes the work developing the controller. First different plant models used during the development process is described. After that, the internal model for the controller is derived. The design of the cost function is presented and at last the implementation is briefly described.
- **Chapter 5 - Analysis**
  Results are shown and analyzed.
- **Chapter 6 - Conclusions and Future Work**
  Conclusions from the results are presented along with a section with suggestions on future work from the thesis workers.

# 2
# Background

The complex system of a car has an underlying computer architecture that performs various tasks in a chain to perform the actual control in the actuators. The actuators in themselves are part of bigger modules. This chapter's aim is for the reader to get an understanding of the system that the controller should be operating within, along with an understanding of the actuators that are used to control it.

## 2.1 System Overview

In Figure 2.1 a brief description of the parts of the system that is used for longitudinal speed control is described. This is, roughly, how it is implemented in the reference vehicle and therefor necessary to understand to be able to compare with the new implementation.

The bigger systems, referred to as nodes, Active Safety Domain Master (ASDM) and Vehicle Dynamics Domain Master (VDDM) are physical components or computers each running a number of tasks. The communication between them is done with a Control Area Network (CAN) bus.

The ASDM module contains functions like ACC and CC. The ACC is a controller that controls the speed depending on the traffic. Typically it is set to keep a desired distance to the vehicle ahead. When the situation changes it sends a request in acceleration to the VDDM node. The CC on the other hand is a controller that keeps a desired speed. When the speed is deviating from the reference the CC also sends a request in acceleration to the VDDM node. There are also other type of request sent by modules in the ASDM node. One examples of such a request is jerk limit requests that should tell the system the maximum jerk that is accepted.

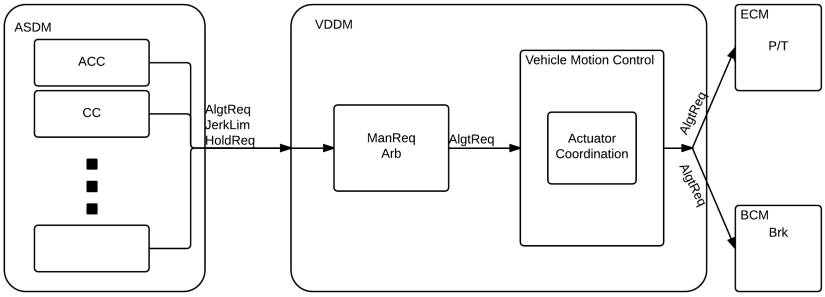


***Figure 2.1:*** *A system overview of the modules that controls the speed in a vehicle in the longitudinal direction in current implementation.*

The different modules in the ASDM node have overlapping interests. In the VDDM node the set of requests sent from the ASDM node are arbitrated to just one request in acceleration. This is then passed to the module Vehicle Motion Control where actuator coordination is performed. The actuator coordination is a set of rules that depending on the situation calculates acceleration requests to send to the Engine Control Module (ECM) and Brake Control Module (BCM). The ECM and BCM module are smart modules and contains controllers that converts the acceleration requests to a generated torque which affects the vehicle in the requested way.

When the problem, stated in section 1.1, is solved the system layout will change into the layout shown in Figure 2.2. The arbitration module and the actuator coordination module is merged to one block, still refered to as actuator coordination in Figure 2.2. This block is implemented as an MPC that handles all request from the ASDM node as inputs. Instead of a set of rules, the coordination is now calculated in a mathematical way to obtain an optimal solution by the MPC. Some of the smart functionallity in the ECM and BCM is now moved into the actuator coordination since the MPC needs to know the dynamics of these systems. Instead of acceleration requests, torque requests are now sent directly to the ECM and BCM.

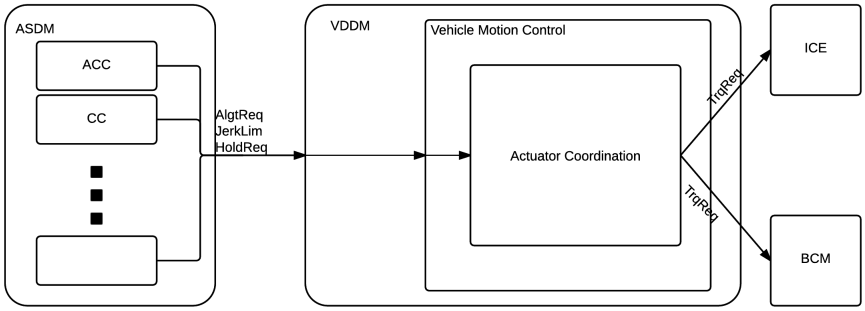


***Figure 2.2:*** *A system overview of the modules that controls the speed in a vehicle in the longitudinal direction if the new actuator coordination is implemented.*

## 2.2 Actuators

When designing a control system using an MPC it is of great importance to know the characteristics of the actuators that is to control the system. A basic understanding of how the actuator is constructed is needed for doing the modelling in the controller as well as the plant models used during the development of the controller.

The three properties of the actuators that is of particular interest is the dynamics, controllability and the ranges of the actuators. The dynamics refer to how fast the system responds to a control signal. The meaning of controllability in this context is the expected difference between the requested and received torque on the system from the actuator. The range here refer to the range of torques an actuator can deliver to the system. An overview of the specifications of the actuators is given in Table 2.1.

| | Combustion Engine | Friction Brake |
|---|---|---|
| Dynamics | Slow | Fast |
| Controllability | Mediocre | Good |
| Range | [ -small , large] | [-large, 0 ] |

***Table 2.1:*** *Overview of the actuator characteristics.*

The range the torques of the different actuators is roughly described in Figure 2.3. The first plot is the range for the ICE. The second plot describes range for friction brake.

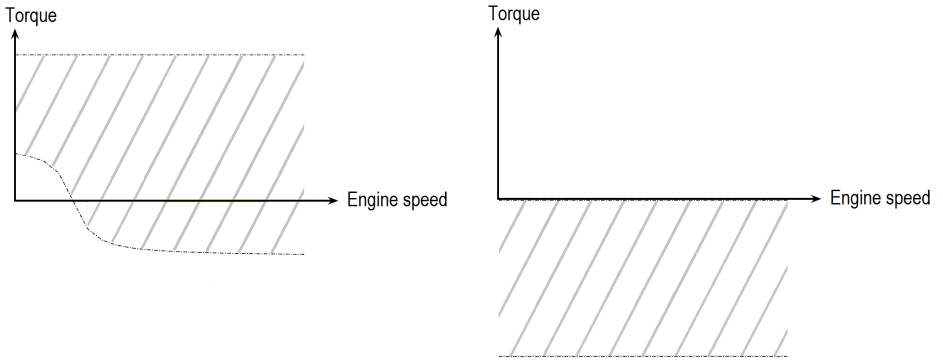


*Figure 2.3: Conceptual plots of intervals which the different actuator is able to generate torque for different engine speeds. To the left is the ICE and to the right is the friction brake.*

### 2.2.1 Combustion Engine

The combustion engine can generate a large torque giving the vehicle acceleration in the forward direction. The lower range is however depending on the engine speed. For low engine speeds the minimum torque that can be generated is positive. This means that the combustion engine as an actuator cannot be used for decelerating the vehicle if the engine speed is low. For high engine speeds however this can be done.

The controllability is mediocre because of the complex nature of the actuator. It is hard to get the exact torque as requested from the actuator the model will be far from exact. The dynamics is also rather slow compared to the friction brake.

### 2.2.2 Friction Brake

The friction brake has the ability to convert kinetic energy to heat energy by friction. When braking the brake system is building up a pressure and then applies the braking pads to create a friction force and slow the car down. When the system requests less braking force on the other hand the pressure must be relieved. That is a faster process than building the pressure but the process of building pressure is also fast compared to the ICE.

The friction brake can obviously not generate a positive torque, as long as the vehicle is moving forward, so its upper limit will always be zero. The lower limit will be a large negative torque that is proportional to the mass of the vehicle and the friction coefficient between the tires and the ground material. Since these parameters are not exact, and estimating them is not a part of this thesis, uncertainties in the range will lead to some effect on the controllability. But the controllability is still considering good compared to the ICE.

## 2.3 Use Cases

As a complement for the problem formulation use cases that put the problem in everyday traffic situations is constructed. In this section visual and verbal descriptions of the use cases are presented. All scenarios that is considered is constructed from an ACC perspective. That is situations where the ACC is the function that is sending the important requests.

There are four scenarios in total. The first two use cases are simple scenarios where both actuators may not be needed. These are constructed to test the basic functionality of the controller. The last scenario however is constructed to test the bridge areas where the actuators are expected to shift roles.

#### Case I - Deceleration Step

Consider driving in a multi-lane road as a host vehicle (H), following a target vehicle (T) using ACC. The controller will try to follow the estimated acceleration of T to keep a desired distance to it. This use case considers another vehicle cutting in between the target vehicle, called cut in target (CIT), and the target vehicle from another lane. At some point the CIT will become the new T and the distance is changed instantly. This will result in a step in the acceleration reference shown in Figure 2.6. The Figures 2.4 and 2.5 are visualizations of the scenario where it is obvious how the distance changes instantly. The case starts in the condition shown in Figure 2.4 and ends in condition shown in Figure 2.5.

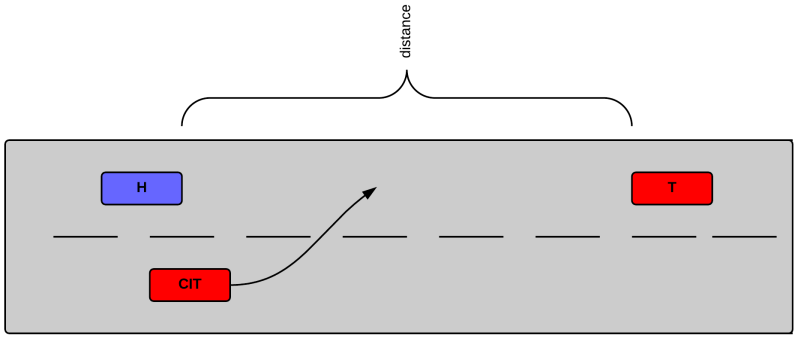


***Figure 2.4:*** *The start of Use Case Scenario I. The host vehicle is driving in a lane following a target vehicle when suddenly a cut in target is entering the lane between them.*

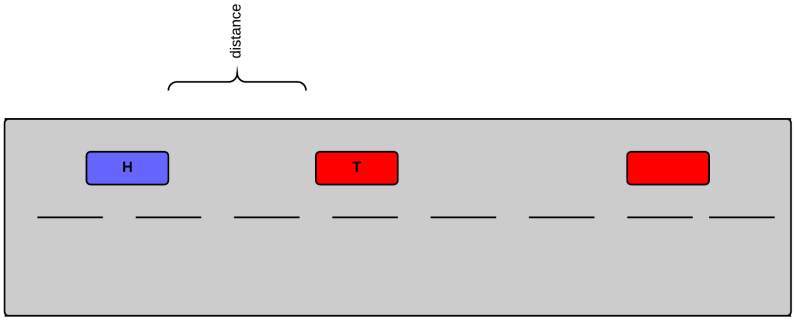


***Figure 2.5:*** *The end of Use Case Scenario I. The cut in target has become the new target vehicle and the distance to the target changes momentarily.*

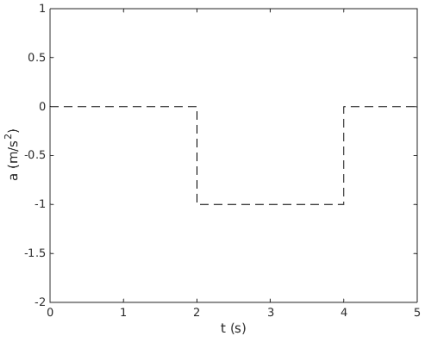


***Figure 2.6:*** *The resulting request in acceleration in use case I. The cut in target becomes the new target at time $t = 2$ and it has reached the desired distance at time $t = 4$.*

### Case II - Acceleration Step

Consider a host vehicle driving in a single lane road following a target vehicle with constant velocity by keeping a constant distance to it. The target vehicle is about to make a turn, and therefor it keeps a low speed. When the target vehicle turns the host vehicle can speed up to the current speed limit. The resulting acceleration reference is shown in figure 2.7.

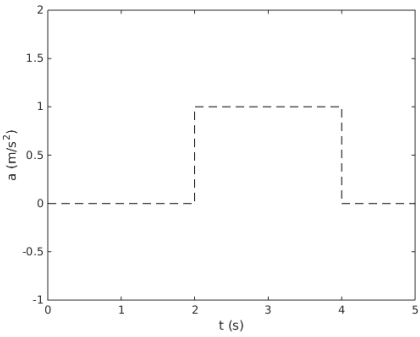


***Figure 2.7:*** *The resulting request in acceleration in use case II.*

### Case III - Deceleration after Acceleration

Consider the same setup as in use case I with the difference that H is accelerating at the start of the scenario. This means that when the CIT enters the lane becoming the new T the reference in acceleration changes from positive to negative instantly. This is shown in figure 2.8.

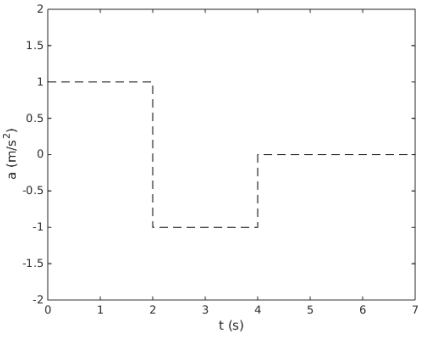


***Figure 2.8:*** *The resulting request in acceleration in use case III.*

# 3
# Theory

This chapter covers the theory needed for this thesis. The basics of mathematical optimization is presented to have just the right amount of theory needed for applying it on the MPC. The MPC which is the single most important theoretical subject in the thesis is covered in more dept. The basic concept of MPC is described together with an algorithm that describes roughly how an MPC should be implemented. After that, a section with extensions to the basic MPC framework is presented. This section also cover the subjects of following a non constant reference path, including integral action in the controller and using slack variables to create soft constraints. Finally stability theory of the MPC is presented.

## 3.1 Optimization

A mathematical optimization problem is expressed on the form (3.1). Where $f_0$ is the objective function, $f_i$ is the inequality constraint functions, $b_i$ is the limits and $x$ is the optimization variable [16].

$$\begin{aligned} &\text{minimize} && f_0(x) \\ &\text{subject to} && f_i(x) \leq b_i \quad , \quad i = 1, ..., m \end{aligned} \tag{3.1}$$

The aim when solving a mathematical optimization problem is to find the optimal solution, $x^*$ for which the objective function in that state, $f_0(x^*)$ is the minimum possible value for among all $x$ satisfying all the inequality constraint functions.

### 3.1.1 Convexity

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if the set of points on and above $f$ is a convex set. A convex set is a set in which it is possible to connect every two points in the set with a straight line without touching a point outside of the set [16]. This is illustrated in the Figures 3.2 and 3.1.

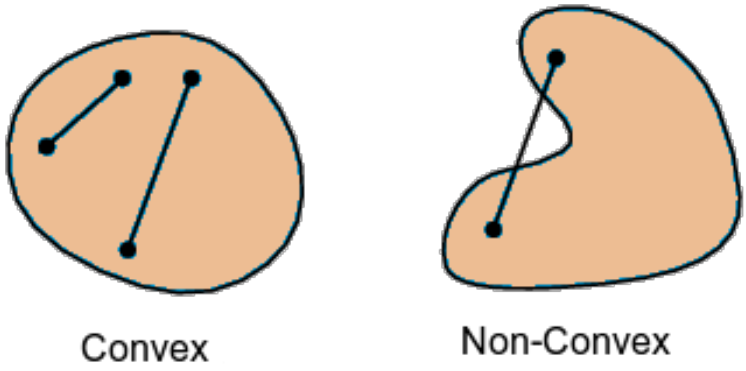


***Figure 3.1:*** *An illustration of the difference between a convex and a non-convex set.*

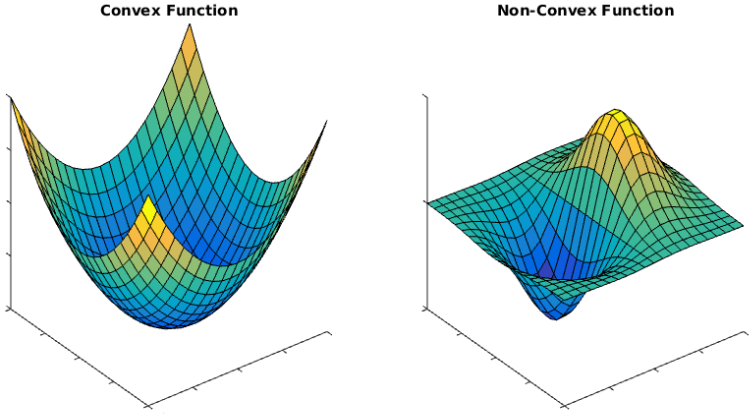


***Figure 3.2:*** *An illustration of the difference between a convex and a non-convex function. This example is in 3D, but the same theory holds for any dimension.*

### 3.1.2 Convex Optimization

The characteristics of the optimization problem (3.1) is determined by the objective function and the inequality constraint functions. Some useful classes of optimization problem is linear programming (LP) where the objective function is linear in $x$ and quadratic programming (QP) where the objective function is on the form (3.2) where $H$ is a constant square matrix and $f$ is a constant vector.

$$f_0(x) = \frac{1}{2}x^T Hx + f^T x \tag{3.2}$$

These two classes of problems is in some cases both part of a greater class, convex optimization problems. This is true if $f_0, ..., f_m$ is convex. The objective function for an LP problem is always convex since it will be a plane. The objective function in the QP problem however, is only convex when $H$ in (3.2) is positive semidefinite [16]. The inequality constrain functions in both LP and QP is often assumed to be linear, and if it is, it is also convex.

A subclass of QP problems is quadratic constraints quadratic programming (QCQP) problems. This is a QP problem with quadratic inequality constraint functions. This can of course also be a convex problem.

The strength with convex optimization problems is that there exists many powerful numerical solvers that can solve this class of problems accurately and fast. LP problems is often solved using simplex method while QP problems often is solved with interior-point methods. Solvers for QCQP problems is unfortunately more unfrequently occurring.

## 3.2 Discretization

A tool that is needed in the implementation of the controller is the ability to transform a continuous time state space model to a discrete time representation. Several methods exists for doing this. Examples of such methods are Euler forward or backwards transformations as well as Tustin's method [3].

### 3.2.1 Euler's Method

By making the transformation in (3.3) a system is discretized with backward Euler's method. Here $T_s$ is the sampling time of the time discrete system and $z$ is a time shift operator. The Euler's transformation projects the entire left half of the s-plane on a circle contained inside the unit circle in the z-plane. This is visualized in Figure 3.3

$$s = \frac{1}{T_s}\left(1 - z^{-1}\right) \tag{3.3}$$

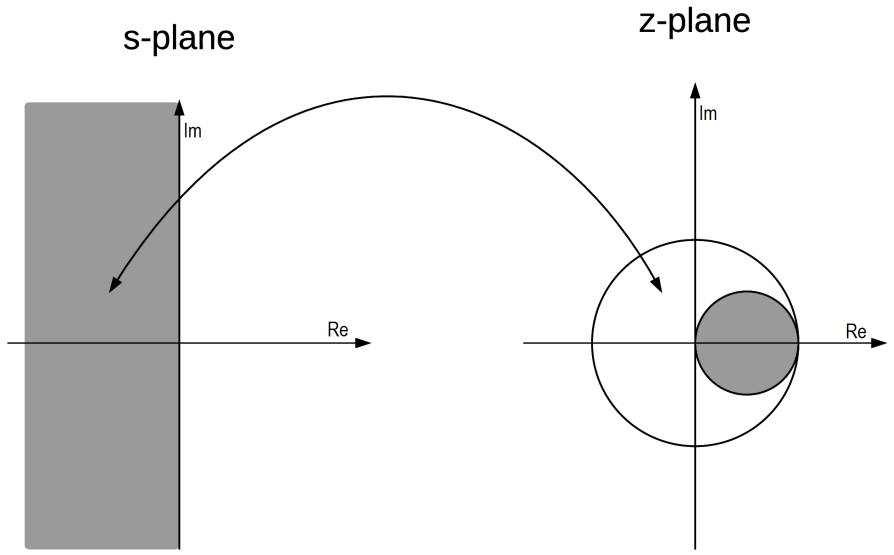


***Figure 3.3:*** *The projection of the s-plane on the z-plane when discretizing a system with Euler's backwards method.*

### 3.2.2 Tustin's Method

Tustin's method or bilinear transformation as it is also called works by replacing the time derivative operator occurring in a time continuous system by the expression (3.4). Here $T_s$ is the sampling time of the time discrete system and $z$ is a time shift operator.

$$s = \frac{2}{T_s} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \tag{3.4}$$

This method is a Möbius transform which projects the entire left half plane on the unit circle. This way stability is guarantied in the discrete system if the continuous system is stable. The transformation is visualized in Figure 3.4.

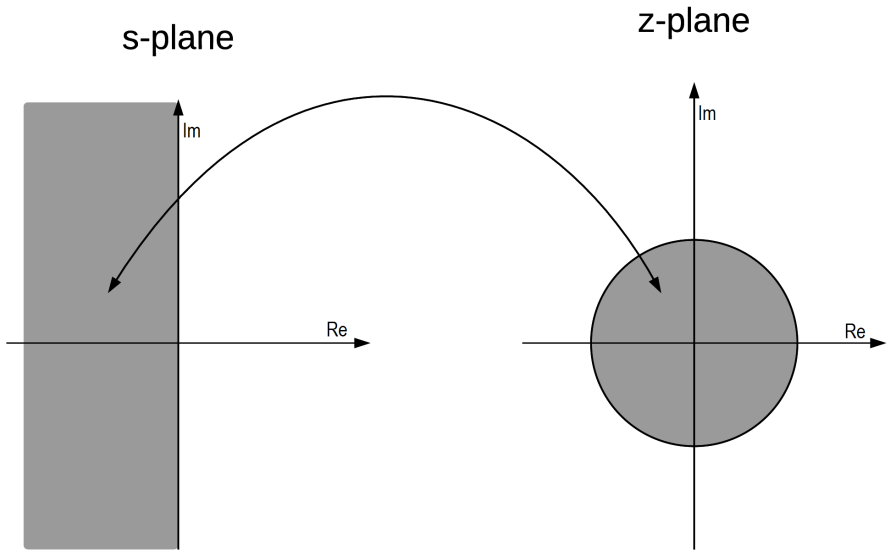


***Figure 3.4:*** *The projection of the s-plane on the z-plane when discretizing a system with Tustin's method.*

## 3.3 Model Predictive Controller

A model predictive controller (MPC), described in [3], is a controller that uses mathematical optimization to calculate a control signal. The controller contains an internal model of the system that is to be controlled that is an approximation of the real system, called a plant. The output of the plant often needs to be processed in some way to estimate the actual states. The setup is described in Figure 3.5.

The idea of the MPC is to solve a mathematical optimization problem in every time sample. This optimization problem is on the form (3.5). $J(x)$ is the objective function, a function that introduces penalties on states and inputs. The equation $Ax \leq b$ describes a set of linear constraints for the variable that should be minimized. A prediction horizon of $N$ samples is used to make use of the information contained in the internal model. But the problem grows with $N$ and thus it can only be as large as the available computational power admits [3].

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & J(x) \\ \text{subject to} \quad & Ax \leq b \end{aligned} \tag{3.5}$$

The limited computational power are in many ways correlated to the limits of the

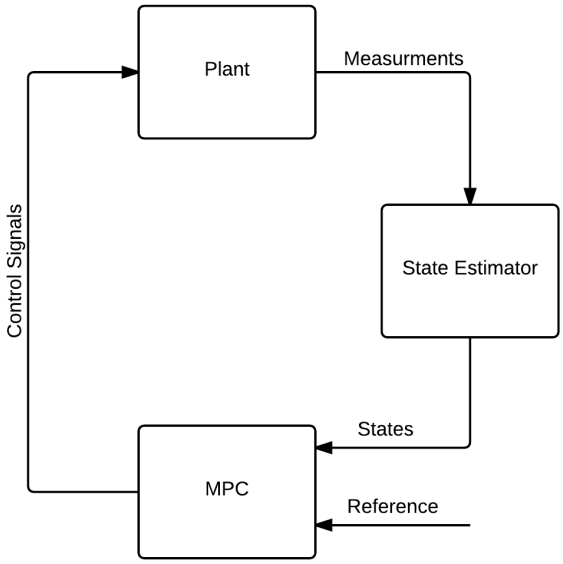


***Figure 3.5:*** *A conceptual overview of an MPC setup.*

MPC. The objective function can in theory be any function. But since minimizing an arbitrary function often requires numerical methods the computational time is too high for real time applications. A workaround for this is to let J be a quadratic function of the form (3.2). The problem will then be a quadratic programming problem for which there exist numerous solvers.

### 3.3.1 MPC Algorithm

The loop in which the MPC runs begins every iteration with estimating the states, $x$. Some states may be possible to measure directly with a sensor of some kind. Others may need to be estimated using measurement of other parameters, models and filters like the Kalman filter. The second step is solving the optimization problem stated in the form (3.5). When the optimization problem is solved a sequence of control signals that covers the entire prediction horizon is obtained. The first control signals in the sequence is applied and the rest of them are thrown away. After this the loop is closed by performing a time update and start from the beginning. The algorithm is summarized in Algorithm 1.

### 3.3.2 Construction of Internal Model

The MPC requires an internal model of the system. This needs do be expressed in state space form to fit the QP problem formulation. The model is formulated as (3.6).

**Algorithm 1** The MPC algorithm

1: Measure x(k)
2: Obtain the control signal sequence u( · ) by solving (3.5)
3: Apply the first element u(k) in the control signal sequence during one sample.
4: Time update, k := k + 1
5: Repeat from step 1

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \\ z(t) &= Mx(t) \end{aligned} \tag{3.6}$$

Where $x(t)$ is the states, $u(t)$ is the control signals, $y(t)$ is the measured output from the plant. $A$, $B$ and $C$ is matrices that describes the system. $z(t)$ is linear combinations of states that should be controlled.

To be able to implement this the state space model needs to be discretized. The bilinear transformation discretization method described in section 3.2 is used to do this. The state space form can then be expressed as equation (3.7). Where $F$ and $G$ is the new system matrices after the discretization.

$$\begin{aligned} x_{k+1} &= Fx_k + Gu_k \\ y_k &= Cx_k \\ z_k &= Mx_k \end{aligned} \tag{3.7}$$

The last step that needs to be done for an implementation supported by the QP problem formulation is repeating the discretized model trough the prediction horizon, $N$, to achieve a vectorized notation.

Given the recursive formula in (3.7) $x_{k+2}$ can be expressed as in (3.8)

$$\begin{aligned} x_{k+2} &= Fx_{k+1} + Gu_{k+1} \\ &= F^2x_k + FGu_k + Gu_{k+1} \end{aligned} \tag{3.8}$$

The same recursive formula can be used for the entire prediction horizon. The recursion formula then gives the expression (3.9).

$$\begin{aligned} x_{k+N} &= Fx_{k+N-1} + Gu_{k+N-1} \\ &= F^Nx_k + F^{N-1}Gu_k + F^{N-2}Gu_{k+1} + \cdots + Gu_{k+N-1} \end{aligned} \tag{3.9}$$

By using defining $\mathcal{F}$, $\mathcal{G}$, $X$ and $U$ according to (3.10) and (3.11) the outcome of the recursive formula in equation (3.9) can be used to express the system in a

vectorized notation over the entire prediction horizon. This is done in equation (3.12).

$$\mathcal{F} = \begin{bmatrix} F \\ F^2 \\ \vdots \\ F^N \end{bmatrix}, \quad \mathcal{G} = \begin{bmatrix} G & 0 & \dots & 0 \\ FG & G & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ F^{N-1}G & F^{N-2}G & \dots & G \end{bmatrix} \tag{3.10}$$

$$U = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-1} \end{bmatrix}, \quad X = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+N} \end{bmatrix} \tag{3.11}$$

$$X = \mathcal{F}x_k + \mathcal{G}U \tag{3.12}$$

When using the vectorized notation in equation (3.12) the penalty matrices, $Q_z$ and $Q_u$, as well as the matrix $M$ must be repeated over the prediction horizon according to equations (3.14) and (3.13).

$$\mathcal{M} = \begin{bmatrix} M & & & \\ & M & & \\ & & \ddots & \\ & & & M \end{bmatrix} \tag{3.13}$$

$$\mathcal{Q}_Z = \begin{bmatrix} Q_z & & & \\ & Q_z & & \\ & & \ddots & \\ & & & Q_z \end{bmatrix}, \quad \mathcal{Q}_U = \begin{bmatrix} Q_u & & & \\ & Q_u & & \\ & & \ddots & \\ & & & Q_u \end{bmatrix} \tag{3.14}$$

With all the definitions above the objective function can now be described in the vectorized notation. This is done in equation (3.15).

$$\begin{aligned} \mathrm{J} &= \sum_{j=0}^{N-1} \|z_{k+j}\|_{Q_z}^2 + \|u_{k+j}\|_{Q_u}^2 = \\ &= Z^T \mathcal{Q}_Z Z + U^T \mathcal{Q}_U U = X^T \mathcal{M}^T \mathcal{Q}_Z \mathcal{M} X + U^T \mathcal{Q}_U U = \\ &= (\mathcal{F}x_k + \mathcal{G}U)^T \mathcal{M}^T \mathcal{Q}_Z \mathcal{M}(\mathcal{F}x_k + \mathcal{G}U) + U^T \mathcal{Q}_U U \end{aligned} \tag{3.15}$$

Equation (3.15) can then be rewritten to fit the QP-problem formulation described in section 3.1.2. This is done in eqation (3.16).

$$\begin{aligned} \mathrm{J} = x_k^T \mathcal{F}^T \mathcal{M}^T \mathcal{Q}_Z \mathcal{M} \mathcal{F} x_k \\ +2x_K^T \mathcal{F}^T \mathcal{M}^T \mathcal{Q}_Z \mathcal{M} \mathcal{G} U + U^T \left( \mathcal{G}^T \mathcal{M}^T \mathcal{Q}_Z \mathcal{M} \mathcal{G} + \mathcal{Q}_U \right) U \end{aligned} \tag{3.16}$$

With the theory presented in this section a basic MPC can be implemented.

### 3.3.3 Extensions to MPC

To extend the MPC framework from the standard MPC described in [3] some areas can be developed further. In this section techniques used in the implementation of the controller is described in detail. All extensions are presented as an extension to the standard framework alone and the extensions are not combined in this section.

#### Following Reference Path

In the objective function, costs are assigned to the states with the cost matrix $Q_z$ for differences in the states in relation to the origin. This works perfectly fine for any constant reference signal since the coordinate system can be adjusted to fit the situation. For a reference signal that changes over time however, the MPC needs support for keeping the system from differing too much from the reference signal.

For a reference signal $R$ in (3.17), the objective function is modified to (3.18). The result of this is that deviations from the reference signal instead of from the origin is penalized with the cost matrix $Q_z$.

$$R = \begin{bmatrix} r_k \\ r_{k+1} \\ \vdots \\ r_{k+N-1} \end{bmatrix} \tag{3.17}$$

$$\begin{aligned} \mathrm{J} = \sum_{j=0}^{N-1} \|z_{k+j} - r_{k+j}\|_{Q_z}^2 + \|u_{k+j}\|_{Q_u}^2 = \\ = (Z-R)^T \mathcal{Q}_Z (Z-R) + U^T \mathcal{Q}_U U = \\ = (\mathcal{M}X - R)^T \mathcal{Q}_Z (\mathcal{M}X - R) + U^T \mathcal{Q}_U U = \\ = (\mathcal{F}x_k + \mathcal{G}U - R)^T \mathcal{M}^T \mathcal{Q}_Z \mathcal{M} (\mathcal{F}x_k + \mathcal{G}U - R) + U^T \mathcal{Q}_U U \end{aligned} \tag{3.18}$$

### Integral Action

If the deviation from the reference and the signal amplitude is penalized as in equation (3.15) there is contradiction between the two wills. At some point the cost for differing from the reference will be equal to the cost for the signal amplitude. The result of this is that there is no gain in using the control signals such that the deviation from the reference becomes smaller. This is the result of having no integral action in the controller.

The solution to this problem is to introduce a cost for increment in the control signals as in equation (3.19). This will serve as integral action.

$$\mathrm{J} = \sum_{j=0}^{N-1} \|z_{k+j}\|_{Q_z}^2 + \|u_{k+j} - u_{k+j-1}\|_{Q_{\Delta u}}^2 \tag{3.19}$$

A vectorized notation of the increment in the control signal can be seen in (3.20).

$$\begin{bmatrix} u_k - u_{k-1} \\ u_{k+1} - u_k \\ \vdots \\ u_{k+N-1} - u_{k+N-2} \end{bmatrix} = \begin{bmatrix} I & & & \\ -I & I & & \\ & \ddots & \ddots & \\ & & -I & I \end{bmatrix} U - \begin{bmatrix} u_{k-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \tag{3.20}$$
$$= \Omega U - \delta$$

With the notation in (3.20) the vectorized objective function can be expressed as in

$$\mathrm{J} = (\mathcal{F} x_k + \mathcal{G} U)^T \mathcal{M}^T \mathcal{Q}_Z \mathcal{M} (\mathcal{F} x_k + \mathcal{G} U) + (\Omega U - \delta)^T \mathcal{Q}_U (\Omega U - \delta) \tag{3.21}$$

### Control Allocation

Consider an overactuated system where several possible combinations of control signals exist for forcing the system to a desired state. If a specific combination is desired it is not obvious that the output will be that combination when solving the optimization problem. To help the controller on its way to a desired state, information indicating a probable behavior of the controller can be used. In situations like that control allocation might be needed.

In [14] and [15] the idea is presented to calculate a virtual control signal, $u_{ref}$. The calculation is made based on the what information the current state of the system together with the reference signal can provide for predicting where the system is heading. This technique is called model predictive control allocation (MPCA). MPCA uses an outer and an inner loop. The inner loop is the standard MPC loop with a controller and a plant model while the outer loop is the one containing the control allocation. This is visualized in Figure 3.6.

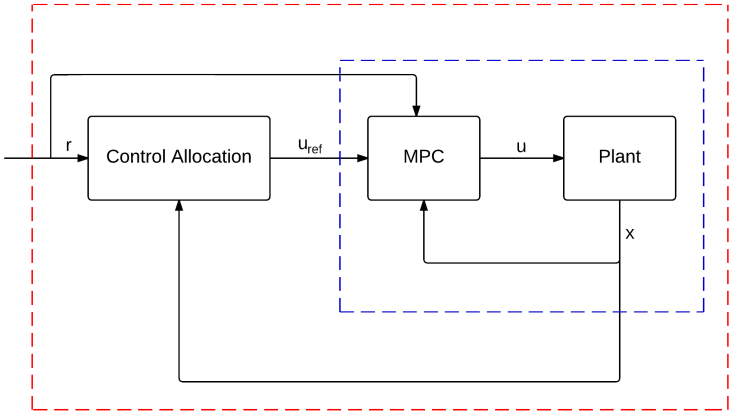


***Figure 3.6:*** *An illustration of the model predictive control allocation. The virtual control signal $u_{ref}$ is calculated in the outer loop and then the actual control signal $u$ is calculated in the inner loop with the MPC. The signal $u_{ref}$ is a rough calculation of the desired behavior of the control signal $u$.*

An additional extension to this is to assume a level of uncertainty in the calculated virtual control signal. That means the MPC should assign a cost to deviations from an interval around $u_{\text{ref}}$. This is motivated to use because if it where believed that the virtual control signal is the best control signal at a specific time it should be used directly as input to the plant. Since that is most probably not the case, the control allocator calculation is used as a rough estimate and then the MPC takes it closer to the optimal solution from there.

In equation (3.22) the objective function of the MPC is extended with the new feature. Here $\epsilon$ is the size of the interval in which the control signal is allowed to deviate from the virtual control signal.

$$\begin{aligned} \text{J} = \sum_{j=0}^{N-1} \|z_{k+j}\|_{Q_z}^2 + \epsilon_{Q_\epsilon}^2 \\ \|u_{ref} - u_{k+j}\| \leq \epsilon \\ \epsilon \geq 0 \end{aligned} \tag{3.22}$$

### Constraints on the control signal

A big benefit with formulation the MPC problem as a QP problem is the possibility to use linear constraints on the variables, in our case the control signals. Constraints can be used to describe limitations on the control signals that should not be violated. A usual technique is to specify an upper and a lower limit on the control signal in each time sample which is a reasonable assumption that many

physical systems have. In (3.24) it is shown how to write the limitations in (3.23) on the QP customized structure.

$$u_{min} \leq u \leq u_{max} \tag{3.23}$$

$$\begin{bmatrix} I & & \\ & \ddots & \\ & & I \\ -I & & \\ & \ddots & \\ & & -I \end{bmatrix} U \leq \begin{bmatrix} u_{max} \\ \vdots \\ u_{max} \\ -u_{min} \\ \vdots \\ -u_{min} \end{bmatrix} \quad = \quad A_u U \leq b_u \tag{3.24}$$

### Constraints on the states

Oftentimes it is also desirable to be able to have constraints on the states and let them vary within a given interval as in (3.25).

$$z_{min} \leq z \leq z_{max} \tag{3.25}$$

However it is only possible in the QP framework to have constraints on the variable, in this case $U$. A method for overcoming that problem is to express the states $Z$ as an function of $U$ as in (3.26). Then the interval from (3.25) can be expressed as (3.27) which is an inequality only containing terms of $U$ or constants. The inequalities can then be expressed as (3.28) which fits in the QP problem framework.

$$Z = \mathcal{M}(\mathcal{F}x_k + \mathcal{G}U) \tag{3.26}$$

$$\begin{aligned} \mathcal{M}(\mathcal{F}x_k + \mathcal{G}U) &\leq \begin{bmatrix} z_{max} \\ \vdots \\ z_{max} \end{bmatrix} \\ -\mathcal{M}(\mathcal{F}x_k + \mathcal{G}U) &\leq -\begin{bmatrix} z_{min} \\ \vdots \\ z_{min} \end{bmatrix} \end{aligned} \tag{3.27}$$

$$\begin{bmatrix} \mathcal{M}\mathcal{G} \\ -\mathcal{M}\mathcal{G} \end{bmatrix} U \leq \begin{bmatrix} z_{max} \\ \vdots \\ z_{max} \\ z_{min} \\ \vdots \\ z_{min} \end{bmatrix} + \begin{bmatrix} -\mathcal{M}\mathcal{F}x_k \\ \mathcal{M}\mathcal{F}x_k \end{bmatrix} \quad = \quad A_z U \leq b_z \tag{3.28}$$

### Soft Constraints

In difference to the constraints described in the previous section, soft constraints is a method used to soften the constraints and bring the controller back to a desired state even if the constraints are violated [6].

The soft constraints is constructed with help from slack variables $\epsilon$. One slack variable is introduced for every constraint that is desired to be softened. The variable connected to a certain constraint is defined to be zero when the constraint is not violated and non-zero when it is.

$$\begin{aligned} z &\leq z_{max} + \epsilon \\ \epsilon &\geq 0 \end{aligned} \tag{3.29}$$

The slack variable, $\epsilon$, is a variable that the optimization problem solver should find an optimal solution for just as the control signals. It is therefor placed in the vector with the real control signals like in (3.30).

$$U_e = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-1} \\ \epsilon_k \\ \epsilon_{k+1} \\ \vdots \\ \epsilon_{k+N-1} \end{bmatrix} \tag{3.30}$$

A quadratic cost term is applied for the slack variable to penalize behavior that violates the constraints as in equation (3.31). The cost can be tuned like any other cost in the cost function. When using this technique it is made sure that the optimization problem solver finds a feasible solution. But by making it expensive to violate the constraints it is stated that it is of greatest importance for the solver not to violate the constraints.

If soft constraints not is used the optimization problem may be too hard for the

solver to handle and the controller eventually will end up in an undefined behavior depending on which solver that is used.

$$J = \sum_{j=0}^{N-1} \|z_{k+j}\|_{Q_z}^2 + \|u_{k+j}\|_{Q_u}^2 + \|\epsilon_{k+j}\|_{Q_\epsilon}^2 \tag{3.31}$$

### Time Delays

In physical systems a request to the actuators does not respond immediately, some delay in the signal is always to expect. Depending on the length of the delay time it might be necessary include in the model of the actuators to get sufficiently high accuracy.

In the discrete state space model it is more relevant to refer to the delay in terms of number of samples, $\eta$, rather then the delay time, $L$, which easily can be calculated by

$$\eta = \text{round}\left(\frac{L}{T_s}\right) \tag{3.32}$$

If we considering a discrete system controlled by only one control signal with input delay, the recursive system can be expressed as (3.33).

$$\begin{aligned}
x_{k+1} &= Fx_k + Gu_{k-\eta} \\
x_{k+2} &= Fx_{k+1} + Gu_{k+1-\eta} = F^2x_k + FGu_{k-\eta} + Gu_{k+1-\eta} \\
&\vdots \\
x_{k+N} &= Fx_{k+N} + Gu_{k+N-\eta} = \\
&F^Nx_k + F^{N-1}Gu_{k-\eta} + F^{N-2}Gu_{k+1-\eta} + \cdots + Gu_{k+N-1-\eta}
\end{aligned} \tag{3.33}$$

In (3.34) this equation is expressed in a more compact manner with vector notation where $\mathcal{F}$, $\mathcal{G}_U$, $\mathcal{G}_\delta$, $X$, $\delta$ and $U$ is defined as in (3.35) - (3.37).

$$X = \mathcal{F}x_k + \mathcal{G}_U U + \mathcal{G}_\delta \delta \tag{3.34}$$

$$\mathcal{G}_U = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ G & 0 & \dots & 0 & 0 & \dots & 0 \\ FG & G & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 & \dots & 0 \\ F^{N-1-\eta}G & F^{N-2-\eta}G & \dots & G & 0 & \dots & 0 \end{bmatrix} \tag{3.35}$$

$$\mathcal{G}_\delta = \begin{bmatrix} G & 0 & \dots & 0 \\ FG & G & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ F^{\eta-1}G & F^{\eta-2}G & \dots & G \\ \vdots & \vdots & & \vdots \\ F^{N-1}G & F^{N-2}G & \dots & F^{N-\eta}G \end{bmatrix} \tag{3.36}$$

$$\delta = \begin{bmatrix} u_{k-\eta} \\ u_{k-\eta+1} \\ \vdots \\ u_{k-1} \end{bmatrix}, \quad X = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+N} \end{bmatrix}, \quad U = \begin{bmatrix} u_{k+1} \\ u_{k+2} \\ \vdots \\ u_{k+N} \end{bmatrix} \tag{3.37}$$

This formula is also valid for a system with more than one control signal but with the same time delay on all inputs. However for a system with more than one control signal and different input delays the matrix $G$ and has to be divided into several matrices, all with the same size as $G$ but every new matrix, $G_i$, is supposed to only include the interaction on the system from actuator $i$.
By creating the new matrices $\mathcal{G}_{U_i}$ and $\mathcal{G}_{\delta_i}$ (3.34) can be replaced by (3.38) where $n$ is the number of control signals.

$$X = \mathcal{F}x_k + \sum_{i=1}^{n} \mathcal{G}_{U_i} U + \sum_{i=1}^{n} \mathcal{G}_{\delta_i} \delta \tag{3.38}$$

### 3.3.4 Stability

Before discussing stability of an MPC one should first define what stability means for an MPC. In [3] stability is defined as the ability to always find a feasible solution without violating the constraints given that the problem initially got a feasible solution.

In the case with no constraints, it is proved in [3] that it exist an optimal state feedback which easily can be analytically calculated off-line and implemented

on-line. However constraints are usually present in physical systems and the unconstrained case is not of interest in this thesis.

With constraints present the problem become more complex and one issue with the finite prediction horizon MPC is the absence of general method to ensure stability. For an infinite prediction horizon it is possible to prove stability according to [3] if it initially exist a feasible solution for the optimization problem, even for a constrained system. Although for obvious reasons that is not an option in practice to have an infinite prediction horizon.

In [3] and [17] it is considered to mainly be a problem from a theoretical view. In practice one can achieve an locally stable controller with some well thought-out strategies.

A common cause for instability is a too short prediction horizon. One should choose a prediction horizon long enough to cover the significant dynamic of the system.

Another important aspect to ensure stability is to formulate the constraints so that a feasible solution always exist. Earlier in section 3.3.3 soft constraints were introduced, an approach that is highly recommended to use on constraints for the states of the system. Due to the fact that the internal model never is perfect and external disturbance may occur the states might assume an illegal values and cause no feasible solution to exist if hard constraints are used.

Even if it doesn't exist any general method to obtain stability, many approaches is developed trying to get a stable closed loop system. Below is a number of strategies to stabilize the system.

#### Terminal Equality Constraint

A very straight forward and simple approach to force the system to stability is to apply a equality constraint on the last state in the prediction horizon, $z_{k+N}$, as can be seen in (3.39)

$$z_{k+N} = 0 \tag{3.39}$$

In [4] and [17] the method is presented. A major drawback with the terminal equality constraint is the fact that feasibility not is guaranteed. With a too short prediction horizon there may not be possible to fulfill the terminal equality constraint given the systems dynamics and constraints on the actuators .

#### Terminal cost function

Another approach to ensure stability for the closed loop system is to add an terminal cost function. The advantage of this method compared to the previous one is that there will always exist a feasible solution to the optimization problem. However the system has to be stable for this approach to work which is proven in [3].

### Contraction Constraint

The final approach that is presented is the contraction constraint approach described in [18]. The idea is to require that the norm of the states are decreasing like in equation(3.40) where $\alpha$ is a scalar that should be chosen such that a feasible solution exists.

$$\|z_{k+1}\| \leq \alpha \|z_k\|, \quad \alpha < 1 \tag{3.40}$$

# 4
# Method

In this chapter the development process of the MPC is described. First the plant models used during the process is described in detail including description of the test vehicle. After that a state space model of the system is derived and reshaped to a form that can be used in the MPC. This is followed by construction of the cost function with motivated choices of techniques from the theory presented in chapter 3. Finally the implementation is presented.

## 4.1 Plant Models

When developing an MPC in a simulation environment like Simulink a plant model is needed. The plant model is the substitute for the real world system and should therefor be as good as it possibly can before the MPC is applied in reality. But it can also be kept simple to facilitate the development process of the MPC. Two different plant models was used before moving on to the implementation of the MPC in a real car. The following is a summary of these.

**Stage 1:** Use a simple model very much like the one used as internal model in the controller.

**Stage 2:** Use a more advanced Simulink model.

**Stage 3:** Use a real car.

### 4.1.1 Road load

The road load is the sum of all the external forces that affects the car that not is seen as a disturbance. That is air resistance, rolling resistance and slope. There are a few differences in how the road load modelling is done across the different

plant models and the internal controller model. The basic concepts are presented here while the variations is described in each section.

The air resistance is modelled as

$$F_{\text{air}} = -\frac{1}{2}\rho C_d A_a v^2 \tag{4.1}$$

where $\rho$ is the density of air, $C_d$ is the drag coefficient, $A_a$ is the maximum cross section area of the car and $v$ is the velocity of the car.

The rolling resistance is modelled as

$$F_{\text{roll}} = -C_{rr} mg \tag{4.2}$$

where $C_{rr}$ is the dimensionless rolling resistance coefficient, $\mu$ is the friction coefficient, $m$ is the mass of the car and $g$ is the gravity constant.

The force from the slope is modelled as

$$F_{\text{slope}} = -mg \sin \alpha \tag{4.3}$$

where m is the mass of the car, g is the gravity constant and $\alpha$ is the angle shown in Figure 4.1. $\alpha$ is assumed to be known at all times.

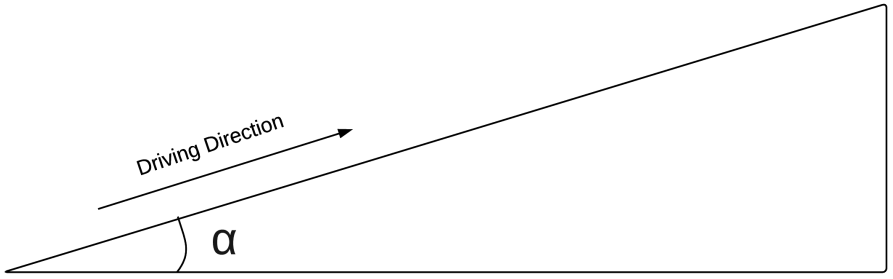


***Figure 4.1:*** *Visualization of the slope angle, $\alpha$.*

The sum of these three are defined as the road load, $F_{\text{road load}}$.

$$F_{\text{road load}} = -\frac{1}{2}\rho C_d A v^2 - C_{rr}\mu mg - mg \sin \alpha \tag{4.4}$$

### 4.1.2 Simple Simulink Model

At the first stage in the development of the MPC-framework a simple model very much like the model used in the controller was used. This uses the exact same models as the controller uses except for the linearization that is performed in the road load internal model (see section 4.2.4). This model is only used in the development process of the MPC-framework because it is a very rough approximation of the real world. But it has many advantages when it comes to evaluation of the controller structure.

### 4.1.3 Advanced Simulink Model

The second stage in the process of developing the controller is using a more advanced Simulink model developed by Volvo Cars as plant model. The complexity of this model makes it a good conversion between the simple model and the real car. Most of the development is done in this environment. The structure of the advanced Simulink model is shown in Figure 4.2. The parts of this system is described further in this section.

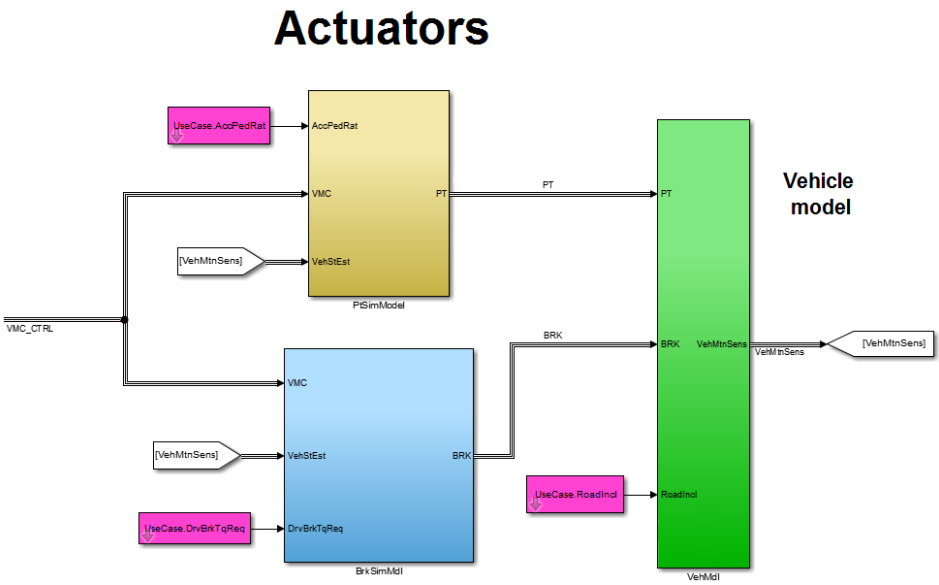


***Figure 4.2:*** *An overview of the subsystems that exists in the advanced Simulink model. The powertrain model (upper left) consists of a engine model and a gearbox model. The lower left subsystem is representing the friction brake. Finally, the right model contains the vehicle dynamics. The requests to the actuators are two of the signals on the bus signal that is the input to both actuators. The signals the are outputs from the actuators and inputs to the vehicle model are torques generated by the actuators.*

### Powertrain

The powertrain model consists of a engine model and a gearbox model. On top of that are several subsystems containing logic for choosing input signals to the engine. The application of the logics lies outside the scope of this thesis. That is why it is desired to keep the control of the input outside of the model at this stage. Therefor these subsystems containing logic is given input such that the control over the input to the engine is kept outside the model.

The powertrain model also contains a model of the force generated by the moment of inertia in the powertrain. This is described by equation (4.5) where $J_{\text{eng}}$ is the moment of inertia in the powertrain, $k_{\text{trans}}$ is the transmission ratio and $r_w$ is the radius of the wheels. The derivation of this expression is done in Appendix A.

$$F_{\text{pt,in}} = -\frac{J_{eng} k_{trans}^2}{r_w} a \tag{4.5}$$

### Friction Brake

The friction brake works by building a pressure that can be applied to the tires by the braking pads. When the braking is no longer requested the pressure that have been built must be released. These are two separate processes that are modelled by two separate second order systems. The system modelling the building of the pressure is slower than the system modelling the release of the pressure.

The input signals to the model is safety, comfort and driver minimum brake torque request. Just like the case with the powertrain model the control is maintained outside the model. The different signals are fed trough an arbitration block that makes the largest negative signal the brake request. To make the control easy, all signals except the safety brake torque request is set to zero.

### Vehicle Dynamics

The vehicle dynamics part of the model is the one implementing the laws of motion. The torques generated in the powertrain and the friction brake will be transferred to motion that will affect the vehicle's states in the longitudinal direction. The model is roughly described by equation (4.6). This is expressed in terms of forces, but can trivially be expressed in torques if desired.

$$F = F_{\text{eng}} + F_{\text{brake}} + F_{\text{road load}} \tag{4.6}$$

Here $F_{\text{eng}}$ and $F_{\text{brake}}$ are the forces from the engine and the brake models that are the outputs from the Simulink models in figure 4.2. The road load, $F_{\text{road load}}$, is modelled by equation (4.7). The term $F_{\text{drag}}$ represents both the aerodynamic drag and the rolling resistance is modelled by the look-up table visualized in Figure 4.3.

$$F_{\text{road load}} = F_{\text{slope}} + F_{\text{drag}} \tag{4.7}$$

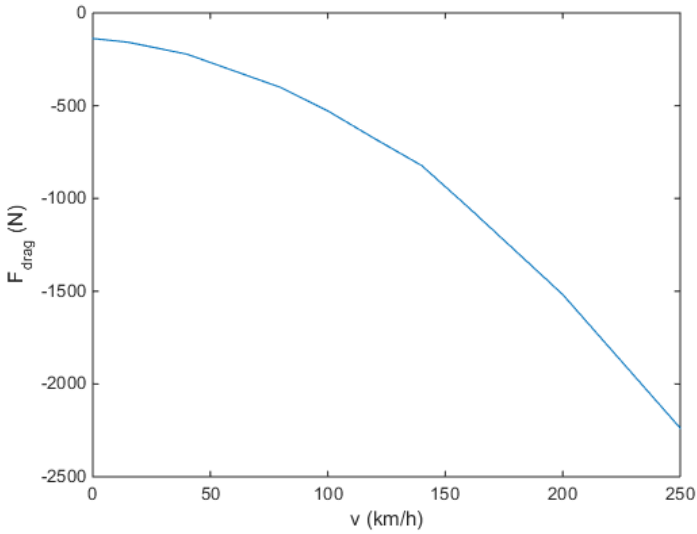


***Figure 4.3:*** *A visualization of the look-up table used for modelling $F_{drag}$ in the advanced Simulink model.*

### 4.1.4 Test Vehicle

The test vehicle is a Volvo V40 Cross Country. It is equipped with a dSpace Autobox on which it is possible to load and execute a program generated from a Simulink model. The Autobox hardware has the ability to read and write to the CAN-bus and by that it has the ability to control the vehicle.

***Figure 4.4:*** *An image of a car of the same model as the vehicle used for testing the controller.*

### Vehicle Data

The controller needs some parameters from the vehicle for its internal model. These are presented in Table 4.1. The parameters mass and wheel radius are not expected to be correct. The mass will be slightly different every time due to different fuel levels and load in the car while the wheel radius will change with different tires and tire pressures. The parameters that are used in the end are reasonable estimates that are close enough for the controller to function as intended. The time constant parameters from the different systems needs to be extracted from experiments. This process is presented later in this section.

| Parameter | Value |
|---|---|
| vehicle mass ($m$) | 2200 kg |
| wheel radius ($r_{\mathrm{w}}$) | 0.37 m |
| inertia in the engine ($J_{\mathrm{eng}}$) | $0.25\ \mathrm{kg}\cdot\mathrm{m}^2$ |
| time constant, engine ($T_e$) | 0.1 s |
| time constant, brake up ($T_{\mathrm{b,up}}$) | 0.1 s |
| time constant, brake down ($T_{\mathrm{b,down}}$) | 0.05 s |
| time delay, brake ($L_b$) | 0.05 s |

***Table 4.1:*** *A summary of vehicle specific parameters that are needed in the controller.*

### Simulink Interface

The Simulink model that handles the communications with the CAN bus is showed in Figure 4.5. To the left in the model is a receiver from which necessary signals are read. To the right is the transmitter block where the writing to CAN is performed. This block has an enable flag that needs to be set to be able to write to the CAN-bus. In between these two blocks, the controller is placed. From this model code is generate and loaded into the vehicle's Autobox.

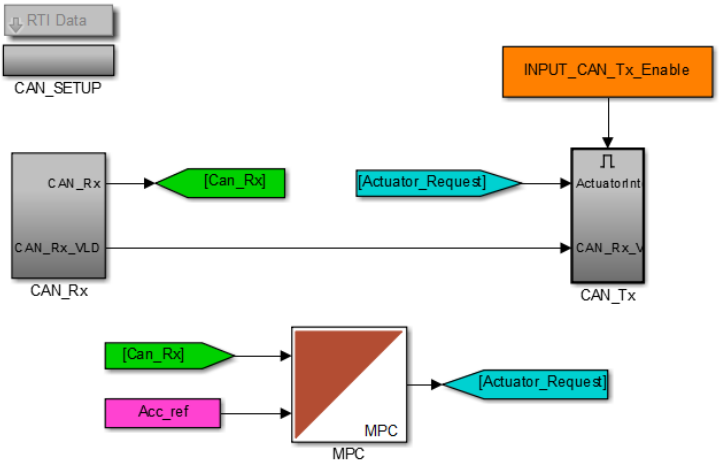


***Figure 4.5:*** *The interface for reading and writing to the CAN-bus.*

The interfaces to the CAN bus that is used to control the vehicle is for the ECM a signal representing minimum torque request. For the BCM however, the interface is a requested brake pressure for each individual wheel. The back pair of wheels will only be able to deliver half the braking torque of the front wheel pairs ability. The torque request that is the controllers interface therefor must be transformed to a request in pressure and divided over the four wheels whit this in mind. The transformation is described in equation (4.8) where $p$ is the pressure needed in each wheel for executing the requested brake torque $\tau_b$, $A_b$ is the area of contact between the tire and the braking pads on the wheels and $r_\mathrm{b}$ is the distance from the wheel center to the point where the brake pads are applies. For this specific vehicle the area of the braking pads on the rear wheels are twice the size than the corresponding area on the front wheels.

$$p = \frac{1}{4}\frac{\tau_b}{A_b r_\mathrm{b}} \tag{4.8}$$

There are a number of signals that are read from the can bus to be fed into the controller. The velocity is read from a CAN and is a fairly good signal that needs

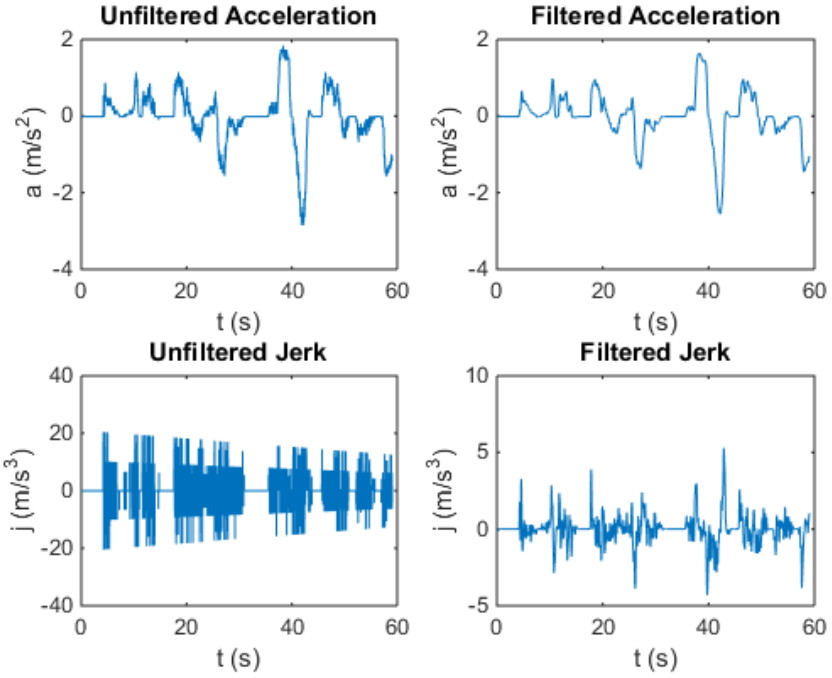


***Figure 4.6:*** *Measurements of the acceleration and jerk before and after filtering for a manual driving sequence. The filter $F_{acc}$ has the bandwidth $\omega = 7.18 rad/s$ and the filter $F_{jerk}$ has the bandwidth $\omega = 8.36 rad/s$*

no post-processing. The acceleration measurement on the other hand is noisy and has to be filtered before used as a state. A second order discrete chebychev filter, described with the transfer function (4.9), is used to obtain this. The jerk is estimated by taking the derivative of the filtered acceleration measurement. This signal is also in need of filtering. The same type of filter, with transfer function (4.10), is used once again for this. An example of the filter effects are shown in Figure 4.6.

$$F_{\text{acc}}(z) = \frac{0.003z^2 + 0.0059z + 0.003}{z^2 - 1.623z + 0.7024} \tag{4.9}$$

$$F_{\text{jerk}}(s) = \frac{0.0188z^2 + 0.0375z + 0.0188}{z^2 - 1.6230z + 0.7024} \tag{4.10}$$

### Step Response Experiments

To identify the time constants of the first order systems that approximates the ICE, $T_e$, and the friction brake, $T_b$, along with the time delays for each system $L_e$ and $L_b$ , step response experiments are performed on the test vehicle.

For the ICE the vehicle is allowed to run on the minimum torque keeping the speed constant. A fixed torque-request is then applied to the vehicle. A first order system is then approximated to fit the curve as well as it could. This is shown in Figure 4.7. The parameters are identified as $T_e = 0.1$s and $L_e = 0.1$s.

The same procedure is applied on the friction brake. The difference is that the vehicle doesn't have to be in motion, a brake pressure is applied at standstill. The dynamics are different when the pressure is building and releasing. That is why step responses for both cases are made and analyzed. This is shown in Figures 4.8 and 4.9. The parameters are identified as $T_b = 0.1$s and $L_b = 0.05$s when building pressure and $T_b = 0.05$s and $L_b = 0.05$s when releasing pressure.

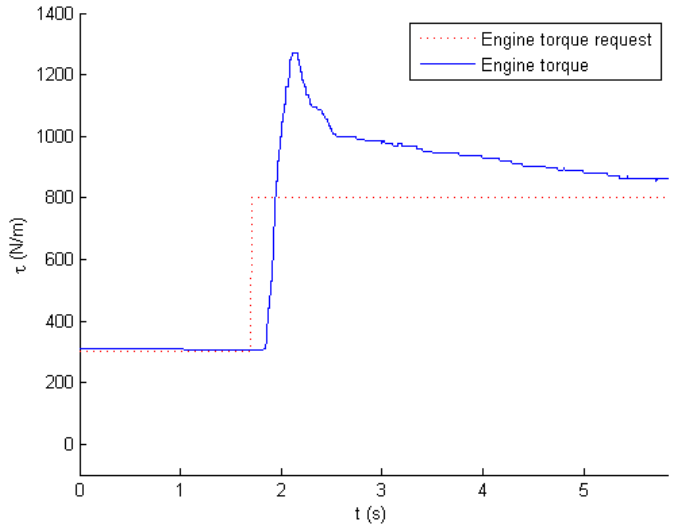


***Figure 4.7:*** *Step in requested engine force.*

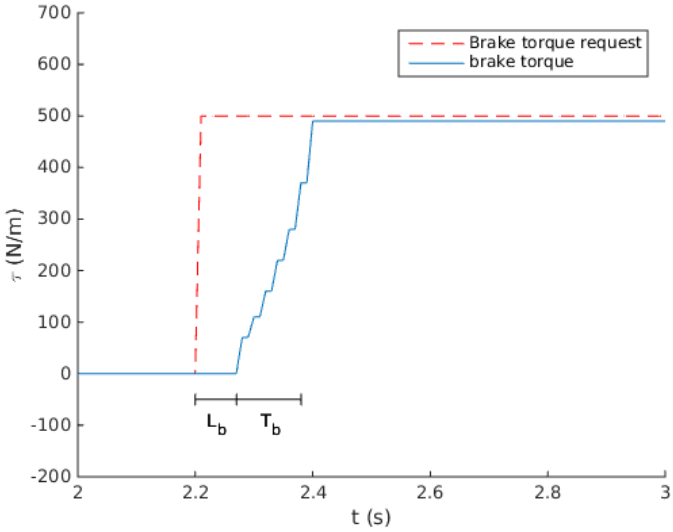


***Figure 4.8:*** *Step in the positive direction requested brake force.*

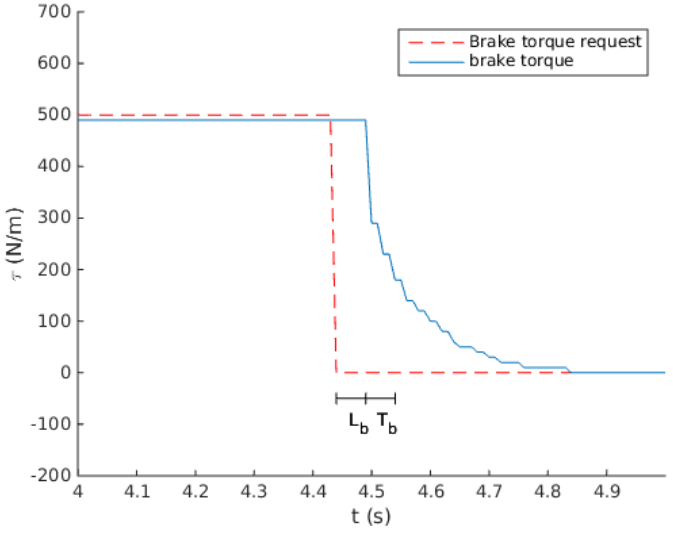


***Figure 4.9:*** *Step in the negative direction requested brake force.*

## 4.2 System Modelling

This section describes in detail how every part of the system is modelled for the internal controller model. The models are very simple in comparison with the plant model for all stages.

### 4.2.1 Car

The car is represented by a point mass where the longitudinal movement is modelled by Newton's second law.

$$m\dot{v} = F_{\text{eng}} + F_{\text{brake}} + F_{\text{road load}} \tag{4.11}$$

All the force components that is non-linear in $v$ by the laws of physics will be linearized around $v = v_0$ where $v_0$ i the speed of the vehicle when the linearization is made. The same linearization will be applied during the prediction horizon and then update in the next time sample. If equation (4.11) is known to be linear in $v$ it can be expressed in state space form.

### 4.2.2 Powertrain

The controller internal model of the powertrain is represented by a first order system from input $u_e$ to the output force $F_{engine}$. On top of that, the inertia of the powertrain, $F_{\text{pt}}$ that is defined in equation (4.5), is modeled.

$$F_{\text{engine}} = \frac{1}{sT_e + 1} u_e + F_{\text{pt,in}} \tag{4.12}$$

### 4.2.3 Friction Brake

The friction brake has the ability to convert kinetic energy to heat energy by friction. When braking the brake system is building up a pressure and then applies the braking pads to create a friction force and slow the car down. When the system requests less braking force on the other hand the pressure must be relieved. That is a significantly faster process than building the pressure. This is why it is modelled as two separate processes as in (4.13). The model will be kept the same under one prediction horizon and can only change in the beginning of each time step.

$$\begin{cases} \frac{e^{-sL_b}}{sT_{b,up}+1} u_b & \text{if } a_{\text{ref}} < a \\ \frac{e^{-sL_b}}{sT_{b,down}+1} u_b & \text{if } a_{\text{ref}} \geq a \end{cases} \tag{4.13}$$

The logic that determines which system to use is an estimate on which side of the reference value the actual acceleration is. If the actual acceleration is higher

than the reference value it is highly probable that the brake, if used, is going to generate a larger negative torque. For the opposite case it is highly probable that the brakes are going to be released if used.

### 4.2.4 Road load

The road load is modelled as the linearization of the expression derived in equation (4.4). The expression is linearized around $v = v_0$ using the first order Taylor expansion.

$$\begin{aligned}\bar{F}_{roadload}\Big|_{v=v_0} &= F_{\text{road load}}(v_0) + \frac{dF_{\text{road load}}}{dv}\Big|_{v=v_0}(v - v_0) = \\ &= -\frac{1}{2}C_d A\rho v_0^2 - mg\sin(\alpha) - C_{\text{rr}}\mu mg - C_d A\rho v_0 (v - v_0) = \\ &= C_1 v + C_2\end{aligned} \tag{4.14}$$

Where

$$\begin{aligned}C_1 &= -C_d A\rho v_0 \\ C_2 &= \frac{1}{2}C_d A\rho v_0^2 - mg\sin(\alpha) - C_{\text{rr}}\mu mg\end{aligned} \tag{4.15}$$

The expression in equation (4.14) is now linear and can be used when expressing the full system in a state space form.

### 4.2.5 State space model using two actuators

All the parts from the internal model is combined to a state space form. This is done by reshaping equation (4.11) when all the submodels are inserted.

$$m\dot{v} = \frac{1}{sT_e + 1}u_e - \frac{J_{\text{eng}}k_{\text{trans}}^2}{r_{\text{w}}}\dot{v} + \frac{e^{sL_b}}{sT_b + 1}u_b + C_1 v + C_2 \tag{4.16}$$

$$\begin{aligned}&\left(m + \frac{J_{\text{eng}}k_{\text{trans}}^2}{r_{\text{w}}}\right)\dot{v}(sT_e + 1)(sT_b + 1) = \\ &= (sT_b + 1)u_e + (sT_e + 1)e^{-sL_b}u_b + (sT_e + 1)(sT_b + 1)(C_1 v + C_2)\end{aligned} \tag{4.17}$$

To simplify further calculations the constant $C_m$ is defined as in equation (4.18).

$$C_m = m + \frac{J_{eng}k_{trans}^2}{r_{\text{w}}} \tag{4.18}$$

Since $s$ is the Laplace transform of a time derivative operator equation (4.17) can be rewritten.

$$\begin{aligned} &C_m v^{(3)} T_b T_e + C_m \ddot{v}(T_b + T_e) + C_m \dot{v} = \\ &= T_b \dot{u}_e + T_e e^{-sL_b} \dot{u}_b + u_e + e^{-sL_b} u_b + T_b T_e C_1 \ddot{v} + (T_b + T_e) C_1 \dot{v} + C_1 v + C_2 \end{aligned} \tag{4.19}$$

$$\begin{aligned} v^{(3)} = &\frac{T_b}{C_m T_b T_e} \dot{u}_e + \frac{T_e}{C_m T_b T_e} e^{-sL_b} \dot{u}_b + \frac{1}{C_m T_b T_e} u_e + \frac{1}{C_m T_b T_e} e^{-sL_b} u_b \\ &- \frac{m(T_b + T_e) - T_b T_e C_1}{C_m T_b T_e} \ddot{v} + \frac{(T_b + T_e) C_1 - C_m}{C_m T_b T_e} \dot{v} + \frac{C_1}{C_m T_b T_e} v + \frac{C_2}{C_m T_b T_e} \end{aligned} \tag{4.20}$$

The state vector $x$ and the input signal vector $u$ is then defined as in equation (4.21). An advantage with the choice of this states is that all of them have a straight forward physical interpretation. The first state is the jerk, the second is the acceleration and the third is the velocity.

$$x = \begin{bmatrix} \ddot{v} \\ \dot{v} \\ v \end{bmatrix} \quad u = \begin{bmatrix} u_e \\ u_b \end{bmatrix} \tag{4.21}$$

From equation (4.20) a state space model can be expressed.

$$\dot{x} = Ax + Bu + D\dot{u} + c \tag{4.22}$$

where

$$A = \begin{bmatrix} -\frac{C_m(T_b+T_e)-T_b T_e C_1}{C_m T_b T_e} & \frac{(T_b+T_e)C_1 - C_m}{C_m T_b T_e} & \frac{C_1}{C_m T_b T_e} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.23}$$

$$B = \frac{1}{C_m T_b T_e} \cdot \begin{bmatrix} 1 & e^{-sL_b} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.24}$$

$$D = \frac{1}{C_m T_b T_e} \cdot \begin{bmatrix} T_b & T_e e^{-sL_b} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.25}$$

$$c = \begin{bmatrix} \frac{C_2}{C_m T_b T_e} \\ 0 \\ 0 \end{bmatrix} \tag{4.26}$$

To express this on a standard state space form, without derivatives on the input signals, a variable substitution is made. A fictive state vector $x'$ is defined as

$$x' = x - Du \tag{4.27}$$

The time derivative of the new state is then

$$\dot{x}' = \dot{x} - D\dot{u} = Ax + Bu + c = A(x' + Du) + Bu + c = Ax' + (B + AD)u + c \tag{4.28}$$

This is now expressed on a state space form

$$\dot{x}' = A'x' + B'u + c \tag{4.29}$$

where $A' = A$ and $B' = AD + B$.

### 4.2.6 Discretization of state space model

The state space model is then discretized using the Euler method presented in section 3.2. This method is chosen before Tustin's method from the same section because of its simplicity in implementation. Tustin's method may be more accurate but the gain is too small to motivate the use of it.

The dynamic system described by equation (4.29) can be rewritten as in equation (4.30) using the Euler forward approximation.

$$\begin{aligned} \dot{x}' \approx \frac{x'_{k+1} - x'_k}{T_s} = A'x'_k + B'u_k + c \Leftrightarrow \\ \Leftrightarrow x'_{k+1} = (T_s A' + I)x'_k + T_s B'u_k + T_s c \end{aligned} \tag{4.30}$$

This can be summarized as in equation (4.31).

$$x'_{k+1} = Fx'_k + Gu_k + H \tag{4.31}$$

where $F = T_s A + I$, $G = T_s\,(AD + B)$ and $H = T_s c$.

To check if the system is stable its poles are determined. It is possible to do the calculations either on the continuous time system or the discrete system since the

region of stability in the continuous time system projects inside the discrete time systems stability region.

The system (4.31) is stable since its poles are contained inside the unit circle. For linearization around different initial speeds, $v_0$, the systems stays stable as long as $v_0 \geq 0$. This is obviously the case since the model only is constructed for forward driving. For increasing $v_0$ the pole closest to the stability border moves slightly closer to the origin. The system is by that slightly more stable as the speed increases. The poles of the system is shown in Figure 4.10 for $v_0 = 30$km/h.

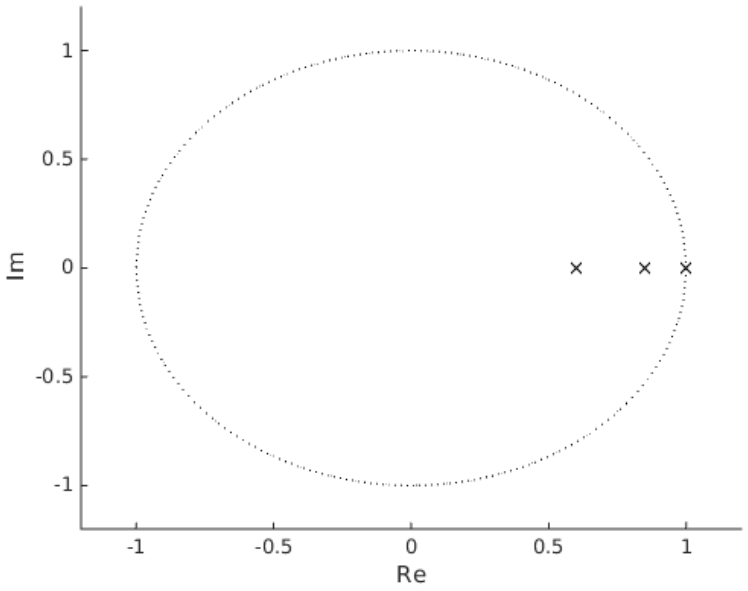


***Figure 4.10:*** *The poles of the discrete system for the case where the vehicle travels in $v_0 = 30km/h$*

## 4.3 Optimization problem

When solving an optimization problem the solution will be optimal for that specific problem formulation if the solver is allowed to run a large number of iterations and if the criteria for stopping is set to be very accurate. Since the computational power is a limited resource it is crucial for the implementation to work that the optimization problem is formulated in a smart way.

The problem needs to be convex for reasons stated in section 3.1.1 and [16] but for the solver to converge to a desired solution the problem cant be too flat either. If the problem is too flat in one ore more dimensions it will lead to a solution that priorities one or more dimension too much making the optimal solution flawed.

The optimization problem can be formulated as equation (4.32).

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \sum_{k=0}^{N} f(x, u, k) \\ \text{subject to} \quad & Au \leq b \end{aligned} \tag{4.32}$$

The design of the objective function, $f$, and the design of the constraints, $A$ and $b$ is what is going to be the design of the controller.

### 4.3.1 Objective Function

In section 3.3 the tools for designing an objective function is presented. The decision to make here is what behaviors to be put a cost on in relation to all the behaviors that are considered to be of interest. These behaviors are listed below.

- How well the reference is followed.
- How much energy is consumed by the actuators.
- How quickly does the actuators signal energy change.
- How well the comfort criterion are fulfilled

A cost will be assigned to each of these properties as a quadratic cost term in the objective function since all of these are only dependent on the magnitude of the quantity and not on the sign.

#### Following the reference

The cost for deviation for the reference is expressed as equation (4.33). Here it is possible to assign a cost to deviation from the reference signal. Inspired by one of the methods to improve the stability, the terminal cost function described in section 3.3.4 this cost is inconstant during the time horizon. Unlike that method with an extra cost on the last sample in the prediction horizon, the cost for the deviation is increased by a linear factor for every sample starting at a specified sample.
This results in a smoother behavior compared to the terminal cost function. The idea is to make sure that the solver gets a larger freedom early in the prediction horizon but, later it gets more important to follow the reference. The parameters that can be changed for tuning the controller is $Q_{\text{ref}}$, $Q_{\text{cone}}$ and $k_{\text{cone}}$. $Q_{\text{ref}}$ is start cost for deviations from the reference. $Q_{\text{cone}}$ is the linear increasing factor which the cost for deviations from the reference increases every sample after $k_{\text{cone}}$.
Compared with other methods like *terminal equality constraint* and *contraction constraint*, both presented in section 3.3.4, this method never cause infeasibility and are very straightforward to implement and tune.

$$
\begin{aligned}
f_r(x,u,k) &= (a_k - a_{\text{ref}})^T Q_{\text{ref,total}} (a_k - a_{\text{ref}}) \\
Q_{\text{ref,total}}(k) &= \begin{cases} Q_{\text{ref}} \left(1 + (k - k_{\text{cone}}) Q_{\text{cone}}\right) & \text{when } k > k_{\text{cone}} \\ Q_{\text{ref}} & \text{when } k \leq k_{\text{cone}} \end{cases}
\end{aligned} \tag{4.33}
$$

#### Integral action

For this system a nonzero control signal is required in most cases to keep the desired acceleration. Therefor adding a cost for the signal energy to the objective function, the result is a trade-off between minimizing the signal energy and following the reference. By substitute the cost for the energy to a cost for change in the control signal, which is described in (4.34), one can avoid this behavior and integral action will be achieved. The method is more detailed motivated in section 3.3.3.

Another benefit with this term in the objective function is that the MPC gets a smoother behavior and it prevents the signals from become too noisy.

$$
f_{\Delta U} = \Delta U^T Q_{\Delta U} \Delta U \tag{4.34}
$$

#### Energy consumption

As already mentioned, energy efficiency have become more crucial for today's vehicles. For an overactuated vehicle as the one considered in this thesis where redundancy exist it is therefor of great importance to find a solution to the optimization problem with a desired behavior. In this case the desired behavior is one that is not consuming more energy than needed to achieve the aim.

In section 3.3.3 a method for control allocation is handled. The idea is to first calculate a desired virtual control signal, $u_{ref}$, and add the deviation from $u_{ref}$ to the objective function as in (4.35)

$$
f_{u_{ref}} = (u - u_{ref})^T Q_{u_{ref}} (u - u_{ref}) \tag{4.35}
$$

To calculate the desired control signal for a stationary reference to follow is fairly straightforward. To minimize the energy consumption the brake should only be used when needed, i.e. the brake should only be used if the wanted deceleration is greater than what the minimum torque and the road load can cause.

$$
u_{\text{ref}} = \begin{cases} u_e = \mathrm{F} \cdot r_{\mathrm{w}}\,, \quad u_b = u_{b,max} & \text{when} \quad \mathrm{F} \cdot r_{\mathrm{w}} \geq u_{e,min} \\ u_e = u_{e,min}\,, \quad u_b = \mathrm{F} \cdot r_{\mathrm{w}} - u_{e,min} & \text{when} \quad \mathrm{F} \cdot r_{\mathrm{w}} < u_{e,min} \end{cases} \tag{4.36}
$$

F in (4.36) is the force needed to achieve the required acceleration and is calculated as in (4.37) where $F_{\text{road load}}$ is the rolling resistance and aerodynamic drag

and $F_{\text{pt,in}}$ describing the inertia in the powertrain.

$$\mathrm{F} = m \cdot a_{\text{ref}} - F_{\text{road load}} - F_{\text{pt,in}} \tag{4.37}$$

### Constraints

A big benefit with the MPC framework is the possibilities to include constraints on the control signals and the states in the optimization problem.

The constraints on the control signals may seems natural. The brake can never generate a propulsive force while the largest force it can generate is estimated as the friction coefficient times the normal force. This is summarized in equation (4.38).

$$-\mu m g \leq u_b \leq 0 \tag{4.38}$$

For the powertrain the minimum and maximum torque that can be generated is calculated from a map depending on the engine speed. The map is visualized in figure 4.11. The map is far from perfect, but will be sufficient for this application in most cases.

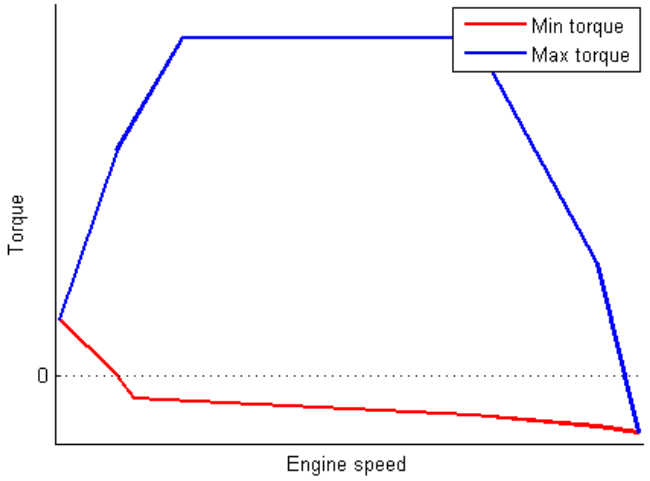


***Figure 4.11:*** *The map that is used for setting the constraints on the engine control signals.*

One factor that contribute to discomfort for the driver is the derivative of the acceleration, jerk. By that reason there is a limit for the maximum level of jerk that are allowed. Unlike the constraints on the control signals a constraint on the jerk

may not always be possible to fulfil. If the states initial violates the constraints it may not be possible to find a feasible solution given the dynamics of the system. The solution to this is to use soft constraints. As described in section 3.3.3 the soft constraints are used to ensure that a feasible solution exist. By adding a slack variable as in (4.40) and add a cost for the size of $\epsilon_j$.

To optimize the problem formulation and make sure that the slack variable only are used when the constraints already are violated, two different solvers are used as shown in (4.39). Else the solver might find a solution to the optimization problem that violate the constraints even when it's not needed.

$$f_j = \begin{cases} f_{\text{j,hard constraints}} & \text{when} \quad |j| \leq j_{\text{lim}} \\ f_{\text{j,soft constraints}} & \text{when} \quad |j| > j_{\text{lim}} \end{cases} \tag{4.39}$$

$$\begin{aligned} -\left(j_{lim} + \epsilon_j\right) \;\leq\; j \;\leq\; j_{lim} + \epsilon_j \\ \epsilon_J \geq 0 \end{aligned} \tag{4.40}$$

## 4.4 Implementation

The implementation differs between the simulation environment and the real vehicle. Both uses the tool CVXgen. CVXgen is a web tool that lets the user enter the optimization problem in a custom language. The syntax of language is very strict to ensure that the problem stated is a convex problem. The outcome of CVXgen is a generated solver optimized for the specific optimization problem that has been stated. The generated solver is available in both C- and Matlab-code.

The implementation in the simulation environment uses the generated Matlab-code. The code that is generated for Matlab is the same problem defined such that the CVX Matlab toolbox can interpret it. The actual optimization problem solver is then chosen in the CVX toolbox.

In the test vehicle the controller is implemented in the Simulink block s-function builder. This block generates a C MEX s-function from the C-code that is provided to it. Hence the controller including the solver for the optimization is implemented in C-code.

The test vehicle implementation is a simplification from the simulation environment. The implantation only uses soft constraints on the jerk limit and does not switch between hard and soft constraints like the simulation implementation does. Another feature that is not implemented in the test vehicle is increasing cost on deviating from the reference. Finally, the prediction horizon is significantly smaller in the test vehicle implementation due to limited computational power.

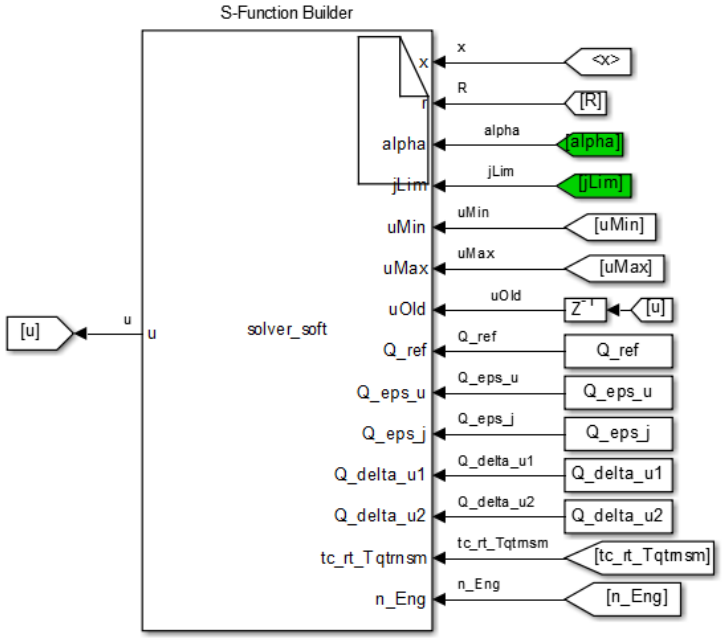


***Figure 4.12:*** *The s-function builder block that implements the controller.*

In Figure 4.12 the implementation of the controller in its Simulink environment is shown. The C-code that is inserted in the s-function builder block is generated from the tool CVXgen. The CVXgen code is presented in Appendix B.

# 5
# Analysis

In this chapter the results from simulations and tests are presented and analyzed. The different use cases, stated in chapter 2, are used when comparing the performance of the controller in simulation and implemented in the vehicle with the performance of the controller available in today's vehicles.

## 5.1 Simulation Results

The use case scenarios from 2.3 are simulated using the developed controller and presented in plots 5.1, 5.3 and 5.5. These result are compared with the performance of the controller existing in today's vehicle that is based on a PID-controller. The result of these are presented along with the corresponding MPC result in figures 5.2, 5.4 and 5.6.

The tests are performed with the same step sizes, jerk limits, initial states and a fixed gear to get a fair comparison. The steps sizes are $0.5\text{m/s}^2$, the jerk limit is set to $j_{\text{lim}} = 1\text{m/s}^3$ and the initial state is set to $x = \begin{pmatrix} \frac{30}{3.6} & 0 & 0 \end{pmatrix}^T$. These are chosen such as the expected behavior of the controller is that both actuators will need to be used.

In figures 5.1 to 5.6 the left column shows the velocity, acceleration compared to reference and jerk with limits while the right plot shows control signals and actual output from the actuators. The blue represent the control signal to the ICE and the red is the control signal to the friction brake. The dotted lines in each color shows the actual output from each actuator.

### Use Case I

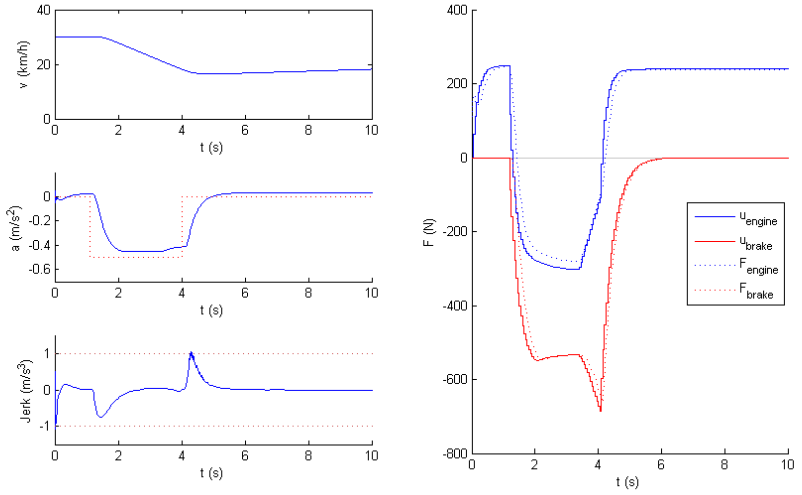


***Figure 5.1:*** *The simulation result using the MPC on scenario I.*

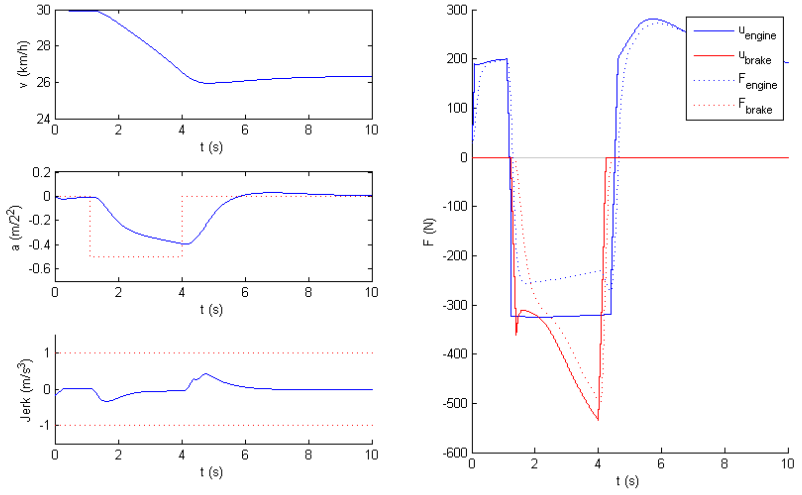


***Figure 5.2:*** *The simulation result using the PID on scenario I.*

When comparing the MPC with the PID in scenario I there are obvious differences in the control signal. In figure 5.2 it can be seen that the control signals is controlled by a rule-set, specially when observing the control signal to the ICE. When the change in reference comes it directly sends the minimum force as control signal. In the MPC in Figure 5.1 on the other hand it has a different shape.

When analyzing the result on the states it can be seen that the MPC takes more advantage on the jerk limits then the PID. Followed by that is that the acceleration decreases faster in the MPC resulting in a lower final velocity.

To summarize the scenario it can be said that the MPC is better on using the jerk limit to maximize the performance. There is a problem with a static error that can be minimized by a better tuning of the controller. The acceleration is still closer to the reference for the MPC than the PID during the step, which is the important part of this scenario.

### Use Case II

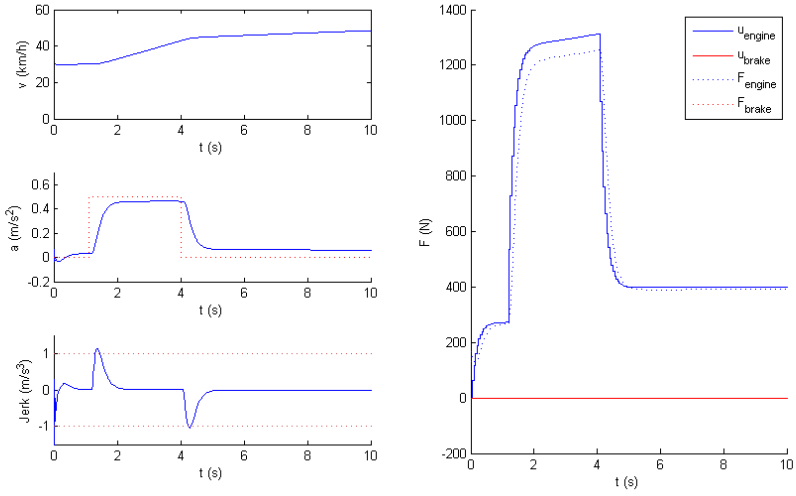


***Figure 5.3:*** *The simulation result using the MPC on scenario II.*

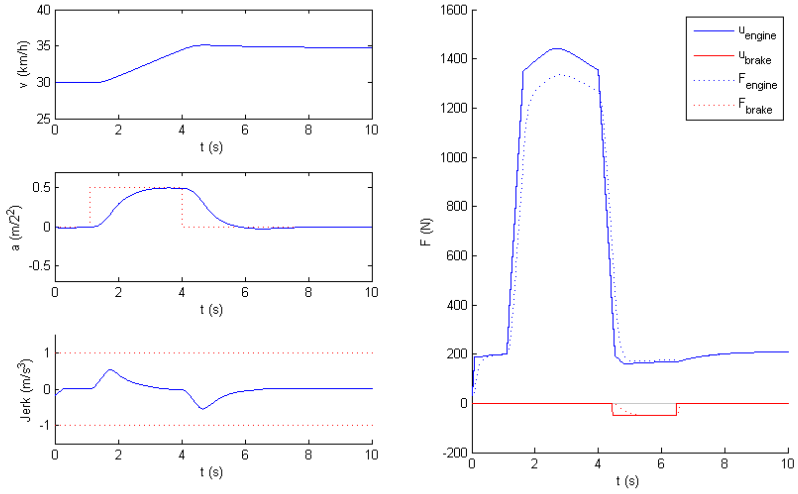


***Figure 5.4:*** *The simulation result using the PID on scenario II.*

The result from the simulation of the second scenario is very similar. Yet again the MPC is better on using the jerk limit to its maximum potential, maybe a little too much since it violates the constraints by a little. But yet again, the controller does not have the optimal tuning.

The MPC deviates from the reference when it is zero after the step. The reason for this is probably that the internal model in the controller is less accurate for higher speeds. More comments on this problem can be read in section 6.2.

### Use Case III

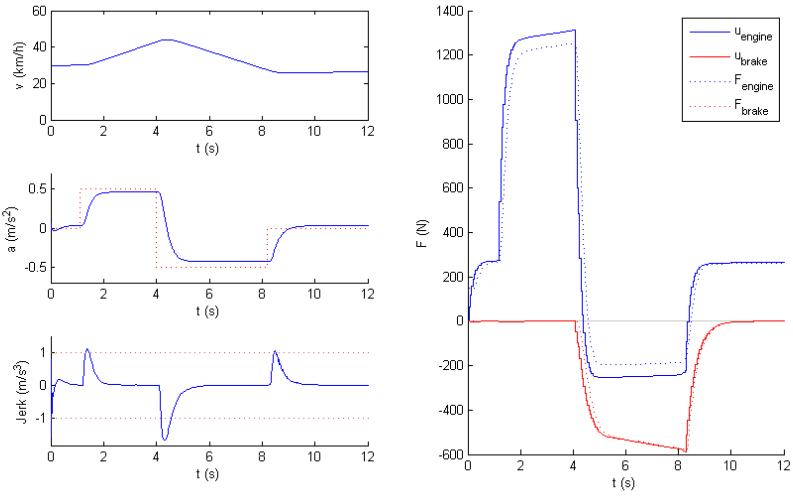


***Figure 5.5:*** *The simulation result using the MPC on scenario III.*

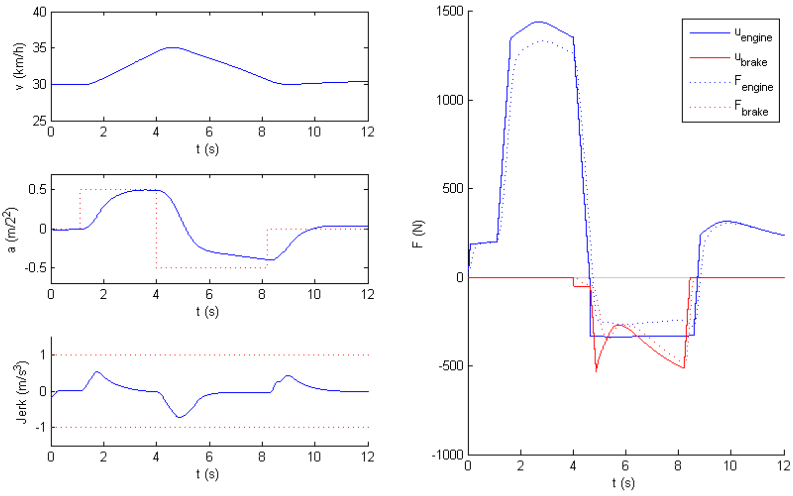


***Figure 5.6:*** *The simulation result using the PID on scenario III.*

The results from the last scenario shows a problem in the MPC in Figure 5.5. The controller violates the lower jerk limit when the engine torque goes to its minimum as the brake is activated. Now it may seem like an unfair comparison

since the MPC is allowed to violate the constraints. Even if the MPC is better at following the reference in this scenario the PID, in Figure 5.6, must be said to be the better controller since it solves the problem without violating the constraints.

## 5.2 Robustness Analysis

To test the robustness of the controller some parameters that are important to the model was changed in the plant model and simulated with both the MPC and the PID. The resulting accelerations was compared to each other and the results from when the correct parameters are used to analyze how dependent the controller is of a correct model.

The parameters that where changed is presented in Table 5.1. These were chosen as examples, there are other parameters in the controller that very well may be important to the robustness of the controller even if they are not presented as examples here.

| Parameter | Original Value | Change |
|---|---|---|
| Mass | 2200 kg | ±1000 kg |
| Drag | - | ±40% |
| Inertia in the engine | 0.25 | $\pm 0.15 \text{kg} \cdot \text{m}^2$ |

***Table 5.1:*** *The parameters that are changed when testing the robustness of the controller. The mass and the inertia of the engine is changed in absolute values while the drag is changed by a percentage of its original value.*

In Figure 5.7 it can be seen that for an error in the modelled mass the behavior for constant acceleration differs and will cause a static error in the controllers output. This is however the case for large differences in mass. If a decent mass estimation exists, the small variations in mass will not ruin the results. The controlled can therefor be said to be robust in the parameter that represents the mass. In Figure 5.8 the corresponding test for the PID-controller is shown. It can be seen that it s not sensitive to changes in mass.

In Figure 5.9 it can be seen that the controller is sensitive for differences in the modelled drag force, which is the sum of the aerodynamic drag and the rolling resistance. This means that the controller would need either a model that corresponds good with reality or introducing an integral action to the controller that can handle this better than today's implementation. For the best result, having both alternatives would be preferred. In Figure 5.10 the corresponding test for the PID-controller is shown. It can be seen that initially it shows sensitivity to changes in the road load, but after a few seconds it does not matter if the road load is modelled bad.

The last robustness test is shown in Figure 5.11. Here it is tested how model errors in the inertia of the engine affects the controllers performance. This test

is made with a constant acceleration reference that is non-zero since the inertia of the engine only has any effect when not accelerating. The test shows that the controller is not very sensitive to model errors in this parameter. The same result can be observed for the PID-controller in Figure 5.12.

To summarize the robustness tests, it can be said that the PID-controller is a lot more robust than the MPC. This is not a surprise since the MPC is a model based controller method and PID is not. Maybe the tuning of the PID is optimized for some assumed model but other than that it should not be dependent of the model at all.

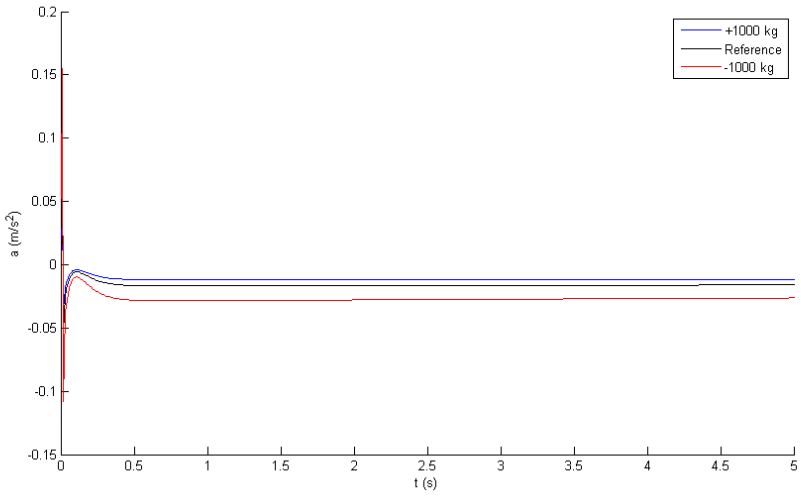


***Figure 5.7:*** *Comparison of simulation results when assuming the vehicle has more or less mass than it actually has using the MPC.*

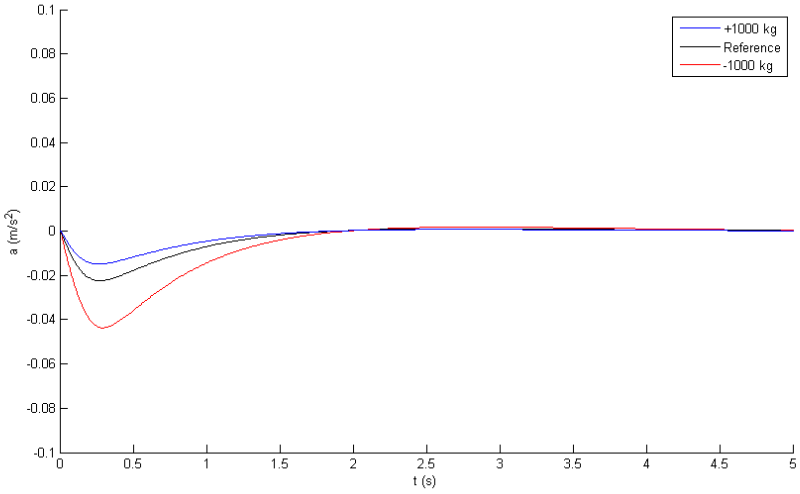


***Figure 5.8:*** *CComparison of simulation results when assuming the vehicle has more or less mass than it actually has using the PID.*

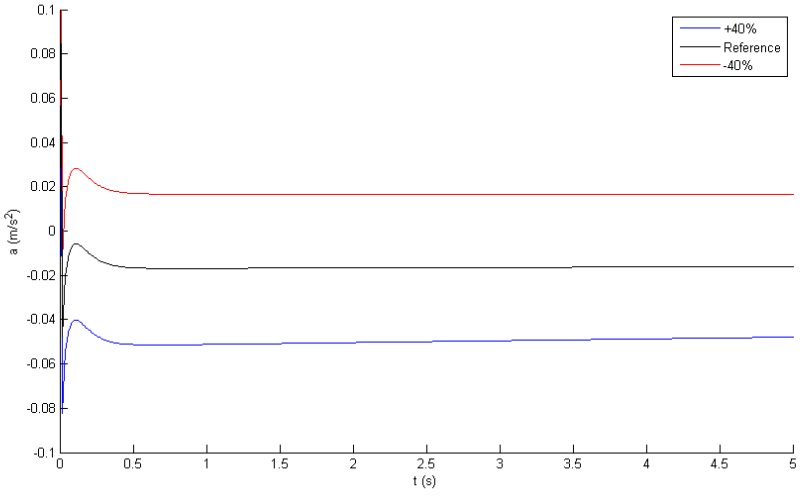


***Figure 5.9:*** *Comparison of simulation results when assuming the rolling resistance has smaller or bigger impact than it actually has using the MPC.*

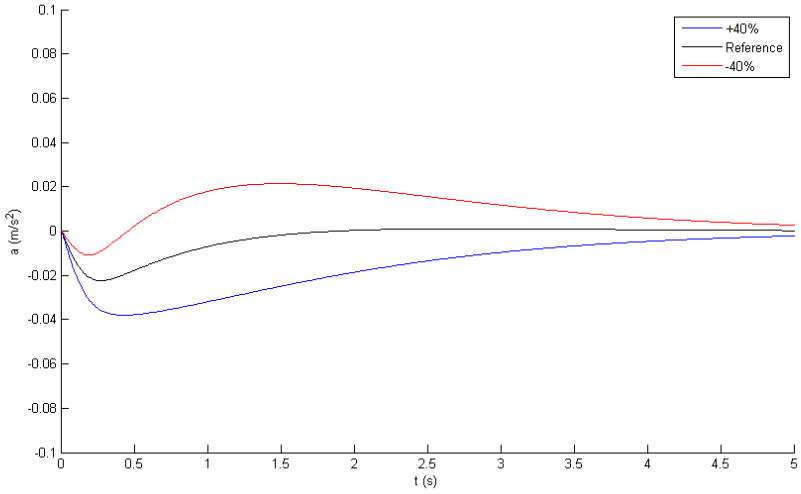


***Figure 5.10:*** *Comparison of simulation results when assuming the rolling resistance has smaller or bigger impact than it actually has using the PID.*

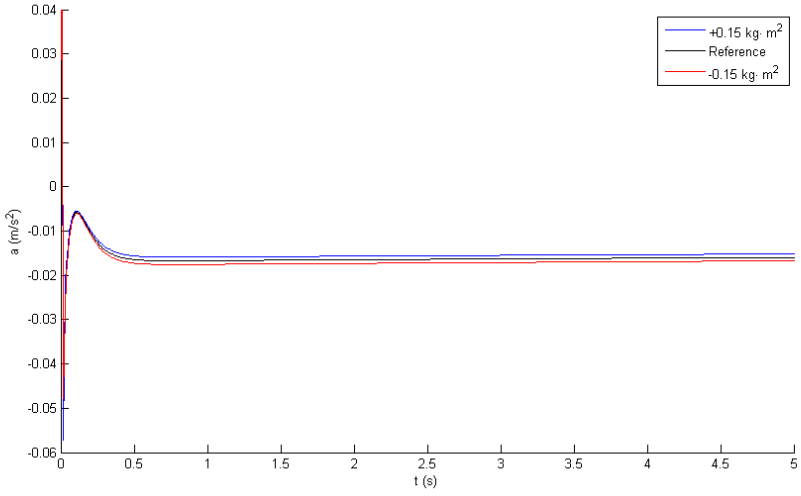


***Figure 5.11:*** *Comparison of simulation results when assuming the inertia in the engine is smaller and bigger than it actually is using the MPC.*

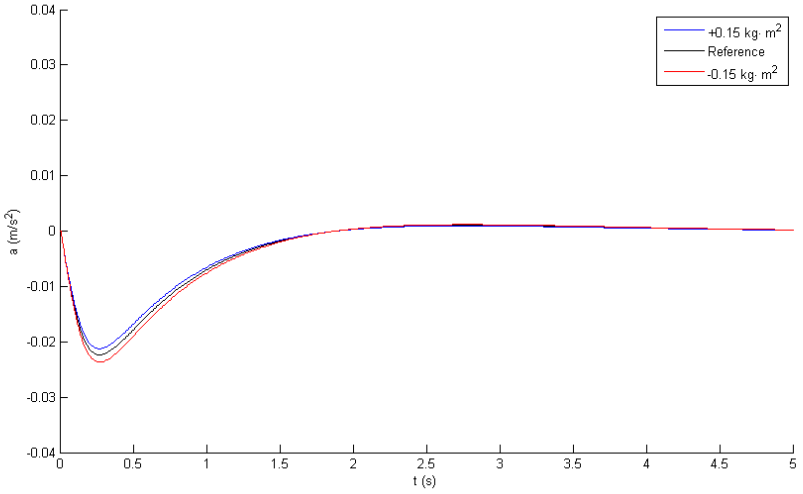


***Figure 5.12:*** *Comparison of simulation results when assuming the inertia in the engine is smaller and bigger than it actually is using the PID.*

## 5.3 Test Vehicle Results

The use case scenarios from 2.3 are tested in a test vehicle, described in section 4.1.4. In Figures 5.13, 5.14 and 5.15 the results from the tests are presented.

The tests are performed on a straight road with a road inclement that is close to zero. The vehicle's gear is fixed such that the result not is flawed by the process of the gear-shift. There was a speed limit on the road that needed to be obeyed and because of that the initial states differs slightly between the different tests. The tests are performed with step heights 0.5 and jerk limit, $j_{\text{lim}} = 1$, just like in the simulations.

What needs to be kept in mind when looking at the results of the tests are that the acceleration is a measurement from a sensor that is sensitive to noise. Even a small bump in the road can cause noteworthy noise on the acceleration measurements even when it is filtered. As a consequence of that the jerk, that is a derivative of the measured acceleration, becomes really noisy.

The results are not optimal in any of the tests due to several reasons. Many of the parameters involved in the controller are hard to estimate in reality. Another difference from the simulation environment is that the prediction horizon of the MPC is shorter than in simulations due to limited computational power. However, the results are good enough that there is no doubt that the concept works in reality as well.

### Use Case I

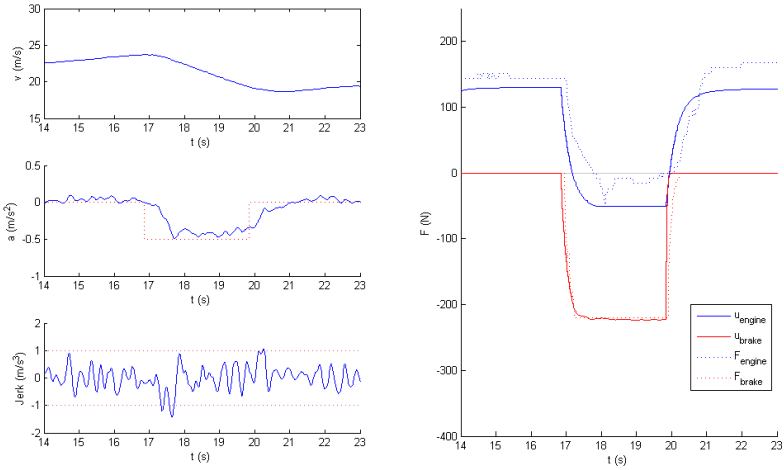


***Figure 5.13:*** *The result of use case I when running the MPC in a test vehicle.*

### Use Case II

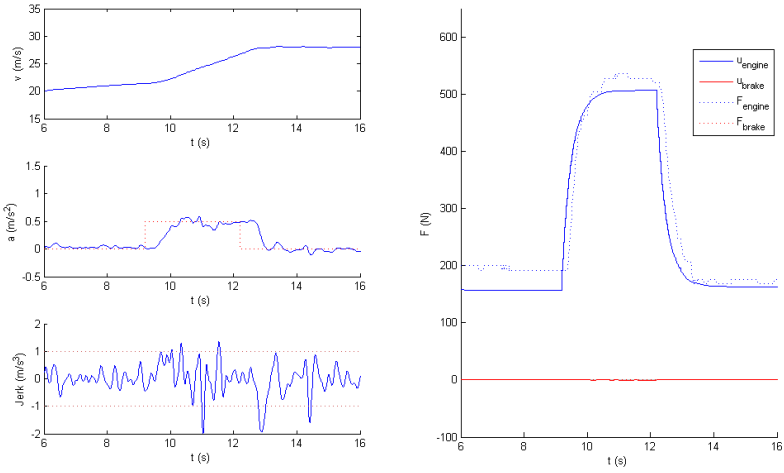


***Figure 5.14:*** *The result of use case II when running the MPC in a test vehicle.*

### Use Case III

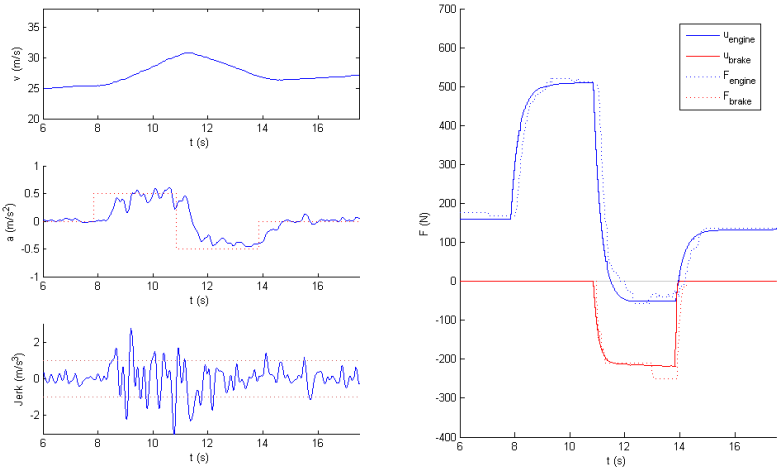


***Figure 5.15:*** *The result of use case III when running the MPC in a test vehicle.*

# 6

# Conclusions and future work

## 6.1 Conclusions

The MPC has been evaluated by comparison to the PID implementation similar to the one implemented in today's vehicles to answer the problem formulation. The main question in the problem formulation is weather it is possible or not to achieve similar or better performance using the MPC than the PID-controller. The answer to this question is yes. However, there are several points that needs to be considered if the question had been if its a motivated switch.

A downside with the MPC is that it is very dependent on a good model. If the internal model in the controller differs too much from reality the performance is quickly affected. For example the reason that the controller differs slightly from the reference when keeping zero acceleration in 5.3 is that the internal models for air resistance and rolling resistance in the internal model differs too much from the plants corresponding models, that in this case represents reality. In section 6.2 suggestions are made on how to solve this problem. Another problem with the MPC is that it sometimes have a hard time keeping the restrictions applied to it by the jerk limits. The reason for this is model errors. The controller predicts that the jerk limits will be met for the upcoming steps in the prediction horizon but when the signals are fed trough the plant model the result is that the constraints are violated. Yet again the importance of good modeling is shown. The advantage of the MPC is that it results in a faster response than the PID does in general. As soon as step in reference is known for the MPC it can prepare how to get there as fast as possible during its prediction horizon and will therefor be faster.

When summarizing the advantages and disadvantages of the MPC as it performs today, conclusions can be made that the MPC outperforms the PID on fast events

like the one in scenario I where a rather fast deceleration is needed still keeping the comfort criteria. In longer events, when a constant acceleration needs to be kept the PID is a better choice while the correctness of the model is this important.

Another factor that always needs to be considered when thinking of implementing an MPC in a real system is how much computational power that is available. When simulating, the prediction horizon of the MPC can be set high, but when implementing it in reality a large prediction horizon is very likely demand too much of the hardware.

## 6.2 Future work

The natural extension to the work presented in this thesis is to extend it to a HEV. It is fairly easy to see by the human eye approximately how to coordinate the actuators when only ICE and friction brake exists. When an electrical machine is inserted to the equation it is not always that obvious. The MPC approach to the problem would make use of the electrical machine in a good way and is therefor a suitable extension to this thesis. When considering a HEV aspects as the state of charge for the battery also have to be included in the equation and a well thought out control strategy such as an MPC might be required to reach the full potential of the HEV.

The robustness of the MPC has been a big drawback of this control strategy. As ca be seen in Section 5.2, the controller is very sensitive to model errors and also have both bias in some operating points and slow convergence. As the performance of the MPC has proved to be very reliant on the internal model of the system some kind of error estimation might be essential to make a robust MPC. Parameters such as the rolling resistance is in reality difficult to estimate but has proven to make a huge impact on the result. In for example [19] methods to include the uncertainty and additive disturbance in the state space model are described. To make the convergence faster a different method to ensure stability may be more successfully. In section 3.3.4 some methods are described and several more exist. With a better model of the system including the uncertainty some other methods might be more rewarding.

The main focus for the controller have been to fulfil the comfort criteria and quick response. Another interesting area to investigate is if it's possible to reduce the energy consumption with an MPC. With a control signal optimized for a finite time horizon it should be possible to make smarter decisions in transients. One way of achieving that can be trough better models of the actuators, especially their limits. Bad modelled limits of the actuators will either result in the controller not being able to use its full potential or that controller overestimates the actuator's capacities. Another way of achieving this would be to consider a HEV with a third actuator. With the possibility to use generative braking with the ERAD the decreased energy consumption with an MPC and a well-optimized control strategy might be even greater.

# Appendix

# A

# Derivation of model for inertia in the driveline

In this section the derivation of the inertia in the driveline is presented. The methods shown here are presented in detail in [20].

Consider a model of the driveline shown in Figure A.1 together with the definitions of torques and angles shown in Figure A.2. Assume that the only part of the system that contains inertia is the engine, $J_{\text{eng}}$. Also assume that the transmission is stiff. If this is true, equation (A.1) to (A.7) describes the powertrain.

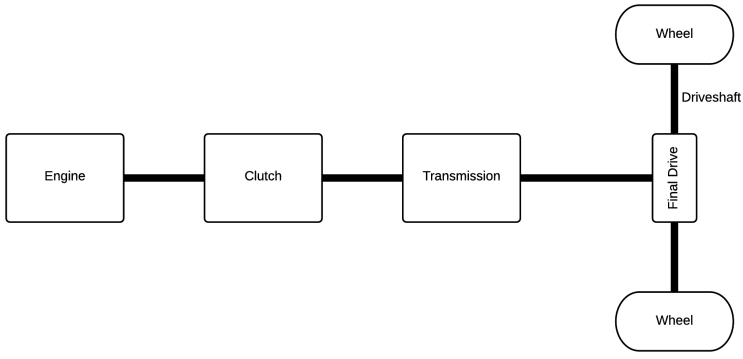


***Figure A.1:*** *A model of the driveline in a car.*

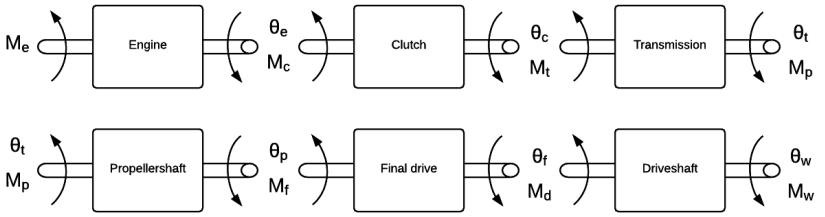


***Figure A.2:*** *A model of a driveline in a car with defined torques and angles in every part of the model.*

$$J_e\ddot{\theta}_e = M_e - M_c \tag{A.1}$$

The equations (A.2) can be stated from the description in Figure A.2.

$$\begin{cases} M_c = M_t \\ \theta_e = \theta_c \\ \theta_c = \theta_t i_t \\ M_t i_t = M_p \\ M_p = M_f \\ \theta_t = \theta_p \\ \theta_p = \theta_f i_f \\ M_f i_f = M_d \\ M_w = M_d \\ \theta_f = \theta_w \end{cases} \tag{A.2}$$

From the equations (A.2) expressions are derived for $M_w$ and $\theta_w$ expressed in $M_c$ and $\theta_e$. This is done in equations (A.3) and (A.4).

$$M_w = M_d = M_f i_f = M_p i_f = M_t \underbrace{i_f i_t}_{i} = M_c i \tag{A.3}$$

$$\theta_w = \theta_f = \frac{\theta_p}{i_f} = \frac{\theta_t}{i_f} = \frac{\theta_c}{i_f i_t} = \frac{\theta_c}{i} = \frac{\theta_e}{i} \tag{A.4}$$

The results from the equations (A.3) and (A.4) is inserted into (A.1).

$$J_e\ddot{\theta}_e = M_e - \frac{M_w}{i} \Leftrightarrow M_w = M_e i - J_e\ddot{\theta}_e i = M_e i - J_e\ddot{\theta}_w i^2 \tag{A.5}$$

The angular velocity of the wheel can be expressed as $\dot{\theta}_w = \frac{v}{r_w}$. Taking the derivative of this gives an expression of the angular acceleration of the wheel. This is shown in equation (A.6).

$$\ddot{\theta}_w = \frac{d}{dt}\dot{\theta}_w = \frac{\frac{d}{dt}v}{r} = \frac{a}{r} \tag{A.6}$$

When this is inserted into equation (A.5) the term that represents the inertia in the driveline appears.

$$M_w = M_e i - \frac{J_e i^2 a}{r} \tag{A.7}$$

# B

# CVXgen Code

```
dimensions
  m = 2
  n = 3
  T = 7

  EngineDelay = 2
  BrakeDelay = 1
  MaxDelay = 1

end

parameters
  u_old_1 (m)
  u_old_2 (m)
  jLim (1) nonnegative
  r (T+1)
  u_ref_e (1)
  u_ref_b (1)

  F (n,n)  # dynamics matrix.
  G_Engine (n,1)
  G_Brake (n,1)
  D_a(1,m)
  D_j(1,m)
  Bc (n,1)

  Q_ref (1,1) psd
  Q_delta_u1 (1,1) psd
  Q_delta_u2 (1,1) psd
  Q_eps_j (1,1) psd
  Q_eps_u (1,1) psd
  Q_energy_u1 (1,1) psd
```

```
  Q_energy_u2 (1,1) psd

  x[0] (n)  # initial state.

  u_max (m)
  u_min (m)

end
variables
  x[t] (n), t=1..T+1  # state.
  u[t] (m), t=0..T  # input.
  epsilon_j[t] (1), t=0..T #Eps
  epsilon_u1[t] (1), t=0..T
  epsilon_u2[t] (1), t=0..T
end
minimize
  sum[t=0..T](quad(x[t+1][2]+D_a*u[t]-r[t+1], Q_ref)
  +quad(epsilon_j[t],Q_eps_j))
  + sum[t=0..T](quad(epsilon_u1[t],Q_eps_u)
  + quad(epsilon_u2[t],Q_eps_u))
  + sum[t=1..T](quad(u[t][1]-u[t-1][1],Q_delta_u1))
  + quad(u[0][1]-u_old_1[1],Q_delta_u1)
  + sum[t=1..T](quad(u[t][2]-u[t-1][2],Q_delta_u2))
  + quad(u[0][2]-u_old_1[2],Q_delta_u2)
subject to
  x[1] == F*x[0] + G_Engine*u_old_2[1] + G_Brake*u_old_1[2] + Bc
  x[2] == F*x[1] + G_Engine*u_old_1[1] + G_Brake*u[0][2] + Bc
  x[t+1] == F*x[t] + G_Engine*u[t-EngineDelay][1]
            + G_Brake*u[t-BrakeDelay][2] + Bc , t=2..T
  u[t][1] <= u_max[1], t=0..T
  u[t][2] <= u_max[2], t=0..T
  u[t][1] >= u_min[1], t=0..T
  u[t][2] >= u_min[2], t=0..T
  abs(x[t+1][1]+D_j*u[t]) - epsilon_j[t]<= jLim, t=0..T
  epsilon_j[t] >= 0, t=0..T
  abs(u[t][1]-u_ref_e)-epsilon_u1[t] <= 0.1*(1+t*0.01)*u_ref_e, t = 0..T
  abs(u[t][2]-u_ref_b)-epsilon_u2[t] <= 0.1*(1+t*0.01)*u_ref_b, t = 0..T
  epsilon_u1[t] >= 0, t=0..T
  epsilon_u2[t] >= 0, t=0..T
end
```

# Bibliography


[1] Thomas A. Badgwell S.Joe Qin. A survey of industrial model predictive control technology. 2013. Cited on page 1.

[2] Raffaello D'Andrea Mark W. Mueller. Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. *IEEE International Conference on Robotics and Automation*, 2014. Cited on page 2.

[3] Svante Gunnarsson Peter Lindskog Lennart Ljung Johan Löfberg Tomas McKelvey Anders Stenman Jan-Erik Strömberg Martin Enqvist, Torkel Glad. *Industriell reglerteknik: Kurskompendium*. Cited on pages 3, 17, 19, 23, 29, and 30.

[4] C.V. Rao P.O.M. Scokaert D.Q. Mayne, J.B. Rawlings. Constrained model predictive control: Stability and optimality. *Automatica Vol. 36*, 2000. Cited on pages 3 and 30.

[5] Lennart Ljung Torkel Glad. *Reglerteori: Flervariabla och olinjära metoder*. Studentlitteratur, 2003. Cited on page 4.

[6] Jan M. Maciejowski Eric C. Kerrigan. Soft constraints and exact penalty function in model predictive control. *UKACC International Conference*, 2000. Cited on pages 4 and 27.

[7] David Di Ruscio. Model predictive control with integral action: A simple mpc algorithm. *Modeling, Identification and Control*, 2013. Cited on page 4.

[8] N. Schofield M.J. West, C.M. Bingham. Predictive control for energy management in all/more electric vehicles with multiple energy storage units. *Electric Machines and Drives Conference*, 2003. Cited on page 4.

[9] M. Morari F. Borrelli, A. Bemporad. *Predictive Control for linear and hybrid systems*. 2014. Cited on page 4.

[10] Giorgio Rizzoni Lorenzo Serrao, Simona Onori. A comparative analysis of energy management strategies for hybrid electric vehicles. *Journal of Dynamic Systems, Measurement, and Control*, 2011. Cited on page 4.

[11] Tao Gao Caiying Shen, Peng Shan. A comprehensive overview of hybrid electric vehicle: Powertrain configurations, powertrain control techniques and electronic control units. *International Journal of Vehicular Technology*, 2011. Cited on page 4.

[12] Huei Peng Jinming Liu. Modeling and control of a power-split hybrid vehicle. *Control Systems Technology Vol. 16*, 2008. Cited on page 4.

[13] A.M. Phillips M.L. Kuang I.V. Kolmanovsky H.A. Borhan, A. Vahidi. Predictive energy management of a power-split hybrid electric vehicle. *American Control Conference*, 2009. Cited on page 4.

[14] Ken Butts Chris Vermillion, Jing Sun. Model predictive control allocation for overactuated systems - stability and performance. *46th IEEE Conference on Decision and Control*, 2007. Cited on pages 4 and 24.

[15] Tor Aksel N. Heirung Bjarne Foss. Merging optimization and control, 2013. Cited on pages 4 and 24.

[16] Lieven Vandenberghe Stephen Boyd. *Convex Optimization*. Cambridge University Press, 2004. Cited on pages 15, 16, 17, and 47.

[17] Johan Löfberg. Linear model predictive control, stability and robustness. 2001. Cited on page 30.

[18] Manfred Morari Alberto Bemporad. *Robust Model Predictive Control: A Survey*. Springer London, 1999. Cited on page 31.

[19] Johan Löfberg. *Minmax approaches to robust model predictive control*. PhD thesis, Linköping University, 581 83 Linköping, Sweden, 4 2003. Cited on page 66.

[20] Lars Nielsen Lars Eriksson. *Modeling and Control of Engines and Drivelines*. Preprint edition, 2014. Cited on page 69.

Linköpings universitet

## Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida `http://www.ep.liu.se/`

## Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: `http://www.ep.liu.se/`

© Mathias Mattsson och Rasmus Mehler