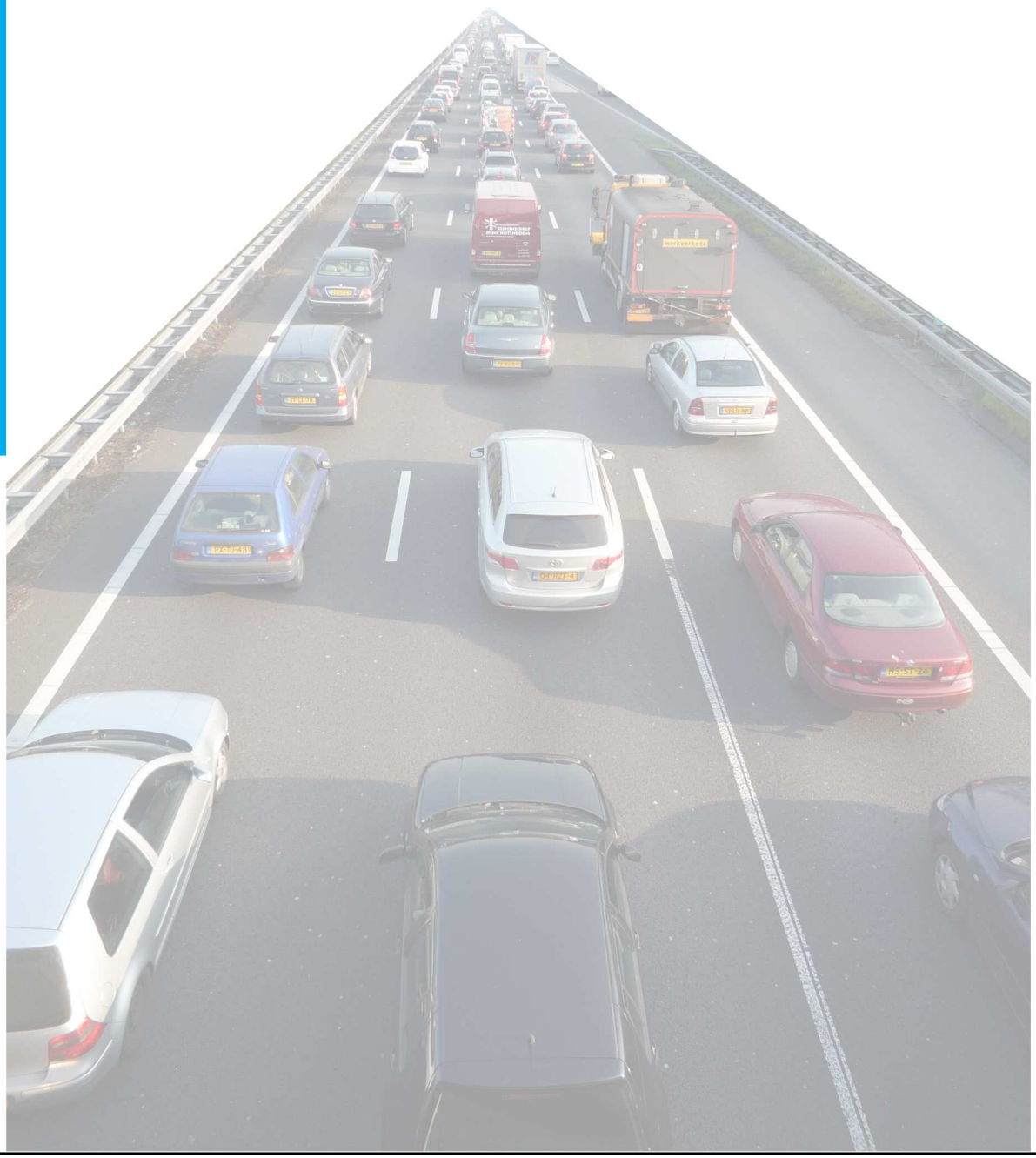
Cooperative Adaptive Cruise Control

Using information from multiple predecessors in combination with MPC

C. Kreuzen

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Cooperative Adaptive Cruise Control

Using information from multiple predecessors in combination with MPC

Master of Science Thesis

For the degree of Master of Science in Systems and Control at
Delft University of Technology

C. Kreuzen

23 May 2012

Faculty of Mechanical, Maritime and Materials Engineering · Delft University of Technology

Abstract

Cooperative adaptive cruise control (CACC) makes the vehicle follow its predecessor at a close but safe distance, and uses information received from other vehicles to accomplish this task. In literature and in practice, the control method mostly applied for CACC is proportional integral derivative (PID) control, possibly with some refinement for gear shifting or comfort. The control method called model predictive control (MPC) can also be used for CACC, and from literature it appears to be more promising than PID, because of its ability to anticipate future situations and to implement constraints directly into the control algorithm. MPC applies the first input of a control input sequence that optimises a performance index calculated from predicted system behaviour, based on a prediction model, subject to operational constraints, in a receding horizon approach. Furthermore, literature has shown that with PID the use of state information from the second predecessor or the platoon leader, in addition to the direct predecessor's states, can improve the CACC performance.
Therefore, in this thesis the approach of using such additional communicated information from either the second predecessor or the platoon leader is combined with the use of MPC as control method. The goal is to investigate whether any of these two configurations give an increase in performance compared with similar configurations with PID as control method, and compared with a more basic configuration that uses just the direct predecessor's state information with either MPC or PID. Also, the possibly added value of using communicated predicted states, in addition to current states, with MPC is investigated.
The CACC controllers are designed to control the throttle, the brakes, and the gears, subject to operational constraints on acceleration, velocity, and vehicle-to-vehicle distance. The PID-based CACC controller contains a proportional feedback of the errors in velocity, position, and acceleration, combined with an automatic transmission scheme, and the control input is restricted at time instants at which a constraint is (almost) violated. The MPC-based CACC controller at each time step minimises the expected errors in position and velocity and the corresponding input variation. The MPC prediction model is obtained by approximating a nonlinear vehicle model by a piecewise affine (PWA) model, and converting the MPC optimisation problem into a mixed integer linear programming (MILP) problem.
In this project, tuning is done by applying simulated annealing for a scenario involving four CACC-controlled vehicles following a platoon leader. Then, the tuned controllers are implemented in a validation scenario comprising a larger platoon undergoing a longer simulation. The results from simulating this validation scenario show that the PID-based CACC controller has a low responsiveness, compared with MPC, and lets the first two vehicles crash. With MPC several peaks and oscillations in throttle/brake input and acceleration occur, and it is expected that with the MPC-based CACC controllers as designed and tuned here, string

stability will not always be achieved for increasing platoon lengths.
It is expected that properly retuning will result in better performing controllers. However, due to limited time, this retuning could not be performed within the scope of this project, and is therefore left as a recommendation. Therefore, only preliminary conclusions can be formulated, which are that MPC should be preferred over PID as a control method for CACC, because it is safer. Moreover, with MPC it should be preferred to, in addition to the current states of the direct predecessor, at least use the current states of the second predecessor and/or the predicted future states from the direct predecessor, in order to achieve better string stability.

Acknowledgements

This project could only have been carried out with the kind assistance of colleagues, family and friends. I would like to thank my supervisor prof.dr.ir. Bart De Schutter for his support and guidance throughout this project, and for his patience and understanding on a more personal level. I was fortunate to be able to profit from his skills, not only in the systems and control theory and research in the field, but also his thorough knowledge of the English language and his precision.
I am also indebted to the other members of my examination committee dr.ir. Ton van der Boom, dr.ir. Riender Happee, and Mohammad Hajiahmadi MSc. for their assessment of my thesis and research.
My thanks go to the teachers, secretaries, and other staff at the Delft Center for Systems and Control for creating a good atmosphere for students in general and for me in particular. My fellow students, with whom I shared a room at the Delft Center for Systems and Control deserve a special mention. Their company, kindness, support and friendship made me enjoy my working hours there and also motivated me to come whenever possible.
I owe a huge debt of gratitude to my other friends for their friendship and support. In particular special thanks go to my best friend Rosalinde who is always there for me, despite the geographical distance, and has helped me to keep going in hard times and has shared many happy moments with me. I have been greatly supported by my parents and my sister, and I would like to give them special thanks for standing by me through all the difficult times, and not giving up on me trying to finish my studies.
And finally, I would like to express my appreciation to all the poor people who regularly get stuck in the daily traffic jam on the A13 near Delft. They (unknowingly) enabled me to just go to a viaduct there at a random day during rush hour to make a suitable picture for my front page. May the future bring intelligent vehicle systems and automated highway systems that make their daily commute between home and work less time consuming, safer and more comfortable.

Delft, University of Technology — Christian Kreuzen
23 May 2012

Table of Contents

Acronyms

ACC adaptive cruise control

CACC cooperative adaptive cruise control

GPS global positioning system

MPC model predictive control

PI proportional integral

PID proportional integral derivative

PWA piecewise affine

MLD mixed logical dynamical

MILP mixed integer linear programming

Chapter 1

Introduction

This report is written as part of the graduation project of the MSc program in Systems and Control at the Delft University of Technology. This chapter first gives the motivation for the choice of this specific project, and then the problem statement, the goal, and the procedure of the project are briefly described, after which an overview of the chapters of this report is given.

1-1 Motivation

In the last couple of decades traffic intensity has grown significantly. One of the results of this is an enormous growth in the amount of congestion, especially on highways. In addition to being very inconvenient, this congestion is financially disadvantageous for the parties involved, and it has harmful environmental effects. The high traffic intensity also has a negative influence on road safety, and together with the resulting irregular driving causes high fuel consumption.
Much research has been done on designing intelligent vehicle systems, which can partially address problems like those mentioned above [5, 37, 42]. Intelligent vehicle systems are systems that sense some elements of the driving environment, and then perform actions to control, or to assist the driver in controlling part of the driving [5]. Another important motivation for most of such systems is to increase the convenience of the driving task for the driver.

There are various intelligent vehicle systems being developed that can act on different levels and different aspects of driving. An example of such a system is a lane departure warning system, which gives a warning sign when the vehicle starts to cross the lane border. Lane keeping assistance goes even one step further by applying a torque to the steering wheel towards the centre of the lane that increases when the vehicle approaches the edge of the lane. Examples of intelligent vehicle systems to be applied in more urban situations are pedestrian detection and avoidance systems, intersection collision avoidance, and parking assist systems.

This project focuses on a system called (cooperative) adaptive cruise control. The purpose of adaptive cruise control (ACC) is to control the velocity of the vehicle, and the distance to its predecessor. The vehicle will maintain a preset speed unless there is a close predecessor, in which case it will try to follow this predecessor at a safe distance. This distance may be

based on a time headway[1] preselected by the driver. The ACC system can, e.g., measure the distance to its predecessor with a radar sensor. However, if there is communication between the vehicles, the controller can receive necessary data (e.g., position, velocity, and acceleration of one, or even more predecessors) faster. Then this data may be more precise, and can possibly give information about future actions of the surrounding vehicles. In this case the ACC system is often called cooperative adaptive cruise control (CACC). Smaller headways can be achieved, and if a group of vehicles is equipped with CACC, platooning is possible. This means that a group of vehicles is driving in a tightly spaced formation. In this report the abbreviation ACC is used as general term for ACC and CACC, and the term CACC will be used when specifically talking about the system with use of vehicle-to-vehicle communication. Early ACC systems would only operate above a speed of about 40 km/h [11]. But much research has already been done with systems that can completely stop the vehicle, if necessary, and accelerate again from a standstill [2, 7, 9, 18, 22, 30].
ACC is primarily a convenience system, relieving the driver in the dreary, but exhausting task of driving in busy or even congested traffic on highways. But it also has the potential to have a positive effect on safety. In addition, the traffic throughput on highways will probably be improved by ACC, because of the smaller following distance and the decrease of collisions. At this moment, only a small percentage of the vehicles on the market, mainly high-end cars [11, 38], is already equipped with (high-speed) ACC. But slowly the introduction of ACC is spreading from the high-end vehicles towards the mid range. For CACC, other vehicles also need to have such a system in order to fully benefit from its possibilities, and reliable vehicle-to-vehicle communication should be implemented as well. Mainly because of this, CACC systems are not on the market yet and it will take longer than for plain ACC before they will be available in average cars [31, 38].
In the literature several strategies for tackling the ACC problem have been proposed [1, 2, 7, 9, 11, 18, 21, 24, 25, 40, 47, 49] (see Chapter 2 for more details). The control methods most frequently used in literature are proportional integral differential (PID) control, and model predictive control (MPC), of which the latter appears to be the most promising one, as will be clarified in Chapter 2. A CACC system can use state information from just its direct predecessor, but it can also be designed to additionally use communicated state information from other vehicles in the platoon. Literature research shows that using states from e.g., the second predecessor or the platoon leader in addition to the states of the direct predecessor can give an improvement of the performance of the controller (see Chapter 2). Such approaches are only found in literature in combination with a PID control method.

1-2 The Project

1-2-1 Problem statement and goal

Problem

The problem that needs to be dealt with is that high traffic intensities on highways cause traffic congestions, with the undesirable (financial, environmental, safety, and convenience related) consequences mentioned before. Therefore, research should be performed to find ways to

[1] The term headway represents the separation of (the front end of) the vehicle from (the rear end of) its predecessor. This separation can be measured in spacial distance (in meters), where the more specific term "distance headway" can be chosen to avoid confusion. The headway can also be given in terms of separation in time, then called the "time headway". The time headway specifies the time that would elapse for the vehicle to reach the current position of the rear end of its predecessor, given the its velocity stays constant.

safely and comfortably increase traffic throughput on highways. CACC systems should—once they are widely introduced in vehicles—increase traffic throughput, because they allow shorter intervehicle distances than possible with manual driven vehicles [1, 5, 42]. However, research has shown that PID-based CACC systems perform worse in comfort for passengers and avoiding collisions, than (less widely researched) MPC-based systems [2, 11]. Therefore, CACC systems using PID generally are suspected to require bigger intervehicle distances than systems using MPC. Furthermore, the approach of applying MPC in combination with using communicated information from more than one predecessor has not been found in literature. This while PID-based CACC performance has shown to improve when using such additional information [22].

Goal

The goal of this project is to design different CACC controllers (further specified below), and to evaluate these controllers by applying them in a simulation of a platoon of CACC-driven vehicles, in order to answer the following question:

> Does the combination of MPC with using communicated information from two other vehicles, being either the first two predecessors or the direct predecessor and the platoon leader, result in an increase of performance, compared with PID (with or without using such additional information) or with MPC using just the direct predecessor's information? And in what aspects, and to what extent, is the performance different?

To be able to answer this question, CACC controllers will be designed in three different information gathering configurations, based on which combinations will be made of other vehicles from which states will be used. The first configuration only uses state information from the direct predecessor. The second configuration uses state information from its two direct predecessors. The third configuration uses states from its direct predecessor and the platoon leader. Each configuration will be designed in two types, one with MPC, and one with PID. MPC needs reference trajectories for the vehicle's states, which are derived from the states of the other vehicles used. Having the vehicles communicate their predicted states (which are calculated in the MPC algorithm) is expected to improve the performance. However this has not been proven yet. Therefore, the MPC version will be designed in two variants, one with using only the current states of the other vehicle(s), and one with also using communicated predicted states.
With each controller to be designed, simulated vehicles should be able to follow at close distance without colliding, in typical everyday traffic scenarios, where the leader's states (which make up the scenario) stay within comfort limits for its passengers. This means that the controllers are not intended to prevent collision in all circumstances. For instance, in the case of the leader braking from high velocity to zero with maximum force, the vehicles (equipped with the CACC controllers designed here) may still collide. For such more extreme situations some sort of collision avoidance system would have to be added. Therefore, in this project the aim is to find which proposed controller performs best for everyday traffic situations, keeping in mind that in a later stadium an additional collision avoidance system could be designed to handle more extreme situations. Furthermore, a platoon of CACC-controlled vehicles should be string stable. This means that if a disturbance enters the platoon, e.g., when the leader suddenly brakes, it should not amplify while propagating towards the tail of the platoon. The CACC system should also ideally obey the following constraints: The velocity of the vehicle should be between zero and a certain maximum velocity, and the acceleration should be between a minimum and a maximum value, representing comfort limits[2].

1-2-2 Procedure

While designing the CACC controller it is assumed that at each time step the states of the own vehicle are known from measurements, and the necessary state information of other vehicles is received through communication. Furthermore it is assumed that the outputs of the CACC controller, which are the throttle/brake pedal position and the current gear, can be appropriately applied to the vehicle. For this project the controllers will be tuned and evaluated by simulating a platoon of vehicles in MATLAB, undergoing some typical traffic scenarios (imposed on the platoon leader). Each of these vehicles, except the leader, will be controlled by a copy of the same CACC controller. Each vehicle will be simulated using a mathematical model of the dynamics of a vehicle.
During the design stage each controller has a few tuning variables which will be tuned by minimising a performance measure calculated from the outputs of a simulation of a range of typical traffic scenarios. This performance measure will assess close following, obeying constraints, the amount of gear switches, smooth pedal position variation, and a string stability margin. These factors will also be assessed when evaluating and comparing the designed controllers in the end.

1-3 Overview

This report is organised as follows. Chapter 2 explains the theory behind (C)ACC in more detail, and discusses state-of-the-art CACC systems presented in literature. Chapter 3 then describes the main objectives of the project, the CACC design methods used here, and a tuning algorithm design. Chapter 4 presents a case study, where the controllers are implemented in a computer simulation of a platoon of vehicles to be tuned and evaluated. Finally Chapter 5 gives some conclusions and recommendations.

[2]The minimum allowed acceleration has a negative value and represents the maximum allowed deceleration.

Chapter 2

State of the art

In literature several proposed ACC systems can be found. Different researchers use different approaches to tackle the various aspects of the ACC problem. Also different methods from control theory have been applied in the design of such systems. Most of these systems have only been tested in simulations, and some have been implemented in test vehicles on test tracks. In this chapter first the ACC problem will further be explained (Section 2-1), after which the state of the art of the proposed ACC systems from literature is presented (Section 2-2). Next, the results from simulations or vehicle tests, as presented in literature, will be discussed (Section 2-3). The chapter will conclude with a short summary.

2-1 The adaptive cruise control problem

In this section, first the task of the system is described, and how it can get its information and apply the control signals when it is used in a real vehicle. Then dynamical models of a vehicle will be presented, after which the control problem will be discussed.

2-1-1 The ACC system in general

The ACC task

The task of ACC is to make the vehicle follow its predecessor at a safe distance, and, in absence of an obstructing predecessor, to drive with a predetermined velocity.
While accomplishing this main task, ACC should be able to achieve string stability. A platoon is string stable if disturbances die out when propagating towards the tail of the platoon. Such a disturbance can for instance be the disturbance in following distance that results when a vehicle suddenly brakes strongly. ACC should also make sure that the (variations in) acceleration and deceleration stay within a comfort zone. Furthermore, frequent switching between applying throttle and brakes should be avoided.

A general representation of the vehicle with the ACC in a block diagram is shown in Figure 2-1. The ACC controls the position and velocity of the vehicle. It aims to make the velocity error and the position error zero. The reference signal in the figure consists of the desired position, velocity, and possibly acceleration. The desired velocity is often equal to the velocity

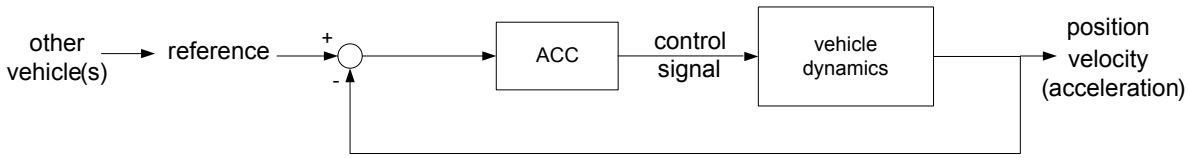


Figure 2-1: General block diagram of an ACC controlled vehicle.

of the predecessor. If following the other vehicle(s) requires a desired velocity that is higher than the preset velocity for free driving, this preset velocity becomes the desired velocity. The desired position is usually derived from a desired distance to the predecessor(s). If there is no preceding vehicle to follow, there is no reference position, and only the velocity (and if desired, acceleration) is controlled. If information from vehicles further along is received through communication and used, this can be used to adapt the desired state values, or communicated states can be used as extra references to which an extended control algorithm can be applied (see Section 2-2-3).

Measurements

The ACC system needs to know at least the velocity of its own vehicle and of the predecessor (if there is one present) and the distance to this predecessor. Some systems also use the acceleration of these two vehicles or the same parameters corresponding to vehicles further ahead. Sometimes radar sensor systems are used, which are fast and give a robust performance for various weather conditions. But these systems are quite expensive [5]. An alternative is the use of lidar systems, which use near-infrared light laser pulses, at wavelengths between 750 nm to 1000 nm, to measure the distance to and the velocity difference with the predecessor [5][1]. These are cheaper, but have trouble handling fast moving images and visibility reduction caused by fog or smoke. Other ACC systems use a global positioning system (GPS) receiver [7, 21, 30]. The position accuracy from GPS is in the order of $10 - 20$ m. Additional differential positioning error correction signals from the nearest ground GPS base station can improve this accuracy up to 1-2 cm [30]. These FM signals are received by a radio receiver mounted on the vehicle. The update frequency of this differential GPS is too low ($\sim$1 Hz) for ACC. Moreover, differential GPS signals can be blocked by, e.g., buildings, trees, and bridges causing the signal to not always be available. To have position information available at all time, the differential GPS data can be fused with data from dead reckoning sensors like wheel encoders and accelerometers[2]. This data fusion can be done by means of an extended Kalman filter, which estimates the parameters for time instants at which no new GPS signal is received, based on information from the dead reckoning sensors [21]. If such a GPS method is used, the states of the preceding vehicle are not measured, but they can be obtained by means of vehicle-to-vehicle communication. From the position data (in time) obtained by the above sensor methods, velocity (and acceleration) data can easily be calculated.

Vehicles can use wireless communication to inform other vehicles about their states and possibly send information about their expected states and (upcoming) throttle and brake actions. For more information on the various technical methods that can be used to achieve such communication, the interested reader is referred to [5].

[1] Lidar stands for "light detection and ranging".

[2] Dead reckoning is the process of estimating a state from a measured value of this state at a known time instant in the past and measurement of a state that ideally represents the derivative of this state during the period since that time instant. For instance, the time-integral of the measured angular speed of a wheel times the wheel's radius should give the travelled distance since the last measurement of the actual position. However, wheel slip or other circumstances can cause the error in estimated position to grow.

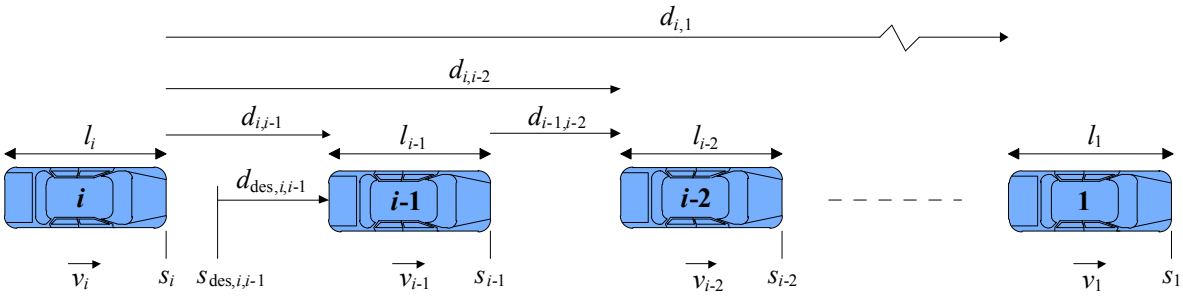


Figure 2-2: ACC vehicle i in a platoon, with important parameters shown.

Actuation mechanisms

The ACC controls the actions of the throttle and the brakes. Some ACC systems take over the control of the gear switching as well, whereas most systems leave that to the automatic transmission of the vehicle.

If the throttle is "by-wire", there is a potentiometer attached to the pedal that converts the pedal position into an analogue signal, which it sends to the motor controller. In this case the ACC throttle controller can send a similar signal to the motor controller, which replaces the pedal signal [30]. For brake-by-wire or gear-by-wire, similar interventions can be made to control braking or gear switching.
If the throttle is mechanically driven, the throttle position can be controlled by a stepper motor [49]. A common hydraulically assisted braking system can be actuated by having a DC motor pulling the brake pedal, or by a solenoid valve controlled electric vacuum booster influencing the pressure in the brake system [49].

2-1-2 Modelling

In order to compare different methods and to avoid confusion, a general standard notation for most of the variables involved is introduced here. This notation will be followed throughout the remaining of this report.

The ACC is being designed for vehicle i (the ACC vehicle, see Figure 2-2). Its predecessor is vehicle $i-1$, counting all the way further up to vehicle 1, which is the platoon leader in case of a platoon. The velocity of each vehicle is given by v with as subscript its number (i, $i-1$, ..., 1) and its acceleration by a with corresponding subscript. The letter s will be used for the position of (the nose of) the vehicle and l is its length. The distances (head to tail) between two vehicles is assigned a d with subscripts indicating which two vehicles the parameter refers to, as shown in Figure 2-2. The distance between the ACC vehicle and its predecessor is called the distance headway of this vehicle[3].
Furthermore, $s_{\text{des},i,i-1}$ refers to the desired position of vehicle i in the case where the only other vehicle (other than vehicle i itself) from which states are used by the ACC system of vehicle i is vehicle $i-1$. When state information from more vehicles is used, the desired position used by the controller can be different from $s_{\text{des},i,i-1}$. The absolute desired position,

[3]In the literature, two definitions can be found for the term headway, differing in whether the distance from head to tail is indicated or the head-to-head distance. In this report, the former version is used to define the term headway.

obtained from the information from all the vehicles considered, is given by $s_{\text{des},i}$.

The relation between the force that the tires exert on the road (wheel force F_{w}), and the velocity and acceleration of the vehicle is given by [11]

$$F_{\text{w}}(t) = ma(t) + (cv^2(t) + \mu mg) \cdot \text{sgn}\big(v(t)\big) \tag{2-1}$$

where m is the mass of the vehicle, s, v, and a respectively give the position, velocity and acceleration of the vehicle, c is the viscous friction coefficient to model the air drag, μ is the Coulomb friction coefficient to model a static friction between the tires and the road, and g is the gravitational acceleration.
The powertrain, transferring the force on the throttle or brake pedal to wheel force contains a delay, which acts as a low-pass filter [18]. This gives the following dynamic vehicle model [2, 18], in which (2-1) can be substituted:

$$F(t) = F_{\text{w}}(t) + \tau \dot{F}_{\text{w}}(t) \tag{2-2}$$

where F is the force at the throttle pedal or, in case of braking, the negative of the force at the brake pedal, and τ is the delay time constant (e.g., $\tau = 0.5\,\text{s}$ [2, 18]).
Most ACC systems in literature give as control input to the vehicle the throttle/brake pedal position. The relation between this pedal position and the force at the pedal, in the simplified case of neglected powertrain delay (i.e. $\tau = 0$), depends on the current gear and the velocity as follows [11]:

$$F(t) = b(j(t), v(t))u_{\text{tb}}(t) \tag{2-3a}$$

with

$$b(j(t), v(t)) = \frac{T_{\text{e}}(\omega)p(j)}{R} \ , \quad v = \frac{\omega R}{p(j)} \tag{2-3b}$$

where $b(j, v)$ is a gear dependence function, j is the gear, $u_{\text{tb}} \in [-1, 1]$ is the pedal position (with a negative value representing brake and a positive value representing throttle action)[4], $T_{\text{e}}(\omega)$ is the engine torque at engine rotational velocity ω, p gives the gear ratio, and R is the average radius of the wheels. The relation between vehicle control inputs (pedal position and gear) and the vehicle's velocity and acceleration can be obtained by combining equations (2-1)-(2-3). For the simplified case of neglected powertrain delay (i.e. $\tau = 0$), this gives as dynamical vehicle model

$$b\big(j(t), v(t)\big) \cdot u_{\text{tb}}(t) = ma(t) + (cv^2(t) + \mu mg) \cdot \text{sgn}\big(v(t)\big) \tag{2-4}$$

In literature, often one or more factors that are modelled above are neglected.

In the above, acceleration caused by braking was modelled by putting the negative of the braking force as the force input into the equations. In reality, the dynamics for braking are somewhat different from the dynamics while applying the throttle. Several models for braking can be found in literature (see, e.g., [9, 24]). But in most ACC research no special model for braking is used.

[4]The variable u_{tb} gives the relative position of the pedals such that a value of -1 corresponds to the brake pedal being pushed all the way down, 0 corresponds to no brake or throttle action, and 1 corresponds to the throttle pushed all the way down. The (undesired) situation of pushing both pedals simultaneously is not modelled here.

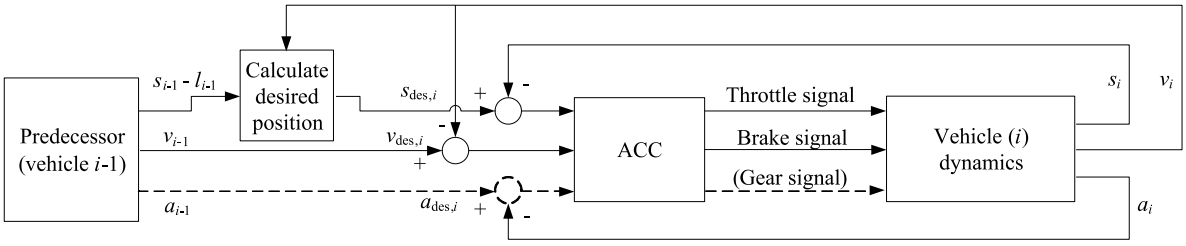


Figure 2-3: Block diagram of an ACC-controlled vehicle. Dashed segments are optional.

2-1-3 Control problem

ACC comprises a control algorithm that consists of some general control method from control theory, like PI(D) or MPC. These methods will be further explained in Section 2-2. Such control methods are constantly working to reduce the error in the outputs of the system to, preferably, zero. In the case of ACC, the system to be controlled consists of the longitudinal motion of the driving vehicle. The outputs of this system are the vehicle's position and velocity. Also its acceleration can be measured and controlled. The control inputs to the (longitudinal) vehicle system are throttle, brake, and gear signals. For the simple situation of just following one predecessor without constraints, this is shown in Figure 2-3, which is a more specific version of the block diagram of Figure 2-1.

Some proposed ACC systems in the literature let one controller directly calculate the throttle/brake (force) command [9, 11, 24, 30, 40, 47]. Other systems have an outer loop with a controller that outputs an acceleration command to an inner loop. The inner loop then controls the dynamics of the vehicle giving a throttle/brake (force) command output [1, 2, 7, 22]. Some ACC systems have different controllers for brake control and for throttle control, combined with some kind of switching logic [9, 24, 30]. However most systems use one controller, that gives as output either the demanded acceleration for an inner loop [2, 21], or a throttle/brake command, where a positive sign corresponds to throttle action and a negative sign corresponds to brake action [11, 18, 25, 40, 47].

When only looking at the direct predecessor, the desired position is determined by the desired distance headway. Without vehicle-to-vehicle communication the desired distance headway needs to be velocity dependent in order to be able to achieve string stability [10, 16, 39]. This means that the desired distance increases with increasing velocity. In literature mostly a constant time headway (h_0) is used, meaning that the desired elapsed time between now and the moment of passing the point where (the rear end of) the preceding vehicle is right now should be independent of velocity. A small minimum distance headway (d_0) is then added as a safety distance. This makes the desired headway equal to [18, 24, 30, 47]:

$$d_{\text{des},i,i-1}(t) = d_0(t) + h_0 v_i(t) \tag{2-5}$$

Sometimes a variable time headway is used. A term that takes the difference in velocity between the two vehicles into account is then added to the constant time headway. Namely, if the leading vehicle is driving faster than its follower (positive velocity difference), the desired headway can be made smaller than in the situation where the following vehicle is approaching the leading one (negative velocity difference). With variable time headway the equation becomes [18, 22, 47],

$$d_{\text{des},i,i-1}(t) = d_0 + \left(h_0 - c_{\text{h}}(v_{i-1}(t) - v_i(t))\right) \cdot v_i(t) \tag{2-6}$$

where c_{h} is a constant.

$v \geq 0\,\mathrm{m/s}$	$v \leq v_{\max}$	$d \geq d_{\min}$	$a \geq a_{\min}$	$a \leq a_{\max}$

Table 2-1: Some general constraints

When communicated information of more vehicles is used, one way to use this information is to adapt the desired headway based on the communicated data (more on this in Section 2-2-3).

Constraints

Ideally, the controller is to be restricted by some general constraints [2, 9, 11, 24], the most important of which are given in Table 2-1 and are explained below. The first constraint ($v \geq 0\,\mathrm{m/s}$) makes sure that the controller does not allow the vehicle to drive backwards. Without such a constraint, e.g., when the desired velocity is close to zero, but the current velocity is significantly greater than that, the controller can try to get to the desired velocity with first having some overshoot, which will make the velocity (in theory) below zero. It is very intuitive that in reality this does not make sense. The other velocity constraint $v \leq v_{\max}$ could be set to the legally maximum allowed velocity or to a different desired maximum velocity.
If the headway (d) would become less than zero, the vehicles would collide. A controller, not knowing this, can make the vehicle arrive at the desired distance after having had some overshoot, which might have been too big, causing d to become negative (which corresponds to a crash). This is the reason for the so called collision avoidance constraint $d \geq d_{\min}$, with a small value for the minimal distance.
Acceleration constraints can be added to keep the movements of the vehicle within the comfort zone. The minimum comfortable acceleration (maximum comfortable deceleration) gives the constraint $a \geq a_{\min}$, and the maximum comfortable acceleration $a \leq a_{\max}$.
It may also be desirable to add constraints on the control inputs, that make sure that the controller does not give control inputs that exceed the physical limitations of the actuators or the vehicle.

2-2 Control methods

Control theory has yielded several control methods. ACC systems in vehicles on the market today still mainly use simple PI controllers, but some other methods are extensively researched, and in the future, versions of those may be implemented in commercial vehicles. The two controller types most frequently found in literature are PID (or just PI) and MPC. An explanation of these methods and an overview of how they are used in adaptive cruise controllers as described in various articles on the subject, will be given in the next subsections.

2-2-1 PI(D)

A proportional integral derivative (PID) controller acts on the error in the output (vector) of the system, which is the difference between the output and the reference value. It calculates the time integral and time differential of the error and feeds a sum of those and the error itself back to the system, each with a gain factor.

In [24, 47] the ACC system uses a PI controller on the velocity error and the position error. The control law is given by

$$u_{\text{tb}}(t) = k_{\text{p}}\big(v_{i-1}(t) - v_i(t) + k_1(s_{\text{des},i}(t) - s_i(t))\big) + k_{\text{int}} \int_0^t \big(v_{i-1}(t') - v_i(t') + k_2(s_{\text{des},i}(t') - s_i(t'))\big)dt' \quad (2\text{-}7)$$

where $s_{\text{des},i}$ is the desired position, k_1, k_2, k_{p} and k_{int} are positive design constants, and u_{tb} is the throttle/brake signal. The letter u will be used to denote the output of the controller from this point on. The other variables are as defined in Section 2-1-2. According to [24] the use of the term $v_{i-1}(t') - v_i(t')$ in the integral part is found to be beneficial in reducing the speed overshoot due to a large position error. The variables k_1 and k_2 can then be used to fine-tune the ratio between the influence of the position error and the speed error for the proportional and the integral term separately. This can be very useful to optimally make use of the benefits of the term $v_{i-1}(t') - v_i(t')$.

In most literature on ACC no integral term as above is used in the PI(D) controller. Most often a proportional action over the position, velocity, and sometimes additionally the acceleration error, is considered [2, 9, 11, 18, 22]. This gives as control law,

$$u_{\text{tb}}(t) = k_s \cdot (s_{\text{des},i}(t) - s_i(t)) + k_v \cdot (v_{i-1}(t) - v_i(t)) + k_a \cdot (a_{i-1}(t) - a_i(t)) \quad (2\text{-}8)$$

where k_s, k_v are positive design constants, and k_a is either positive or zero, the last of which is the case in literature that does not consider acceleration. This can be seen as a proportional (P) controller with as input a vector containing the errors in the two or three outputs of the system[5].

For heavy-duty vehicles some model parameters can vary quite strongly. The mass variation between a lightly and heavily loaded truck can be more than 300%. Also road grade has much more influence than it has on light passenger vehicles. This causes fixed-gain controllers to have limited performance. A way to deal with this problem is to use adaptive gains [47]. For this, a reference model is used, and at each time step the gains of the controller are adapted online such that the system and the reference model have the same response to the same input signal. To achieve this, the error between the system and the reference model responses is used, and a Lyapunov function is applied, after which an update law for the gains is found. For the exact method, see [24, 47].

If constraints should be implemented with a PID control method, this can only be done by cutting off the output at the appropriate values. This means that a simple algorithm should decide, at every instant, whether a constraint is (about to be) violated. If so, it should calculate the corresponding maximum or minimum allowed output and give that as output, instead of the output it would give with no constraints. To be able to do this, a model of the system is needed.
For the constraints on the acceleration this means that if the output of the PID control law would make the vehicle accelerate faster than the maximum allowed acceleration, the controller output will be set to the value that will make the vehicle assume this maximum acceleration. Similarly, if the expected acceleration resulting from applying the PID law is too small, violating of the minimum acceleration constraint will be prevented by setting the controller output to the value resulting in the smallest allowed acceleration (maximum allowed

[5]This can also be seen as a PID controller over the velocity, because acceleration is the derivative of velocity and position is the integral of it.

deceleration).
For some of the constraints this is implemented in [9], which uses different controllers for braking and accelerating. The throttle control law is given by

$$u_{\mathrm{t}}(t) = \begin{cases} \min\left(\frac{(a_{\max}+k_{\mathrm{f}}v_i(t))}{b_u}, k_s\big(s_{\mathrm{des},i}(t) - s_i(t)\big) + k_v\big(v_{i-1}(t) - v_i(t)\big)\right), & \text{if } v_i < v_{\max} \\ \frac{k_{\mathrm{f}}v_i(t)}{b_u}, & \text{if } v_i = v_{\max} \end{cases} \tag{2-9}$$

Here u_{t} is the throttle pedal position, b_u is a positive constant, and $k_{\mathrm{f}}v_i$ gives the (simplified) friction. The simplified vehicle model for positive throttle input used here is[6] $b_u u_{\mathrm{t}}(t) = a_i(t) + k_{\mathrm{f}}v_i(t)$. So the first line of the control law makes sure the maximum acceleration is obeyed. The second line prevents the vehicle to go above the maximum velocity. Similar constructions can be made to obey some of the other constraints.
In [1], the ACC system tries to obey the maximum velocity by taking the minimum of two calculated acceleration commands, to give as input to an inner loop:

$$a_{\mathrm{ref}} = \min(a_{\mathrm{ref}_v}, a_{\mathrm{ref}_d}) \tag{2-10}$$

where a_{ref} is the acceleration command, used as acceleration reference by the inner loop, a_{ref_v} is the acceleration command based on following the maximum velocity, so a little overshoot can occur here, and a_{ref_d} is the acceleration command based on following the predecessor.

In more extreme situations, e.g., when the predecessor suddenly brakes strongly or when a vehicle suddenly changes lane right in front of the ACC-controlled vehicle, there is an increased chance of collision. To take care of this problem, some PID-based ACC designs have extended the algorithm with a special part to avoid collisions in situations like these. See [9, 22, 30] for the different approaches.

2-2-2 MPC

Model predictive control (MPC) uses a mathematical model of the system to be controlled and, if available, knowledge of the future values of the reference signals, to predict the effect of a sequence of control inputs. By finding the control sequence that results in the most optimal predicted system behaviour, restricted by the given constraints, the next control input is determined.

MPC often uses a state space model to calculate state and output values corresponding to the input values that are explored during the optimisation. The model can be nonlinear, but often it is desirable to use a linear model for reasons of low computational expense. A discrete-time linear state space model is of the following form[7]:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2-11}$$

where $x(k)$ is the state vector at time step k, $u(k)$ is the input, and $y(k)$ is the output vector. MPC uses a performance index to determine the cost of every input-predicted output sequence

[6] Although [9] fails to mention the mass with respect to the positive-throttle vehicle model, it is assumed that the coefficients k_{f} and b_u in (2-9) are values considered per unit mass.

[7] In this thesis the simulation period (T_{sim}), i.e., the period between two time steps in the discrete simulation model used by the MPC controller to simulate predicted future dynamics, is set equal to the sampling period T_{s}, i.e., the time between subsequent output measurements performed to be used by the controller. Therefore, from now on the term sampling period will be used when referring to this period.

[8, 28, 33]. Such a performance index can take different forms, which usually are quite similar in essence. In the case when there are reference signals to track, a possible performance index is

$$J(u,k) = \sum_{n=N_{\mathrm{m}}}^{N} \left\| Q(\hat{y}(k+n \mid k) - r(k+n)) \right\|_{p_1}^{p_2} + \sum_{n=1}^{N_{\mathrm{c}}} \left\| Ru(k+n-1) \right\|_{p_1}^{p_2} \tag{2-12a}$$

or

$$J(u,k) = \sum_{n=N_{\mathrm{m}}}^{N} \left\| Q(\hat{y}(k+n \mid k) - r(k+n)) \right\|_{p_1}^{p_2} + \sum_{n=1}^{N_{\mathrm{c}}} \left\| R\Delta u(k+n-1) \right\|_{p_1}^{p_2} \tag{2-12b}$$

where $\hat{y}(k+n \mid k)$ means the prediction of the output y at time step $k+n$, based on knowledge up to time step k (current time), $r(k+n)$ is the reference signal at time $k+n$, u is the input control signal to the system, $\Delta u(k) = u(k) - u(k-1)$ is the input increment, Q is the weighting matrix for the output error, R the weighting matrix for the input (increment), $N_{\mathrm{m}} \in \{1, \dots N\}$ is called the minimum cost horizon, N the prediction horizon, $N_{\mathrm{c}} \in \{1, \dots N\}$ the control horizon, and p_1 and p_2 determine the order of the norm. Often 1-norms ($p_1 = p_2 = 1$) or squared 2-norms ($p_1 = p_2 = 2$) are used, but other norms can be used too, like the ∞-norm ($p_1 = \infty,\ p_2 = 1$). It should be noted that if C in (2-11) is an identity matrix, then the output y used in (2-12a) is equal to the state. In the following, the "$\mid k$" part will be omitted in the equations for ease of notation.

The first part of the performance index from (2-12a) calculates a weighted quadratic sum of the predicted output error over time steps from N_{m} up to the prediction horizon N. The second part is a weighted sum of the inputs over the length of the control horizon. Hence, keeping the performance index low is making the output follow the reference well, while also trying to keep the input control signal small. The second performance index (2-12b) takes as second part a sum of the input increments. This is because sometimes it is preferable to keep the variation in the input control signal small.

MPC uses an optimisation algorithm to find a sequence of control inputs that minimises the performance index for fixed prediction, control, and minimum cost horizons. It then takes the first value of this optimal input sequence and feeds this to the system. At the next time step, this is done again including data from the new measurements of the output, and new reference values, and so on. Every time step, after a new control signal has been applied, it stays constant until its value is updated to match the calculated control input of the next time step.

In the explanation above, it is assumed that the sampling frequency and the control update frequency, i.e., the frequency of computing and updating the control signal, are equal. However, it is also possible to have a sampling period that is a multiple of the control update period. This way, modelled system dynamics occurring between control-update time instants can be detected by the optimisation. A performance index of the type of (2-12b) would, in the case that the control-update period T_{ctrl} is bigger than the sampling period T_{s} according to $T_{\mathrm{ctrl}} = m_{\mathrm{c}} T_{\mathrm{s}}$ ($m_{\mathrm{c}} \neq 0 \in \mathbb{N}$), look like

$$J(u,k) = \sum_{n=N_{\mathrm{m}}}^{N} \left\| Q(\hat{y}(k+n \mid k) - r(k+n)) \right\|_{p_1}^{p_2} + \sum_{n=1}^{N_{\mathrm{c}}/m_{\mathrm{c}}} \left\| R\Delta u(k_{\mathrm{c}}+n-1) \right\|_{p_1}^{p_2} \tag{2-13a}$$

with

$$k_{\mathrm{c}} = \mathrm{floor}\left(\frac{k-1}{m_{\mathrm{c}}} + 1 \right) \tag{2-13b}$$

Here "floor" means rounding down to the nearest integer, and k_{c} is the control update step counter. In the model of (2-11) $u(k)$ is then replaced by $u\left(\mathrm{floor}\left(\frac{k-1}{m_{\mathrm{c}}} + 1\right)\right)$[8].

[8] Although the performance index (2-13) and the system model representation for the case of $T_{\mathrm{ctrl}} = m_{\mathrm{c}} T_{\mathrm{s}}$ may look unusual, they are easy to transform into a prediction form (defined in Section 3-3-3) as is needed for the application of the MPC algorithm.

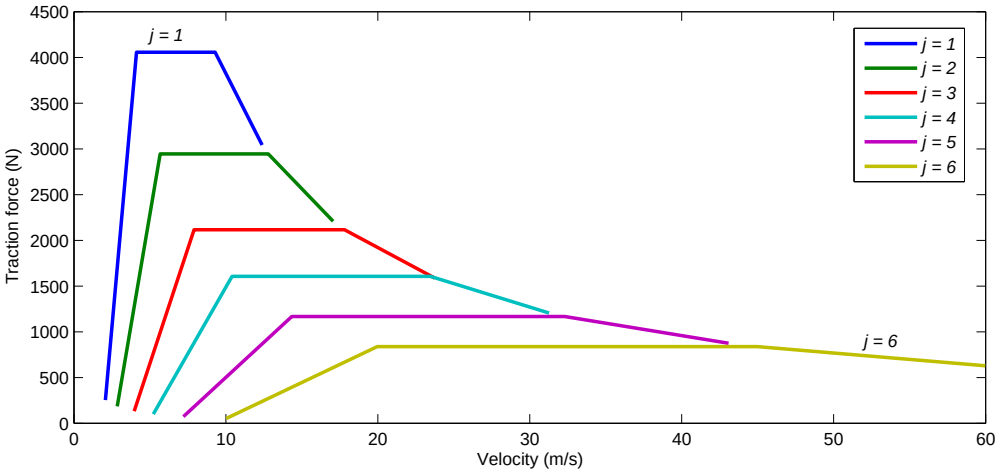


Figure 2-4: Relation between the vehicle's velocity and *maximum* wheel traction force, i.e., for $u_{\text{tb}} = 1$, for each gear. Here the model parameters of a SMART are used, according to [11].

If the computations are too complex to be done online, the optimisation can also be done offline instead. Then a look-up table has to be constructed, which can be used online to find the right values corresponding to the current situation. The disadvantage of off-line MPC is that it requires more on-board memory. If the future reference signals are not known a priori, in the prediction they will usually be assumed to be equal to the current reference signal, i.e., $r(k+n) = r(k)$ for $n = N_{\text{m}}, \ldots, N$ (or estimated in some other manner).
Because some optimisation algorithms can cope with several kinds of constraints on the variables, MPC using such an algorithm can directly deal with constraints, without having to apply extra (non-sufficient) tricks as has to be done with PID (discussed in Section 2-2-1).

Because in a model predictive controller the system model plays an important role, the controller will perform better if the model is a closer representation of the reality. However, a model that is too complex results in too much computation needed to obtain the control signal, which is undesirable, especially when the computations have to be done online. Also, often not much will be gained by using a very complex model, because it will not make the controller more robust against changing circumstances.
For the ACC application it can be chosen to use a simplified model that neglects the nonlinear influence of the air drag, and to assume that the force is proportional to the pedal position independent on the gear. This is done in [2], which also neglects the tire friction. A disadvantage of just neglecting such factors can be that the control input at some points is far from optimal, because it is based on a model that is simplified too much[9].

Another article [11] takes the more complex model from (2-3), as a basis, and then presents several ways to approximate this model for use as a prediction model for MPC. Two factors make this model very complex to use. The first is the presence of the gear dependence function ($b(j, v)$), which is a nonlinear function of the gear and the velocity. The second is the nonlinearity introduced by modelling the air drag giving a squared velocity.
The (approximated) relation between the velocity and the traction force for each gear for

[9] In [2] the model simplifications are also applied in the model used to simulate the vehicles in the validation scenarios. Therefore it is not clear from this paper what the consequences of these simplifications in the model used by MPC are when applying the controller to a more realistic vehicle simulation.

Gear j	δ_1	δ_2	δ_3	$b(j)$(N)
1	0	0	0	4057
2	1	0	0	2945
3	0	1	0	2116
4	1	1	0	1607
5	0	0	1	1166
6	1	0	1	838

Table 2-2: Relation between the gear number and the values of the three binary variables that encode the gear, and the values of the gear dependence function $b(j)$ after it was simplified to only depend on j.

$u_{\text{tb}} = 1$ and $\omega_{\min} \leq \omega \leq \omega_{\max}$ is given by [11] as shown in Figure 2-4[10]. As can be seen from the figure, for each gear there is a region where the traction force is approximately constant at a maximum value. To simplify the model, in [11] for each gear the traction force is assumed to be equal to this maximum (see Table 2-2), where the velocity bounds for choosing each gear are put at the edges of the respective maximum value region (as can be seen by comparing Figure 3-2b with Figure 2-4). So the minimum and the maximum allowed velocity for each specific gear number j are given by constants $v_{\text{low},j}$, and $v_{\text{high},j}$ respectively. In this approximated model the traction force factor (b) is no longer (directly) dependent on the velocity, but only dependent on the gear, replacing $b(j, v)$ by $b(j)$.
The resulting model is given by

$$b(j(t))u_{\text{tb}}(t) = ma(t) + (cv^2(t) + \mu mg) \cdot \text{sgn}\big(v(t)\big) \tag{2-14}$$

After discretising and translating to a state space form this becomes

$$x(k+1) = \begin{bmatrix} x_1(k) + T_{\text{s}} x_2(k) \\ x_2(k) - \frac{T_{\text{s}} c}{m} x_2^2(k) - T_{\text{s}} \mu g \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{T_{\text{s}}}{m} b(j(k)) \end{bmatrix} \cdot u_{\text{tb}}(k) \tag{2-15}$$

where $x = [x_1, x_2]^{\text{T}} = [s, v]^{\text{T}}$, and T_{s} is the sampling period.
Then [11] approximates the nonlinear gear dependence $b(j)$ by a PWA function. Here, instead of the gears being controlled actively, gear switching is governed by dividing the velocity domain into regions. In each region only one gear is allowed, which is defined by the following inequality constraints:

$$v_{\text{g},0} + v_{\text{g},1} j(k) \leq v(k) \leq v_{\text{g},0} + v_{\text{g},1} \cdot (j(k) + 1) \tag{2-16}$$

where $v_{\text{g},1} > 0$ and $v_{\text{g},0}$ are constants. For each gear, between its velocity bounds, the gear dependence is approximated by a linear function of the gear number, $b(j(k)) \approx \beta_0 + j(k)\beta_1$, with β_0 and β_1 constants. Then j is encoded as

$$j(k) \triangleq \delta_1(k) + 2\delta_2(k) + 4\delta_3(k) + 1 \tag{2-17}$$

where $\delta_i(k) \in \{0, 1\}$ are binary variables (used to be able to transform the model into an MLD form, as described below, see (2-26)). This way each value of the gear corresponds to a

[10] Here the relation $T_{\text{e}}(\omega)$ (representing the engine torque at engine rotational velocity ω, see (2-3b)) is approximated by a piecewise affine (PWA) function consisting of three lines, which for the vehicle model of a Smart have their edges at the points (105 N·m, 5 rad/s), (209 N·m, 80 rad/s), (471 N·m, 80 rad/s), and (628 N·m, 60 rad/s). Hence the straight lines in Figure 2-4. The function $p(j)$ (giving the gear ratio for each gear in (2-3b)) for the Smart has values 14.203, 10.31, 7.407, 5.625, 4.083, and 2.933 for the respective gears in increasing order.

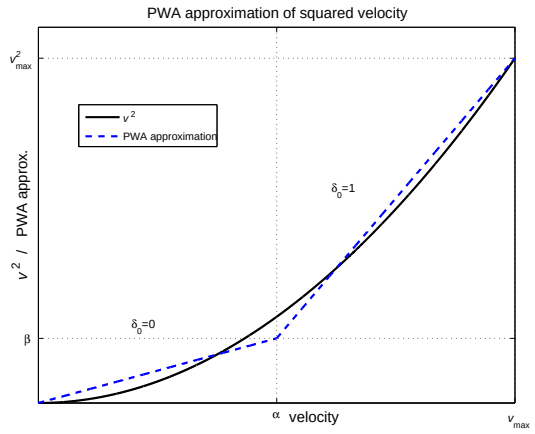


Figure 2-5: PWA approximation of the squared velocity, applied by [11].

unique combination of values for the three corresponding binary variables, as shown in Table 2-2. The gear dependence function now becomes

$$b(j(k)) \approx (\beta_0 + \beta_1) + \beta_1\delta_1(k) + 2\beta_1\delta_2(k) + 4\beta_1\delta_3(k) \tag{2-18}$$

Because of the combination of continuous and discrete variables, this is a hybrid system, which can be dealt with by selecting appropriate MPC algorithms [3, 11].

Several methods are applied by [11], for comparison purposes, where the model is further approximated in different ways and to different extents. First applies MPC is applied with a nonlinear discrete-time model with approximated gear dependence. This has very high computational costs. Therefore, some other methods are applied with linearised models. One of those uses the affine system tangent to the current operating point. This way, the linearised model has to be calculated at each time step. Another method (applied in two variants, online and offline) approximates the nonlinear air drag by a PWA function, as shown in Figure 2-5. The expressions for the two lines (P_1, and P_2) of the PWA function are given by:

$$\begin{aligned} P_1 &: \frac{\beta}{\alpha}x_2, && x_2 \in [0, \alpha] \\ P_2 &: \frac{v_{\max}^2-\beta}{v_{\max}-\alpha}(x_2-\alpha)+\beta, && x_2 \in [\alpha, v_{\max}] \end{aligned} \tag{2-19}$$

with

$$\alpha = \frac{v_{\max}}{2}, \qquad \beta = \frac{3v_{\max}^2}{16} \tag{2-20}$$

Each line is valid in a specific region on the domain of x_2. This means that replacing x_2^2 in the model by its PWA approximation results in a different state space equation for each region. The term $x_2 - \frac{T_s c}{m}x_2^2$ from (2-15) can now be replaced by

$$\begin{aligned} x_2 - \tfrac{T_s c}{m}x_2^2 \quad &\rightarrow \quad (1 - \tfrac{T_s c\beta}{m\alpha}) \cdot x_2 \;=\; a_1 x_2 && \iff \quad v_{\min} \le x_2 \le \alpha \\ x_2 - \tfrac{T_s c}{m}x_2^2 \quad &\rightarrow \quad (1 - \tfrac{T_s c(v_{\max}^2-\beta)}{m(v_{\max}-\alpha)}) \cdot x_2 + \tfrac{T_s c\alpha(v_{\max}^2-\beta)}{m(v_{\max}-\alpha)} - \tfrac{T_s c\beta}{m} \\ &\qquad = a_2 x_2 + r && \iff \quad \alpha < x_2 \le v_{\max} \end{aligned} \tag{2-21}$$

where the constants a_1, a_2, and r are introduced here for clarity. After substituting this, a PWA state space model is obtained of the form

$$\begin{cases} x(k+1) = A_1 x(k) + F_1 + B u_{\text{tb}}(k), & \text{for } x_2(k) < \alpha \\ x(k+1) = A_2 x(k) + F_2 + B u_{\text{tb}}(k), & \text{for } x_2(k) \ge \alpha \end{cases} \tag{2-22}$$

Some tools are developed in [3] to deal with such PWA models. This paper shows that a binary variable can be introduced, which encodes the two regions of the model. For the model of (2-22) this gives $\delta_0(k) = 0$ for $x_2(k) < \alpha$, and $\delta_0(k) = 1$ for $x_2(k) \geq \alpha$. The following rules from [45] are presented (by [3]) to be able to further process models with such variables. This first rule shows how to construct a system of linear inequalities to alternatively express the definition of these (region-encoding) binary variables:

$$([f(x) \leq 0] \iff [\delta = 1]) \quad \iff \quad \begin{cases} f(x) & \leq & M(1-\delta) \\ f(x) & \geq & \varepsilon + (m-\varepsilon)\delta \end{cases} \tag{2-23}$$

with tolerance $\varepsilon > 0$ having a very small value[11], $M = \max_{x \in X} f(x)$ and $m = \min_{x \in X} f(x)$, where X is the domain of x. The next rule can be used when in the system model a product occurs between a binary input or state variable and a continuous input or state variable. Such a product makes the model nonlinear. By replacing it by a new continuous auxiliary variable, and alternatively expressing the definition of this variable by a system of linear inequalities, the model can be linearised. The corresponding rule is [45]

$$z = \delta f(x) \quad \iff \quad \begin{cases} z & \leq & M\delta \\ z & \geq & m\delta \\ z & \leq & f(x) - m(1-\delta) \\ z & \geq & f(x) - M(1-\delta) \end{cases} \tag{2-24}$$

with z a continuous auxiliary variable. Similarly, a product between two binary input or state variables, making a model nonlinear, can be replaced by a new binary variable of which the definition can also be expressed by linear inequalities according to [45]

$$\delta_3 = \delta_1 \delta_2 \quad \iff \quad \begin{cases} -\delta_1 + \delta_3 & \leq & 0 \\ -\delta_2 + \delta_3 & \leq & 0 \\ \delta_1 + \delta_2 - \delta_3 & \leq & 1 \end{cases} \tag{2-25}$$

It is proposed in [3] to transform a PWA model into a mixed logical dynamical (MLD) model, which is done in [11] with the model of (2-22). The general MLD model looks like

$$\begin{cases} x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) + B_5 \\ E_1 u(k) + E_2 \delta(k) + E_3 z(k) \leq E_4 x(k) + E_5 \end{cases} \tag{2-26}$$

with state vector $x = [x_\mathrm{c}^\mathrm{T}, x_\mathrm{b}^\mathrm{T}]^\mathrm{T}$, $x_\mathrm{c} \in \mathbb{R}^{n_\mathrm{c}}$, $x_\mathrm{b} \in \{0,1\}^{n_\mathrm{b}}$, input vector $u = [u_\mathrm{c}^\mathrm{T}, u_\mathrm{b}^\mathrm{T}]^\mathrm{T}$, $u_\mathrm{c} \in \mathbb{R}^{m_\mathrm{c}}$, $u_\mathrm{b} \in \{0,1\}^{m_\mathrm{b}}$, $\delta \in \{0,1\}^{r_\mathrm{b}}$ a vector of binary auxiliary variables, and $z \in \mathbb{R}^{r_\mathrm{c}}$ a vector of continuous auxiliary variables.

Using this approach, (2-22) can be transformed into

$$x(k+1) = (1 - \delta_0(k))(A_1 x(k) + F_1) + \delta_0(k)(A_2 x(k) + F_2) + B u_\mathrm{tb}(k) \tag{2-27}$$

After defining a new continuous auxiliary variable z, where $z(k) = \delta_0(k)x(k)$, the model becomes

$$x(k+1) = A_1 x(k) + B u_\mathrm{tb}(k) + (F_2 - F_1)\delta_0(k) + (A_2 - A_1)z(k) + F_1 \tag{2-28}$$

[11] In (2-23) the statement $[f(x) \leq 0] \iff [\delta = 1]$ (with $\delta \in \{0,1\}$) implies $[f(x) > 0] \iff [\delta = 0]$. The statement $[\delta = 1] \implies [f(x) \leq 0]$ is represented by $f(x) \leq M(1-\delta)$ (if $\delta = 0$ this inequality stays valid, by stating that $f(x)$ is smaller than or equal to it maximum value). The statement $[\delta = 0] \implies [f(x) > 0]$ can be represented by $f(x) > m\delta$. To rewrite this strict inequality into a non-strict inequality, the small tolerance ε is introduced, which typically has the size of the machine precision, giving the inequality $f(x) \geq \varepsilon + (m-\varepsilon)\delta$. If, here, the value of $f(x)$ is smaller than ε the constraint $f(x) > 0$ is considered to be violated [3].

The inequalities that arise from applying (2-23)-(2-25) become part of the MLD model (2-26), and can be incorporated into the MPC algorithm as constraints to the optimisation.

The performance index used in [11] is based on 1-norms. The advantage of this over squared 2-norms is that it lowers the complexity of the optimisation problem, allowing the use of mixed integer linear programming (instead of mixed integer quadratic programming). It is preferable to also minimise the number of gear switchings (Δj), which is why this is also entered in the performance index:

$$J(x(k), \tilde{u}_{\mathrm{tb}}(k), \tilde{j}(k)) = \sum_{i=1}^{N} \left(\parallel Qe(k+i) \parallel_1 + \parallel R_u \Delta u_{\mathrm{tb}}(k+i-1) \parallel_1 + \parallel R_j \Delta j(k+i-1) \parallel_1 \right) \tag{2-29}$$

where $e(k) = x(k) - x_{\mathrm{des}}(k)$ is the state error vector, Q, R_u, and R_j are weighting matrices, $\tilde{u}_{\mathrm{tb}}(k) = [u_{\mathrm{tb}}(k), \ldots, u_{\mathrm{tb}}(k+N-1)]^{\mathrm{T}}$, and $\tilde{j}(k) = [j(k), \ldots, j(k+N-1)]^{\mathrm{T}}$. Because the acceleration is not used as an output or state, the acceleration constraint is written as function of the state x_2 and sampling period T_{s} as $a_{\min} T_{\mathrm{s}} \leq x_2(k+1) - x_2(k) \leq a_{\max} T_{\mathrm{s}}$. Extra constraints are necessary here, because of the introduction of the gears:

$$\begin{aligned} 1 \leq j(k) \leq 6 \qquad &\rightarrow \quad 0 \leq \delta_1(k) + 2\delta_2(k) + 4\delta_3(k) \leq 5 \\ -1 \leq j(k+1) - j(k) \leq 1 \qquad & \\ \hookrightarrow \quad -1 \leq \delta_1(k+1) + 2\delta_2(k+1) + 4\delta_3(k+1) &- \delta_1(k) - 2\delta_2(k) - 4\delta_3(k) \leq 1 \end{aligned} \tag{2-30}$$

The first is because there are only 6 gears in the specific car used. The second constraint, which can be applied if desired, prohibits gear changes that jump to two or more gears higher or lower.
The optimisation problem becomes a mixed integer nonlinear programming or a mixed integer linear programming (MILP) problem depending on whether the final model still contains nonlinearities.

As seen above, the structure of the MPC method is very suited to add constraints and include extra features, e.g., introducing discrete variables into the model, like the gear. Also, because the minimisation of the performance index can be done online, the MPC method can deal better with model variation than the PID method. It just needs an observer to estimate online the value of parameters that might vary, like the mass in case of a truck. These values can then be adapted in the model that is used for the prediction of future states and outputs. With the forward looking ability of MPC, choosing the appropriate constraints should be sufficient to avoid collision, at least in everyday driving situations[12].

2-2-3 Using communicated information from more vehicles

Sometimes it can be very useful to already act directly on what a vehicle further downstream is doing. For instance in the situation where the second vehicle ahead suddenly brakes strongly, it is safer to start braking right away instead of waiting until the direct predecessor reacts to it. When there is communication between the vehicles, information from vehicles downstream (or upstream if necessary) can be received by wireless communication, and the vehicle can also react to changes further ahead. Including these changes in the controller can be implemented

[12] Extreme situations, like a suddenly maximally braking predecessor are not researched in the articles about MPC considered here. In such cases an MPC-based ACC system following at close distance may still crash, since ACC is designed as a vehicle following system and not as a collision avoidance system.

by designing a special cooperative adaptive cruise control (CACC) algorithm.
One method to use communicated state information is proposed in [22]. For just following the direct predecessor (vehicle $i-1$) it uses a PID control law to determine the commanded acceleration ($a_{\text{ref},i-1}$) (to pass to an inner loop). To include communicated information from the second vehicle in front, it now uses a similar control law to keep a desired distance

$$d_{\text{des},i,i-2} = d_{\text{des},i-1,i-2} + l_{i-1} + d_{\text{des},i,i-1} \tag{2-31}$$

to this second predecessor ($i-2$). Here $d_{\text{des},i-1,i-2}$ is the desired headway from vehicle $i-1$ to vehicle $i-2$, and l represents vehicle length. This results in reference acceleration $a_{\text{ref},i-2}$. Then an algorithm computes the resulting commanded acceleration from the outputs of the two PID rules:

$$a_{\text{ref},i} = \begin{cases} \min(a_{\text{ref},i-1}, a_{\text{ref},i-2}), & \text{if } \min(a_{\text{ref},i-1}, a_{\text{ref},i-2}) < 0 \\ \max\left(a_{\text{ref},i-1}, \frac{a_{\text{ref},i-1}+a_{\text{ref},i-2}}{2}\right), & \text{otherwise} \end{cases} \tag{2-32}$$

with a smooth transition (unspecified by [22]) designed between the outputs of the two rules.

Most changes in acceleration of the vehicles in a platoon of ACC vehicles are an indirect response to actions of the platoon leader. So in the case of a CACC platoon, it can be useful for all vehicles to react also to the movements of the platoon leader in addition to already reacting to the direct predecessor. To implement this, [18] extends the PID equation by adding terms to it that perform a similar PID action, but this time with the references based on the platoon leader's states. The values of the PID gains of this extra part can then be tuned to be lower than the gains of the part based on keeping a desired distance to the direct predecessor. Instead of first applying a PID to different sets of desired states, it is also possible to use information from vehicles further ahead to directly apply a correction term to the desired distance to the predecessor ($d_{\text{des},i,i-1}$), as is done in [7].

2-3 Evaluation of strategies from literature

In literature, the ACC systems are tested by implementing them in simulations or sometimes with real vehicles that undergo some typical traffic scenarios. The scenarios used are catching up with a distant predecessor, following the predecessor when it accelerates or decelerates to a new constant velocity, or coming to a stop safely behind a suddenly strongly braking predecessor. The focus is usually on vehicle following, meaning monitoring the situation between two subsequent vehicles, or on platoon behaviour, like evaluating string stability.

2-3-1 Effects on vehicle following

Literature shows that when using a PID controller, close following often results in either violating (desired) acceleration constraints or collision in some situations [2, 11]. This is caused by the fact that either no acceleration bounds are included, or the PID controller output has to be cut off to obey the minimum acceleration. Because PID cannot predict the results of its actions, it can crash when the deceleration is bounded. Using MPC without constraints, or only with the minimum acceleration constraint (both applied in [2] to gain more insight), can also result in a crash in some situations (with closer following). This is because the minimum distance constraint is necessary for the controller to "know" that the position overshoot should not become too big to avoid collision. With the right constraints

applied, MPC-based ACC can handle several types of scenarios without collision or uncomfortably strong decelerations [2, 11]. Because the MPC algorithm can include constraints like the minimum distance constraint, closer following distances can be achieved compared to PID.

In [11] robustness to model variation is tested by performing a number of simulations with different systems, each time changing different parameters in the simulation vehicle model. The conclusion drawn is that in case of model variation, online methods (like online MPC) are not affected, but a look-up table or a fixed gain PI method only finds sub-optimal solutions and seems to violate the constraints (even) more.

It is shown in [22] that with CACC and using information from the first two predecessors (compared with ACC that just looks at the direct predecessor), vehicles can react very quickly to changes ahead. Two SMARTs using the algorithm of (2-32) following a Peugeot that suddenly brakes strongly to stop, react fast and simultaneously by also braking. The second SMART only needs to brake with half the intensity that the first SMART uses in order to stop at $d_0 = 8\,\text{m}$ behind the first SMART (with $h_0 \approx 1.5\,\text{s}$ in (2-6)). This means that the minimum acceleration constraint does not need to be violated to avoid a crash, even though PID is used here. This also means that the brake intensity decreases with vehicles more upstream, so string stability will also be achieved in a platoon of such vehicles. Because of the quicker reactions, with CACC smaller safe intervehicle distances can be achieved.

2-3-2 Effects on platoon behaviour and string stability

In the literature different definitions of the term "string stability" can be found [6, 12, 31, 40, 46]. Based on these definitions, a general description would be the following. A platoon of vehicles is string stable if disturbances that are introduced anywhere in the platoon die out when propagating upstream (i.e., towards the tail of the platoon).
Different papers focus on different measures to represent the propagating disturbances mentioned above. Some papers consider the propagation of the position error from one vehicle to the next towards the tail of the platoon to study string stability [12, 46]. Others consider the propagation of position and velocity errors [40], of position, velocity, and acceleration errors [6], of velocity [12], of acceleration [12], or of a combination of control input, position error, velocity, and acceleration [31]. Sometimes, a definition of string stability is given that is a relaxed version of the description above, and states that, provided that the initial state errors of all the vehicles are bounded, the state errors of all vehicles stay bounded in time [40][13]. A few papers make a distinction between strong string stability, which requires (norms of) the measure(s) considered to attenuate from each vehicle to its follower, and weak string stability, which only requires (norms of) the measure(s) to decrease from the platoon leader to each of its followers, provided the disturbance started with the platoon leader [17, 31, 39].

In the literature investigation of whether a platoon of ACC vehicles is string stable has been done analytically and empirically.
The analytical method calculates a frequency domain transfer function ($G(j\omega)$) for the considered measures from one vehicle to the next one. This can be calculated after the closed-loop system of the vehicle with the ACC controller is determined. Papers that only consider

[13]The relaxed condition for string stability considering bounded errors or states should not be used for a platoon with bounded length (especially when relatively short), because even if this condition is satisfied for the bounded platoon, it does not guarantee boundedness for longer platoons

the position error, state that for (strong) string stability, the following should hold in the frequency domain [18, 24, 46]:

$$|G(j\omega)| = \left| \frac{S_{\text{des},i}(j\omega) - S_i(j\omega)}{S_{\text{des},i-1}(j\omega) - S_{i-1}(j\omega)} \right| < 1, \ \forall \omega > 0 \tag{2-33}$$

where $S_i(j\omega)$ and $S_{\text{des},i}(j\omega)$ are the frequency domain representations of, respectively, the position and the desired position of vehicle i. To investigate the propagation of other measures (or from the leader to vehicle i), similar transfer functions as (2-33) could be derived for these other measures (or for this other vehicle combination) [31]. By using condition (2-33), it is proven that no string stability can be achieved if the reference distance is constant and there is no vehicle-to-vehicle communication [24, 37, 42, 46]. In fact, it is found that either vehicle-to-vehicle communication (not only from each vehicle to its direct follower)[14], or use of rear end sensor information [46, 50], or a speed-dependent distance headway [10, 16, 24, 39, 46] is needed in order to be able to achieve string stability.
To empirically study the way disturbances propagate through the platoon, the platoon leader accelerates or decelerates from an initial constant velocity to another velocity, which it keeps constant for some period of time. Then the propagation of the measures considered upstream through the platoon and in time is evaluated. In the case of string stability, the peak value over time of the measure(s) considered should decrease for increasing vehicle index (or, in the case of the weaker form of string stability, at least decrease for each vehicle compared to the platoon leader, or the first ACC controlled vehicle). It has been shown, by conducting such empirical analysis, that indeed stability is achieved in simulations of ACC methods that use communication or speed-dependent distance headways [9, 18, 24, 47]. Furthermore it is found that with using a variable time headway as in (2-6) instead of a constant time headway, string stability is easier to achieve and the resulting errors are smaller in magnitude [47].

A comparison between the two basic control methods, PI(D) and MPC, with respect to string stability cannot be given here, because none of the papers on MPC found in literature have done tests on this matter.

Simulations have shown that for vehicles from neighbouring lanes it may be hard to merge into a close-following CACC platoon [1]. This is caused by the small intervehicle spaces and the fact that CACC only considers the situation in the longitudinal direction. A solution to this problem could be to design some co-operative merging procedure. Furthermore from simulations it is concluded that ACC significantly reduces the probability of collision between the ACC vehicle and its predecessor, but it slightly increases the chance of collision for the followers of the ACC vehicle [41, 42].

[14] Often, when in the literature statements are made about vehicle-to-vehicle communication being needed to achieve string stability with a constant desired distance headway, it is added that this means that the *platoon leader* can communicate its velocity (and acceleration) to all other vehicles in the platoon [18, 36, 46]. Additionally, [46] specifically states that it is impossible to achieve string stability with only receiving communicated state information from the direct predecessor, and not from other vehicles. However, no information has been found on the possibility or impossibility of achieving string stability when the communication is not between the platoon leader and all its followers, but, e.g., between each combination of three subsequent vehicles. Furthermore, such statements—although expressed in general terms—are derived as conclusions from research with PID-like methods. It seems incorrect to extend such statements, without further research, to ACC systems that use different control methods, such as MPC.

2-4 Summary

In this chapter, several approaches from literature to tackle the ACC problem are reviewed. Here a short summary will be given.

It is found that both reviewed control methods, proportional integral (derivative) (PI(D)) and model predictive control (MPC), appropriately implemented with possibly necessary extra features, can give reasonably acceptable results with respect to vehicle following when applied in average circumstances. However, when the circumstances diverge from average vehicle following, or when comfort issues are considered, MPC performs better than PID. This is because with PID constraints cannot be properly included, and the controller cannot predict the consequences of its actions. This results in a higher chance of collisions in more critical circumstances, or a greater following distance. MPC on the other hand can easily include constraints in its algorithm. Moreover, because at each control update time step MPC performs an optimisation based on system predictions, the effects of the constraints can be incorporated and be accounted for when deciding the optimal control input. This makes MPC-based ACC safer and more comfortable than a PID-based one. Another advantage of MPC is that online MPC is more robust to variations in the vehicle model. Online MPC is much more computationally complex than PID, but with the use of a simplified model, and with the increase of computational power of computers, it does not have to be too complex to be implemented.
Cooperative ACC (CACC) uses vehicle-to-vehicle communicated information. The use of a special CACC algorithm (incorporating information from more vehicles than just the direct predecessor) was only found in the literature in combination with PID, so at this point no comparison with MPC can be done on this issue. It can be concluded from the literature that with PID closer following can be achieved when communication is involved, especially with the use of specially designed CACC methods.

So, MPC appears to be the control method with the most potential to include several important aspects to make the ACC systems of the future even better at their task. Introducing vehicle-to-vehicle communication and using communicated data in a cooperative ACC algorithm is expected to further improve the benefits of ACC.
Therefore, the project that will be described in the remaining of this report will combine the use of MPC and vehicle-to-vehicle communication.

Chapter 3

Designing control systems for cooperative adaptive cruise control

Now that the state of the art CACC systems have been reviewed, ideas to combine promising approaches in the design of a new CACC controller arise on which this and the following chapter will report[1]. This chapter covers the theoretical part of this new research, independent of the specific vehicle or exact traffic situation the controllers are applied to. First a detailed description of the design objectives is given in Section 3-1. In Section 3-2 a slightly simplified vehicle model is presented to be used for vehicle simulation, and to base the controller design on. Then control algorithms are designed in Section 3-3, after which a way to measure the performance and tune the controllers is presented in Section 3-4.

3-1 Design objectives

Within this project several variants of CACC controllers are to be designed that ideally meet a set of performance requirements that will be given below. The measurement and communication data, which consist of position, velocity, and acceleration states (present and, if needed, predicted future states) of the own vehicle and other vehicles in the platoon, will be assumed available. It will also be assumed that lower-level control is implemented that appropriately applies the control signals, i.e., throttle/brake pedal positions and current gear number, to the vehicle. So the CACC design here will be limited to the level of designing the algorithms that take the measured and communicated data as input and give the control signals as output.

3-1-1 CACC controllers to be designed

Recall that for ease of the exposition the vehicles in a platoon are numbered, starting with the platoon leader as vehicle 1, its follower being vehicle 2, and so on (see Figure 2-2). When designing the CACC controller for an unspecified vehicle in the platoon, this vehicle will be

[1] From this point on, the term "CACC controller" (or just "controller") is used to denote the combination of PID/MPC control law and the processing of the communicated information from other vehicles. Both of these steps can differ for the different CACC controllers. So, as will be explained below, in this chapter nine CACC controllers are designed, and for the sake of clarity, the term "CACC system" will not be used to indicate these specific CACC controllers.

called vehicle i, with $2 \leq i \leq$ total number of vehicles in the platoon. This makes, e.g., its direct predecessor vehicle $i-1$.
Concerning how to use the opportunity of communicating with other vehicles in the platoon, the CACC controllers will be designed in three different information gathering configurations. These configurations will differ in which other vehicles they use communicated state information from:

Configuration I. Only communicated data from the direct predecessor is used. So vehicle i uses data from vehicle $i-1$.

Configuration II. Communicated data from the first two predecessors is used. So vehicle i uses data from vehicles $i-1$ and $i-2$.

Configuration III. Communicated data from the direct predecessor and from the platoon leader is used. So vehicle i uses data from vehicles $i-1$ and 1.

Furthermore, for each of the configurations above, two main controller types will be designed, based on which basic control methods will be used:
- **PID** is used as control method.
- **MPC** is used as control method.

As mentioned in Chapter 2, the promising performance of the use of MPC as well as vehicle-to-vehicle communication in the design of ACC systems, led to the idea of combining the two approaches. Because in the literature and in practice PID is the control method most frequently used, this method will be applied for comparison with MPC. The reasons for the choice of these three specific configurations (I-III) above are the following. Configuration I is very commonly found in literature and, with measuring the states of the predecessor instead of using communication, already applied in real-life commercial vehicles, and is therefore chosen as reference to compare the performance of the other configurations with. Configuration II, documented in [7, 22] (with PID), is chosen because it is reported to achieve better performance than configuration I, since the vehicle can react quicker to upcoming changes, making shorter following distances possible. No literature has yet been found that documents using this configuration in combination with MPC, which is why this promising combination is going to be researched here. Configuration III, proposed in [18] (with PID), is also expected to give a better performance than configuration I, since every (significant) change in velocity introduced by the leader travels through the platoon and will reach each following vehicle at some point. Therefore, incorporating state data from the platoon leader in the CACC controller of each follower is expected to help the vehicles react quicker to such changes. Just as the second configuration, the third configuration, too, is not found in combination with MPC yet, and since it is a considerably different approach compared with configuration II, this configuration is also chosen to investigate.

Because MPC uses predictions of the vehicles' states, the question arises whether or not the CACC controller will perform even better if the vehicles also communicate their predicted states and the controllers use these predicted states from other vehicles. Therefore, the MPC types, for all three configurations I-III, will be designed in two variants:

$\mathbf{MPC_1}$: Only current state data from other CACC-controlled vehicles is used by the MPC-based CACC controller.

$\mathbf{MPC_2}$: In addition to current states, also predicted state data from other CACC-controlled vehicles is received via communication and used by the MPC-based CACC controller.

It will be assumed that the leader of the platoon knows its own future actions (at least up to the prediction horizon of the MPC controllers of the other vehicles), and that for both of the above MPC variants the leader communicates its expected states for the near future to the vehicles that use its states[2].

[2]MPC controllers of vehicles that receive future states from the platoon leader through communication use this information for their state reference trajectories corresponding to future time steps up to the length of the

v	$\geq$	$v_{\min}$	$=$	0
v	$\leq$	$v_{\max}$	$>$	$0\,\mathrm{m/s}$
d	$\geq$	$d_{\min}$	$>$	$0\,\mathrm{m}$
a	$\geq$	$a_{\min}$	$<$	$0\,\mathrm{m/s^2}$
a	$\leq$	$a_{\max}$	$>$	$0\,\mathrm{m/s^2}$

Table 3-1: Constraints that give performance requirements.

Furthermore it is chosen for the CACC controllers to be designed to, in addition to the throttle and brake pedals, also include the gear choice as a control variable. This will be done, because it is expected that the MPC-based CACC controllers may find a more efficient time-gear trajectory if the gear choice along with its consequences are incorporated in the MPC scheme, which looks forward in time.

3-1-2 Performance requirements

To tune, and later to evaluate the controllers, a simulation needs to be designed. Such a simulation should represent a platoon of vehicles moving in longitudinal direction only, each of which (except the leader) is controlled by a CACC controller. To obtain a closer match to reality, the vehicle dynamics simulation is preferably based on a continuous time vehicle model, and the CACC controller is in discrete time. This means that measured state data (and communicated data) is received by the CACC controller at time steps, and the control output is fed to the vehicle simulation through a zero-order hold filter. For the evaluation of the performance of the controllers, different types of traffic scenarios should be simulated in order to cover the main range of typical longitudinal traffic situations on highways.

The controllers have to be designed subject to some performance requirements. First, driving with CACC should be safe, while the following distance should be be small. If the following distance will be taken very big, e.g., a distance corresponding with a time headway of five seconds, then it will probably not be so hard to design a controller that is safe in all circumstances. But the smaller the preferred following distance, the harder it is to make a design that is still safe in all kinds of typical traffic scenarios. So the aim is to design the controllers such that the following distance can be as small as possible, without collisions occurring in the range of daily traffic situations. Also, the controller should obey some constraints, which are shown in Table 3-1. Here, the constraints on the acceleration are for comfort purposes, the constraints on velocity to only drive in the forward direction and to obey a desired maximum velocity, while the distance constraint is to prevent collision, as was explained in more detail in Section 2-1-3. Exact values should be assigned to the parameters $v_{\max}$, $d_{\min}$, $a_{\min}$, and $a_{\min}$ before the controllers are tuned, as will be done in Section 4-2-1. The distance constraint ($d = s_{i-1} - s_i - l_{i-1} \geq d_{\min}$ with $d_{\min}$ having a small value) is one that should not be broken, because otherwise the vehicle will almost definitely crash into its predecessor. The minimum velocity constraint should not be broken, because in a real vehicle a negative velocity in vehicle following makes no sense, and is very undesirable. For the maximum velocity it can be chosen (in the designing stage), e.g., to allow the driver to choose the value for it, or to fix

prediction horizon. Therefore, receiving state information from the platoon leader at least for this period of time into the future is assumed to be beneficial (otherwise missing future (platoon leader) states would have to be predicted in some manner). Since PID cannot handle predictions, the fact that the platoon leader knows its future actions is not relevant for PID-based CACC controllers.

its value to, e.g., the legally allowed maximum velocity on highways. This maximum velocity should not be exceeded too strongly, because, e.g., one could get a fine at some point if it is broken frequently. Though probably if it is approximately obeyed with possibly some very small overshoots (say $< 0.5\,\mathrm{m/s} = 1.8\,\mathrm{km/h}$), it will not pose too much problems in real traffic situations. For the acceleration comfort constraints it will also not suddenly cause a highly uncomfortable situation if relatively small excesses of the constraints (say $< 0.05\,\mathrm{m/s^2}$) occur at some points. Still, the aim will be to strictly obey these constraints. However, knowing from literature research this may not be achievable with PID, the above reasoning will give some extra guidelines for designing the PID controller.
Furthermore, the CACC controller should achieve string stability for a platoon of vehicles that use this controller.

Another comfort issue is to try to make the input variation go smoothly, without too sudden changes or unnecessary changes back and forth (i.e., chattering) resulting in more gradual movements. This does not give hard constraints, but rather motivates to keep changes in the throttle or brake pedal positions or the gear low.

The performance requirements given above should be taken into account while designing and tuning the controllers. Also the evaluation of the controller performance should be based on these requirements.

3-2 Vehicle model

Equation (2-14), from [11], will be taken as the model to represent the vehicle dynamics:

$$b(j(t))u_{\mathrm{tb}}(t) = ma(t) + (cv^2(t) + \mu mg) \cdot \mathrm{sgn}\big(v(t)\big) \tag{3-1}$$

where v is the vehicle's velocity, a its acceleration, m its mass, c the viscous friction coefficient, μ the coulomb friction coefficient, g the gravitational acceleration, $u_{\mathrm{tb}} \in [-1, 1]$ the throttle or negative of the brake pedal position (normalised), $j \in \{1, 2, 3, 4, 5, 6\}$ is the current gear with each value of j only allowed in a given speed interval (see Section 2-2-2), and the function $b(j)$ gives for each gear the maximum traction force (that is the traction force when $u_{\mathrm{tb}} = 1$) as described in Section 2-2-2.
In [11], within this model the lower velocity velocity bound for the first gear ($v_{\mathrm{low},1}$) is set to the starting point of the constant traction force region in Figure 2-4 (which, in that figure, is 3.94 m/s). Because the vehicle should also be able to drive with velocities lower than this value, without further complicating the model again, $v_{\mathrm{low},1}$ will be set to zero. The velocity bounds for the other gears, as well as the mass of the vehicle, depend on what vehicle is used and therefore are given in Chapter 4, which describes the case study. It is assumed, for simplicity, that all the vehicles in the platoon have the same vehicle model, and therefore each vehicle has the same length, denoted by l_{veh}.

3-3 CACC controller algorithms

This section presents the algorithms that make up the CACC controllers. First it is shown how desired values for the states at each time step are obtained from the states of other vehicles serving as reference. Then it is shown how each of the control methods, PID and MPC, are implemented, and strategies are presented that deal with different problems that arise.

3-3-1 Handling the reference values

Depending on which of the three information gathering configurations is concerned, one or two other vehicles in the platoon are used as reference to derive the desired state values for the own vehicle. Looking first at the situation between two subsequent vehicles in the platoon, the velocity of the predecessor is used as desired velocity for its follower, and (if used) the desired acceleration of the follower is the acceleration of the predecessor. The *ideal* following distance (not knowing what is happening elsewhere in the platoon) is then given by

$$d_{\text{des},i,i-1}(k) = d_0 + h_0 v_i(k) \tag{3-2}$$

Here, d_0 represents the intervehicle distance at zero velocity, where this value is a reasonable value for traffic situations. How small the value of time headway h_0 can be chosen, depends on how well the controller can perform. The desired headway is chosen to be velocity-dependent so as to have bigger following distances at higher velocities, which is preferable for safety.
When there is no communication with vehicles other than the direct predecessor (configuration I), the desired states are derived as just described, where the desired position is at the point which lies the desired distance behind the predecessor:

$$\begin{array}{rclcl} s_{\text{des},i}(k) & = & s_{\text{des},i,i-1}(k) & = & s_{i-1}(k) - l_{\text{veh}} - d_{\text{des},i,i-1}(k) \\ & = & s_{i-1}(k) - l_{\text{veh}} - d_0 - h_0 v_i(k) & & \\ v_{\text{des},i}(k) & = & v_{\text{des},i,i-1}(k) & = & v_{i-1}(k) \\ a_{\text{des},i}(k) & = & a_{\text{des},i,i-1}(k) & = & a_{i-1}(k) \end{array} \tag{3-3}$$

When the CACC controller uses communicated state information from specific other vehicles in the platoon, this information will be used to adapt the desired state values. From the states of each other vehicle of which information is used, a set of alternative desired states will be derived in a similar way as is done before with using the states of the direct predecessor. The resulting set represents the desired states as if the focus were on following only this other vehicle, knowing its relative place in the platoon. Then the different sets are linearly combined, with normalised weights, to form one resulting set of desired states, which will be used by the PID or MPC algorithm. This is the general idea, but in this project only states from at most one other vehicle, in addition to the direct predecessor, are used.
Specifically, the alternative set of desired states for vehicle i derived from the second predecessor's (vehicle $i-2$) information are (see also (2-31))

$$\begin{array}{rcl} s_{\text{des},i,i-2}(k) & = & s_{i-2}(k) - 2l_{\text{veh}} - d_{\text{des},i,i-1}(k) - d_{\text{des},i-1,i-2}(k) \\ & = & s_{i-2}(k) - 2l_{\text{veh}} - 2d_0 - h_0(v_i(k) + v_{i-1}(k)) \\ v_{\text{des},i,i-2}(k) & = & v_{i-2}(k) \\ a_{\text{des},i,i-2}(k) & = & a_{i-2}(k) \end{array} \tag{3-4}$$

where the subscript "des, $i, i-2$" means that this is the desired value for vehicle i when only taking into account vehicle $i-2$. The desired values when only taking vehicle $i-1$ into account are as in (3-3). Now, when the first two predecessors states are used as references (configuration II), the final set of desired states will be derived as follows

$$\begin{array}{rclcl} s_{\text{des},i}(k) & = & q_{s,\text{p1}} s_{\text{des},i,i-1}(k) + q_{s,\text{p2}} s_{\text{des},i,i-2}(k) & & \\ v_{\text{des},i}(k) & = & q_{v,\text{p1}} v_{\text{des},i,i-1}(k) + q_{v,\text{p2}} v_{\text{des},i,i-2}(k) & = & q_{v,\text{p1}} v_{i-1}(k) + q_{v,\text{p2}} v_{i-2}(k) \\ a_{\text{des},i}(k) & = & q_{a,\text{p1}} a_{\text{des},i,i-1}(k) + q_{a,\text{p2}} a_{\text{des},i,i-2}(k) & = & q_{a,\text{p1}} a_{i-1}(k) + q_{a,\text{p2}} a_{i-2}(k) \end{array} \tag{3-5a}$$

with

$$q_{s,\text{p1}} + q_{s,\text{p2}} = 1, \quad q_{v,\text{p1}} + q_{v,\text{p2}} = 1, \quad \text{and} \quad q_{a,\text{p1}} + q_{a,\text{p2}} = 1 \tag{3-5b}$$

where the weights are represented by the parameters q, with subscript "p1" corresponding to the references from the direct predecessor, and subscript "p2" corresponding to the references

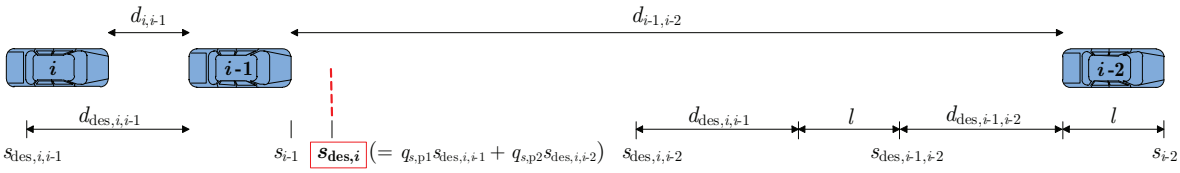


Figure 3-1: Illustration of how, without applying (3-7), the desired position (of vehicle i) could lie beyond the predecessor (vehicle $i-1$). In this example $q_{s,\text{p1}} = q_{s,\text{p2}} = 0.5$

from the second predecessor.
For the configuration where the direct predecessor and the platoon leader are used as references, the position of the leader is not used by the vehicles that do not have the leader as direct predecessor (only its velocity). This is because the relevance of the own position relative to the leader diminishes when there are more vehicles between the leader and the own vehicle, because of position errors of predecessors. The leader is added as reference because of its function in the platoon, where usually the rest of the platoon will follow the leader. This means the leader will probably induce most of the velocity changes in the platoon. Therefore, it is expected that using the velocity and acceleration of the leader to slightly influence the desired velocity and acceleration respectively, will have a positive effect on quick reaction to changes induced by the leader. So in the case of configuration III (using states of direct the predecessor and the platoon leader) the desired state values are given by

$$\begin{array}{lcll} s_{\text{des},i}(k) & = & s_{i-1}(k) - l_{\text{veh}} - d_{\text{des}i,i-1}(k) & = s_{i-1}(k) - l_{\text{veh}} - d_0 + h_0 v_i(k) \\ v_{\text{des},i}(k) & = & q_{v,\text{p}} v_{i-1}(k) + q_{v,\text{l}} v_1(k), & \text{with } q_{v,\text{p}} + q_{v,\text{l}} = 1 \\ a_{\text{des},i}(k) & = & q_{a,\text{p}} a_{i-1}(k) + q_{a,\text{l}} a_1(k), & \text{with } q_{a,\text{p}} + q_{a,\text{l}} = 1 \end{array} \tag{3-6}$$

where weights with the subscript "p" correspond to references from the direct predecessor, and the subscript "l" represents weights corresponding to the references from the platoon leader.

Above for configuration II the desired position is calculated as a weighted average of the desired position when just looking at the direct predecessor, and the desired position when just looking at the second predecessor. This means that when this second predecessor is driving away from the direct predecessor, at some point the desired position (of the vehicle's front tip) can go beyond the rear end of the direct predecessor or even exceed that vehicle's front tip position, as is illustrated in Figure 3-1. To prevent this from happening, a minimum for the desired distance is introduced. When the desired position according to (3-5a) becomes greater than the point lying at the minimum desired distance behind the direct predecessor, this latter point will be used instead as desired position. In that situation, if the direct predecessor's velocity or acceleration is less than the desired velocity or acceleration of the own vehicle respectively, as calculated according to (3-5a), the value of this predecessor's state will replace the calculated value of the respective desired state. This is expressed by

$$\begin{array}{lcll} d_{\text{des,min},i}(k) & = & c_{\text{dmin}}(d_0 + h_0 v_i(k)), & \text{with } 0 \leq c_{\text{dmin}} < 1 \\ s_{\text{des,max},i}(k) & = & s_{i-1}(k) - l_{\text{veh}} - d_{\text{des,min},i}(k) & \end{array} \tag{3-7}$$

and

$$\text{if } s_{\text{des},i}(k) > s_{\text{des,max},i}(k), \quad \text{then redefine:} \quad \begin{cases} s_{\text{des},i}(k) & := & s_{\text{des,max},i}(k) \\ v_{\text{des},i}(k) & := & \min\left(v_{\text{des},i}(k), v_{i-1}(k)\right) \\ a_{\text{des},i}(k) & := & \min\left(a_{\text{des},i}(k), a_{i-1}(k)\right) \end{cases} \tag{3-8}$$

where the value of c_{dmin} may be chosen to be different for the MPC-based CACC controller than for the PID type. The lower bound for c_{dmin} is set to 0 to avoid a desired position that

exceeds the position of the rear end of the predecessor. Here it is assumed that the platoon will not completely split up, so a big gap in the platoon will only be a temporary situation.

The time headway h_0 can have different values for each of the CACC controllers. The second vehicle in the platoon only has one predecessor. When a controller with configuration II or configuration III is used in the vehicles in the platoon, this second vehicle only receives state information from the leader. Its desired states are then calculated in a similar way as with configuration I (as in (3-3), including using the value of h_0 for configuration I with the same controller type. For a platoon in which all vehicles from the third and further ($i \geq 3$) have configuration II CACC, the third vehicle applies, to the first line of (3-4), the fact that the second vehicle uses this other value of h_0. This changes the second line of (3-4) accordingly. The PID controller will use the error in all three states (position, velocity, and acceleration), whereas the MPC controller will only use the position and velocity errors. This choice is made because the MPC algorithm is more powerful than the PID algorithm, since it uses predicted state errors over a prediction period along with the ability to obey constraints. Because of this, using also the acceleration error in the MPC algorithm is considered to be less important (than for PID). Furthermore, not using it will lower the amount of computation at each time step, which is relatively high for an online MPC controller.

3-3-2 CACC with PID

At each time step k the CACC controller will receive state data from the own and other vehicle(s), calculate the control inputs, and apply these to the throttle or brake pedal, and to the gearbox. The PID control law is given by

$$u_{\text{PID}}(k) \ \big(= b(j(k)) \cdot u_{\text{tb}} \big) = k_s \big(s_{\text{des}}(k) - s(k) \big) + k_v \big(v_{\text{des}}(k) - v(k) \big) + k_a \big(a_{\text{des}}(k) - a(k) \big) \quad \text{(3-9)}$$

Here, the desired states ($s_{\text{des}}(k)$, $v_{\text{des}}(k)$, and $a_{\text{des}}(k)$) are ideal states which serve to compare to the already fixed states ($s(k)$, $v(k)$, and $a(k)$) to calculate the state errors. These state errors are needed to determine—by applying the PID law above—how to adjust the input signal. One could call this a proportional (P) control law (or even PDD or PII), but it will be called a PID control law here, because of the three states that are fed back, one ($s(k)$) is actually the integral of one of the other states ($v(k)$), and one ($a(k)$) is the differential of this state $v(k)$. The output of the PID law gives the wheel traction force that is needed, which is the product of the input variable for the throttle/brakes ($u_{\text{tb}}(k)$) with $b(j(k))$, which in its turn is a function of the other input variable the gear. It is assumed here that a model of the vehicle dynamics, including a representation of gear dependency $b(j(k))$, like (3-1) is available.

Now the question is how to divide the value of $u_{\text{PID}}(k)$ over $b(j(k))$ and $u_{\text{tb}}(k)$. First the gear number is obtained by applying some sort of shifting scheme. Equation (2-16) showed how [11] divided the velocity domain into regions, each of which only allows for one specific gear. One problem with this approach is that the gear could be chattering up and down if the velocity is only slightly varying around a shifting value. To solve this problem, hysteresis can be added to such a shifting scheme, making the new gear value not only depend on velocity but also on the previous gear value. Another problem with this scheme is that it does not take into account the amount of wheel traction force demanded. That is, if the PID output ($u_{\text{PID}}(k)$) is high it is preferable to choose a lower gear than in case of a low PID output, since throttle input is cut off at 1 (physical limit) and a lower gear gives higher traction. Conversely choosing a higher gear is favoured in case of low PID output, because of lower fuel consumption rates. This is why conventional automatic transmission schemes also use the

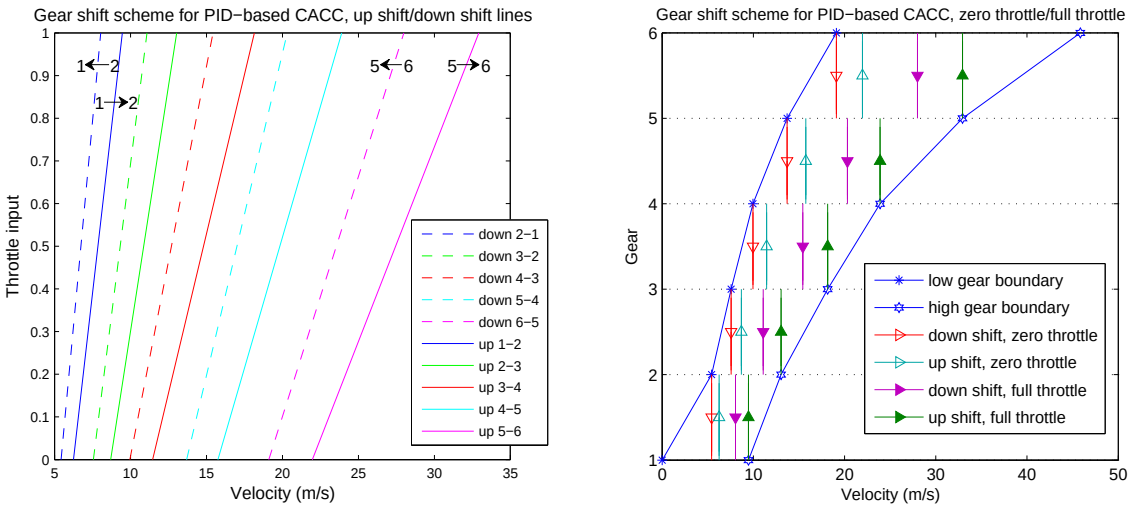


Figure 3-2: The gear scheduling algorithm as applied by the PID-based CACC controller, illustrated by two schematic plots. (The values are based on the model of a SMART, which is used in the case study of Chapter 4)

throttle input (usually exerted by a human driver) as input to the shifting scheme [27, 29, 35]. In these schemes the gear shift point is based on the velocity and the throttle input, with hysteresis added such that the up-shift point from gear $j^\star$ to $j^\star + 1$ lies at a higher velocity or lower throttle input than the down-shift from $j^\star + 1$ to $j^\star$ with the same throttle input or the same velocity respectively. The gear scheduling algorithm that will be applied here uses a simplified gear shift scheme based on these conventional automatic transmission schemes. Recall (from Section 3-2) that in the vehicle model (3-1) each gear is only allowed between its lower and upper velocity bounds, and that while in one gear the maximum achievable traction force (equal to the value of $b(j)$) stays constant. To make full use of these velocity regions, for each gear $j^\star$ the velocity for the up shift to gear $j^\star + 1$ while at full throttle ($u_{\text{tb}} = 1$) is set to the upper velocity bound for this gear ($v_{\text{high},j^\star}$). The up-shift velocity for zero throttle ($u_{\text{tb}} \leq 0$, so no throttle action, but possibly brake action) is set to a value 15 % lower, at $0.85 \cdot v_{\text{high},j^\star}$. The velocity for a shift down from $j^\star$ to $j^\star - 1$ while $u_{\text{tb}} \leq 0$ is set to the lower velocity bound for this gear ($v_{\text{low},j^\star}$). The velocity for this down shift with full throttle is set at a 15 % higher value[3] (at $1.15 \cdot v_{\text{low},j^\star}$). Figure 3-2 shows this scheme graphically.
Now the situation is such that the gear scheduling algorithm needs $u_{\text{tb}}(k)$ as an input, while to calculate $u_{\text{tb}}(k)$ the gear number is required. This is handled by first calculating $u_{\text{tb}}(k)$ assuming that the gear stays constant, according to

$$u_{\text{tb}}(k) = \frac{u_{\text{PID}}(k)}{b(j(k))} \tag{3-10}$$

Then (after, if necessary, having cut of the throttle input to try to obey the maximum velocity and maximum acceleration constraints, which will be explained below, around (3-11)) apply the gear scheduling algorithm with this value. Then, only if the gear changes, once more calculate u_{tb}, now with the new value of the gear. Since it is not preferred to have the gear change to much, these steps are not repeated again (which could cause an additional change

[3]No specific information could be found in literature on how to choose the precise shape and location of the gear shift lines. This is why here the lines in Figure 3-2a are chosen to be straight and the endpoints are chosen as explained in the text, where the 15 % approach is chosen to make these lines diverge for increasing throttle input (as they do in the literature).

in the gear), thereby keeping the gear change $\|\Delta j\| \leq 1$.

As explained before, the constraints from Table 3-1 should ideally be obeyed, and the throttle/brake signal should be between $u_{\text{tb}} = -1$ and $u_{\text{tb}} = 1$. A PID controller cannot anticipate such saturation and constraints, nor can it, in its basic form, obey them. The problem with no anticipation cannot be solved, and strictly obeying the constraints (from Table 3-1) in all situations is not achievable. However, some tricks can be applied to try to meet some of the constraints by approximation. Say, the situation is such that a constraint is violated and the PID control signal is expected to make the situation even worse. Then the initially computed control signal can be replaced by a signal with a magnitude that is expected to (partly) abolish the violation of the constraint for the next time step, as was done in [9] (see also Section 2-2-1). Assuming a continuous-time model is available, it can be used to calculate a replacing throttle/brake signal that aims at keeping the state (approximately) at the boundary for the violated constraint. So after the initial PID output is calculated, a few constraints are checked one by one in a specific order, after each check giving the control signal a new restrained value if necessary, according to

$$\begin{array}{llllll}
1. & \text{if } v(k) & \geq & v_{\max} & \longrightarrow & u_{\text{tb}} := \min\left(\frac{1}{b(j)}(cv_{\max}^2 + m\mu g), u_{\text{tb}}\right) \\
2. & \text{if } u_{\text{tb}} & > & u_{\text{tb}}(a_{\max}, k) & \longrightarrow & u_{\text{tb}} := u_{\text{tb}}(a_{\max}, k) \\
3. & \text{if } v(k) & \leq & v_{\min} = 0 & \longrightarrow & u_{\text{tb}} := \max\left(0, u_{\text{tb}}\right) \\
4. & \text{if } u_{\text{tb}} & < & u_{\text{tb}}(a_{\min}, k) & \longrightarrow & u_{\text{tb}} := u_{\text{tb}}(a_{\min}, k) \\
5. & \text{if } u_{\text{tb}} & < & u_{\text{tb,min}} = -1 & \longrightarrow & u_{\text{tb}} := -1 \\
6. & \text{if } u_{\text{tb}} & > & u_{\text{tb,max}} = 1 & \longrightarrow & u_{\text{tb}} := 1
\end{array} \tag{3-11a}$$

with

$$u_{\text{tb}}(a, k) = \frac{1}{b(j)}(ma + (cv(k)^2 + m\mu g)\text{sgn}(v(k))) \tag{3-11b}$$

where $u_{\text{tb}}(a, k)$ gives the pedal input at time step k that would theoretically result in the instantaneous acceleration a. This approach was inspired by [9] (see also Section 2-2-1). In rule 1 of (3-11a), for the case when the velocity has reached the maximum allowed velocity and the PID output tries to increase the velocity even further, the new control signal is calculated such that the acceleration a becomes zero. Rules 2 and 4 of (3-11a) try to (approximately) obey the acceleration constraints. The input is restricted by rules 5 and 6 by just cutting of the control signal value at ± 1.
In fact, the first two rules of (3-11a) are already applied once before the new gear is calculated. This is because these are the only constraint related rules that affect the throttle signal[4] ($u_{\text{tb}} > 0$). Then after the definite value $j(k)$ is obtained and (3-10) is applied, if the gear has changed, the first two rules are applied again (not if the gear stays constant). Then the remaining rules are applied.

Now for almost all constraints an algorithm has been designed that tries to keep from violating the constraint (strongly), except for one. The constraint $d \geq d_{\min}$ is one that cannot be properly dealt with. Namely this constraint exists because breaking it represents colliding with the vehicle in front, after which no control actions can be taken any more. Just as with

[4]The other rules that are related to constraints only affect the brake signal. The throttle input saturating rule 6 of (3-11a) should not be applied before the new gear is obtained, because the unsaturated value of u_{tb} represents how much traction is needed. The throttle (/brake) input saturation is only applied completely at the end of the CACC algorithm.

the other constraints, the PID algorithm cannot foresee breaking this one[5]. To avoid breaking this minimum distance constraint the desired distance should be made big enough such that, even with restricted negative acceleration, the vehicles will not collide.

3-3-3 CACC with MPC

MPC model

As explained in Section 2-2-2, at each control update time step the MPC determines the input sequence that minimises a certain performance index defined over a certain prediction period, and takes from this minimising sequence the input vector corresponding to the current time step (k) to apply to the system. To be able to predict future outputs resulting from applying a sequence of inputs to the system, the MPC algorithm uses a model of the system.

Because the CACC controller uses samples of the vehicle's states and other necessary variables, the controller will be acting in discrete time. Therefore, the model that the controller uses to predict the system behaviour must be in discrete time too. First the model from (3-1) will be written in state-space form, and then discretised. For the MPC application the position and velocity of the vehicle will be joined in one state vector:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} s \\ v \end{bmatrix} \tag{3-12}$$

The state-space formulation looks as follows,

$$\dot{x}(t) = \begin{bmatrix} x_2(t) \\ -\frac{c}{m}x_2^2(t) - \mu g \cdot \mathrm{sgn}(x_2(t)) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m}b(j(t)) \cdot u_{\mathrm{tb}}(t) \end{bmatrix} \tag{3-13}$$

Here u_{tb} as well as j are inputs, and from the original term $\frac{c}{m}x_2^2(t) \cdot \mathrm{sgn}(x_2)$ (from (2-14)) the "sgn" operation is omitted. This is because the minimum velocity constraint will prevent the velocity from becoming negative, and when the velocity is zero, making $\mathrm{sgn}(x_2)$ zero, x_2^2 is also zero. So omitting the $\mathrm{sgn}(x_2)$ factor from that term does not change anything within the operational region.

Now, the model will be discretised according to the Euler rule:

$$\dot{x}(kT_{\mathrm{s}}) \approx \tfrac{x(k+1)-x(k)}{T_{\mathrm{s}}} \quad \longrightarrow \quad x(k+1) \approx x(k) + T_{\mathrm{s}} \cdot \dot{x}(kT_{\mathrm{s}}) \tag{3-14}$$

where T_{s} represents the sampling time. This results in the discrete-time model

$$x(k+1) = \begin{bmatrix} x_1(k) + T_{\mathrm{s}}x_2(k) \\ x_2(k) - \frac{T_{\mathrm{s}}c}{m}x_2^2(k) - T_{\mathrm{s}}\mu g \cdot \mathrm{sgn}(x_2) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{T_{\mathrm{s}}}{m}b(j(k)) \cdot u_{\mathrm{tb}}(k) \end{bmatrix} \tag{3-15}$$

The model from (3-15) contains several nonlinearities (resulting from x_2^2, $b(j)$, and $\mathrm{sgn}(x_2)$), which makes it highly computationally expensive to be used in online MPC. This is because the optimisation problem to be solved within the MPC algorithm is nonconvex. For the vehicle to react quickly, the control signal should be determined and applied quickly. To significantly

[5]The difference with the other constraints is that even if they are violated, the vehicle can still drive and the CACC controller can try to keep the state close to the limit value (instead of significantly breaking the constraint). Contrarily to this, when breaking the minimum distance constraint, it is too late for any action, which is why no such rule as the ones from (3-11a) is designed for the minimum distance constraint.

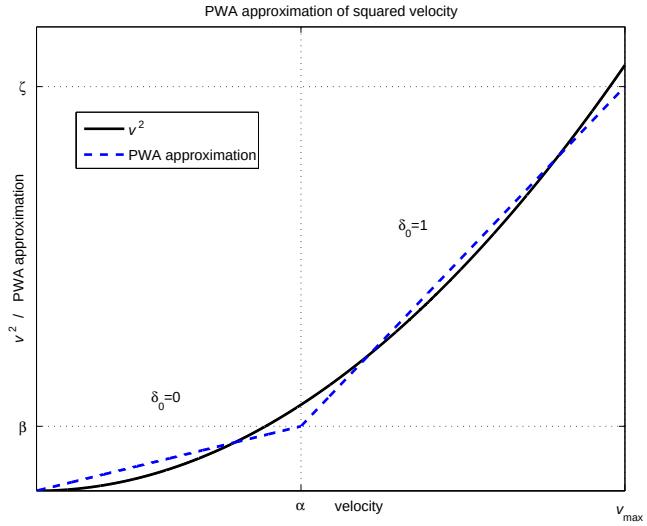


Figure 3-3: Piecewise affine approximation of the squared velocity, applied for the MPC model of the vehicle.

shorten the computation time, the strategy to approximate linearities by PWA functions and apply some other manipulations to obtain a mixed logical dynamical (MLD) model (2-26) is adopted from [11] (see also Section 2-2-2). However, some different choices are made, compared to [11], in the application of this strategy. The optimisation problem will then become a mixed integer linear programming (MILP) problem, reducing the optimisation running time.

The first linearisation is to approximate the nonlinear term x_2^2, representing the increase of the air drag with velocity, by a PWA function (see Figure 3-3). A PWA function consisting of two straight lines should give a fair trade-off between good approximation and low mathematical complexity. The optimal values for the parameters defining these lines are derived from an optimisation procedure to fit the lines as well as possible to the parabola between the minimum and maximum allowed velocities. While searching for these optimal values, the start of the first line is fixed on the origin. This is because any other value at zero velocity would represent air drag while the vehicle is standing still. This would cause the controller to still give unwanted throttle/brake signals unequal to zero to try to counteract this air drag in the prediction model, in the case where the vehicle should stay still. Contrarily to the approach of [11] (see Figure 2-5), it is chosen not to fix the point for $v_{\max}$ at $v_{\max}^2$, but to find the optimal value for it. This way a closer approximation can be achieved. Now, the expressions for the two lines (P_1, and P_2) of the PWA function can be given by:

$$\begin{array}{lll} P_1 & : & \frac{\beta}{\alpha}x_2, \qquad\qquad\qquad\quad x_2 \in [0, \alpha] \\ P_2 & : & \frac{\zeta-\beta}{v_{\max}-\alpha}(x_2-\alpha)+\beta, \quad x_2 \in [\alpha, v_{\max}] \end{array} \tag{3-16}$$

Because the system needs only to be modelled for velocities between the minimum and maximum allowed velocities, these values are used as the lower bound for the first region ($v_{\min} = 0$) and the upper bound for the second region ($v_{\max}$) respectively. The optimal values for the unknown parameters that define the two lines can be found by minimising the mean squared error between the PWA approximation and x_2^2. These optimal values are

$$\alpha = (-2+\sqrt{6}) \cdot v_{\max}, \quad \beta = (7\frac{1}{2} - 3\sqrt{6}) \cdot v_{\max}^2, \quad \zeta = (-1\frac{1}{2} + \sqrt{6}) \cdot v_{\max}^2 \tag{3-17}$$

The term $x_2(k) - \frac{T_s c}{m} x_2^2(k)$ from (3-15) can now be replaced similarly as was shown in (2-21)

according to

$$\begin{aligned} x_2(k) - \tfrac{T_s c}{m} x_2^2(k) \quad &\rightarrow \quad (1 - \tfrac{T_s c \beta}{m\alpha}) \cdot x_2(k) \;=\; a_1 x_2(k) \qquad \iff \quad v_{\min} \leq x_2(k) \leq \alpha \\ x_2(k) - \tfrac{T_s c}{m} x_2^2(k) \quad &\rightarrow \quad (1 - \tfrac{T_s c(\zeta-\beta)}{m(v_{\max}-\alpha)}) \cdot x_2(k) + \\ &\qquad \tfrac{T_s c \alpha(\zeta-\beta)}{m(v_{\max}-\alpha)} - \tfrac{T_s c \beta}{m} = a_2 x_2(k) + r \quad \iff \quad \alpha < x_2(k) \leq v_{\max} \end{aligned} \tag{3-18}$$

where the constants a_1, a_2, and r are introduced here for clarity. Now a dummy binary variable $\delta_0(k) \in \{0, 1\}$ will be defined as follows:

$$\begin{aligned} \delta_0(k) = 0 \quad &\iff \quad x_2(k) \leq \alpha \\ \delta_0(k) = 1 \quad &\iff \quad x_2(k) > \alpha \end{aligned} \tag{3-19}$$

This definition as such cannot be implemented into the MLD model, or anywhere else in the MPC algorithm. Therefore, using (2-23) (from [3]), the definition of δ_0 will be rewritten into the form of the system of inequalities

$$\begin{aligned} x_2(k) \quad &\geq \quad (\alpha + \varepsilon)\delta_0(k) + (1 - \delta_0(k))v_{\min} = (\alpha + \varepsilon)\delta_0(k) \\ x_2(k) \quad &\leq \quad \alpha(1 - \delta_0(k)) + \delta_0(k) v_{\max} \end{aligned} \tag{3-20}$$

where $\varepsilon > 0$ has a very small value (as explained in Section 2-2-2, e.g., $\varepsilon = 10^{-14}$). The introduction of the variable $\delta_0(k)$ makes it possible to join the two expressions from (3-18) into a single expression valid on the whole velocity domain:

$$\begin{aligned} x_2(k) - \tfrac{T_s c}{m} x_2^2(k) \quad \rightarrow \quad &(1 - \delta_0(k)) a_1 x_2(k) + \delta_0(k)(a_2 x_2(k) + r) \\ &= a_1 x_2(k) + (a_2 - a_1)\delta_0(k) x_2(k) + r\delta_0(k) \end{aligned} \tag{3-21}$$

This expression contains the product of a binary variable and a state variable ($\delta_0(k) \cdot x_2(k)$), which makes the expression still nonlinear. Replacing this product by the new auxiliary variable $z_0(k) = \delta_0(k) x_2(k)$, and alternatively expressing this definition by a system of inequalities, removes the nonlinearity. This system of inequalities, derived from (2-24) is given by

$$\begin{aligned} z_0(k) \quad &\leq \quad \delta_0(k) v_{\max} \\ z_0(k) \quad &\geq \quad \delta_0(k) v_{\min} = 0 \\ z_0(k) \quad &\leq \quad x_2(k) - v_{\min}(1 - \delta_0(k)) = x_2(k) \\ z_0(k) \quad &\geq \quad x_2(k) - v_{\max}(1 - \delta_0(k)) \end{aligned} \tag{3-22}$$

In the state-space model (3-15) there is another nonlinearity that should be taken care of, which is the term $T_s \mu g \cdot \operatorname{sgn}(x_2(k))$. To get rid of this nonlinearity, a new binary variable $\delta_\mu(k) \in \{0, 1\}$ is introduced, with $\delta_\mu(k) = 0 \iff x_2(k) = 0$, and $\delta_\mu(k) = 1 \iff 0 < x_2(k) \leq v_{\max}$. This can be put into the inequalities

$$\begin{aligned} x_2(k) \quad &\geq \quad \varepsilon \delta_\mu(k) \\ x_2(k) \quad &\leq \quad v_{\max} \delta_\mu(k) \end{aligned} \tag{3-23}$$

Now the term $T_s \mu g \cdot \operatorname{sgn}(x_2(k))$ can be replaced by $T_s \mu g \delta_\mu(k)$. After the introduction of the variables $\delta_0(k)$, $z_0(k)$, and $\delta_\mu(k)$, the new, now partly linearised, state space model becomes

$$\begin{aligned} x(k+1) = \begin{bmatrix} 1 & T_s \\ 0 & a_1 \end{bmatrix} \cdot x(k) + \begin{bmatrix} 0 \\ \frac{T_s}{m} b(j(k)) \cdot u_{\mathrm{tb}}(k) \end{bmatrix} \\ + \begin{bmatrix} 0 & 0 \\ r & -T_s \mu g \end{bmatrix} \cdot \begin{bmatrix} \delta_0(k) \\ \delta_\mu(k) \end{bmatrix} + \begin{bmatrix} 0 \\ a_2 - a_1 \end{bmatrix} \cdot z_0(k) \end{aligned} \tag{3-24}$$

As was done in [11], the gear is encoded by binary variables according to (2-17), for convenience shown again here:

$$j(k) = 1 + \delta_1(k) + 2\delta_2(k) + 4\delta_3(k) \tag{3-25}$$

where $\delta_1(k)$, $\delta_2(k)$, $\delta_3(k) \in \{0, 1\}$. Now these three binary variables become control inputs for the MPC model instead of j, which can easily be calculated from these binary variables using (3-25). The constraints that restrict the gear to stay between 1 and 6 are given by (2-30). Recall that each gear number can only be chosen between corresponding velocity bounds. For each gear number $j^\star$, this restriction can be given by a pair of inequalities as follows:

$$j(k) = j^\star \;\rightarrow\; x_2(k) \geq v_{\text{low},j^\star} \;\text{ and }\; x_2(k) \leq v_{\text{high},j^\star} \tag{3-26}$$

This can alternatively be represented by a pair of inequalities involving $x_2(k)$ and the binary variables $\delta_1(k)$, $\delta_2(k)$, and $\delta_3(k)$. Doing this for all gears results in a total of twelve inequalities. Here, e.g., the inequality pair corresponding to the second gear looks like

$$\begin{array}{rcl} x_2 & \geq & \delta_1(1-\delta_2)(1-\delta_3)\cdot v_{\text{low},2} + \big(1-\delta_1(1-\delta_2)(1-\delta_3)\big)\cdot v_{\min} \\ x_2 & \leq & \delta_1(1-\delta_2)(1-\delta_3)\cdot v_{\text{high},2} + \big(1-\delta_1(1-\delta_2)(1-\delta_3)\big)\cdot v_{\max} \end{array} \tag{3-27}$$

where $v_{\min} = 0$ is the minimum allowed velocity, $v_{\max}$ is the maximum allowed velocity, and the dependence on k has been omitted for clarity. For the other ten inequalities the reader is referred to Appendix A-1. These inequalities contain several instances of the products $\delta_2(k)\delta_3(k)$, and $\delta_1(k)\delta_2(k)\delta_3(k)$. However, as can be seen from Table 2-2, within the operating region, $\delta_2(k)$, and $\delta_3(k)$ are never both equal to 1 simultaneously. Therefore, these products are always zero, and can be omitted from the inequalities. The inequalities also contain the products $\delta_1(k)\delta_2(k)$, and $\delta_1(k)\delta_3(k)$, which makes them nonlinear. Therefore, these products will be replaced by new binary dummy variables, according to

$$\delta_4(k) = \delta_1(k)\delta_2(k) \tag{3-28a}$$

$$\delta_5(k) = \delta_1(k)\delta_3(k) \tag{3-28b}$$

the definition of which will alternatively be expressed by systems of inequalities by applying (2-25) (see Appendix A-1).

The final nonlinearity to linearise is the dependence of the traction force factor on the gear ($b(j)$), which is shown in Figure 3-4. When approximating this relation, the final point (value for $j = 6$) should be fixed at the original point ($b(6)$). This is because otherwise it will lie below that, causing the controller to give higher throttle inputs when close to $v_{\max}$. This can then result in the velocity exceeding its maximum allowed value with the MPC scheme not being able to find an input that theoretically would decrease the velocity to below its upper bound before the next sample step. That would then result in an infeasible optimisation problem for the MPC controller[6]. A possible linearisation approach is to approximate this relation $b(j)$ by one straight line, as was done by [11] (see (2-18)). However, this results in a significant model mismatch. A second possible approach, which gives smaller model errors compared to the first approach, is to perform a two-region-PWA approximation, similar to what was done for the approximation of the squared velocity in Figure 3-3. A third approach, which employs the fact that the gear is a (already binary decoded) discrete variable, is to keep the six values of $b(j)$, and make a six-region-PWA approximation, as it were. For each of these areas the respective line is just reduced to one point that lies on $b(j)$. The arithmetics of applying these three approaches are shown in Appendix A-2. It turns out that all three approaches

[6] Applying experiments with $b(j)$ approximated such that the final point has a lower value than $b(6)$ indeed resulted in the infeasible situation described when trying to maintain the velocity at (approximately) $v_{\max}$.

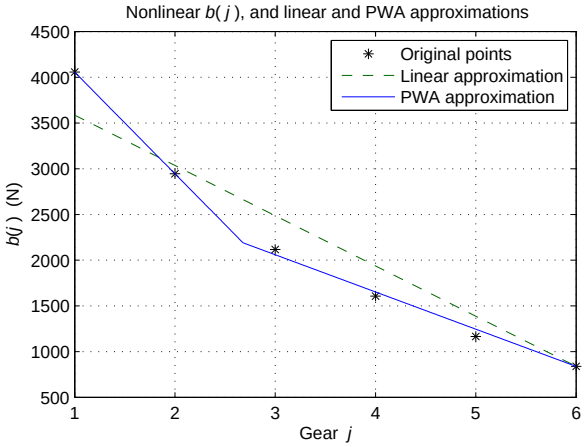


Figure 3-4: The gear dependence factor (equal to traction force at maximum pedal input) is nonlinear with the gear. Values are derived from the vehicle model of the SMART, which is used later on for the simulation setup (in Section 4-1). Approximations (not used, see text) are obtained with mean squared error minimisation method.

can be applied without the need to introduce new binary dummy variables. However, all require the introduction of three new continuous auxiliary variables, $z_1(k) = \delta_1(k)u_{\mathrm{tb}}(k)$, $z_2(k) = \delta_2(k)u_{\mathrm{tb}}(k)$, and $z_3(k) = \delta_3(k)u_{\mathrm{tb}}(k)$, and both the second and third approach require two more of such variables, $z_4(k) = \delta_4(k)u_{\mathrm{tb}}(k)$ and $z_5(k) = \delta_5(k)u_{\mathrm{tb}}(k)$. It is decided to apply the third approach, because this approximation is better than the second one, and compared with the first approach the approximation is so much better, that it should outweigh the computational demands of the two additional real-valued variables. The three binary variables that encode the gear can be used to specify each of the six areas of the PWA function, which are actually the six (discrete) gears. This way $b(j)$ can be replaced by

$$\begin{aligned} b_\delta(\delta_1,\delta_2,\delta_3) = {} & (1-\delta_1)(1-\delta_2)(1-\delta_3)b_1 + \delta_1(1-\delta_2)(1-\delta_3)b_2 \\ & + (1-\delta_1)\delta_2(1-\delta_3)b_3 + \delta_1\delta_2(1-\delta_3)b_4 + (1-\delta_1)(1-\delta_2)\delta_3 b_5 + \delta_1(1-\delta_2)\delta_3 b_6 \end{aligned} \tag{3-29}$$

where $b_1, \ldots, b_6$ represent the respective values of the traction force gear dependence factor $b(j)$ for each gear, and for ease of notation the dependence of all the binary variables (and j) of k is omitted (as will be done in the following lines). After substituting for δ_4 and δ_5 (from (3-28)) and expanding, the function becomes

$$\begin{aligned} b_\delta(\delta_1,\delta_2,\delta_3,\delta_4,\delta_5) = {} & (b_2-b_1)\delta_1 + (b_3-b_1)\delta_2 + (b_5-b_1)\delta_3 \\ & + (b_1-b_2-b_3+b_4)\delta_4 + (b_1-b_2-b_5+b_6)\delta_5 + b_1 \end{aligned} \tag{3-30}$$

After replacing $b(j(k))$ in (3-24) by $b_\delta(\delta_1(k),\delta_2(k),\delta_3(k),\delta_4(k),\delta_5(k))$, the vehicle model will contain the product $b_\delta(\delta_1(k),\delta_2(k),\delta_3(k),\delta_4(k),\delta_5(k)) \cdot u_{\mathrm{tb}}(k)$, which means it contains products between binary (input and dummy) variables and the throttle/brake input $u_{\mathrm{tb}}(k)$. This makes the model still nonlinear. To linearise these nonlinearities, five continuous auxiliary variables are introduced, defined as

$$\begin{aligned} & z_1(k) = \delta_1(k)u_{\mathrm{tb}}(k), \quad z_2(k) = \delta_3(k)u_{\mathrm{tb}}(k), \quad z_3(k) = \delta_3(k)u_{\mathrm{tb}}(k), \\ & z_4(k) = \delta_4(k)u_{\mathrm{tb}}(k), \quad z_5(k) = \delta_5(k)u_{\mathrm{tb}}(k) \end{aligned} \tag{3-31}$$

each of which will alternatively be expressed by a system of inequalities resulting from applying (2-24) (see (A-16) in Appendix A-2).

After applying the above, the model can easily be written in the form of an MLD model as was given by (2-26), with

$$u(k) = \begin{bmatrix} u_{\text{tb}}(k) \\ \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \end{bmatrix}, \quad \delta(k) = \begin{bmatrix} \delta_0(k) \\ \delta_4(k) \\ \delta_5(k) \\ \delta_\mu(k) \end{bmatrix}, \quad z(k) = \begin{bmatrix} z_0(k) \\ z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix}, \tag{3-32}$$

which is demonstrated in Appendix A-3. All the inequalities that alternatively express the definition of (continuous and binary) dummy variables together with the constraints that express the physical limits on the throttle brake inputs,

$$u_{\text{tb}}(k) \geq u_{\text{tb,min}} = -1, \quad u_{\text{tb}}(k) \leq u_{\text{tb,max}} = 1, \tag{3-33}$$

are gathered into one big inequality that forms the second line of the MLD model formulation of (2-26).

MPC optimisation problem

The MPC controller task for time step k can be formulated as solving the following problem:

$$\min_{\tilde{u}_{\text{tb}}(k), \tilde{j}(k)} \; J(x(k), \tilde{u}_{\text{tb}}(k), \tilde{j}(k))$$

$$\begin{aligned} \text{s.t.} \quad & \text{eq. (2-26)}, \\ & d_{\min} \leq d(k), \\ & 0 \leq x_2(k) \leq v_{\max}, \\ & a_{\min}(k) \leq a(k) \leq a_{\max} \end{aligned} \tag{3-34a}$$

with

$$\begin{aligned} J(x(k), \tilde{u}_{\text{tb}}(k), \tilde{j}(k)) = \sum_{n=1}^{N} \Big(& \left\| Q(x(k+n) - x_{\text{des}}(k+n)) \right\|_1 \\ & + \left\| R_{u_{\text{tb}}} \cdot \Delta u_{\text{tb}}(k+n-1) \right\|_1 + \left\| R_j \cdot \Delta j(k+n-1) \right\|_1 \Big) \end{aligned} \tag{3-34b}$$

and

$$\begin{aligned} \Delta u_{\text{tb}}(k+n) &= u_{\text{tb}}(k+n) - u_{\text{tb}}(k+n-1) \\ \Delta j(k+n) &= j(k+n) - j(k+n-1) \\ \tilde{u}_{\text{tb}}(k) &= [u_{\text{tb}}(k), \ldots, u_{\text{tb}}(k+N-1)]^{\text{T}} \\ \tilde{j}(k) &= [j(k), \ldots, j(k+N-1)]^{\text{T}} \\ d(k) &= x_{1,i-1}(k) - x_1(k) - l_{\text{veh}} \end{aligned} \tag{3-34c}$$

Here a represents acceleration, $x_{1,i-1}$ the position of the direct predecessor, the desired state vector at time step k, $x_{\text{des}}(k) = [s_{\text{des}}(k), v_{\text{des}}(k)]^{\text{T}}$, is derived as described in Section 3-3-1, and $Q \in \mathbb{R}^{2\times 2}$, $R_{u_{\text{tb}}} \in \mathbb{R}$, and $R_j \in \mathbb{R}$ are weights. The performance index J here has the same form as the one used by [11], given by (2-29). Minimising this index means trying to find a trade-off between following the desired states, and keeping the variation of both the inputs low. The second part is to make the driving experience feel more smoothly. Here 1-norms are chosen, instead of, e.g., 2-norms, because 1-norms can be recast as PWA functions, as can be seen later on, making the optimisation problem less complex.
It is chosen to set the control update frequency equal to the sampling frequency. Hence, each

time step k represents a sample step as well as a control update step.
The linearised model (as derived above) should be used to predict future states as function of present and (theoretical) future inputs. It is preferable to write the whole optimisation problem in the following form, such that it can be solved by an MILP optimisation algorithm,

$$\min_{y} \; M^{\mathrm{T}} \cdot y, \quad \text{s.t. } E' \cdot y \leq F \tag{3-35}$$

where $y = [y_{\mathrm{c}}, y_{\mathrm{b}}]^{\mathrm{T}}$ with $y_{\mathrm{c}} \in \mathbb{R}^{m_{\mathrm{c}}}$, $y_{\mathrm{b}} \in \{0,1\}^{m_{\mathrm{b}}}$, M, and F are column vectors, and E' is a matrix. The problem statement as given in (3-34) needs to be rewritten in several steps to obtain the MILP form of (3-35).

First, the predicted future states can be written as function of the current state and current and future inputs, according to

$$\tilde{x}(k) = A^* x(k) + B_1^* \tilde{u}(k) + B_2^* \tilde{\delta}(k) + B_3^* \tilde{z}(k) \tag{3-36a}$$

with

$$\tilde{x}(k) = \begin{bmatrix} \hat{x}(k+1) \\ \vdots \\ \hat{x}(k+N) \end{bmatrix}, \quad \tilde{u}(k) = \begin{bmatrix} u(k) \\ \vdots \\ u(k+N-1) \end{bmatrix} \tag{3-36b}$$

and $\tilde{\delta}(k)$ and $\tilde{z}(k)$ defined similarly, which is shown in Appendix A-4, where also the definition of the matrices A^*, B_1^*, B_2^*, and B_3^* can be found. Here, $\hat{x}(k+n)$ represents the predicted state at time step $k+n$, resulting from theoretically applying inputs $u(k)$ to $u(k+n-1)$. Equation (3-36) is the prediction form of the first line of the MLD model (2-26), which means that it represents this line of the model for each of the time steps within the prediction horizon. By applying the prediction form of the variables in the second line of (2-26) and expanding the matrices $E_1, \ldots, E_5$ accordingly, the part of the MLD model that is represented by inequality constraints can be rewritten in prediction form. The constraints from the MPC control problem formulation (3-34) should also be written in prediction form, and combined with the rest of the inequalities (in prediction form) to construct one big inequality. First it should be noted that, since the acceleration is not used as a state, the acceleration constraint from (3-34) needs to be rewritten, as follows

$$a_{\min} T_{\mathrm{s}} \leq x_2(k+1) - x_2(k) \leq a_{\max} T_{\mathrm{s}} \tag{3-37}$$

where T_{s} is the sampling time. The general inequality in prediction form looks like this:

$$E_1^* \cdot \tilde{u}(k) + E_2^* \cdot \tilde{\delta}(k) + E_3^* \cdot \tilde{z}(k) \leq E_4^* \cdot \tilde{x}(k) + E_5^* \tag{3-38}$$

where the derivation of the matrices $E_1^*, \ldots, E_5^*$ can be found in Appendix A-4.

In order to write the problem of minimising the performance index of (3-34) into the MILP form of (3-35), this index should first be formulated as function of $\tilde{x}$ and $\tilde{u}$. Doing this, it can be shown that J in (3-34) is equivalent to[7]

$$\begin{aligned} J(\tilde{u}(k), \tilde{x}(k), k) &= \|Q^* \tilde{x}(k) + Q_{xr}^*(k)\|_1 + \|R_u^* \tilde{u}(k) + R_{up}^*(k)\|_1 + \|R_j^* \tilde{u}(k) + R_{jp}^*(k)\|_1 \\ &= \|M_{\mathrm{perf}}(\tilde{u}(k), \tilde{x}(k), k)\|_1, \\ &\quad \text{with } M_{\mathrm{perf}}(\tilde{u}(k), \tilde{x}(k), k) = \begin{bmatrix} Q^* \tilde{x}(k) + Q_{xr}^*(k) \\ R_u^* \tilde{u}(k) + R_{up}^*(k) \\ R_j^* \tilde{u}(k) + R_{jp}^*(k) \end{bmatrix} \end{aligned} \tag{3-39}$$

[7] The full notation of the performance index $J(\tilde{u}(k), \tilde{x}(k), k)$ in (3-39) would be $J(\tilde{u}(k), \tilde{x}(k), Q_{xr}^*(k), R_{up}^*(k), R_{jp}^*(k))$, but since $Q_{xr}^*(k)$, $R_{up}^*(k)$, and $R_{jp}^*(k)$ are known at time step k before this performance index is minimised, and depend on k, these variables are omitted from this notation and instead just k is added as a separate variable.

The definition of the matrices $Q^* \in \mathbb{R}^{2N\times 2N}$, $R_u^* \in \mathbb{R}^{N\times 4N}$, $R_j^* \in \mathbb{R}^{N\times 4N}$, and the vectors $Q_{xr}^*(k)$, $R_{up}^*(k)$, $R_{jp}^*(k)$ is given in Appendix A-5. The prediction of the states of the other vehicle(s) used as reference is contained in $Q_{xr}^*(k)$. As described in Section 3-1-1, with MPC for each configuration (I, II, and III) two variants will be designed. One variant has the vehicles (except the leader) only communicate their current states. Then the future states of the other vehicle(s) will be predicted assuming their velocity stays constant. The second variant has the vehicles communicate their predicted states from the previous time step, corresponding to their optimising input vector. In that case these predicted states, received through communication, will be used for the reference sequence by the CACC controllers of vehicles that receive this information. These communicated future states were predicted one time step before they are used by the controller of the receiving vehicles. This means that the final prediction over the prediction horizon (corresponding to time step $k+N$) of the states of the other vehicles (except the platoon leader) needed as reference for the MPC algorithm is not received through communication. This final reference value is gained from predicting the states of the other vehicle(s) assuming their respective velocities stay constant from time step $k+N-1$ to $k+N$. Only the platoon leader's states are received for the entire prediction horizon. This is because with both MPC variants the platoon leader communicates its current and future states up to N time steps into the future, which are assumed to be known by the leader.

In order to be able to minimise this performance index, the 1-norm should be eliminated. This can be done by applying the fact that the problem

$$\min_{\theta\in\mathbb{R}^m} \|\theta\|_1 \quad \text{s.t. } A\theta \leq b \tag{3-40a}$$

is equivalent to the linear problem

$$\min_{\rho,\theta\in\mathbb{R}^m} \sum_{i=1}^{m} \rho_i, \quad \text{s.t. } \rho_i \geq \mid \theta_i \mid \text{ and } A\theta \leq b \tag{3-40b}$$

So the optimal solution of (3-40a) is equal to the optimal solution of (3-40b). For the problem of minimising the performance index of (3-39), this means

$$\left.\begin{array}{r} \min\limits_{\tilde{u}(k),\tilde{x}(k),\tilde{\delta}(k),\tilde{z}(k)} J(\tilde{u}(k),\tilde{x}(k),k) \\ \text{s.t. (3-36), and (3-38)} \end{array}\right\} \equiv \left\{\begin{array}{ll} \min\limits_{\tilde{u}(k),\tilde{x}(k),\tilde{\delta}(k),\tilde{z}(k),\rho(k)} & \sum\limits_{i=1}^{m} \rho_i(k) \\ \text{s.t.} & \text{eqns. (3-36) and (3-38),} \\ & \rho(k) \geq M_{\text{perf}}(\tilde{u}(k),\tilde{x}(k),k), \\ & \rho(k) \geq -M_{\text{perf}}(\tilde{u}(k),\tilde{x}(k),k) \end{array}\right. \tag{3-41}$$

where $M_{\text{perf}} \in \mathbb{R}^m$ with $m = 4N$, and $\rho = [\rho_1, \ldots, \rho_m]^{\text{T}} \in \mathbb{R}^m$ is a vector of new dummy variables[8].

Now, the inequality constraints that have been generated in (3-41) should be combined with the other inequality constraints from (3-38), which results in

$$E_{1,\rho}\cdot\tilde{u}(k) + E_{2,\rho}\cdot\tilde{\delta}(k) + E_{3,\rho}\cdot\tilde{z}(k) + E_{6,\rho}\rho(k) \leq E_{4,\rho}\cdot\tilde{x}(k) + E_{5,\rho}(k) \tag{3-42}$$

The definitions of the vector and matrices in this equation are given by (A-29). Now the expression for $\tilde{x}(k)$ from (3-36) can be substituted for $\tilde{x}(k)$ in (3-42). After some rearranging, this results in

$$E_{1,\text{subs}}\cdot\tilde{u}(k) + E_{2,\text{subs}}\cdot\tilde{\delta}(k) + E_{3,\text{subs}}\cdot\tilde{z}(k) + E_{6,\rho}\rho(k) \leq E_{5,\text{subs}}(k) \tag{3-43}$$

[8] M_{perf} has length $m = 2N + N + N = 4N$. The first $2N$ corresponds to the predicted states up to $k+N$, the next N corresponds to the throttle/brake input sequence over the prediction horizon, the final N corresponds to the gear sequence over the prediction horizon (See also (3-39)).

where $E_{1,\text{subs}}, \ldots, E_{5,\text{subs}}$ are defined accordingly (shown in Appendix A-6)[9]. This inequality now gives the linear inequality constraints which together with the linear objective function (being $\rho(k)$) to be minimised, form the MILP problem that needs to be solved at control update time step k (see Appendix A-6 for the exact formulation of this MILP problem).

The mixed-integer linear and quadratic programming solver TOMLAB /CPLEX can be applied to solve this problem [23]. The optimisation results in finding an optimiser $y_{\text{opt}}(k) = [\tilde{u}^{\text{T}}_{\text{opt}}(k), \tilde{\delta}^{\text{T}}_{\text{opt}}(k), \tilde{z}^{\text{T}}_{\text{opt}}(k), \rho^{\text{T}}_{\text{opt}}(k)]^{\text{T}}$ for the problem in the form of (3-35). The starting point for the search will be the previous optimiser, where the entries corresponding to time step $k-1$ are omitted and the entries corresponding to $k+N$ are taken equal to the entries corresponding to $k+N-1$. The first four entries of the optimising vector $y_{\text{opt}}(k)$ represent the first input of the optimising input sequence ($u_{\text{opt}}(k) = [u_{\text{tb,opt}}(k), \delta_{1,\text{opt}}(k), \delta_{2,\text{opt}}(k), \delta_{3,\text{opt}}(k)]^{\text{T}}$). The value of $u_{\text{tb,opt}}(k)$ will be given as throttle/brake pedal position input at the current control update time step ($u_{\text{tb}}(k)$). The gear choice at this time step will be $j_{\text{opt}}(k) = 1 + \delta_{1,\text{opt}}(k) + 2\delta_{2,\text{opt}}(k) + 4\delta_{3,\text{opt}}(k)$. The predicted output sequence for time steps $k+1$ to $k+N$ can be calculated from substituting the optimal prediction vectors from $y_{\text{opt}}(k)$ into the prediction model of (3-36).

3-4 Platoon performance measures for tuning and evaluation

The tunable parameters of each controller will be tuned for a series of typical everyday traffic scenarios, covering a major part of the range of possible types of traffic situations. The goal here is to tune the controllers such that the intervehicle distances are as small as possible while safe at all time, and the performance requirements as described in Section 3-1-2 are satisfied. The tuning will be achieved by optimising a measure of the performance of a simulated platoon of vehicles using the same CACC controller. The same measure, which will be called the platoon performance measure, will later be used to evaluate the tuned controllers.

For tuning and evaluation, as will be done in the case study of Chapter 4, preferably the following four typical traffic scenarios described in general terms will be used, which cover most of the vehicle following situations.

a. The initial distance between the platoon leader and its direct follower (second vehicle) is significantly greater than the desired distance. The initial velocity of the second vehicle may be different from the leader's velocity. The other vehicles in the platoon are initially following at steady state. The leader keeps its velocity constant at its initial value.
b. From initial steady following at some velocity[10], the platoon leader accelerates to a higher velocity, after which its velocity stays constant.
c. From initial steady following at some velocity, the platoon leader decelerates to a lower velocity, after which its velocity stays constant.
d. From initial steady following at some velocity, the platoon leader decelerates to a halt, after which it keeps standing still.

[9] $E_{5,\text{subs}}(k)$ depends on k (only) because it contains the current state $x(k)$, so at one time step k it is not a variable, but just a constant vector (after the value of $x(k)$ has been inserted).

[10] Steady following means that a steady state has been reached. In other words, as long as the platoon leader does not change its velocity, the intervehicle distances and the velocity of all the vehicles will stay constant as well. Thus, initial steady following means that the initial intervehicle distances and vehicles' states are such that, for the specific CACC controller used, the platoon is initially at steady-state.

e. Initially all vehicles are standing still at a proper intervehicle distance[11]. Then the platoon leader accelerates to some velocity, after which it keeps its velocity constant.

Also, the leader's acceleration should obey the comfort bounds given for the CACC-controlled vehicles. Furthermore, the leader's trajectory should obey the limits that a vehicle would encounter when simulated with the simulation model used for the CACC-controlled vehicles (i.e., limited acceleration feasibilities for higher velocities). To get a representative measure of the general performance of the controller, the series of traffic scenarios to be actually used for tuning (and evaluation) should at least cover variations of the scenarios above. The more variations of scenarios are applied, and the more vehicles are in the platoon, the better the general performance of the controller can be evaluated. It can be chosen, for during tuning, to let the acceleration of the leader in some of the scenarios slightly cross the acceleration comfort bounds, or the acceleration feasibility limits valid for the CACC-controlled vehicles[12]. This to give the tuned CACC controllers of the followers a little more robustness especially against strong decelerations.

The platoon performance measure combines assessment of performance at platoon level (string stability) with assessment of performance at vehicle level (e.g., throttle/brake input variation). To calculate the platoon performance measure, implementations of the traffic scenarios above should be simulated subsequently as subscenarios to form one scenario, and from the result, information on several performance-indicating elements should be extracted. The choice of these elements is based on the performance requirements that are formulated in Section 3-1-2. These elements are intervehicle distances, amount of gear switches, changes in pedal position, amount and severity of constraint violations, and string stability (margin). The information on each of these elements is averaged over all vehicles and over time, and normalised (based on implementation in simulations) to obtain values for the different elements that are of the same order of magnitude. Then each element is given a weight that represents the importance of optimising this particular element. Finally a weighted sum of the element-specific platoon performance measures is taken, which gives a measure of the overall platoon performance. The computation of the element specific platoon performance measure $P_{\{\text{element}\}}$ for the different elements is shown into detail now.

- Intervehicle distance:

$$\begin{aligned} d_{\text{av}} &= \frac{\int_0^{t_{\text{end}}} \left(s_1(t) - s_{N_{\text{veh}}}(t)\right) dt}{t_{\text{end}}(N_{\text{veh}}-1)} - l_{\text{veh}} \\ P_d &= \frac{d_{\text{av}}}{d_{\text{av,norm}}} \end{aligned} \tag{3-44}$$

where N_{veh} is the total number of vehicles in the platoon, t_{end} is the ending time of the simulation, d_{av} is the average head-to-tail intervehicle distance, and $d_{\text{av,norm}}$ is a normalising factor[13].

[11] When it is stated that the vehicles are standing still at a proper intervehicle distance, this means that the distance between each pair of subsequent vehicles is the desired distance for zero velocity, i.e., d_0.

[12] For higher gears the gear dependence factor $b(j)$ is lower, limiting the acceleration and deceleration feasibilities of the vehicles. The effect of this overpowers the effect of the increase of air drag on deceleration, and adds to it for positive acceleration.

[13] The normalising factors in (3-44)-(3-46) ($d_{\text{av,norm}}$, $u_{\text{tb,av1.5,norm}}$, and $j_{\text{av1.5,norm}}$) should be chosen such that for the types of traffic scenarios as will be used for tuning and evaluation of the controllers, and for acceptably tuned controllers, the values of the respective performance measures (P_d, $P_{u_{\text{tb}}}$, and P_j) are (based on simulations) expected to be in the order of 1.

- Pedal position variation:

$$\begin{aligned} \Delta u_{\text{tb,av1.5}} &= \frac{\sum_{i=2}^{N_{\text{veh}}} \left(\sum_{k=1}^{k_{\text{end}}} \mid u_{\text{tb},i}(k) - u_{\text{tb},i}(k-1) \mid^{1.5} / t_{\text{end}} \right)^{1/1.5}}{(N_{\text{veh}}-1)} \\ P_{u_{\text{tb}}} &= \frac{\Delta u_{\text{tb,av1.5}}}{u_{\text{tb,av1.5,norm}}} \end{aligned} \tag{3-45}$$

  where i is vehicle number, t_{end} is the ending time of the simulation, k is the sample step counter (with sampling time T_{s}, $k = 0$ at initial situation), k_{end} is the number of the final sample step (at t_{end}), $u_{\text{tb},i}(k)$ is the value of the discrete-time throttle/brake signal for vehicle i at time step k, and $u_{\text{tb,av1.5,norm}}$ is a normalising factor. Instead of the plain average (over vehicles and time) of the absolute change in pedal position input over all (CACC-controlled) vehicles, the power 1.5 is applied to the changes and the 1.5-power-root is taken. This is to cause sudden big changes to be punished more severely than the same change spread over more time steps, but not as severely as would be the case if the root mean square (power 2) was taken.

- Gear variation:

$$\begin{aligned} \Delta j_{\text{av1.5}} &= \frac{\sum_{i=2}^{N_{\text{veh}}} \left(\sum_{k=1}^{k_{\text{end}}} \mid j_i(k) - j_i(k-1) \mid^{1.5} / t_{\text{end}} \right)^{1/1.5}}{(N_{\text{veh}}-1)} \\ P_j &= \frac{\Delta j_{\text{av}}}{j_{\text{av1.5,norm}}} \end{aligned} \tag{3-46}$$

  where $j_i(k)$ is the discrete-time gear choice for vehicles i at time step k, and $j_{\text{av1.5,norm}}$ is a normalising factor. The reasoning behind this equation is similar as for (3-45).

- String stability:
  When an (online) MPC controller is applied, a closed-loop frequency domain transfer function of the vehicle with the controller, as used in (2-33) to determine string stability, cannot be obtained. Therefore, another function is designed to assess the performance concerning string stability, which takes its inputs from the simulation results. Only a limited part of the various (almost unlimited) thinkable traffic scenarios can be simulated. Therefore, for the platoon performance measure it is not only important whether the platoon is string stable in the simulated scenarios, but also the *margin* of stability should not be too small, in order to improve the likelihood of stability in other scenarios. Recall from Section 2-3-2 that string stability can be assessed based on the propagation of state errors (then at least the position error), or of the velocity or acceleration or a combination of those. Because the stability margin will be compared between the different designed CACC controllers (instead of just determining whether the controller achieves stability), it is chosen not to use state errors in the stability margin function. This is because these errors depend on the desired states ($s_{\text{des}}(i)$), which are internal variables of the controller and are derived differently for different controllers (compare the derivations of the desired states for configuration I with configurations II and III in Section 3-3-1). Also, the PID controller keeps a nonzero position error in steady state driving (except at zero velocity) such that the resultant throttle input counteracts the tire friction and air resistance. Therefore, for the PID-based CACC controllers the propagation of position errors does not exactly represent the propagation of disturbances. This is why it is chosen to use the acceleration of the vehicles, or more precisely the maximum deceleration during periods when all following vehicles have to brake. For such a braking period, of the relative decline of the maximum deceleration from each vehicle to its follower the root mean square is taken instead of just the mean. This is to encourage a faster decline.
  For each period during which all following vehicles have to brake, the minimum acceleration

of each following vehicle ($i \geq 2$) over this period is taken. For braking period p and for vehicle i this minimum is called $a_{\min,p,i}$. Then

$$\begin{aligned} f_{\text{stab},p} &= 1 - \sqrt{\sum_{i=3}^{N_{\text{veh}}} \left(\frac{a_{\min,p,i}}{a_{\min,p,i-1}} \right)^2} \Bigg/ (N_{\text{veh}} - 2), \text{ for } p = 1, \ldots, N_{\text{per}} \\ f_{\text{stab}} &= \frac{1}{\sum_p w_p} \sum_p w_p \cdot f_{\text{stab},p}, \quad \text{with } w_p = \begin{cases} \frac{1}{2} N_{\text{per}} & \text{if } f_{\text{stab},p} < 0 \\ 1 & \text{if } f_{\text{stab},p} \geq 0 \end{cases} \\ P_{\text{stab}} &= \begin{cases} c_{\text{stab}} / f_{\text{stab}} & \text{if } f_{\text{stab}} > 0 \\ \infty & \text{if } f_{\text{stab}} \leq 0 \end{cases} \end{aligned} \tag{3-47}$$

where f_{stab} (< 1) is a stability margin measure, N_{per} is the number of braking periods considered here, and the constant c_{stab} represents an acceptable value for the stability margin measure. An unstable braking period is given extra weight in (3-47) before the average is taken, because the occurrence of this is very undesirable[14]. For f_{stab} it holds that, the bigger its value the bigger the stability margin, with $f_{\text{stab}} = 0$ representing marginal stability, and $f_{\text{stab}} < 0$ no string stability.

- Constraint violations:
  Since not only breaking a constraint is important, but also how strongly it is broken, the extent of violation is used for the calculation of the platoon performance measure. Here a small extent of violation will be punished mildly, but growing extent will be punished extra strongly to avoid violation to such a larger extent. For each constraint of Table 3-1 (except for the minimum distance constraint), and for each vehicle, the difference between the state and its bound is calculated for all violation periods. This is then normalised through dividing by a value (c_x) that is considered as an (almost) acceptable small violation for the constraint considered. The square of this is integrated over the violation periods. After having done this for each CACC-controlled vehicle, the results are averaged over all vehicles. Then the result is divided by the total period of time that the platoon leader's state approaches the bound[15]. This last action is to obtain a sort of time-average based on the time period most likely to cause the specific constraint to be violated. For a maximum value constraint on parameter x this looks like

$$P_{x_{\max}} = \sum_{i=2}^{N_{\text{veh}}} \frac{\displaystyle\int_{\{t|x_i(t)>x_{\max}\}} \left(\frac{x_i(t) - x_{\max}}{c_x} \right)^2 dt}{\varepsilon_0 + (N_{\text{veh}} - 1) \displaystyle\int_{\{t|x_1(t)>x_{\max}-x_{\text{p}}\}} dt} \tag{3-48}$$

such that $x_{\max}$ represents either $v_{\max}$ (with x_i representing v_i), or $a_{\max}$ (with x_i representing a_i), and with $\varepsilon_0 > 0$ a small constant to avoid dividing by zero. For a minimum value constraint on parameter x this looks like

$$P_{x_{\min}} = \sum_{i=2}^{N_{\text{veh}}} \frac{\displaystyle\int_{\{t|x_i(t)<x_{\min}\}} \left(\frac{x_i(t) - x_{\min}}{c_x} \right)^2 dt}{(N_{\text{veh}} - 1) \displaystyle\int_{\{t|x_1(t)<x_{\min}+x_{\text{p}}\}} dt} \tag{3-49}$$

[14] If one of the braking periods is unstable, this of course means that the controller fails in achieving string stability in all cases. Nevertheless it was chosen not to give the stability performance measure P_{stab} an infinite value in this situation, in order for P_{stab} to be a continuous function. The chosen approach in this matter, in retrospect, is not the right approach. In Section 4-4 an approach is given that would have been better.

[15] The reason why the total time period that the platoon leader *approaches* the bound is used in (3-48) and (3-49) instead of the time period that it *exceeds* this bound is because the leader approaching the bound can result in the follower exceeding this value. Even if the leader never exceeds any of these bounds, with poor CACC controllers followers may still do so.

such that $x_{\min}$ represents either $v_{\min}$ (with x_i representing v_i), or $a_{\min}$ (with x_i representing a_i). The parameters c_x and x_{p} from (3-48) and (3-49) have different values for different constraints.
The minimum vehicle-to-vehicle distance constraint ($d \geq d_{\min}$) exists because breaking it means (practically) the occurrence of collision. Violation of this constraint should be punished very strongly, with a distance smaller than or equal to zero (collision) given a performance measure of infinity. Therefore, the inverse distance is used. For the distance approaching zero, the penalty starts increasing (from zero) at $d = d_{\mathrm{pun0}}$, and is chosen to rise such that it has the value of 1 for $d = d_{\mathrm{pun1}}$. This results in the performance measure equation,

$$P_{d_{\mathrm{margin}}} = \sum_{i=2}^{N_{\mathrm{veh}}} \frac{\int_{\{t|d_{i,i-1}(t)<d_{\mathrm{pun0}}\}} \frac{d_{\mathrm{pun1}}}{d_{\mathrm{pun0}} - d_{\mathrm{pun1}}} \left(\frac{d_{\mathrm{pun0}}}{d_{i,i-1}(t)} - 1 \right) dt}{(N_{\mathrm{veh}} - 1) f_{d\mathrm{m}} t_{\mathrm{end}}} \tag{3-50}$$

where $f_{d\mathrm{m}} t_{\mathrm{end}}$ should represent a rough estimate of the fraction of the time (t_{end}) that for the scenarios used, without optimally tuned parameters, it would be possible for the intervehicle distance to approach $d_{\min}$.

The overall platoon performance measure is given by

$$\begin{aligned} P_{\mathrm{platoon}} = \; & W_d P_d + W_{u_{\mathrm{tb}}} P_{u_{\mathrm{tb}}} + W_j P_j + W_{\mathrm{stab}} P_{\mathrm{stab}} + W_{v\,\min} P_{v\,\min} + \\ & W_{v\,\max} P_{v\,\max} + W_{a\,\min} P_{a\,\min} + W_{a\,\max} P_{a\,\max} + W_{d\mathrm{margin}} P_{d\mathrm{margin}} \end{aligned} \tag{3-51}$$

where W_d, $W_{u_{\mathrm{tb}}}$, W_j, W_{stab}, $W_{v\,\min}$, $W_{v\,\max}$, $W_{a\,\min}$, $W_{a\,\max}$, and $W_{d\mathrm{margin}}$ are the weights of the element-specific platoon performance measures for each of the respective elements.

To tune each of the controllers, an optimisation method is applied to an objective function that needs to be optimised. This function comprises a simulation of variations of each of the scenarios mentioned above. The inputs to the function are the parameters that need to be tuned. The output of the function is the platoon performance measure (3-51) calculated from the simulation results. This objective function is not convex, causing local search algorithms to only find a local optimum. Therefore, in Chapter 4 it will be chosen to use a more global search method, called simulated annealing [14, 48]. Because at the time of designing and execution of most of this part of the project the simulated annealing toolbox of MATLAB was not yet available, a self-made simulated annealing function is designed in MATLAB, which is described in Appendix A-7.

3-5 Summary

This chapter has described the design objectives of the project into further detail, given the vehicle model that will be representing a real vehicle in this project, presented designs of the CACC controllers, and proposed a way to rate the performance and an approach to tune the controllers. The contents are summarised here.
In short, the CACC controllers to be designed as part of the project goal are MPC-based as well as PID-based CACC controllers, with for each of these controller types three information gathering configurations, one that only looks at the direct predecessor for its reference, one that looks at its first two direct predecessors, and one that looks at its direct predecessor and the platoon leader. The state data from the vehicles is assumed to be obtained from vehicle-to-vehicle communication. The controllers have to be designed according to performance

requirements such as keeping a distance that is as small as possible while still being safe, achieving string stability, and obeying constraints on velocity, acceleration, and distance. A model from literature is chosen to represent vehicle dynamics. The inputs of this model are pedal input (positive for throttle, negative for brakes) and gear number. The CACC controller combines the states from other vehicles to form a set of reference states. The PID controller uses proportional gains for the position, velocity, and acceleration errors respectively. To try to (approximately) obey the constraints, the calculated PID output is cut off when it is expected, from model calculation, that a constraint will otherwise be broken. For its predictions, the MPC controller uses an approximation of the vehicle model that results in an MLD model. The optimisation problem for determining the optimal control inputs then takes the form of an MILP problem. To rate the performance of the CACC controllers a performance measure is developed, based on the various aspects of the performance requirements given earlier in this chapter. This measure will be used for the tuning and the final evaluation of the controllers in the next chapter.

This chapter has presented the theory of the project, independent of the parameter values (mass, length, etc.) of the specific vehicle used, the choice of constraint values, the values of the parameters that are part of the performance measure for tuning and evaluation (e.g., weights and normalisation factors), or the exact values of the parameters defining the traffic scenarios for tuning or evaluating the controllers. These parameters are assigned values in the next chapter, where they define the specifics of the case study.

Chapter 4

Case study

The CACC controllers that are described in Chapter 3 will be implemented into simulated vehicles that are part of a simulation of a platoon of vehicles. The controllers will be tuned and evaluated using this simulation setup. They are not evaluated with real vehicles, because that is beyond the scope of this project. This chapter first describes the simulation setup, after which the scenarios used to tune the controllers with are given and the tuning results are presented. The final section then evaluates the tuned controllers.

4-1 Simulation setup

In MATLAB a simulation of a platoon of vehicles is developed, where the vehicles are simulated according to the vehicle model presented in Section 3-2. Figure 4-1 depicts such a platoon. All of the vehicles, except the platoon leader, are controlled by a copy of the same CACC controller. They can share their state information, and in case of the MPC_2 variant also their predicted states, via communication. The platoon leader trajectory is given by a time-velocity sequence, which is defined by periods of constant acceleration that can be altered before starting the simulation. It is assumed that the leader knows its future states for at least up to the prediction horizon of the MPC-CACC-driven followers, and that it communicates its current, and in the case of MPC its future states (up to time instant $t = (k + N) \cdot T_s$), to the vehicles that require this information for their CACC controller. The simulation is one-dimensional. This means that only the longitudinal direction is considered, and the lateral (steering) direction is ignored. So the vehicles are assumed to be perfectly aligned longitudinally, and driving on a straight, flat, horizontal road.

The simulation consists of a main simulation function in MATLAB, which calls other functions. This main function contains some simulation settings that can be altered. The first of these settings are the information gathering configuration, the controller type (MPC or PID), and the MPC variant (communication of just states, or also predictions). For the information gathering configuration, there can be chosen between using state information from just a number ($f <$ total number of vehicles in the platoon) of direct predecessors, or in addition to that also the platoon leader or even the direct follower. In the case of also using the platoon leader's information, the vehicles to whom the platoon leader is also one of their ($\leq f$) direct predecessors concerned consider the leader only as in this latter capacity (one of the direct

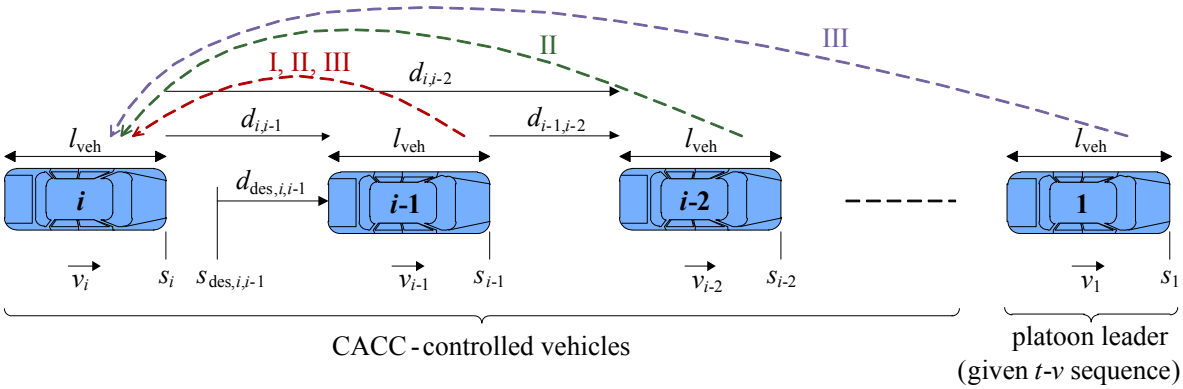


Figure 4-1: A platoon of vehicles as simulated in MATLAB. The coloured dashed lines illustrate from which vehicles vehicle i receives information for the three different information gathering configurations (I, II, and III).

predecessors)[1]. This means that the simulation has room for more configurations than just the three configurations that are described in Chapter 3. Even though the simulation is developed such that it is possible to choose settings that give other configurations, only these three (with their different controller types) are tuned and investigated within the scope of this project. The other simulation settings are the total number of vehicles in the platoon, the leader time-acceleration sequence, and the initial states of the other vehicles, together defining the traffic scenario to be simulated, the sampling period (T_s), and the simulation sampling period[2] ($T_{s,sim} \leq T_s$). Recall that in this project, for simplicity[3], $T_{ctrl} = T_{sim} = T_s$. The leader time-velocity sequence is defined by sets of time instants and corresponding accelerations. Between two subsequent given time instants ($t_{a,n-1} < t \leq t_{a,n}$, with $t_{a,n} = c_{a,n} \cdot T_{s,sim}$, $c_{a,n} \in \mathbb{N}$, $c_{a,n-1} < c_{a,n}$, $c_{a,0} = 0$), the leader is accelerating with the corresponding given acceleration (a_n), which can be negative, positive, or zero (see, e.g., Figure 4-2). The main simulation function also assigns values to all the tunable parameters (from Table 4-4), which will be set to the optimal values found after being tuned. During the tuning stage, the values of the tunable parameters will be passed to the main simulation function by the function that contains the tuning algorithm. This file runs the main simulation function in every iteration with varying values of the tunable parameters, in order to find the values that optimise the platoon performance measure calculated on the simulation results.

The main simulation function simulates the platoon over the simulation period (ended at the final leader-acceleration time instant given) by executing a loop of which each iteration represents one sampling period (T_s). At each sampling step each vehicle is simulated for the time period between the current and the next sampling step by calling a vehicle simulation function. The time-state and control action data that are given as outputs by these vehicle function executions are, in addition to being used for the next time step, also gathered to be used to calculate performance information after the simulation has finished, and to graphically

[1] To illustrate, e.g., when $f = 3$ and information from the platoon leader is used, vehicle 2 to 4 do not use the leader's states the way that all followers of vehicle 4 do, but operate as if only information from the first three predecessors is used.

[2] The simulation sampling period ($T_{s,sim}$) gives the interval between time instants at which the simulated states that are calculated according to the continuous-time simulation model (the vehicle model of (3-1)) are sampled. These state samples are used after the simulation has finished to calculate the platoon performance measure, and to plot time-state graphs representing the simulated vehicles.

[3] Recall that the control update period (T_{ctrl}) gives the interval between control update time instants, i.e., time instants at which the controller calculates a new control input and applies it to the vehicle. The sampling period (T_s) gives the time interval between samples taken to be used by the controller. The simulation period T_{sim} would represent the time between two time steps used by MPC to simulate the results of possible future control actions, to be used for the calculation of its performance index.

Vehicle mass m	Vehicle length l	Viscous friction c	Coulomb friction μ	Gravitational constant g
800 kg	2.5 m	0.5 kg/m	0.01	9.8 m/s^2

Gear j	Traction force and velocity bounds for the gears $b(j)$ (N)	$v_{\text{low},j}$ (m/s)	$v_{\text{high},j}$ (m/s)
1	4057	0	9.46
2	2945	5.43	13.04
3	2116	7.56	18.15
4	1607	9.96	23.90
5	1166	13.70	32.93
6	838	19.10	45.84

Table 4-1: Model parameter values (for a SMART) [11].

show the simulation results. At each time step the vehicle simulation function receives as input the data needed by the CACC controller. It then processes this data according to the theory of Section 3-3, and applies the PID or MPC controller[4]. The vehicle function then passes the control signal to a function that contains the continuous model of the vehicle from (3-1). To retrieve the state trajectories for the time period between the current and the next time step ($k \cdot T_\text{s} < t \leq (k+1) \cdot T_\text{s}$), the differential equation is solved, with the control inputs kept constant over this period. The states at times $T_{\text{s,sim}} = 0.01\,\text{s}$ apart are then calculated (by means of the MATLAB function "ode45") to obtain vectors with samples of the simulated continuous-time state trajectories.
The vehicles that are simulated all have the same vehicle model. In this project the model parameters of a SMART, taken from [11] are used for the vehicle simulation, and therefore the controllers are designed and tuned to be used in a (simulated) platoon of SMARTs. The values of all the parameters of the vehicle model (3-1) for the SMART and the vehicle length are shown in Table 4-1.

4-2 Tuning

Limiting the computation time for tuning

For the tuning of parameters in the CACC controllers of Chapter 3, repeatedly a platoon of vehicles undergoing a traffic scenario is simulated in order to minimise the platoon performance measure, using an optimisation algorithm. The more different subscenarios the traffic scenario contains (i.e., the longer it gets) or the more vehicles it comprises, the more time the computer needs for the simulation. Similarly, the shorter the chosen sampling period T_s, the more computation time is needed to simulate a scenario that represents the same time range. Also, when tuning, the more parameters are tuned simultaneously, the more iterations are (ideally) needed (growing exponentially). Therefore, in the following subsections some non-ideal choices will be made in order to keep the computation time needed for tuning relatively low. These choices will concern the following (for further details see the subsections below):

[4]For the MPC controller, the largest part of the matrices is defined outside of the major loop in the main simulation function to minimise the amount of calculation, to be done at each iteration of this loop.

Parameter	Value
$d_{\min}$	$0.1\,\mathrm{m}$
$v_{\min}$	$0\,\mathrm{m/s}$
$v_{\max}$	$120\,\mathrm{km/h} \approx 33.3\,\mathrm{m/s}$
$a_{\min}$	$-2.5\,\mathrm{m/s^2}$
$a_{\max}$	$2.5\,\mathrm{m/s^2}$
T_s	$0.5\,\mathrm{s}$

Table 4-2: Chosen values for parameters associated with the CACC design, partly based on [1, 11, 13, 15, 20].

- The value of the control update time (here equal to T_s) will be chosen higher than it should be for practical implementations (Section 4-2-1).
- The prediction horizon (N) will be tuned separately, only after the other parameters are tuned (Section 4-2-2).
- Several parameters that occur in all three configurations, are tuned with configuration I, and then the optimal value found is also given to these parameters in the corresponding configurations II and III (i.e., with the same controller type/MPC variant, Section 4-2-2).
- The number of vehicles in the tuning scenario, and the number of subscenarios making up the tuning scenario are kept low (Section 4-2-3).
- The separate subscenarios that make up the tuning scenario are given a smaller time span when tuning MPC, compared with when tuning PID, i.e., every time after the platoon leader has just accelerated/decelerated in the MPC tuning scenario less time passes before it starts accelerating/decelerating again, compared with the PID tuning scenario (because MPC demands more computation time than PID, Section 4-2-3).
- The maximum allowed number of iterations and the number of starting points for the multi-start simulated annealing optimisation applied for tuning, are kept realtively low (Section 4-2-4).

4-2-1 Chosen parameter values

In Chapter 3 several parameters have been defined that need to be assigned values for this case study. These parameters are the constraint values that give performance requirements (introduced in Table 3-1), the sampling period, and platoon performance measure related parameters (introduced in Section 3-4). The chosen values for these parameters are given in Tables 4-2 and 4-3. The minimum velocity ($v_{\min}$) was already fixed to zero, necessary for the controller design in general (to avoid negative velocity, as explained in Section 3-1-2). For the maximum velocity ($v_{\max}$) the maximum allowed velocity on highways in The Netherlands ($120\,\mathrm{km/h}$) is used. The values for minimum and maximum comfortable acceleration ($a_{\min}$ and $a_{\max}$ respectively) are chosen based on the range of values used for these parameters in literature [1, 11, 13, 15, 20]. The control update period—chosen equal to the sampling period, T_s—is set to $0.5\,\mathrm{s}$ for simplicity.
The chosen value or $v_{\max}$ fixes the values of the parameters concerning the PWA approximation of the squared velocity in Figure 3-3. For future reference it may be useful to know that now $\alpha \approx 14.98\,\mathrm{m/s}$, and the three (nonzero) intersections between v^2 and its PWA approximation lie at respective (approximate) velocities $11.24\,\mathrm{m/s}$, $18.86\,\mathrm{m/s}$, $29.45\,\mathrm{m/s}$.

Performance element	Weights		Other parameter values	
	Name	Value	Name	Value
intervehicle dist.	W_d	10	$d_{\text{av,norm}}$	22.5 m
pedal variation	$W_{u_{\text{tb}}}$	2.0	$\Delta u_{\text{tb,av1.5,norm}}$	$\frac{1}{15}$
gear variation	W_j	2.5	$\Delta j_{\text{av1.5,norm}}$	$\frac{1}{9}$
string stability	W_{stab}	1.25	c_{stab}	0.03
distance margin	$W_{d\text{margin}}$	10	d_{pun0} d_{pun1} $f_{d\text{m}}$	1.25 m 1 m $\frac{1}{25}$
			Parameters related to constraint violations	
			c_x	x_{p}
$v_{\min}$	$W_{v\,\min}$	100	0.05 m/s	3.3 m/s
$v_{\max}$	$W_{v\,\max}$	10	0.5 m/s	3.3 m/s
$a_{\min}$	$W_{a\,\min}$	5	0.05 m/s^2	0.05 m/s^2
$a_{\max}$	$W_{a\,\max}$	5	0.05 m/s^2	0.05 m/s^2

Table 4-3: Chosen values for parameters associated with the platoon performance measure.

4-2-2 Parameters to be tuned

In Sections 3-3-1 to 3-3-3 many CACC controller parameters have been defined that still need to be tuned. Some of these parameters are mutually dependent since the sum of specific sets of parameters should be equal to 1. This is why for configurations II and III the *ratio* of importance of the two other vehicles considered is tuned. The weight matrix Q from (3-34b) can be represented by

$$Q = \begin{bmatrix} Q_s & Q_{vs} \\ Q_{sv} & Q_v \end{bmatrix}, \quad (Q_{vs} = Q_{sv} = 0 \text{ for simplification}) \tag{4-1}$$

It is chosen to set $Q_{vs} = Q_{sv} = 0$ to simplify the tuning task. For Q_s, Q_v, $R_{u_{\text{tb}}}$, and R_j only the ratio between the values of these parameters is important. Therefore, the first of these is set to $Q_s = 1$. Then, tuning the other three means finding the optimal values relative to Q_s. For reasons given at the start of Section 4-2, the prediction horizon N is assigned a constant value ($N = 5$), while tuning the other parameters. Then, after all other parameters are tuned, N will be varied between 1 and 14 to see which increase of N still gives a significant decrease of the platoon performance measure, while also considering the average computation time needed for the MPC optimisation at each time step. The parameters that are only tuned with configuration I (for each controller type), and then are assigned the same values for configuration II and configuration III (see the start of Section 4-2) are k_s, k_v, and k_a for PID, and Q_v, $R_{u_{\text{tb}}}$, and R_j for MPC. Because for configurations II and III the information of the second other vehicle considered is expected to help the controller react quicker to changes, making closer following possible, the time headway is retuned for each of the three configurations (recall that the second vehicle in the platoon always uses the value for h_0 corresponding to the appropriate configuration I).

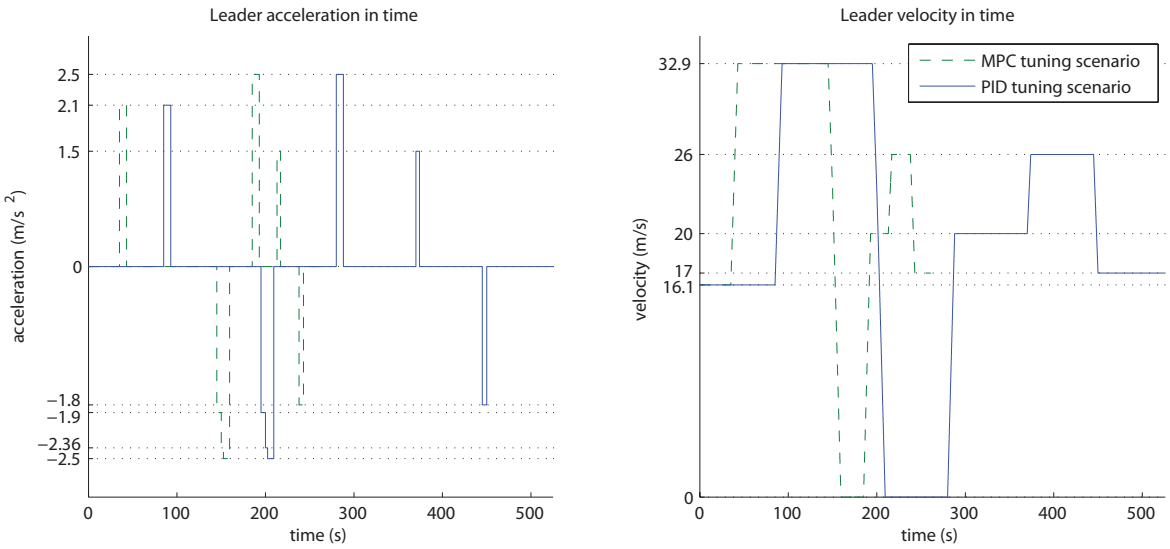


Figure 4-2: Acceleration and velocity trajectories of the platoon leader for the tuning scenario. When tuning MPC, shorter subscenarios are used (see the start of Section 4-2).

4-2-3 Traffic scenario for tuning

To obtain the platoon performance measure used for tuning the CACC controllers, a platoon of vehicles will undergo a series of subscenarios which represent variations of all the traffic scenario types a-e listed in Section 3-4. The more variations are used, the better the space of all possible traffic scenarios is represented. Here, a platoon should consist of at least three vehicles in order to also take string stability into account, although it is preferable to have more vehicles for this. If there are at least four vehicles, then for each of the three configurations there is at least one vehicle that does not use states from all its predecessors. The traffic scenario used for tuning will consist of seven of the before mentioned subscenarios, simulated for a platoon of five vehicles.
The velocity and acceleration trajectories of the leader for the tuning scenario are shown in Figure 4-2. The initial situation mimics vehicle 2 and its followers up to the time instant $t = 0$ being one platoon catching up with another vehicle that, from $t = 0$ on, takes the place of platoon leader (a variant of scenario type a from Section 3-4). All followers have an initial velocity of 25 m/s, while the platoon leader has an initial velocity of 16.1 m/s. Initially, the distance between the leader and the second vehicle is 100 m. Between the other three subsequent vehicle pairs the initial distance is the distance they would keep if the platoon leader were not there and the second vehicle were the leader, i.e., $d_{i,i-1}(t{=}0) = d_0 + h_0 \cdot v_i(t{=}0)$ for $i = 3, 4, 5$. As can be seen in Figure 4-2, the scenario types from Section 3-4 are represented in the following order: a-b-d-e-b-c. When braking from 32.9 m/s to zero velocity, here the deceleration increases in steps in order not to exceed (too greatly) the maximum deceleration that can be achieved for the corresponding velocity with the vehicle model (3-1).

4-2-4 Values of the tuned parameters

For reasons described in Section 3-4, as optimisation method for tuning, simulated annealing is used. Although this method is not guaranteed to find the global optimum for a problem when only given finite time, theoretical results show that if it is given sufficient time, ultimately

configuration	tuned parameters	PID	MPC_1	MPC_2
I	h_0 (s)	2.731	0.82	0.437
	k_s (N/m)	141	—	—
	k_v (N·s/m)	1182	—	—
	k_a (N·s^2/m)	18	—	—
	Q_v	—	1.727	1.286
	$R_{u_{\mathrm{tb}}}$	—	0.262	0.582
	R_j	—	0.45	0.48
	N	—	5	5
II	h_0 (s)	0.492	0.662	0.299
	c_{dmin}	0.32	0	0.45
	$\frac{q_{s,p2}}{q_{s,p1}}$	0.719	0.439	6.25
	$\frac{q_{v,p2}}{q_{v,p1}}$	0.375	0.441	0.021
	$\frac{q_{a,p2}}{q_{a,p1}}$	103.73	—	—
	N	—	5	5
III	h_0 (s)	0.417	0.78	0.325
	$\frac{q_{v,l}}{q_{v,p}}$	0.089	0.013	0.175
	$\frac{q_{a,l}}{q_{a,p}}$	206.04	—	—
	N	—	5	5

Table 4-4: Tuned values for the tunable parameters[7].

a global optimum can be found[5] [14]. Here, multi-start simulated annealing is applied, with only two different random starting points, with a maximum number of performance measure evaluations per starting point of 5000 with MPC and 7500 with PID (because its simulations require less time). Only with PID-II three starting points are applied, with a maximum of 10000 performance measure evaluations per starting point, because one additional parameter is tuned simultaneously compared with the other configuration II and I controllers.
Tuning the tunable parameters, i.e., optimising the platoon performance measure (3-51) for simulations of the tuning scenario (where the states are "measured" with simulation sampling period $T_{\mathrm{s,sim}}$)[6], results in the values given in Table 4-4. Some interesting aspects of these tuned parameter values will be discussed now.

When looking at Table 4-4, one thing that stands out are the different values of the time headway h_0. As will be seen in Section 4-3-2 the tuned PID controllers react more slowly to changes than MPC does. The fact that the platoon performance measure (minimised by tuning) punishes big sudden changes in throttle/brake signal must have caused the PID gains after tuning to be such that the responsiveness is as low as it is. This low responsiveness, combined with the fact that the PID controller cannot anticipate the leader's, hard to follow,

[5]Since it is not known whether the time given here to the simulated annealing optimisations is actually sufficient to find the global optima, (some of) the optima found might actually be local optima, i.e., giving suboptimal results. However, for the sake of simplicity, from now on the general term "optimal" is used, also when referring to what might actually be suboptimal.

[6]Recall that the *platoon performance measure* (P_{platoon} from (3-51)) evaluates a CACC controller by evaluating the performance of a whole simulated platoon of vehicles using this CACC. This should not be confused with the MPC *performance index* (J from (3-34)), which is minimised at each control update time step as part of the MPC algorithm in order to determine the control input to feed to the vehicle.

[7]Recall that the position weight in the MPC performance index was set to $Q_s = 1$, and that the parameters k_s, k_v, k_a, Q_v, $R_{u_{\mathrm{tb}}}$, and R_j are tuned with configuration I, and then given the same values for the corresponding configurations II and III. See Section 3-1-1 for the meaning of configurations I-III, MPC_1, and MPC_2.

future actions causes the second vehicle in the platoon ($i = 2$) to need a large time headway to keep a safe distance to the platoon leader throughout the tuning scenario. Hence the high value of h_0 for PID with configuration I. The second vehicle's trajectory varies more slowly than the leader's, making it easier for the third vehicle to follow closely with configurations II and III. This combined with the advantage of the extra features of configurations II and III, i.e., using information from one additional predecessor would explain the much lower values of h_0 for these configurations[8].
For MPC_2 following at close distance appears to be easier than for MPC_1, which suggests that the communication of predicted states might indeed improve the controller. When comparing the values of h_0, it should be noted that at constant velocity, the same value of h_0 causes a larger steady-state intervehicle distance for PID than it does for MPC. This is because with PID a positive position error is needed in order to have the controller give a positive throttle input to counteract the air resistance and tire friction components in the vehicle model.

From the PID gains (shown in Table 4-4) it can be seen that, in PID terms, the proportional action (corresponding to k_v) is here the most important one, while the integral action k_s is just needed to adjust the control action to avoid drifting. The differential action k_a is of almost no importance, probably because a high value of k_a would have made the throttle/brake input trajectory much less smoothly, caused by following the sudden changes in acceleration of the platoon leader. Note that, although the ratios $(q_{a,p2}/q_{a,p1})$ for configuration II and $(q_{a,l}/q_{a,p})$ for configuration III are relatively big, because of the low value of k_a the influence of this is on the control action is still very limited.

The value of $c_{\text{dmin}} = 0.32$ for PID-II means that the desired position will not become closer to the direct predecessor than roughly one third of the desired distance resulting from only considering this predecessor, and not the second predecessor (according to (3-7)-(3-8)). A similar reasoning holds for $c_{\text{dmin}} = 0.45$ with MPC_2-II. For MPC_1-II $c_{\text{dmin}} = 0$, which means that the desired position (of the vehicle front end) can approach the rear end of the direct predecessor, but cannot exceed it. For this controller, the optimum found corresponds to a line on which $c_{\text{dmin}} \leq 0.15$, and the values for the other tunable parameters as shown in Table 4-4. It turns out that, for the tuning scenario, at this optimum the part of the CACC algorithm that applies c_{dmin} ((3-7) and (3-8)) is not executed (because also without this part, the desired position does not exceed the rear end of the predecessor). It was therefore chosen to set the parameter c_{dmin} to zero for MPC_1-II[9].

Recall that the prediction horizon was set to $N = 5$ while all the other parameters of the controllers where first tuned. After that, N was varied to find the optimal value, while the other tunable parameters were fixed to their optimal values, just found. Using this approach, instead of tuning N simultaneously with the other parameters, increased the likelihood of $N = 5$ to give the best performance to be found for the tuning scenario (see section 4-4 for reflection on this). Indeed, for all MPC controllers $N = 5$ does give the best platoon performance measure.

[8] Retuning, afterwards as a test, only h_0 for vehicles $i \geq 3$ with configuration I, gave with PID and MPC_2 a lower value for this h_0 (0.646 s and 0.364 s respectively). However, this lower value of h_0, and the corresponding decreased value of the platoon performance measure, were not as low as those of the corresponding configurations II and III. With MPC_1 no such improvement was found. This shows that, indeed, using information from an extra predecessor (in part) causes the lower value of h_0 for configurations II and III.

[9] A smaller value than zero for c_{dmin} would be undesirable, because it could put the desired position during another traffic scenario on or beyond the predecessor.

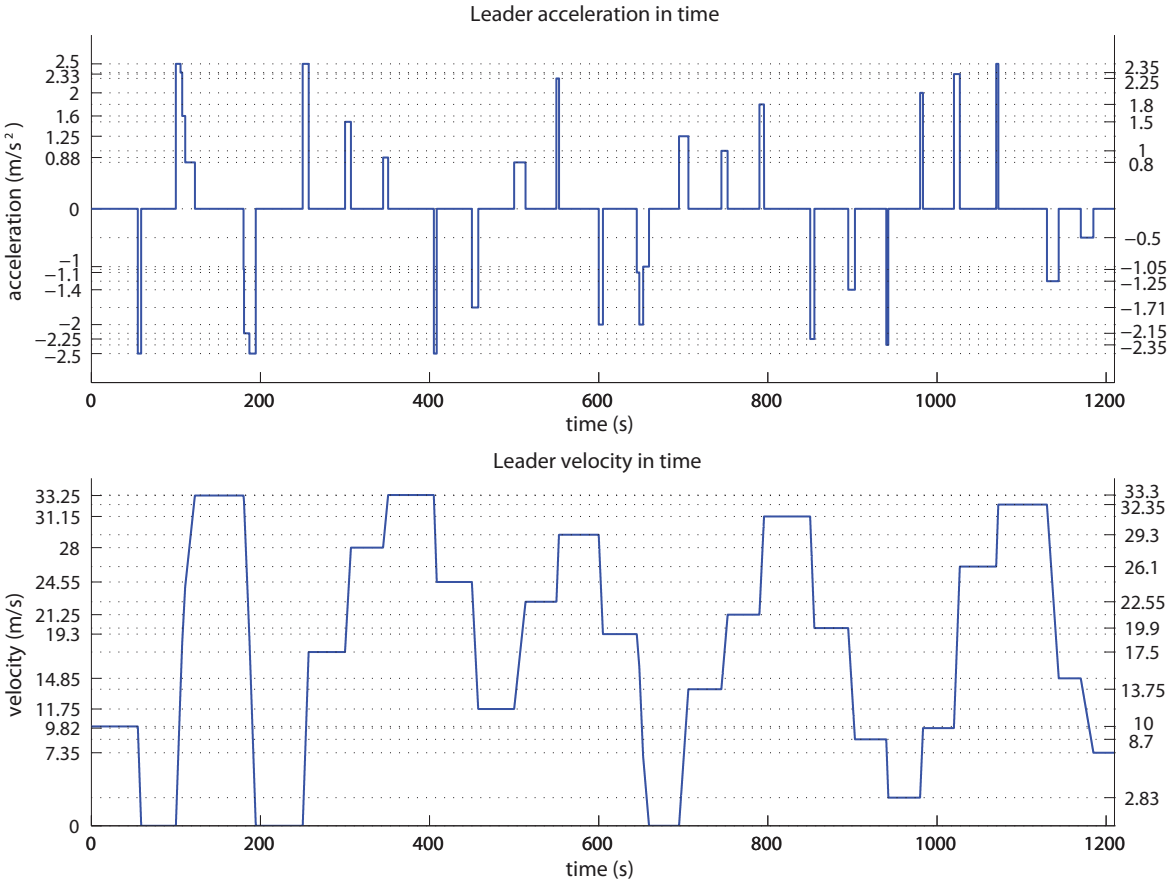


Figure 4-3: Acceleration and velocity trajectories of the platoon leader for scenario 2 (used for evaluation of both PID and MPC-based CACC controllers).

4-3 Evaluation

4-3-1 Results from simulating different scenarios

To be able to evaluate the CACC controllers that have been designed, they are implemented in a platoon of vehicles that undergoes two traffic scenarios, each of which is composed of several subsequent subscenarios. The first scenario, which will be called scenario 1, is the scenario used for tuning of the PID-based controllers (again with five vehicles)[11]. The second scenario (the validation scenario, scenario 2) is built up from similar types of subscenarios, but with different values for velocity and acceleration. Because this scenario is only used for validation (i.e., not for tuning), there is no need for it to be small. Furthermore, the longer the platoon and the more different situations are simulated, the better the controllers can be

[10] When looking at Table 4-5, it should be noted that the average distance is for the entire simulation period, including the period of time when, with the PID-based controllers, the second vehicle's position lies beyond the leader's rear-end position, giving a negative vehicle-to-vehicle distance, i.e., the vehicles crashed. Furthermore, it should be noted that since the heavy oscillations in acceleration with MPC_1 configurations I and III are all bounded at $a_{\min}$ ($=2.5\,\text{m/s}$), which inhibits the decrease of minimum acceleration reached with increasing vehicle number, their string stability measure does not fully show the severity of their string instability.

[11] Recall that the PID tuning scenario only differs from the MPC one in the length of the leader's constant velocity periods. The acceleration and constant-velocity values during the acceleration periods are the same for the two scenarios.

Performance aspect →	platoon performance measure P_{platoon}	average distance (m)	number of gear switches /vehicle /100s (s^{-1})	throttle /brake variation $\Delta u_{\text{tb,av1.5}}$	string stability margin f_{stab}	total constraint violation time /vehicle (s)				
						$v_{\min}$	$v_{\max}$	$a_{\min}$	$a_{\max}$	$d_{\min}$
Type/ Configuration	Scenario 1 (PID tuning scenario):									
PID-I	30.35	58.93	2.95	0.055	0.126	10.67	0	0	0	0
PID-II	16.30	25.20	2.95	0.057	0.032	31.66	0	0	0	0
PID-III	15.83	24.65	2.95	0.055	0.036	24.44	0	0	0	0
MPC_1-I	13.15	19.86	2.57	0.058	0.064	0	0	0.78	0.25	0
MPC_1-II	11.81	17.53	2.57	0.052	0.086	0	0	1.05	0.22	0
MPC_1-III	13.15	19.29	2.57	0.060	0.050	0	0	1.14	0.29	0
MPC_2-I	10.61	12.49	2.90	0.066	0.043	0	0	0.64	0.13	0
MPC_2-II	8.91	10.47	2.62	0.059	0.088	0	0	1.09	0.76	0
MPC_2-III	9.18	10.87	2.90	0.059	0.100	0	0	1.16	0.35	0
	Scenario 2 (validation scenario):									
PID-I	∞	55.15	3.94	0.040	0.133	3.72	0	0	0	1.99
PID-II	∞	17.23	4.26	0.056	0.016	18.19	18.74	0	0	1.99
PID-III	∞	16.58	4.20	0.053	0.045	9.93	8.25	0	0	1.99
MPC_1-I	∞	18.00	3.55	0.145	-0.021	0	0	1.22	1.17	0
MPC_1-II	∞	15.38	3.55	0.063	-0.001	0	0	1.20	0.16	0
MPC_1-III	∞	17.33	3.55	0.179	-0.036	0	0	1.46	2.86	0
MPC_2-I	∞	10.98	3.55	0.088	-0.006	0	0	1.82	3.32	0
MPC_2-II	∞	8.73	3.55	0.095	-0.021	0	0	1.30	1.98	0
MPC_2-III	∞	9.16	3.58	0.088	-0.032	0	0	1.79	4.24	0

Table 4-5: For each controller the platoon performance measure and the performance measures according to each of the performance aspects, for both traffic scenarios. Here measures that give values per vehicle are averaged over all following vehicles. The distance (third column) is taken from head to tail. See (3-51), (3-45), and (3-47) for the definitions of P_{platoon}, $\Delta u_{\text{tb,av1.5}}$, and f_{stab}. The value $P_{\text{platoon}} = \infty$ is caused with PID by the violation of the minimum distance constraint (vehicles crash), and with MPC by a negative string stability margin (indicating string instability). These unsatisfactory results are indicated by a red shade[10].

evaluated. Therefore, the platoon contains more vehicles than with the tuning scenario, and the scenario contains more variations of the proposed (sub)scenarios from Section 3-4[12].
The velocity and acceleration trajectories of the platoon leader for scenario 2 are shown in Figure 4-3. In this scenario the platoon consists of twelve vehicles, with initial conditions: $v_i(t{=}0) = 17\,\text{m/s}$ for $i = 2, \ldots, 12$, intervehicle distance $d_{2,1}(t{=}0) = 65\,\text{m}$, $d_{i,i-1}(t{=}0) = d_0 + h_0 \cdot v_i(t{=}0)$ for $i = 2, \ldots, 12$. The stepwise increase or decrease in acceleration that occurs at some points is introduced to approximately obey the maximum/minimum achievable acceleration with the vehicle model used for simulation, which depends on velocity (caused by air resistance) and gear number.
From the results, all elements of performance shall be evaluated. For each of these two scenarios, comparing the value of the platoon platoon performance measure from (3-51) will tell which of the controllers performs best in general. In addition to that, each aspect that is graded by the platoon performance measure will also be analysed separately to be able

[12]The number of vehicles in the platoon for the validation scenario (scenario 2) is chosen based on what was still feasible to simulate with the memory available on the computers used (7.93 GB), keeping in mind that the more vehicles in the platoon, the better it can be seen if the platoon is string stable.

to get better insight into the performance of the controllers. The performance measures of the controllers for these two scenarios are given in Table 4-5. Graphs showing parts of the simulation results (trajectories in time) for scenario 2 are given in the next section. For a more general overview, simulation results for some of the controllers for a few subscenarios of scenario 2 are plotted in Figure B-1 in Appendix B.

4-3-2 Evaluation of the designed CACC controllers

Intervehicle distance

It can be seen from Table 4-5 that with MPC_2-II the vehicles keep the smallest average intervehicle distance. PID-I achieves by far the highest average intervehicle distance. In Section 4-2-4 is was explained how the tuned value of time headway (h_0) for PID-I became significantly higher than those of PID configurations II and III. The differences in the resulting average intervehicle distances between these three configurations reflect these different values of h_0.
While with PID, configuration III keeps the smallest average intervehicle distance, it is configuration II that keeps the smallest average intervehicle distance with the two MPC variants.

The minimum distance constraint is violated with the PID-based controllers for scenario 2 by the second vehicle, of which the front end (for $t \in [197.7\,\text{s}, 219\,\text{s}]$) exceeds the rear-end position of the platoon leader (reaching a peak of $d_{1,2}$ = -1.7 m, see Figure 4-4). So even though this second vehicle tries to keep a relatively big time headway, it still crashes.

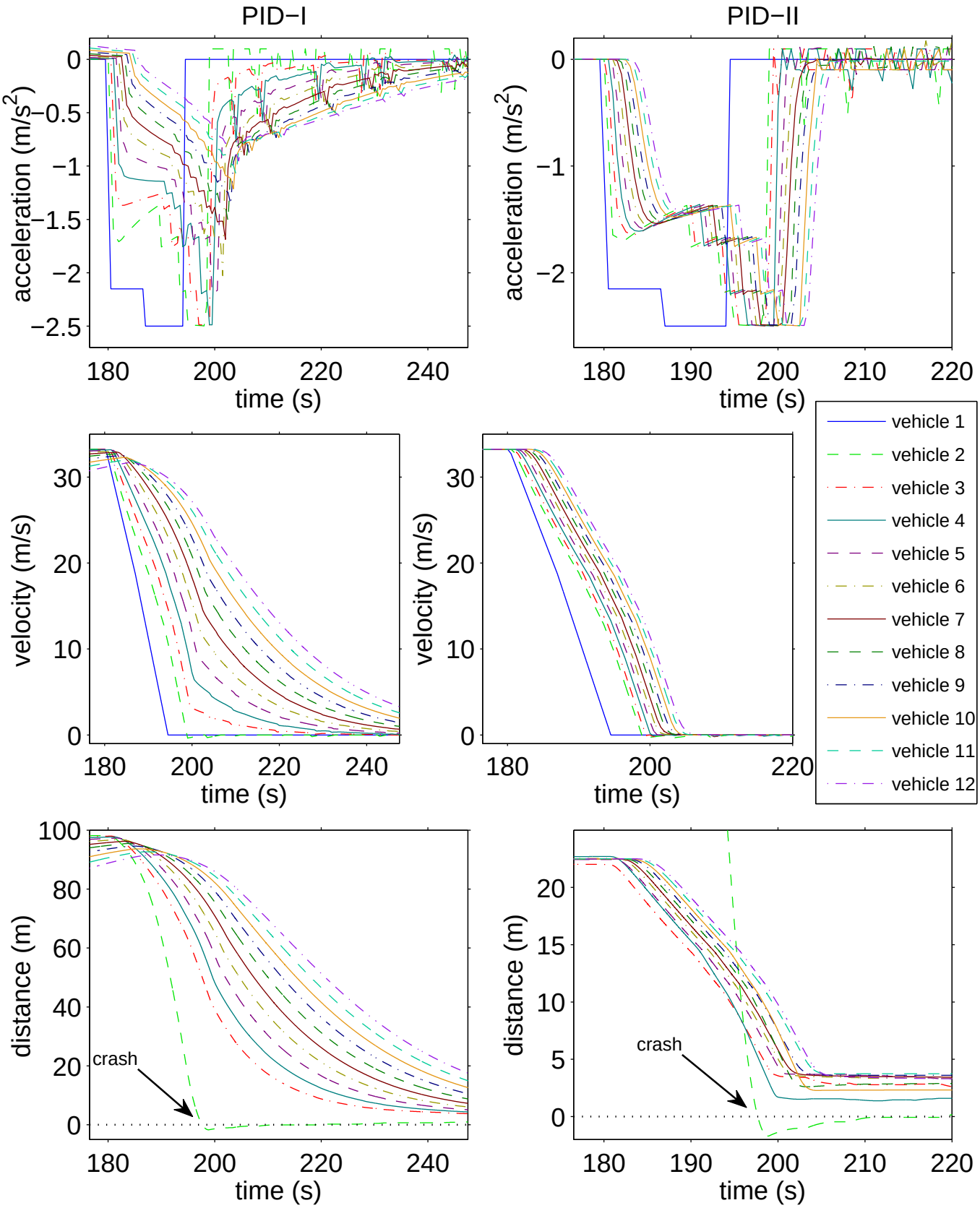
During this same period within scenario 2 (around $t = 200\,\text{s}$), with MPC_2 configurations I and III, the vehicles come to a halt at different vehicle-to-vehicle distances (as close as $d = 0.15\,\text{m}$ for $i = 7$ with MPC_2-I, see Figure 4-4c, and $d = 0.19\,\text{m}$ for $i = 5$ with MPC_2-III). Contrarily to what PID would do, MPC does not try to drive backwards, but keeps its vehicle steady at zero velocity. In all other cases all MPC-controlled vehicles come to a halt at a proper 3 m ($= d_0$) distance from their stalled predecessors.

Velocity trajectories

The PID-based controllers do not brake fast enough such that, for scenario 2, in the first and third of the subscenarios where the leader decelerates to zero, the followers do not reach zero velocity before the leader starts to accelerate again. During the second time the leader brakes to a halt, and when the leader decelerates to zero during scenario 1 (in both cases around $t = 200\,\text{s}$), not braking fast enough causes vehicles $i = 2$ with PID-I, and $2 \leq i \leq 4$ with PID-II and PID-III to come (approximately) to a halt beyond the desired position, which at zero velocity is at $d_0 = 3\,\text{m}$ behind the predecessor. In fact, in these cases the velocity of the vehicles, just like the position, has an overshoot, causing the violations of the minimum velocity constraint for the PID-based CACC controllers that can be seen in Table 4-5[13].

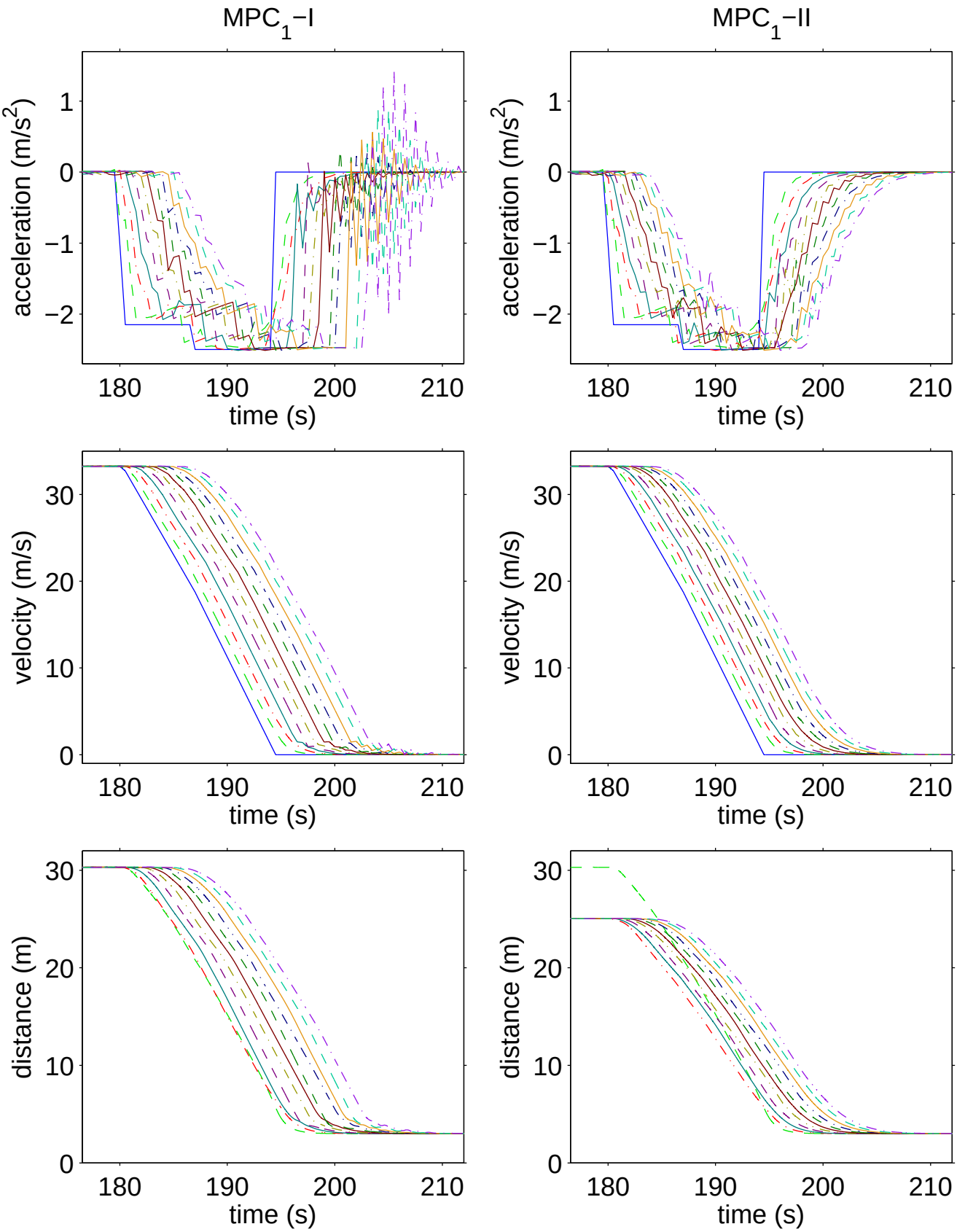
With PID-I it takes a long time for all the vehicles to reach a new target velocity. PID configurations II and III are quicker, because the vehicles further along start earlier with accelerating.

[13]The PID-based CACC algorithm ((3-11) line 3) prevents the velocity to become significantly negative. The air resistance causes minor oscillation of the velocity around zero (amplitude $\sim 10^{-5}\,\text{m/s}$). During these oscillations, a positive velocity at a sample time instant causes the controller to give a negative pedal input, hence the negative peaks in acceleration in Figure 4-4a.



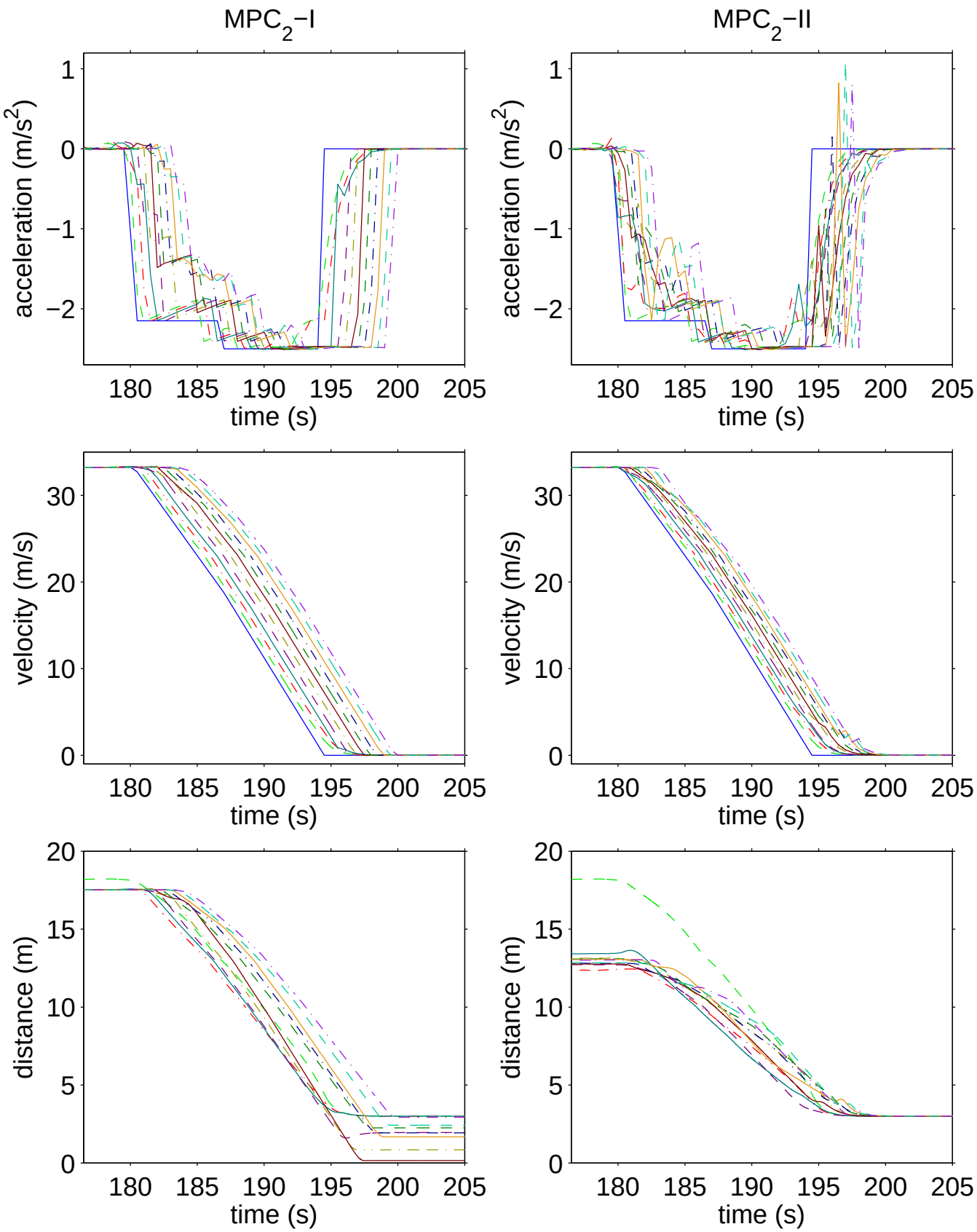
(a) PID I and II, note: vehicle 2 crashes, oscillations around $a = 0$, PID-I is slow.

Figure 4-4: Overview of a part of a subscenario of scenario 2 where the platoon leader decelerates to zero. (The acceleration in this and the other figures is sampled with T_s (=0.5 s) for clarity, avoiding vertically overlapping lines at each sample instant.)



(b) MPC_1, note: no crash, oscillations with MPC_1-I (MPC_1-III is similar here), MPC_1-II shows lighter fluctuations and looks the best of all in this subscenario. (see Figure 4-4a for the legend)

Figure 4-4: (continued) Overview of a part of a subscenario of scenario 2 where the platoon leader decelerates to zero.



(c) MPC_2, note: no crash, close intervehicle distances with MPC_2-I (the small distances staying constant until the reaction to the leader accelerating again, similar with MPC_2-III), slight positive acceleration before deceleration, unwanted acceleration peaks with MPC_2-II. (see Figure 4-4a for the legend)

Figure 4-4: (continued) Overview of a part of a subscenario of scenario 2 where the platoon leader decelerates to zero.

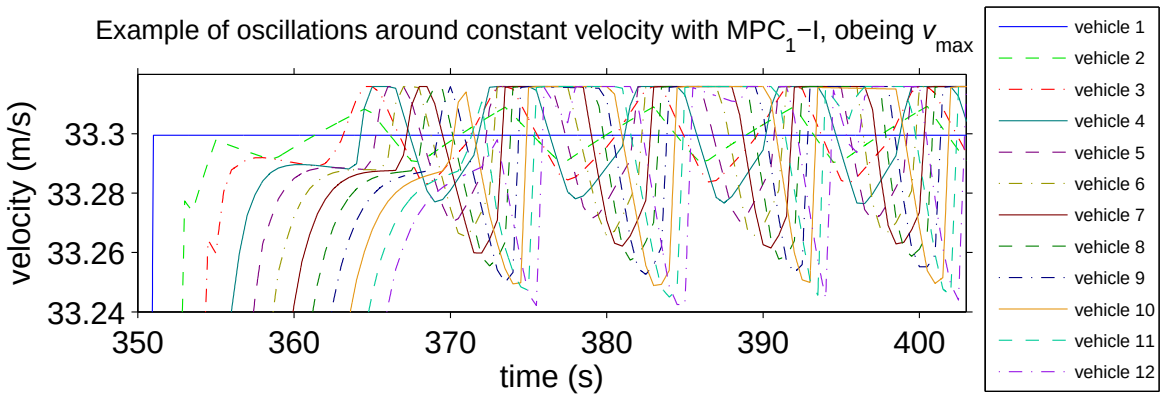


Figure 4-5: Example of mild oscillations around a constant velocity with MPC_1. In this figure it can also be seen that the MPC controller obeys the maximum-velocity constraint. This example, which shows a part of scenario 2, clearly demonstrates the effect of string instability, i.e., the oscillation gets stronger with higher vehicle number.

The velocity trajectories for the MPC-based controllers show unexpected fluctuations, ripples, and in some cases oscillations (see, e.g., Figure 4-4b left middle, and Figure 4-4c right middle), that are the result of undesirable fluctuations and oscillations in acceleration that will be described below (under the heading "Acceleration trajectories"). An example of oscillations in the velocity trajectories are oscillations that occur with the MPC_1-based controllers during a few periods of constant leader-velocity. The amplitude of oscillation grows with vehicle number, but not in time. In Figure 4-5 this is shown for MPC_1-I, during a subscenario, which is chosen, because it also shows that the controller obeys the maximum velocity constraint.

PID-II violates the maximum velocity constraint with scenario 2, which is caused by an overshoot (from $t = 361\,\text{s}$) up to $v = 33.361\,\text{m/s}$ (recall that $v_{\max} \approx 33.33\,\text{m/s}$). According to the performance requirements of Section 3-1-2, a small overshoot like this is acceptable, and its penalty in the calculation of the platoon performance measure P_{platoon} is therefore negligible.

With MPC, the highest velocity the controller allows the vehicle to reach is $33.316\,\text{m/s}$ (see Figure 4-5), which lies just below the velocity $v_{\max}$ used in the maximum velocity constraint. At this velocity, when the MPC "thinks" it gives a force that causes the vehicle to exactly reach $v_{\max}$ by the next sample step ($T_s = 0.5\,\text{s}$ from now), this force exactly cancels out the air drag and friction forces. This difference between expected effect and real effect of the control input is caused by underestimation of the air drag, according to (3-16) and Figure 3-3.

Acceleration trajectories

With PID, some vehicles show fluctuation in acceleration around zero with accidental negative peaks, when having reached zero velocity beyond their desired position, as described above. With the PID-based CACC controllers the vehicles do not reach the maximum acceleration for the simulated scenarios. They do reach the minimum acceleration of $-2.5\,\text{m/s}^2$, but do not exceed this value. This is because the acceleration is bounded according to line 4 of (3-11). The acceleration trajectories for the PID-based controllers are significantly smoother than those for the MPC-based controllers. Apart from the added constraints, the PID controller is nothing more than a weighted sum of the state errors. This means that as long as the desired states, and hence the state trajectories of its predecessors, vary smoothly in time or the PID gains are sufficiently low, the throttle/brake input and the resulting acceleration also

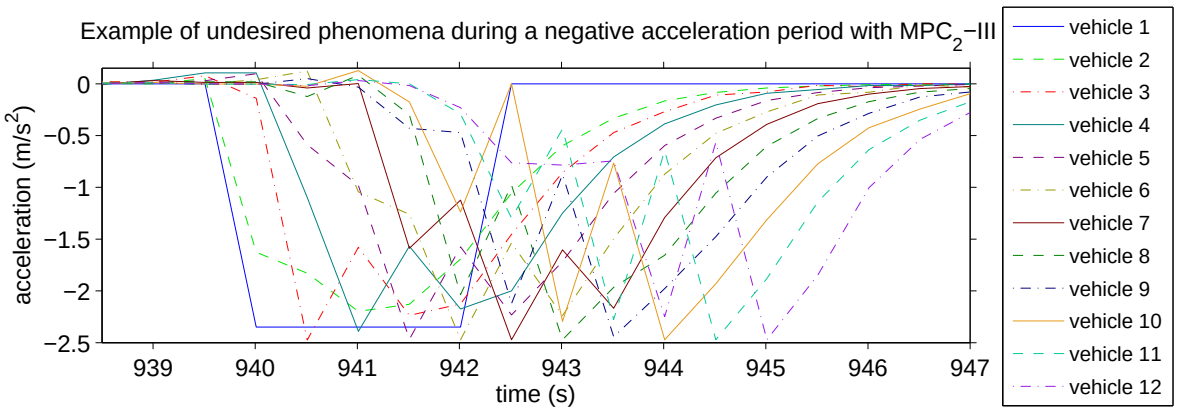


Figure 4-6: First a light increase in acceleration before decreasing, as occurring with all configurations with MPC_2. Also some brief undesirable fluctuations in accelerations is visible here. (This example is part of scenario 2.)

vary smoothly. The platoon leader's acceleration does not always vary smoothly, but the PID acceleration gain after tuning is so low that it does not have a significant negative effect on the smoothness of the state trajectories of the followers.

With MPC_1 configurations I and III, during scenario 2, every time the platoon leader stops accelerating (positively or negatively), and often already when *starting* accelerating, as a (indirect) reaction, the acceleration of the followers from vehicle 3 and onward starts oscillating. For the first vehicles this is a very moderate oscillation (only viewable when closely inspecting), but it grows with vehicle number to an extreme oscillation for the vehicles at the rear end of the platoon. Every time such oscillations occur, the amplitude first grows in time, and then decreases again to zero (see Figures 4-4b top left, 4-8 top, and B-1c). From vehicle $i = 2$ on, the slope of the time-acceleration trajectory of each next vehicle deviates more than that of its follower (for each next vehicle with one time step more delay), resulting in the observed oscillations in acceleration, which indicates string stability. To examine this phenomenon an MPC_2-I controller has been given the same tuning settings as the tuned MPC_1-I controller and the two have been compared for their response to initial situations from which with MPC_1-I vehicles start oscillating. This showed that the described oscillations do not occur when the vehicles communicate their predicted future states (the feature of MPC_2 that MPC_1 lacks). For the reference for its predicted states MPC_1 assumes constant velocity of its predecessors. This is why the control action of the third vehicle ($i = 3$), when its predecessors are accelerating, is not as optimal as expected, which it has to correct for at the next time step. The assumption that vehicle $i = 4$ makes about the future states of $i = 3$ are false to an even greater extent than the assumption $i = 3$ made about the future states of $i = 2$, because of the attempt that $i = 3$ will make to correct for its poor action (unforeseen by $i = 4$). At following time steps, $i = 4$ needs to correct for its poor control action, and so forth. This causes the amplitude of the oscillations to grow with vehicle number.
This phenomenon is the most severe for MPC_1-III (with amplitudes up to $2.5\,m/s^2$, only constrained by the acceleration bounds); with MPC_1-I it is only slightly less severe (with amplitudes up to $2.3\,m/s^2$). In both cases the oscillations occur in all of the subscenarios of scenario 2, whether it should be an easy to follow subscenario, like the final one (leader decelerates in 15 s from 14.85 m/s to 7.35 m/s, then keeps its velocity constant), or a more difficult one. With MPC_1-II these oscillations also occur, but to a much more moderate extent, not necessarily amplified with growing vehicle number, and not during all subscenarios. Still also with this controller the oscillations can be experienced by passengers as unpleasant (mostly

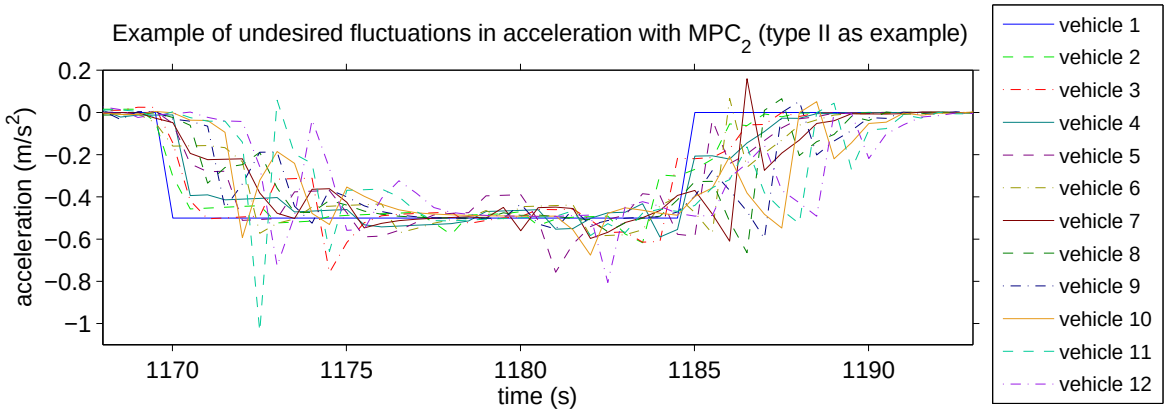


Figure 4-7: Irregular fluctuations in acceleration, as occurring with the MPC_2-based controllers during scenario 2. In this example the vehicles decelerate from 14.85 m/s to 7.35 m/s.

with amplitudes up to $0.25\,\mathrm{m/s^2}$, but occasionally up to $1\,\mathrm{m/s^2}$).
A few times with MPC_1 when the leader has a constant velocity and the followers have already reached that velocity, a longer lasting light oscillation of the acceleration around zero occurs (see Figure 4-5 for corresponding oscillations in velocity).

With MPC_2, every time before starting to decelerate, as a (indirect) reaction to the platoon leader starting to decelerate, the acceleration of the vehicles first increases slightly (see Figure 4-6). Vehicle $i = 2$ starts slightly accelerating 2 seconds ($(N-1)\cdot T_\mathrm{s}$) before the leader starts decelerating. Then each following control update time step ($T_\mathrm{s} = 0.5\,\mathrm{s}$) each next follower starts accelerating. The peak of this acceleration usually first grows with vehicle number i (up to a value between $1\cdot10^{-3}\,\mathrm{m/s^2}$ and $0.12\,\mathrm{m/s^2}$), but from some value of i on it stays approximately constant or decreases with i. Each vehicle then starts decelerating after having had such low positive acceleration for a period of 0.5 s–1 s. The explanation of this phenomenon would be that the expected decrease in velocity within the prediction horizon, that will result from following the deceleration of the predecessor, implies that the desired distance to this predecessor is also expected to decrease. It seems that the weight ratios R_{u_tb}/Q_s and Q_v/Q_s in the MPC performance index (see also (4-1)) are such that it is considered cheaper to first accelerate in order to already decrease the intervehicle distance slightly, before starting to decelerate. This is unexpected, and not ideal behaviour.
This phenomenon occurs as well during scenario 1 as during scenario 2, and the reverse also occurs after the leader starts a period of positive acceleration (i.e., the followers first decelerate slightly before starting to positively accelerate), but with negligible intensity. The MPC_1-based controllers also show this behaviour, however to a much less significant extent (peaks of up to $10^{-2}\,\mathrm{m/s^2}$) and the peaks mainly decrease with increasing vehicle number. This difference between MPC_1 and MPC_2 is caused by the fact that with MPC_1 only $i = 2$ receives future states from its predecessor. The other vehicles do not expect their predecessor to decelerate during the prediction horizon, so they do not perform actions anticipating that.

With the MPC_2-based CACC controllers often the followers' acceleration trajectories show unwanted fluctuations that usually last only up to a few sample steps, but sometimes continue during a longer period (see Figures 4-4c top right, and 4-6 to 4-8). The intensity of these phenomena usually fluctuates with vehicle number (i), and only occasionally as a general trend increases with i. These unwanted fluctuations in acceleration are significantly less extreme

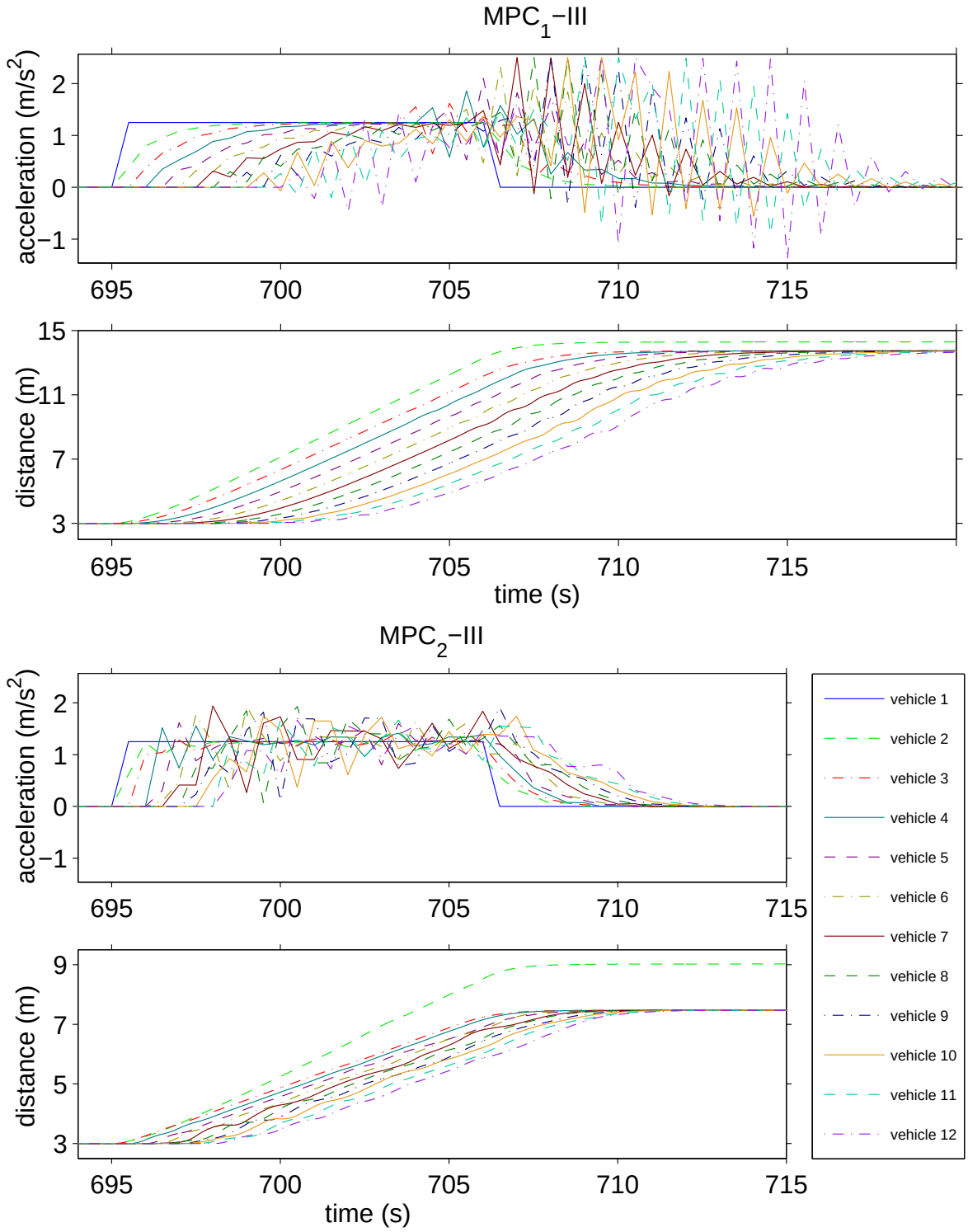


Figure 4-8: Oscillations in acceleration during scenario 2. The plot of MPC_1-III is an example of oscillations that are typical for MPC_1 configurations I and III with this scenario. The plot of MPC_2-III is an example for oscillations that last longer than a few sample periods, as occurring with all three MPC_2-based controllers. In this example the vehicles accelerate from 0 m/s to 13.75 m/s.

than the oscillations that occur with MPC_1 configurations I and III. They do not occur during all of the subscenarios, but they do occur in different situations, i.e., when following a strongly decelerating platoon leader, as well as a very lightly decelerating leader, or a positively accelerating leader. The amplitudes of these oscillations (with MPC_2) usually reach values between $0.15\,\mathrm{m/s^2}$ and $1.0\,\mathrm{m/s^2}$, but are sometimes as large as $1.75\,\mathrm{m/s^2}$.

All MPC-based controllers violate the minimum and maximum acceleration constraints at times, with some vehicles. This is, except for additional violations for MPC_1 configurations I and III caused by their heavy oscillations, always due to respectively underestimation or overestimation of the air drag by the MPC prediction model. This underestimation occurs when the velocity is close to α ($\approx 16.67\,\mathrm{m/s}$), and this overestimation occurs when the velocity has a value that lies between zero and the first intersection (at $v \approx 11.2\,\mathrm{m/s}$) of the PWA approximation of v^2 and the original v^2 plot in Figure 3-3. The violations of the minimum and maximum acceleration constraints by the MPC controllers are acceptably small with absolute violations by at most $0.035\,\mathrm{m/s^2}$ and $0.020\,\mathrm{m/s^2}$ respectively.

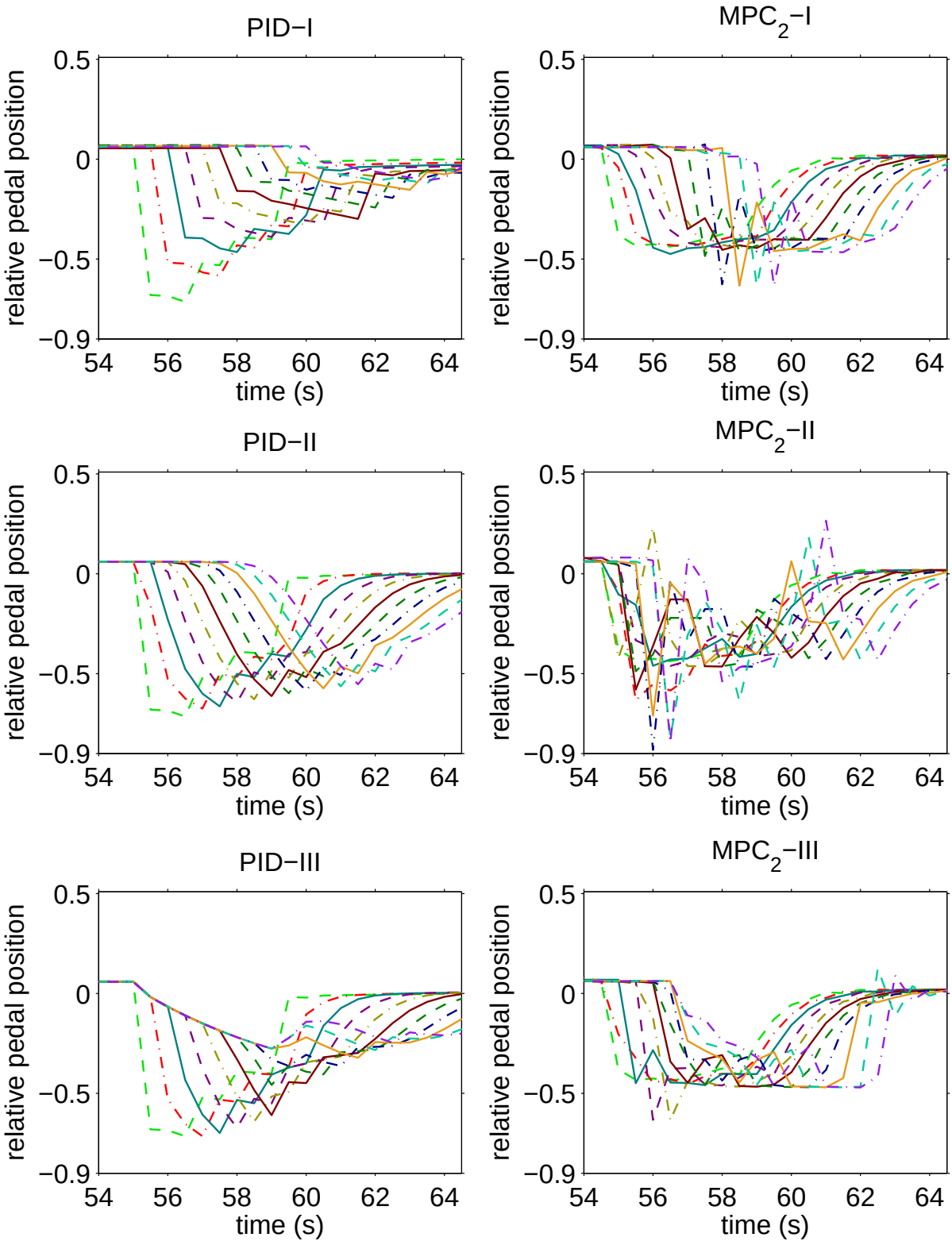
Gear variation

The differences in number of gear changes between the MPC-based controllers with scenario 1 occur at a certain point where different numbers of vehicles perform a down shift in order to increase the traction force while positively accelerating. The PID-based controllers switch gear more than the MPC-based controllers, because they follow a switching scheme that is based on velocity and throttle input, without any effort to try to keep the number of gear switches low, as the MPC algorithm does.

Throttle/brake input variation

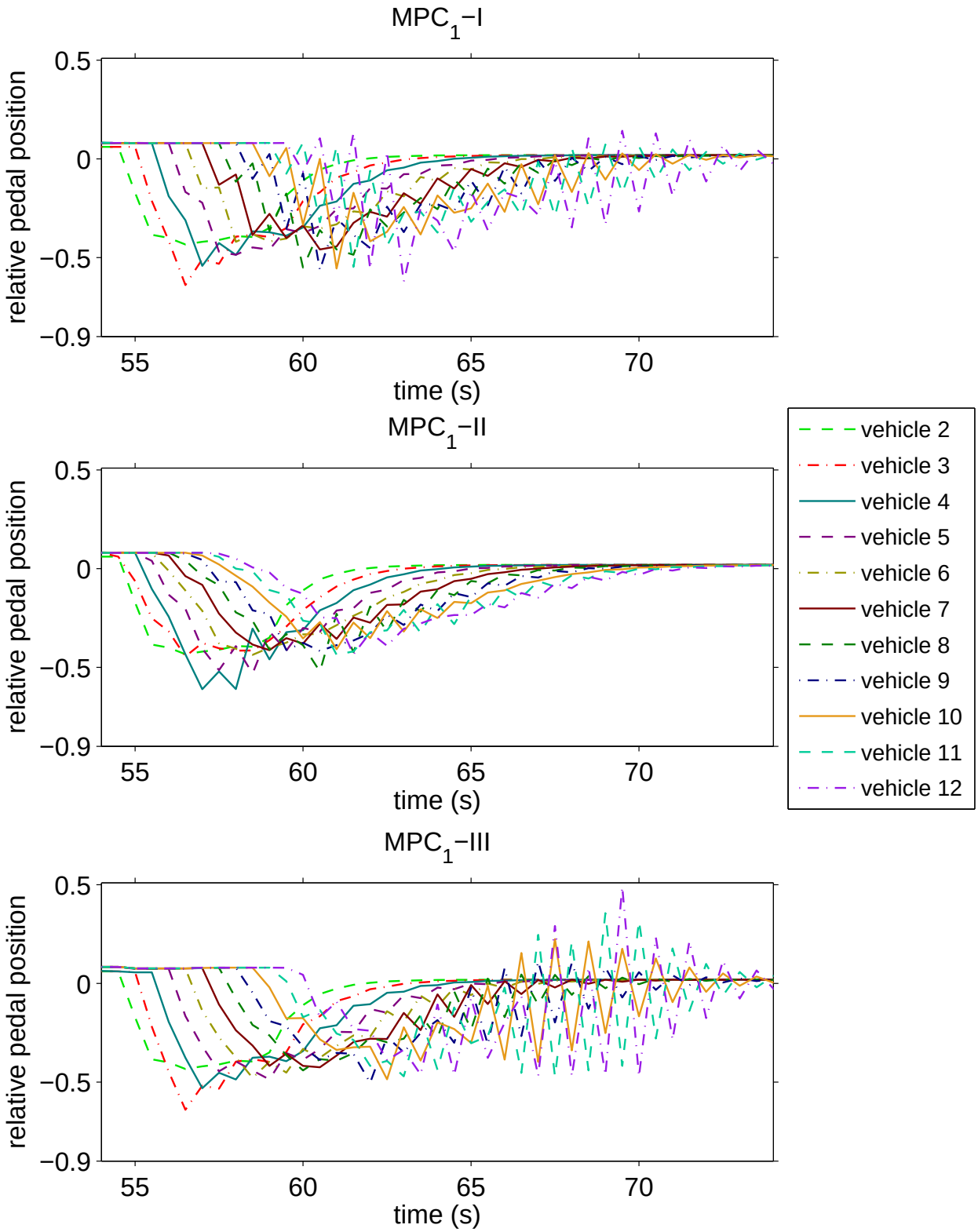
With PID-I, after a constant velocity period every vehicle starts pushing the throttle/brake pedal one time step after its direct predecessor. With configuration II the vehicles start changing their throttle/brake input in pairs, meaning vehicles $i = 2$ and $i = 3$ 0.5 s after the leader, $i = 4$ and $i = 5$ after 1 s, $i = 6$ and $i = 7$ after 1.5 s, and so on, while with PID-III all the followers ($i \geq 2$) start pushing the throttle/brake pedal simultaneously, one sample step after the platoon leader starts accelerating/braking. These differences between the three configurations with PID can be seen in Figure 4-9.

The fluctuations occurring with the MPC-based controllers, that were described above in the subsection on acceleration, can be traced back to similar behaviour of the throttle/brake pedal position sequences in time (see e.g., Figure 4-9). This means that sudden peaks in acceleration are induced by sudden peaks in throttle/brake input, i.e., not by a gear change. Closer examination shows that indeed such fluctuations almost always occur a significant period of time away from gear shift moments. So, even though the MPC algorithm tries to minimize the throttle/brake input variation, the result with respect to this task is very poor. This implies that the tuned weight $R_{u_{\mathrm{tb}}}$ corresponding to this task (as part of the MPC performance index J, see (3-34))—although it is high enough for the tuning scenario—should have been higher, in order to give good performance for other scenarios.
Also, with scenario 2, with PID the throttle/brake variation measure, before being averaged over all vehicles, decreases with vehicle index i, while with MPC the general trend is for this measure to increase with i (except with MPC_1-II, for which it only increases for $i \geq 11$), which is another indication of string instability with MPC (more on this in the next paragraph).



(a)

Figure 4-9: Braking behaviour during a period (of scenario 2) where the platoon leader brakes to decelerate from $v = 10\,\mathrm{m/s}$ to $v = 0\,\mathrm{m/s}$. With PID it can be seen how the vehicles start braking (pairwise) simultaneously or subsequently, depending on the information gathering configuration. With MPC the plots show fluctuations in throttle/brake pedal position that result in similar fluctuations in acceleration.



(b)

Figure 4-9: (Continued) Braking behaviour during a period (of scenario 2) where the platoon leader brakes to decelerate from $v = 10\,\mathrm{m/s}$ to $v = 0\,\mathrm{m/s}$. With PID it can be seen how the vehicles start braking (pairwise) simultaneously or subsequently, depending on the information gathering configuration. With MPC the plots show fluctuations in throttle/brake pedal position that result in similar fluctuations in acceleration.

String stability margin

Recall, from (3-47), that a positive string stability margin means that when in front of the platoon a vehicle decelerates, the intensity of the deceleration of vehicles as a reaction to this manoeuvre generally decreases with increasing vehicle number. If the value of f_{stab} in Table 4-5 (see (3-47)), representing the string stability margin, is negative, then during the simulation there is at least one braking period during which the trend is that the intensity of the deceleration increases towards the back of the platoon, or several vehicles decelerate significantly stronger than their direct predecessors. Therefore, from a negative string stability margin it is concluded that the platoon is string unstable.

PID has no problem keeping a positive string stability margin for the two traffic scenarios. With the desired distance depending on velocity, it was already expected that, after proper tuning, the controllers would make the platoons string stable (as stated in Section 2-3-2). The high stability margin for PID-I can be ascribed to indirectly resulting from the relatively big time headway (h_0).

With the MPC-based controllers it appears to be much more common to have some vehicles brake more intensely than others independent of their index i. That does not necessarily pose a problem, as long as these fluctuations in maximum deceleration with i are not too big, and the trend is for the maximum of the deceleration to decrease with vehicle number, for each braking period. However, the unwanted oscillations and other fluctuations in acceleration, growing or otherwise fluctuating with vehicle number (already being a sign of string instability), cause deceleration peaks that cause the string stability margin measure (f_{stab}) to become negative. This indeed means that the MPC-based controllers are assessed not to bring string stability. As reported above, inspection of the time-acceleration plots showed that the intensity of the oscillations with MPC_1 configurations I and III always grow with increasing vehicle number. For the other MPC-based controllers the unwanted fluctuations do not consistently grow with vehicle number. The plots therefore show that MPC_1 configurations I and III have in fact the worst string instability, which is not obvious from the values in Table 4-5, due to the imperfection of the stability margin measure.

Overall performance

Just from the results with traffic scenario 1 it could be said that the general performance of MPC_2 is better than that of MPC_1, of which the performance in turn is better than that of PID, and that configuration II is the best configuration to use with MPC, while configuration III is the best in combination with PID, and MPC_2-II performs the best of all. However, all of the controllers perform unsatisfactory with validation scenario 2, hence the infinite platoon performance measure. The PID-based controllers have an infinite platoon performance measure value (P_{platoon}) because of the distance between the second and first vehicle in the platoon becoming negative at some point, representing a crash. The MPC-based controllers receive the infinite value of P_{platoon} for their inability to achieve string stability. These controllers have been tested with increased prediction horizon, up to $N = 9$, while the other tuning settings were still fixed to their values in Table 4-4). However, none of the increased values of N resulted in string stability for any of the MPC-based controllers for scenario 2.

From scenario 2, even though all controllers have an infinite platoon performance measure, from the measures of the different performance aspects and from the plots showing the simulation results, still the overall performance of the different controllers can be compared. First of

all it can be stated that PID performs worse than MPC, because a crash is significantly more severe than (even very) uncomfortable peaks or oscillations in acceleration. Furthermore, it can be seen that with PID, configuration II performs much better than configuration I on the average distance, but worse on the other performance aspects. Overall PID-II seems better than PID-I. The same holds for PID-III, which, on top of that, performs better on all aspects than PID-II. Therefore the overall performance of configuration III can be considered the best of the three configurations with PID. With MPC_1, however, configuration II clearly performs the best overall, and configuration I and III perform more or less equally badly. This while with MPC_2 there are not such big overall differences between the three configurations. The performance of configurations I and III with MPC_2 might be considered worse than that of configuration II, because with both of these former configurations, a vehicle comes uneasily close to its predecessor at some point.
These differences may be explained as follows. PID is a pretty straightforward control method, such that two vehicles having the same CACC controller more or less apply similar control actions. Therefore, if with PID the control actions of all vehicles are partly based on the leader's states, the vehicles can closely follow, especially since the second vehicle in the platoon undergoes only slow changes. This was more or less expected after the literature review. MPC, using a prediction model that contains discrete—in this case binary—variables, on the other hand, might give a significantly different control signal for an only marginally different situation. This makes that with such an MPC-based CACC controller the control actions of each individual vehicle in the platoon will differ more, no matter which of the information gathering configurations is used. It is probably this difference between MPC and PID that causes the difference in added value of using the platoon leader's states. The second predecessor is much closer in the platoon than the leader (for most vehicles), which makes using its information still advantageous for MPC_1. MPC_2, however, receives expected future states from its predecessor, which in turn result from predictions that were based on communicated expected future states of the second predecessor (and so on). Therefore, indirectly MPC_2 has much more information on what will happen in front than MPC_1 has, reducing the added value of directly communicated information from its second predecessor.

4-4 Recommendations for the case study

After having reviewed the results, an important question is why do the MPC-based CACC controllers designed here give such instability. For the MPC_1 variant it was explained above that the oscillations result from assuming the velocity of its predecessor(s) to stay constant. For the MPC_2 variant it is not that obvious what causes the problems (in a direct sense). In both cases it is clear that the weighting factor for input variation ($R_{u_{\text{tb}}}$) in the MPC performance index (J) is not high enough to *prevent* such oscillations. Because the value of this weight was determined by tuning, the fact that $R_{u_{\text{tb}}}$ has been assigned too low a value could be caused by choices made in relation to any (or a combination) of the following four items:

1. the parameters to be tuned (simultaneously)
2. the (extent of the) traffic scenario used for tuning
3. the construction of the platoon performance measure (minimised when tuning)
4. the settings for the optimisation algorithm

Therefore, below, these four subjects are addressed, and suggestions are given about alterations that may result in better performing controllers, and, with that, results from which more relevant conclusions can be drawn. Unfortunately, the suggested improvements could not be implemented within the scope of this project, due to the limited time span.

1. **Tuning all tunable parameters simultaneously for each controller.**
In the case study, when tuning the controllers, some weight parameters (k_s, k_v, k_a, Q_v, and $R_{u_{\text{tb}}}$, R_j) were tuned with configuration I and then given the same value for the corresponding configurations II and III. Moreover, the parameter N (prediction horizon) was kept constant during the tuning of all other tunable parameters, and only after that tuned for each controller separately. This is not the best approach. The optimal values for the weight parameters, just mentioned, for configuration I may not be optimal for configuration II or III. Furthermore, a higher prediction horizon would have most probably resulted in a better performance, when combined with some different values for the other tunable parameters.
Therefore, all the parameters just mentioned should be tuned simultaneously with the other tunable parameters for each controller in which they play a part. This means retuning the weight parameters (mentioned above) for each configuration, and tuning N simultaneously with the other parameters. Of course, then a penalty needs to be added for using a higher value of N, since this results in higher computational costs. For the tuning it should be taken into account that the optimisation computation time for such a nonlinear, nonconvex optimisation problem in general increases exponentially with the number of parameters to be tuned.

2. **Expanding the tuning scenario.**
In order to improve the probability of detecting unwanted behaviour, more different subscenarios and more vehicles should be added to the tuning scenario. This will in principle only linearly increase the computation time.

3. **Improving the platoon performance measure.**
For its performance assessment of all the controllers on both traffic scenarios (and on several test scenarios) the platoon performance measure appears to be grading all the performance aspects as intended, except for the stability margin (see also Table 4-5). It was already observed in Section 4-3-2 that the difference in severity of the string instability between the controllers is not in all cases adequately represented in the value of the string stability margin measure. Looking at (3-47) it can be seen that the string stability performance measure P_{stab} can have a good (low) value even if one of the braking periods proved the controller to be slightly string unstable. This is undesirable, because if a CACC-vehicle combination is said to be string stable, a platoon of (any number of) vehicles should be stable in any circumstances within the operational region specified. This means that if any of the braking periods in the simulation gives instability, P_{stab} should have infinite value, just as it now has if all the braking periods give instability, or one or more such periods are unstable to a higher extent. The derivation of the stability performance measure (3-47) can be improved by first deriving such a measure for each braking period (p), before averaging these measures to obtain one overall performance measure for stability. The proposed improved equation for P_{stab} is

$$\begin{aligned} f_{\text{stab},p} &= 1 - \sqrt{\sum_{i=3}^{N_{\text{veh}}} \left(\frac{a_{\min,p,i}}{a_{\min,p,i-1}} \right)^2} \Bigg/ (N_{\text{veh}} - 2), \text{ for } p = 1, \ldots, N_{\text{per}} \\ P_{\text{stab,p}} &= \begin{cases} c_{\text{stab}} / f_{\text{stab,p}} & \text{if } f_{\text{stab,p}} > 0 \\ \infty & \text{if } f_{\text{stab,p}} \leq 0 \end{cases} \\ P_{\text{stab}} &= \frac{1}{N_{\text{per}}} \sum_{p=1}^{N_{\text{per}}} P_{\text{stab},p} \end{aligned} \tag{4-2}$$

Furthermore, the derivation of a stability measure has been based on the braking periods, because often in literature for empirical evaluation of string stability only braking is considered. However, it should also be based on the positive acceleration periods, because the platoon

leader suddenly positively accelerating is also a disturbance that travels upstream[14]. Then $a_{\min,p,i}$ in (4-2) should be replaced by $a_{\text{peak},p,i}$, where for braking periods $a_{\text{peak},p,i}$ will be the same as $a_{\min,p,i}$, and for positive acceleration periods $a_{\text{peak},p,i}$ will represent $a_{\max,p,i}$, i.e., the maximum acceleration for vehicle i during period p.
Having an improved measure to rate the performance with respect to string stability margin should improve the chance of finding a set of values for the tunable parameters, during the tuning stage, that induces a better margin of string stability for other traffic scenarios.
Furthermore, the occurrence of all the unwanted fluctuations in acceleration and throttle input means that the value of $R_{u_{\text{tb}}}$ in the MPC performance index (3-34b) is too low, relative to the other weights in this index. Although for the tuning scenario no such fluctuations occurred with the tuned controllers, increasing the weight ($W_{u_{\text{tb}}}$ in eq:generalP) of the pedal position variation measure that is part of the platoon performance measure might help find more optimally tuned parameters with respect to keeping the input variation low.

4. **Adding more starting points to the multi-start search, and increasing the total number of allowed iterations within the optimisation algorithm during tuning.**
Because the variation of the platoon performance measure with varying values of the tunable parameters is highly non-convex, the simulated annealing optimisation algorithm may not have found the global optimum within the time given (the maximum allowed number of function evaluations, i.e., tuning scenario simulations, was in the order of 10^4, divided over searches from two starting points). Using more starting points with simulated annealing may (further) improve the tuning results. Additionally, if the available time permits, it could be attempted to improve the tuning results by allowing for more function evaluations within the optimisation algorithm (in addition to already increasing the number of iterations allowed in accordance with the increased number of parameters to be tuned simultaneously). Also, if available, the MATLAB function "simulannealbnd" can be used instead of the simulated annealing function for MATLAB as designed for this case study (described in Appendix A-7). However, for the tuning scenario the optimised platoon performance measure does appear to have a low value, corresponding to promising results. Therefore, it is expected by the author that the suggestions concerning the previous three subjects are more important when retuning in order to find better results.

4-5 Summary

In this chapter a case study has been performed with simulated platoons of SMARTs, undergoing scenarios that consist of a small series of common traffic scenarios. In these platoons all followers use the same CACC controller, while the leader has been assigned a specific time-velocity sequence. The tunable parameters have been tuned by minimising a performance measure applied to the results from simulating a five vehicle platoon, undergoing a certain traffic scenario. Then, for each tuned controller a simulation of a twelve-vehicle platoon undergoing another, longer, traffic scenario has been carried out, after which the performance of the controllers has been evaluated and compared. The results show that all controllers behave very poorly. The PID-based controllers allow a collision to happen, while the MPC-based controllers cause sudden peaks and oscillations in throttle/brake pedal input and acceleration, and string instability. Finally, the results have been discussed and recommendations have been formulated that should result in a better performance.

[14] In literature, when the concept of empirically studying string stability is described in general, the peaks in time are taken regardless whether these peaks are minima or maxima [12].

Chapter 5

Conclusions and recommendations

5-1 Project summary

The goal of this project was to answer the following question concerning CACC (as stated in Chapter 1):

> Does the combination of MPC with using communicated information from two other vehicles, being either the first two predecessors or the direct predecessor and the platoon leader, result in an increase of performance, compared with PID (with or without using such additional information) or with MPC using just the direct predecessor's information? And in what aspects, and to what extent, is the performance different?

In order to answer this question, for each of these combinations a CACC controller has been designed. Several constraints have been presented, which the controllers should—at least approximately—obey, representing acceleration comfort bounds, velocity bounds (minimum velocity being zero), and a minimum intervehicle distance bound. The control variables are the throttle and brake pedal positions, and the gears. To the author's best knowledge, this thesis is the first to explicitly include the gears as control variable in CACC.
It is chosen for the desired distance to the predecessor(s), as used by the controller, to be proportional to the vehicle's velocity. When only the direct predecessor states are used by the CACC controller (configuration I), the desired velocity and acceleration are equal to this predecessor's velocity and acceleration respectively. When the states of the second predecessor are used (configuration II), the desired states are a weighted average of the desired states based on following the direct predecessor and the desired states based on following the second predecessor at a larger distance. When the platoon leader's states are used (configuration III), a similar construction is applied, except that the leader's position is not used in the algorithm. The second vehicle always uses the configuration that only looks at the direct predecessor (i.e., configuration I).
With PID, velocity, position, and acceleration errors are fed back, with a simple automatic gear transmission scheme added. Approximately obeying constraints (except the minimum distance constraint) has been achieved by cutting off the output of the PID control law at values that according to an available vehicle model would cause the vehicle's states to stay at the upper or lower bounds for their constraints.
The model used for vehicle simulation in the case study is linearised to obtain the model that

the MPC controllers use for their predictions. One of the nonlinearities, the air drag, which is a function of the squared velocity, is approximated by a piecewise affine function consisting of two straight lines. To convert the hybrid model that results from this into a model of a linear nature, and to similarly handle other nonlinearities, the model is rewritten into a so called mixed logical dynamical form [3]. It is chosen to have the MPC minimise the predicted future state errors and input variation. The MPC-based CACC controllers for simplicity use only position and velocity states. These controllers are designed in two variants, one that uses communicated predicted future states to derive the reference signals from, and one that assumes the velocity of its predecessors constant when calculating future desired states.
For the case study, each controller has been tuned by applying multi-start simulated annealing to optimise the performance of a simulated platoon of five vehicles undergoing a sequence of everyday (non-extreme) traffic scenarios. The leader is given a time-acceleration sequence with an initial velocity, and the other four vehicles use CACC. The platoon performance includes the averaged intervehicle distance, the throttle/brake pedal position variation, the gear variation, the number and severity of constraint violations, and the margin in platoon string stability. To save time, the weight factors of the states (and inputs with MPC) that occur in the CACC algorithm are only tuned for the configuration I controllers, and given the same value for the corresponding configuration II and III controllers, and the prediction horizon for MPC is only tuned afterwards, when all the other tunable parameters have been given their already tuned values.
After tuning, for each controller a simulation has been performed with the tuned CACC controller implemented in vehicles undergoing a validation traffic scenario. This validation scenario consists of a different and larger set of typical everyday traffic scenarios than the set used for tuning, and is simulated with a platoon of twelve vehicles. The performance of each controller is evaluated by considering the platoon performance calculated from these simulations, as well as by looking at the time-state and time-input graphs of the simulations.

The most important results found are:

- PID
  With PID, the time-state sequences are relatively smooth, but the vehicles are rather slow in their following task, and during the validation scenario the second vehicle crashes into the platoon leader. When only information from the direct predecessor is used, the vehicles keep a very big distance to their predecessors (an average distance of 55.2 m for the validation scenario). Also using information from the second predecessor improves the controller (the average distance decreases to 17.2 m). Also using information from the platoon leader gives an even better improvement (with an average distance of 16.6 m). With PID the trend is for vehicles to brake less intensively than their predecessors, meaning string stability is achieved in the simulated scenarios.
- MPC
  With MPC, when no information about future states is being shared (variant 1, with average distances for the validation scenario of 18.0 m, 15.4 m and 17.3 m for configurations I, II, and III respectively), during the validation scenario heavy oscillations occur frequently in the throttle/brake input and the acceleration when only the direct predecessor's states or in addition also the leader states are used. The intensity of these oscillations grows from the front to the rear of the platoon. When the states of the two direct predecessors are used, only much smaller fluctuations occur (not necessarily growing from the front to the rear of the platoon). These are also less severe than the fluctuations occurring with MPC variant 2, although they would sometimes still be unpleasant for passengers. Also communicating predicted future states between vehicles with MPC (variant 2), resulted in the most responsive controllers with the smallest average intervehicle distances (11.0 m, 8.7 m, and 9.2 m for configurations I, II, and III respectively for the validation scenario). However, also unwanted

short-term fluctuations in the throttle/brake input and the acceleration are observed, usually not specifically growing from the front to the rear of the platoon, and of which several are expected to be uncomfortable. With all six MPC-based controllers, for each individual vehicle these fluctuations do not keep growing in time, but diminish. However, they do sometimes—with some of the controllers more, and more consistently, than with others—increase from vehicle to vehicle, indicating string instability.

5-2 Conclusions

Although all the designed CACC controllers performed poorly with the validation scenario, it is expected that retuning with properly improved tuning conditions will result in better performing controllers. Therefore, it cannot be stated indefinitely that conclusions drawn from the results of the case study performed here will hold in general. That is why only preliminary conclusions will be formulated here, based on the case study and the expectation that the observed qualitative differences in performance between the controllers, that these conclusions are based on, will be similar in the case of better tuned versions of these controllers.
With respect to the research question presented in Section 5-1 the preliminary conclusions are:

1. **MPC-based CACC is safer than PID-based CACC, and therefore preferred as a control method for CACC.**
   Even though the PID-based CACC controllers were safe for the tuning scenario, they let two vehicles crash during the validation scenario. PID, by itself, cannot anticipate dangerous situations as well as MPC can. The first follower of the vehicle that introduces a disturbance (usually the platoon leader) has the highest risk of colliding into its predecessor. Its followers have a lower risk for that because they apply similar control actions, especially when they use information from more than one predecessor, combined with the fact that the platoon shows string stability for the simulated scenarios. The MPC-based controllers kept the vehicles safe (although not always comfortable) for the relatively small platoon lengths simulated in the case study. The MPC-based controllers that caused the worst string stability would, without retuning, probably also cause collision in some larger platoons. However, for the other MPC-based controllers, the peaks and oscillations in acceleration did mostly only *fluctuate* from the front to the rear of the platoon, but not necessarily grow. Therefore, it is expected that MPC-based CACC can also be safe for larger platoons. Because safety is more important than comfort, from the results of this case study it can be deduced with a high degree of certainty that MPC is indeed superior over PID as control method for CACC.

2. **With MPC-based CACC it is preferred to receive through communication, in addition to the current states of the direct predecessor, at least the current states of the second predecessor and/or the predicted future states from the direct predecessor, in order to achieve better string stability.**
   This means that better string stability can be achieved if the CACC controller has some indication of what the direct predecessor is going to do. Here, the current states of the second predecessor give an indirect indication of the near-future states of the direct predecessor. This conclusion is deduced from the fact that the MPC-based CACC controllers that do not use predicted future state information from predecessors and that do not use any state information from the second predecessor, caused by far the worst string instability in the case study.

Furthermore, with respect to designing CACC controllers, based on the case study the following conclusion can be formulated:

3. **It is not easy to design a CACC controller that obtains smooth throttle/brake trajectories with MPC. With PID this seems easier to achieve, but at the cost of a slower response.**
   The right tuning of the weight factor that punishes throttle/brake increments as part of the MPC performance index is very important, because the results from the case study showed how too low a value for this weight can cause sudden peaks and oscillation in acceleration. Because too high a value of this weight could give other problems (e.g., low responsiveness to braking predecessors, possibly causing the need for excessive braking later on, past the initial prediction horizon), it is not easy to find the right value for it. With PID the observed throttle/brake trajectories were much smoother. But it should be noted that in order to achieve this, the acceleration gain after tuning was negligible. This also resulted in a slow response, which in turn, resulted in the need for a large headway for the second vehicle in the platoon, and in the fact that this vehicle still could not avoid a crash during the validation scenario.

5-3 Recommendations for further research

During the course of this project several ideas for further research have been formed. The most relevant of these will be addressed briefly here.

5-3-1 Recommendations concerning the specific CACC controllers as designed in this project

First, to possibly achieve more useful results, from which better conclusions can be drawn about the CACC controllers as designed in Chapter 3, these controllers would need to be retuned with applying the following improvements (as suggested, and further explained, in Section 4-4)[1]:
- Tuning all tunable parameters simultaneously for each controller.
- Expanding the tuning scenario (to include more different subscenarios and more vehicles).
- Defining a platoon performance measure that better detects string instability and more strongly punishes throttle/brake input variation.
- Changing the settings for the optimisation algorithm used for tuning, in order to introduce more starting points for multi-start simulated annealing, and possibly to allow for more function evaluations.

Some of these will increase the time needed to tune the controllers (even exponentially for the first improvement listed).
After retuning, if some of the controllers give satisfying results, a natural continuation would be to study the performance of the better ones on simulations with more realistic vehicle models, and, after that, in case of positive results, while implemented in real test vehicles. Also, the vehicle mass or the road/air resistance could be varied to investigate robustness.

Then, still, some suggestions can be given for alterations to the CACC controllers that might further improve such controllers:

[1]The insight that these issues should have been handled differently was gained during a later stage of the project (when processing the simulation results, after tuning), at a point when the time needed to implement these improvements and retune all controllers was no longer available within the scope of this project.

- **Using more regions for the piecewise affine approximation in the model for MPC.**
  There is no direct evidence that suggests that there is a relation between the fact that the linearised MPC prediction model deviates from the vehicle simulation model, and the observed problems with MPC. However, it can be investigated whether using more regions in the piecewise affine approximation—causing the model to be more accurate—improves the CACC performance. When doing that, it should be considered whether an observed improvement in performance outweighs the increase in computational costs.
- **Allowing for emergency braking.**
  CACC controllers as designed here should be combined with some sort of a collision avoidance algorithm that takes over in emergency situations, obviously then being allowed to break the comfort constraint on deceleration. For configuration II, a simple example of such a collision avoidance algorithm could be the following:

  > When at least one of the two predecessors brakes with an acceleration lower than $a_{\min} - \varepsilon_1$, with $\varepsilon_1 > 0$ (e.g., $\varepsilon_1 = 0.1\,\mathrm{m/s^2}$ or $\varepsilon_1 = 0.05 \cdot |a_{\min}|$), then the constraint $a \geq a_{\min}$ is temporarily removed from the set of constraints. After that, this constraint is reinstated as soon as
  >
  > $$\begin{aligned} &a(k) \geq a_{\min} + \varepsilon_2 \quad \& \quad \min(a_{i-1}(k-n), a_{i-2}(k-m)) \geq a_{\min} \\ &\forall n \in \left\{0, 1, \ldots, \mathrm{floor}\left(\tfrac{T_1}{T_\mathrm{s}}\right)\right\} \quad \forall m \in \left\{0, 1, \ldots, \mathrm{floor}\left(\tfrac{T_2}{T_\mathrm{s}}\right)\right\} \end{aligned} \tag{5-1}$$
  >
  > where $\varepsilon_2 > 0$ (e.g., $\varepsilon_2 = 0.1\,\mathrm{m/s^2}$), $T_2 > T_1 \geq T_\mathrm{s}$ (T_s is the sampling time, $T_\mathrm{s} = 0.5\,\mathrm{s}$ in this project, e.g., $T_1 = 1\,\mathrm{s}$, $T_2 = 1.5\,\mathrm{s}$), a represents acceleration, and indices $i-1$ and $i-2$ refer to the direct and second predecessor respectively.

  With MPC, another option to avoid collisions in more extreme circumstances is to make the minimum acceleration constraint a soft constraint [3], instead of the hard constraint it was in this project, i.e., replace the constraint

  $$a(k) \geq a_{\min} \tag{5-2}$$

  by

  $$\begin{cases} a(k) & \geq & a_{\min} - \varepsilon(k) \\ \varepsilon(k) & \geq & 0 \end{cases} \tag{5-3}$$

  To the MPC performance index of (3-34), which is minimised at each control time step, the following term is then added:

  $$\sum_{n=1}^{N} \|Q_\varepsilon \varepsilon(k+n)\|_1 \tag{5-4}$$

  where Q_ε is a penalty weight that should be relatively large, such that only if the problem would otherwise become infeasible the acceleration will become lower than $a_{\min}$.
- **Only looking at the direct predecessor when slowing down to a halt.**
  It is suggested, if information from other vehicles besides the direct predecessor is used, to lower the influence of this information to none when approaching zero velocity. This is to prevent the second predecessor's position to cause the own "desired position" to become dangerously close to, or unnecessarily far from the direct predecessor when coming to a halt. The latter situation may unnecessarily cause an infeasible optimisation problem.
- **Finding a better braking model.**
  In the vehicle models used here for simulation and for MPC prediction, braking is modelled as following the same dynamics as positive acceleration, but with a negative pedal input value. This way, e.g., the influence of braking is dependent on current gear in the same way as positive acceleration is. If problems arise while implementing a CACC controller based on such a model in a more realistic simulation or in a real vehicle, it is advisable to find a better braking model to use for the MPC prediction model, and for simulations.

- **Favouring higher gears to protect the environment.**
  If desired, the MPC-based CACC controller could be made more environmentally friendly by adding a term to the MPC performance index that comprises the negated gear number with a small weight factor.
- **Considering more predecessors.**
  If using communicated information from the second predecessor indeed turns out to improve the CACC performance, compared with only using the direct predecessor's information, then it could be investigated whether, and up to which value of n, using information of the closest n predecessors ($n \leq$ total number of predecessors) gives an even further improvement.
- **Comparing with other control methods.**
  A CACC controller that uses another control method than PID or MPC, e.g., fuzzy logic, can be designed and applied with the same configurations as applied here, with respect to which other vehicles to use information from as reference, in order to investigate if this control method can give a better performance than MPC can.

5-3-2 Recommendations concerning CACC systems in general

The CACC controllers designed in this project assume that all the vehicles in the platoon (except maybe the leader) are equal and use the same type of CACC. In addition, the CACC controllers assume that the vehicle weight and the road conditions are non-changing, and that all vehicles will stay in the platoon, and no new vehicles will join. Besides these assumptions, everyday traffic situations are assumed, meaning no extreme actions of predecessors. It is recommended to extend the CACC system with features (or combine them with another intelligent systems) that can handle the following situations:

- **Platoon splitting.**
  When the platoon splits, caused by vehicles in front accelerating permanently away from the remaining part of the platoon, or vehicles leaving by changing lanes, this could result in the remaining platoon having a new leader or in the direct predecessor(s) that the CACC controller needs information from being a different vehicle than before. The CACC controller should be made able to adapt to the new situation by re-identifying vehicles from which information is needed. Furthermore, the CACC controller can be integrated with a higher-level controller that coordinates the platoon splitting procedure and the (relative) movements of platoons.
- **Platoon merging.**
  Two platoons or one platoon and a new vehicle should be able to join to form one new platoon. This situation can occur as a result of two platoons catching up or vehicles merging from a neighbouring lane or an on-ramp. In the latter situation, not only the CACC systems of (some) vehicles need to adjust to the insertion and reordering of vehicles in the platoon. Also, to maintain autonomous longitudinal driving, the CACC system would need to be combined with a higher level cooperative merging system. This means that the merging vehicles should—in a direct or indirect way—communicate with the vehicles in the platoon, and the platoon should make room for the new vehicles to merge.
- **Different parameter values for different types of vehicles or for changing circumstances.**
  Usually vehicles within a platoon will differ from each other in several characteristics, e.g., mass, size, aerodynamics. Moreover, such characteristics may even change for one vehicle with changing circumstances. Therefore, the values of the time headway h_0 and possibly other CACC parameters should depend on such characteristics. Furthermore, to improve the performance of MPC-based CACC systems for changing circumstances, the value of parameters from the vehicle model corresponding to some of these characteristics can be measured (or otherwise identified) at regular time intervals, and be adapted in the vehicle model used by MPC.

- **Looking at the direct follower.**
  An idea that could be investigated, especially for its value in mixed traffic is for the CACC to also use communicated/measured data from its direct follower. This way, the vehicle can avoid close encounters with a follower that has a much slower reacting (C)ACC or no ACC at all.
- **Designing CACC for emergency situations as well as for daily following.**
  CACC should not only be able to safely perform close following in daily traffic scenarios, but also to react adequately to suddenly strongly braking predecessors, so as to avoid collisions (as already mentioned above). Since also lateral vehicle control systems are being studied and developed, at some point, the combination of longitudinal and lateral control can provide for even better collision avoidance features, enabling the CACC controller to steer around the predecessor if that provides for a better chance of avoiding a (severe) collision under the given circumstances (i.e., the presence or absence of a neighbouring lane with or without nearby vehicles, the velocity difference with the predecessor, the road friction, etc.).
- **Applicability in mixed traffic.**
  It will be a long time before all vehicles on the road will be equipped with some version of CACC. If CACC controllers are to be introduced in vehicles that take part in everyday traffic, the percentage of traffic that is equipped with these controllers will at first only grow slowly. This means that these controllers have to be able to longitudinally control the vehicle in mixed CACC/manual traffic. If the direct predecessor is not able to communicate the required information, some of it (e.g., its position and relative velocity) should be measured. If the predecessor is not using CACC, or a CACC controller with different settings, it will behave differently than is to be expected of a CACC-driven predecessor with similar CACC settings as those of the own vehicle. If the second predecessor cannot communicate, its states cannot be used by the controller, unless the direct predecessor's measurements of the second predecessor's states are used in some way. Solutions to these and other problems that are consequences of mixed traffic should be incorporated in an extended design of CACC controllers.
- **Multi-level control for the vehicle following within a platoon.**
  In this thesis each vehicle had its own CACC controller. At each time step the MPC-based CACC controller receives (expected) states from the predecessor(s). Then it optimises a sequence of control actions over a prediction horizon. All vehicles do this simultaneously. This means that, even though the CACC controller can react to the previous actions of its predecessors, and possibly to previously optimised paths of the predecessor(s), it cannot instantaneously anticipate the control actions the other vehicles are calculating at the same moment. Also, each vehicle computes only its own optimal trajectory, based on a given or assumed trajectory of other vehicles. This raises the idea that the trajectories of the vehicles might be more optimal if the control actions of all vehicles in a platoon are optimised simultaneously by one MPC algorithm. This algorithm then optimises the state trajectories and input sequences of all the vehicles over the prediction horizon. It can be executed by some form of roadside network. The centralised controller would receive the vehicles' states through communication, and at each control update step send the respective throttle/brake position and gear choice to each of the vehicles. For this, the controller also should have received vehicle model information of all vehicles in the platoon. The effect of such centralised control in a simulation of a platoon of vehicles can be investigated. In reality the corresponding optimisations would be highly computationally demanding. Therefore, if simulations indeed show that centralising the longitudinal vehicle control bring significant benefits, then probably strategies need to be devised and investigated that lower the computational demands. E.g., it could be considered to investigate multi-level control, where a higher-level controller computes optimal paths for the vehicles in the platoon, using MPC, for a small number of relatively large time steps (e.g., $T_{\text{contr,central}} = 1\,\text{s}$, $N_{\text{central}} = 5$). Here, the MPC alorithm still tries to keep the corresponding throttle, brake, and gear changes low. Then between these subsequent time steps lower-level controllers within the vehicles can control the throttle/brake

and gear to follow the received optimal velocity and distance trajectories, with a higher control update frequency (e.g., $T_{\text{contr,vehicle}} = 0.1\,\text{s}$, and $N_{\text{vehicle}} = 5$ or a PID controller). This way, e.g., the higher-level controller can use a simplified model, and the lower-level on-board controllers can be more precise on a local level.

Furthermore, strategies similar to the ones applied in this project may be useful to apply, in an adapted manner, to control problems other than longitudinal vehicle following on roads. Recommendations along this line are:

- **Applying cooperative control to laterally control a number of neighbouring vehicles.**
  The concept of cooperative control to keep a safe distance, as applied in this thesis, may also be applicable in the lateral direction when there is room for more than two vehicles side by side. If the vehicles in neighbouring lanes communicate and coordinate their lateral and longitudinal motions, they may be able to drive closer together laterally. This does not just mean that the lanes may be made smaller, but even that eventually, in the long term, there is no need to have the road divided in stationary lanes. The number of vehicles driving side by side, and the lateral distances between these vehicles can just be based on circumstances such as velocity, road surface quality, straightness of the road, and vehicle saturation. This can be combined with a tidal flow system, i.e., a system that devides the entire width of the road between the two opposing driving directions, depending on the sizes of the two opposing traffic flows.
- **Application of CACC to railway traffic.**
  Usually for railway traffic a signalling block system is used [32]. A railway track is divided into blocks with, in the Netherlands, a length of 1 km to 1.5 km each. Safety is preserved by denying trains access to a block, by means of signalling, if a part of this block is already occupied by another train. This causes the minimum space between any two trains on the same track (or crossing tracks) to be relatively large. Therefore, for some railway networks, a moving block system has already been implemented. This means that the distance between two trains must be at least the sum of a predetermined safety distance and the required braking distance of the following train [19, 32]. This enables closer following distances. However, it seems likely that introducing platooning with MPC-based CACC and possibly the use of information from more than one predecessor can further increase the throughput on railway tracks. This would then replace the (moving) signalling block system. So, it is recommended to investigate the possible benefits of the application of CACC (combined with platoon forming/platoon splitting procedures) with configurations like the ones designed here, possibly with increasing the number of trains to use communicated information from, to railway traffic.
- **Application of CACC to aircraft in formation flight.**
  Research has shown that formation flight of aircraft can reduce fuel consumption, airspace congestion, and operator workload [4, 34]. It is therefore proposed to apply the strategy of using communicated states of other vehicles to obtain state reference values in combination with MPC, as designed in this thesis, to design an onboard control system to control the velocity and relative position of the aircraft.

Appendix A

Additional information on MPC models and methods

A-1 Gear-velocity restrictions

In this section the dependence of the velocity (state x_2) and the binary variables (δ_n with $n = 1, \ldots, 5$) on the time step k will be omitted for ease of notation. The twelve inequalities that alternatively represent the restriction that each gear can only be chosen between its corresponding velocity bounds as given in (3-26) (two of which are shown by (3-27)) are

$$\begin{aligned}
x_2 &\geq (1-\delta_1)(1-\delta_2)(1-\delta_3)\cdot v_{\text{low},1} + \big(1-(1-\delta_1)(1-\delta_2)(1-\delta_3)\big)\cdot v_{\min}\\
x_2 &\leq (1-\delta_1)(1-\delta_2)(1-\delta_3)\cdot v_{\text{high},1} + \big(1-(1-\delta_1)(1-\delta_2)(1-\delta_3)\big)\cdot v_{\max}\\
x_2 &\geq \delta_1(1-\delta_2)(1-\delta_3)\cdot v_{\text{low},2} + \big(1-\delta_1(1-\delta_2)(1-\delta_3)\big)\cdot v_{\min}\\
x_2 &\leq \delta_1(1-\delta_2)(1-\delta_3)\cdot v_{\text{high},2} + \big(1-\delta_1(1-\delta_2)(1-\delta_3)\big)\cdot v_{\max}\\
x_2 &\geq (1-\delta_1)\delta_2(1-\delta_3)\cdot v_{\text{low},3} + \big(1-(1-\delta_1)\delta_2(1-\delta_3)\big)\cdot v_{\min}\\
x_2 &\leq (1-\delta_1)\delta_2(1-\delta_3)\cdot v_{\text{high},3} + \big(1-(1-\delta_1)\delta_2(1-\delta_3)\big)\cdot v_{\max}\\
x_2 &\geq \delta_1\delta_2(1-\delta_3)\cdot v_{\text{low},4} + \big(1-\delta_1\delta_2(1-\delta_3)\big)\cdot v_{\min}\\
x_2 &\leq \delta_1\delta_2(1-\delta_3)\cdot v_{\text{high},4} + \big(1-\delta_1\delta_2(1-\delta_3)\big)\cdot v_{\max}\\
x_2 &\geq (1-\delta_1)(1-\delta_2)\delta_3\cdot v_{\text{low},5} + \big(1-(1-\delta_1)(1-\delta_2)\delta_3\big)\cdot v_{\min}\\
x_2 &\leq (1-\delta_1)(1-\delta_2)\delta_3\cdot v_{\text{high},5} + \big(1-(1-\delta_1)(1-\delta_2)\delta_3\big)v_{\max}\\
x_2 &\geq \delta_1(1-\delta_2)\delta_3\cdot v_{\text{low},6} + \big(1-\delta_1(1-\delta_2)\delta_3\big)\cdot v_{\min}\\
x_2 &\leq \delta_1(1-\delta_2)\delta_3\cdot v_{\text{high},6} + \big(1-\delta_1(1-\delta_2)\delta_3\big)\cdot v_{\max}
\end{aligned} \tag{A-1}$$

These are six inequality pairs, each of which corresponds to a specific gear number. Each such pair states that for the combination of binary variables that correspond to this gear number, the velocity can only lie between the velocity bounds of this gear. Furthermore, for any other combination of the three binary variables, each inequality should remain valid. Therefore, in (A-1) the terms containing $v_{\min}$ or $v_{\max}$ are added, expanding the bounds to $v_{\min}$ and $v_{\max}$ respectively (instead of zero, which would be the case if these terms were omitted).
The systems of inequalities that arise from applying (2-25) to the definition of δ_4 and δ_5 by

(3-28) are

$$\delta_4 = \delta_1\delta_2 \quad \rightarrow \quad \begin{cases} -\delta_1 + \delta_4 & \leq & 0 \\ -\delta_2 + \delta_4 & \leq & 0 \\ \delta_1 + \delta_2 - \delta_4 & \leq & 1 \end{cases} \tag{A-2}$$

$$\delta_5 = \delta_1\delta_3 \quad \rightarrow \quad \begin{cases} -\delta_1 + \delta_5 & \leq & 0 \\ -\delta_3 + \delta_5 & \leq & 0 \\ \delta_1 + \delta_3 - \delta_5 & \leq & 1 \end{cases} \tag{A-3}$$

For the gears up to 6 the variables δ_2 and δ_3 are never both equal to 1 simultaneously, as can be seen in Table 2-2. Therefore $\delta_2\delta_3$ can be set to zero. After expanding the inequalities of (A-1), substituting for $\delta_4 = \delta_1\delta_2$ and $\delta_5 = \delta_1\delta_3$, and setting $\delta_2\delta_3 = 0$ and $v_{\text{low},1} = v_{\min} = 0$, the system of linear inequalities that constrain the allowed velocity range for each gear becomes

$$\begin{aligned} 0 &\leq x_2 \\ (v_{\text{high},1} - v_{\max})(\delta_1 + \delta_2 + \delta_3 - \delta_4 - \delta_5) &\leq v_{\text{high},1} - x_2 \\ v_{\text{low},2}(\delta_1 - \delta_4 - \delta_5) &\leq x_2 \\ (v_{\text{high},2} - v_{\max})(-\delta_1 + \delta_4 + \delta_5) &\leq v_{\max} - x_2 \\ v_{\text{low},3}(\delta_2 - \delta_4) &\leq x_2 \\ (v_{\text{high},3} - v_{\max})(-\delta_2 + \delta_4) &\leq v_{\max} - x_2 \\ v_{\text{low},4}\delta_4 &\leq x_2 \\ -(v_{\text{high},4} - v_{\max})\delta_4 &\leq v_{\max} - x_2 \\ v_{\text{low},5}(\delta_3 - \delta_5) &\leq x_2 \\ (v_{\text{high},5} - v_{\max})(-\delta_3 + \delta_5) &\leq v_{\max} - x_2 \\ v_{\text{low},6}\delta_5 &\leq x_2 \\ -(v_{\text{high},6} - v_{\max})\delta_5 &\leq v_{\max} - x_2 \end{aligned} \tag{A-4}$$

A-2 Linearisation of gear dependence for the MPC model

In this section the dependence on the time step k of u_{tb}, j, δ_n, and z_n with $n = 1, \ldots, 5$ will be omitted for ease of notation. In Section 3-3-3, in the part leading up to (3-29), three approaches to linearise the nonlinear relation between the gear (discrete variable j) and the gear dependence factor ($b(j)$), shown in Table 2-2 and Figure 3-4, are proposed. Here, each of these approaches will be worked out to demonstrate which new variables would need to be introduced for each method, and to clarify why the third method is chosen.

1. Straight-line approximation (applied in [11]):
   Approximate $b(j)$ by $b_{\text{s}}(j)$, where

   $$b_{\text{s}}(j) = \beta_0 + j\beta_1 \tag{A-5}$$

   with β_0 and β_1 real valued scalars. After substituting for (3-25), this becomes

   $$b_{\text{s}}(\delta_1, \delta_2, \delta_3) = \beta_0 + \beta_1 + \beta_1\delta_1 + 2\beta_1\delta_2 + 4\beta_1\delta_3 \tag{A-6}$$

   Recall that the model (3-24) contains the product $b\big(j(k)\big) \cdot u_{\text{tb}}(k)$. This product can now be replaced by

   $$b_s(j)u_{\text{tb}} = (\beta_0 + \beta_1)u_{\text{tb}} + \beta_1\delta_1 u_{\text{tb}} + 2\beta_1\delta_2 u_{\text{tb}} + 4\beta_1\delta_3 u_{\text{tb}} \tag{A-7}$$

Since δ_1, δ_2, and δ_3 are now inputs to the model (instead of j), products between any of these binary variables and the other input u_{tb} still contribute nonlinearities to the model. To get rid of these nonlinearities, new continuous auxiliary variables can be defined according to

$$z_1 = \delta_1 u_{\text{tb}}, \quad z_2 = \delta_2 u_{\text{tb}}, \quad z_2 = \delta_3 u_{\text{tb}} \tag{A-8}$$

Then (2-24) can be applied to alternatively express these definitions with systems of linear inequalities.

2. Two-region PWA approximation:
The nonlinear function $b(j)$ can be approximated by two affine lines, each of which is valid in a different region of the gear domain, intersecting at the boundary between the two regions, as shown in Figure 3-4. From optimising the mean squared errors between the possible two-region PWA approximations and the original values for the SMART model, it is found that for the best approximation the boundary between the two regions lies between $j = 2$ and $j = 3$. This turns out to be very convenient concerning the number of variables, since now instead of needing to introduce a new binary variable to encode the regions, the binary variables that already encode the gear can be used. From Table 2-2 it can be seen that the variables δ_2 and δ_3 are both zero for each of the first two gears (first region), while they are never zero simultaneously for any of the other gears (second region). Now the approximation of $b(j)$ can be given by

$$b_{\text{pwa2}}(j) = \begin{cases} \beta_{01} + j\beta_{11}, & \text{for } j \leq 2 \\ \beta_{02} + j\beta_{12}, & \text{for } j \geq 3 \end{cases} \tag{A-9}$$

This can be put into a single-line equation as

$$b_{\text{pwa2}}(j, \delta_2, \delta_3) = (1 - \delta_2)(1 - \delta_3)(\beta_{01} + j\beta_{11} - \beta_{02} - j\beta_{12}) + \beta_{02} + j\beta_{12} \tag{A-10}$$

After eliminating j by substituting for (3-25), and simplifying (applying $\delta_2\delta_2 = \delta_2$, $\delta_3\delta_3 = \delta_3$, and $\delta_2\delta_3 = 0$) , this becomes

$$\begin{aligned} b_{\text{pwa2}}(\delta_1, \delta_2, \delta_3) = \delta_1\beta_{11} + \delta_2(\beta_{02} + 3\beta_{12} - \beta_{01} - \beta_{11}) + \delta_3(\beta_{02} + 5\beta_{12} - \beta_{01} - \beta_{11}) \\ + \delta_1\delta_2(\beta_{12} - \beta_{11}) + \delta_1\delta_3(\beta_{12} - \beta_{11}) + \beta_{01} + \beta_{11} \end{aligned} \tag{A-11}$$

After substituting for $\delta_4 = \delta_1\delta_2$ and $\delta_5 = \delta_1\delta_3$, which were already defined in (3-28), this becomes

$$\begin{aligned} b_{\text{pwa2}}(\delta_1, \delta_2, \delta_3, \delta_4, \delta_5) = \delta_1\beta_{11} + \delta_2(\beta_{02} + 3\beta_{12} - \beta_{01} - \beta_{11}) \\ + \delta_3(\beta_{02} + 5\beta_{12} - \beta_{01} - \beta_{11}) + \delta_4(\beta_{12} - \beta_{11}) \\ + \delta_5(\beta_{12} - \beta_{11}) + \beta_{01} + \beta_{11} \end{aligned} \tag{A-12}$$

From this it is clear to see that the product $b_{\text{pwa2}}(\delta_1, \delta_2, \delta_3, \delta_4, \delta_5) \cdot u_{\text{tb}}$, replacing the term $b(j(k)) \cdot u_{\text{tb}}(k)$ in the model (3-24), gives five products between binary input or dummy (-state) variables and the continuous input variable u_{tb}. These products still make the model nonlinear. To get rid of these nonlinearities, these products need to be replaced by new continuous auxiliary variables, defined as follows

$$z_1 = \delta_1 u_{\text{tb}}, \quad z_2 = \delta_2 u_{\text{tb}}, \quad z_3 = \delta_3 u_{\text{tb}}, \quad z_4 = \delta_4 u_{\text{tb}}, \quad z_5 = \delta_5 u_{\text{tb}} \tag{A-13}$$

As with the first approach above, (2-24) can be applied to derive five corresponding systems of linear inequalities, which are given below by (A-16).

3. Six-region PWA approximation:
   This approach is applied similar to the previous approach, except that now six regions are defined, where each region lies around another value of j. Since j is discrete, each region can be reduced to just the corresponding value of j, and each corresponding affine line is then just reduced to one point being the value of $b(j)$ at that gear. The regions are encoded the same way as the gear by δ_1, δ_2, and δ_3. So now, $b(j)$ can be replaced by

$$\begin{aligned} b_{\text{pwa6}}(\delta_1,\delta_2,\delta_3) &= (1-\delta_1)(1-\delta_2)(1-\delta_3)b_1 + \delta_1(1-\delta_2)(1-\delta_3)b_2 \\ &\quad +(1-\delta_1)\delta_2(1-\delta_3)b_3 + \delta_1\delta_2(1-\delta_3)b_4 \\ &\quad +(1-\delta_1)(1-\delta_2)\delta_3 b_5 + \delta_1(1-\delta_2)\delta_3 b_6 \qquad \text{(A-14)} \\ &= \delta_1(b_1-b_2) + \delta_2(b_3-b_1) + \delta_3(b_5-b_1) \\ &\quad +\delta_1\delta_2(b_1-b_2-b_3+b_4) + \delta_1\delta_3(b_1-b_2-b_5+b_6) + b_1 \end{aligned}$$

   where $\delta_2\delta_3 = 0$ is applied. After substituting for the already defined $\delta_4 = \delta_1\delta_2$ and $\delta_5 = \delta_1\delta_3$, this becomes:

$$\begin{aligned} b_{\text{pwa6}}(\delta_1,\delta_2,\delta_3,\delta_4,\delta_5) &= \delta_1(b_1-b_2) + \delta_2(b_3-b_1) + \delta_3(b_5-b_1) \\ &\quad + \delta_4(b_1-b_2-b_3+b_4) + \delta_5(b_1-b_2-b_5+b_6) + b_1 \qquad \text{(A-15)} \end{aligned}$$

   Similar as with the second approach, above, here the product $b_{\text{pwa6}}(\delta_1,\delta_2,\delta_3,\delta_4,\delta_5)\cdot u_{\text{tb}}$ results in products of input or dummy variables and the continuous input variable u_{tb}. Therefore also here the five continuous auxiliary variables of (A-13) need to be defined. The linear inequalities that result from applying (2-24) to these definitions are

$$\begin{aligned} z_n &\leq \delta_n u_{\text{tb,max}} = \delta_n \\ z_n &\geq \delta_n u_{\text{tb,min}} = -\delta_n \\ z_n &\leq u_{\text{tb}} - u_{\text{tb,min}}(1-\delta_n) = u_{\text{tb}} + (1-\delta_n) \\ z_n &\geq u_{\text{tb}} - u_{\text{tb,max}}(1-\delta_n) = u_{\text{tb}} - (1-\delta_n) \end{aligned} \qquad \text{(A-16)}$$

   for $n = 1,\ldots,5$.

So, to summarise, for none of the three approaches to linearise the gear dependence new binary dummy variables are needed. However, all approaches require the introduction of new continuous auxiliary variables, and corresponding systems of inequalities. The first approach (straight-line approximation) only requires the introduction of three such variables, while the other two approaches (respectively two, and six region PWA approximation) need an additional two more of such variables, i.e., five in total. However, in the decision which approach to choose (in Section 3-3-3) it is also taken into consideration how well the linearised model compares with the original model.

A-3 MLD representation of linearised vehicle model used for MPC

After all the nonlinearities in the vehicle model have been linearised in Section 3-3-3 (when arriving at (3-32)), the linearised model can be written in MLD form (as presented in (2-26)). The first line of this model is given by

$$x(k+1) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \qquad \text{(A-17a)}$$

with

$$A = \begin{bmatrix} 1 & T_s \\ 0 & a_1 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{T_s}{m} b_1 & 0 & 0 & 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ r & 0 & 0 & -T_s \mu g \end{bmatrix},$$

$$B_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ a_2 - a_1 & \frac{T_s}{m}(b_2 - b_1) & \frac{T_s}{m}(b_3 - b_1) & \frac{T_s}{m}(b_5 - b_1) & \frac{T_s}{m}(b_1 - b_2 - b_3 + b_4) & \frac{T_s}{m}(b_1 - b_2 - b_5 + b_6) \end{bmatrix},$$

$$u(k) = \begin{bmatrix} u_{tb}(k) \\ \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \end{bmatrix}, \quad \delta(k) = \begin{bmatrix} \delta_0(k) \\ \delta_4(k) \\ \delta_5(k) \\ \delta_\mu(k) \end{bmatrix}, \quad z(k) = \begin{bmatrix} z_0(k) \\ z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix} \tag{A-17b}$$

A-4 MPC prediction model

Based on (A-17a), the predicted future states can be written as function of the current state and current and future inputs, according to

$$\begin{bmatrix} \hat{x}(k+1) \\ \vdots \\ \hat{x}(k+N) \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix} x(k) + \begin{bmatrix} I & 0 & \cdots & 0 \\ A & I & & 0 \\ \vdots & & \ddots & \\ A^{N-1} & A^{N-2} & \cdots & I \end{bmatrix} \cdot \left(\begin{bmatrix} B_1 & & 0 \\ & \ddots & \\ 0 & & B_1 \end{bmatrix} \cdot \begin{bmatrix} u(k) \\ \vdots \\ u(k+N-1) \end{bmatrix} + \begin{bmatrix} B_2 & & 0 \\ & \ddots & \\ 0 & & B_2 \end{bmatrix} \cdot \begin{bmatrix} \hat{\delta}(k) \\ \vdots \\ \hat{\delta}(k+N-1) \end{bmatrix} + \begin{bmatrix} B_3 & & 0 \\ & \ddots & \\ 0 & & B_3 \end{bmatrix} \cdot \begin{bmatrix} \hat{z}(k) \\ \vdots \\ \hat{z}(k+N-1) \end{bmatrix} \right) \tag{A-18}$$

where $\hat{x}(k+n), \hat{\delta}(k+n)$, and $\hat{z}(k+n)$ are the predictions of respectively x, δ, and z for prediction time step $k+n$. After defining

$$\tilde{x}(k) = \begin{bmatrix} \hat{x}(k+1) \\ \vdots \\ \hat{x}(k+N) \end{bmatrix}, \quad \tilde{u}(k) = \begin{bmatrix} u(k) \\ \vdots \\ u(k+N-1) \end{bmatrix}, \quad \tilde{\delta}(k) = \begin{bmatrix} \hat{\delta}(k) \\ \vdots \\ \hat{\delta}(k+N-1) \end{bmatrix},$$

$$\tilde{z}(k) = \begin{bmatrix} \hat{z}(k) \\ \vdots \\ \hat{z}(k+N-1) \end{bmatrix}, \quad A^* = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad A_b^* = \begin{bmatrix} I & 0 & \cdots & 0 \\ A & I & & 0 \\ \vdots & & \ddots & \\ A^{N-1} & A^{N-2} & \cdots & I \end{bmatrix}, \tag{A-19}$$

$$B_1^* = A_b^* \cdot \begin{bmatrix} B_1 & & 0 \\ & \ddots & \\ 0 & & B_1 \end{bmatrix}, \quad B_2^* = A_b^* \cdot \begin{bmatrix} B_1 & & 0 \\ & \ddots & \\ 0 & & B_2 \end{bmatrix}, \quad B_3^* = A_b^* \cdot \begin{bmatrix} B_3 & & 0 \\ & \ddots & \\ 0 & & B_3 \end{bmatrix}$$

the predicted states equation becomes

$$\tilde{x}(k) = A^* x(k) + B_1^* \tilde{u}(k) + B_2^* \tilde{\delta}(k) + B_3^* \tilde{z}(k) \tag{A-20}$$

The prediction form of the inequality part of an MLD model formulation was given in (3-38), and is repeated here as

$$E_1^* \cdot \tilde{u}(k) + E_2^* \cdot \tilde{\delta}(k) + E_3^* \cdot \tilde{z}(k) \leq E_4^* \cdot \tilde{x}(k) + E_5^* \tag{A-21}$$

The matrices of this inequality are built from matrices $E_1, \ldots, E_5$ that make up the second line of the MLD model (2-26) giving the model inequalities, and are given by

$$E_1^* = \begin{bmatrix} E_1 & & 0 \\ & \ddots & \\ 0 & & E_1 \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \begin{matrix} \updownarrow N \text{ blocks} \\ \\ \updownarrow 5N \end{matrix}, \quad E_2^* = \begin{bmatrix} E_2 & & 0 \\ & \ddots & \\ 0 & & E_2 \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \begin{matrix} \updownarrow N \text{ blocks} \\ \\ \updownarrow 5N \end{matrix}, \quad E_3^* = \begin{bmatrix} E_3 & & 0 \\ & \ddots & \\ 0 & & E_3 \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \begin{matrix} \updownarrow N \text{ blocks} \\ \\ \updownarrow 5N \end{matrix}$$

$$E_4^* = \left[\begin{array}{cccccccccc}
0 & & & & & & & & & \\
E_4 & & & & & & & \mathbf{0} & & \\
 & & E_4 & & & & & & & \\
 & & & & \ddots & & & & & \\
 & \mathbf{0} & & & & & E_4 & & & \\
 & & & & & & & E_4 & & 0 \\
\hline
1 & 0 & & & & & & & & \\
 & & 1 & 0 & & & & \mathbf{0} & & \\
 & & & & \ddots & & & & & \\
 & & & & & 1 & 0 & & & \\
 & \mathbf{0} & & & & & & 1 & 0 & \\
 & & & & & & & & 1 & 0 \\
\hline
0 & 1 & & & & & & & & \\
 & & 0 & 1 & & & & \mathbf{0} & & \\
 & & & & \ddots & & & & & \\
 & & & & & 0 & 1 & & & \\
 & \mathbf{0} & & & & & & 0 & 1 & \\
 & & & & & & & & 0 & 1 \\
\hline
0 & -1 & & & & & & & & \\
 & & 0 & -1 & & & & \mathbf{0} & & \\
 & & & & \ddots & & & & & \\
 & & & & & 0 & -1 & & & \\
 & \mathbf{0} & & & & & & 0 & -1 & \\
 & & & & & & & & 0 & -1 \\
\hline
0 & 1 & & & & & & & & \\
0 & -1 & 0 & 1 & & & & \mathbf{0} & & \\
 & & & & \ddots & & & & & \\
 & \mathbf{0} & & & & 0 & -1 & 0 & 1 & \\
 & & & & & & & 0 & -1 & 0 \;\; 1 \\
\hline
0 & -1 & & & & & & & & \\
0 & 1 & 0 & -1 & & & & \mathbf{0} & & \\
 & & & & \ddots & & & & & \\
 & \mathbf{0} & & & & 0 & 1 & 0 & -1 & \\
 & & & & & & & 0 & 1 & 0 \;\; -1
\end{array}\right]
\begin{matrix} \\ \\ \\ \\ \\ \\ \left.\vphantom{\begin{matrix}1\\1\\1\\1\\1\\1\end{matrix}}\right\} d_{\min} \\ \left.\vphantom{\begin{matrix}1\\1\\1\\1\\1\\1\end{matrix}}\right\} v_{\min} \\ \left.\vphantom{\begin{matrix}1\\1\\1\\1\\1\\1\end{matrix}}\right\} v_{\max} \\ \left.\vphantom{\begin{matrix}1\\1\\1\\1\\1\end{matrix}}\right\} a_{\min} \\ \left.\vphantom{\begin{matrix}1\\1\\1\\1\\1\end{matrix}}\right\} a_{\max} \end{matrix}, \quad
E_5^*(k) = \left[\begin{array}{c}
E_5 + E_4 x(k) \\ E_5 \\ \vdots \\ E_5 \\ \hline
\hat{x}_{1,i-1}(k+1) - l_{\text{veh}} - d_{\min} \\ \vdots \\ \hat{x}_{1,i-1}(k+N) - l_{\text{veh}} - d_{\min} \\ \hline
-v_{\min} \\ \vdots \\ -v_{\min} \\ \hline
v_{\max} \\ \vdots \\ v_{\max} \\ \hline
-T_{\text{s}} a_{\min} - x_2(k) \\ -T_{\text{s}} a_{\min} \\ \vdots \\ -T_{\text{s}} a_{\min} \\ \hline
T_{\text{s}} a_{\max} + x_2(k) \\ T_{\text{s}} a_{\max} \\ \vdots \\ T_{\text{s}} a_{\max}
\end{array}\right]
\begin{matrix} \updownarrow N \text{ blocks} \\ \updownarrow N \\ \updownarrow N \\ \updownarrow N \\ \updownarrow N \\ \updownarrow N \end{matrix} \tag{A-22}$$

When looking at these matrices, recall from (A-19) that the entries of $\tilde{x}(k)$ range from $\hat{x}(k+1)$ to $\hat{x}(k+N)$, while $\tilde{u}(k)$, $\tilde{\delta}(k)$, and $\tilde{z}(k)$ contain expected values for the time steps k to $k+N-1$[1].

[1] If the matrices for the state constraints were attached at the bottom of the E matrices before the extension to prediction form, then after this extension the matrices for these inequalities for the final time step $(k+N)$ would still need to be attached to the E^* matrices separately. Also then for time step k these inequalities would also be checked, even though the states for this time step are already fixed. That is why the prediction form inequalities of the state constraints are all combined with the inequality after it has been extended to prediction form for all the other inequalities. Because $E_4 x(k)$ cannot result from the product of a matrix (put within E_4^*) and $\tilde{x}(k)$ (since $\tilde{x}(k)$ does not contain $x(k)$), it is calculated within E_5^*.

A-5 Derivation of the prediction form of the MPC performance index

To show how (3-39) is derived from (3-34b), first the sum ($\sum$) is eliminated from this latter equation. To be able to do that, the performance index is written as function of the state and input prediction vectors, resulting in

$$J(\tilde{u},k) = \left\| \begin{bmatrix} Q & & 0 \\ & \ddots & \\ 0 & & Q \end{bmatrix} \left(C\tilde{x}(k) - x^*_{\text{ref}}(k) \right) \right\|_1 + \tag{A-23}$$

$$\left\| R_{u_{\text{tb}}} \left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & & & & & & & & & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & & & & & & & & & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & & & & & & & & & & 0 \\ \vdots & & & & & & & & & & & & & & & & & & \\ 0 & & & & & & & & & & & & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \cdot \tilde{u}(k) + \begin{bmatrix} -u_{\text{tb}}(k-1) \\ \vdots \\ 0 \end{bmatrix} \right) \right\|_1$$

$$+ \left\| R_j \left(\begin{bmatrix} 0 & 1 & 2 & 4 & 0 & 0 & 0 & 0 & \\ 0 & -1 & -2 & -4 & 0 & 1 & 2 & 4 & \\ & & & & & & & & \ddots \end{bmatrix} \cdot \tilde{u}(k) + \begin{bmatrix} -\delta_1(k-1) - 2\delta_2(k-1) - 4\delta_3(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right) \right\|_1$$

where $C \in \mathbb{R}^{2N \times 2N}$ and $x^*_{\text{ref}}(k) \in \mathbb{R}^{2N}$ are different for each of the information gathering configurations that are being designed.
For configuration I (using communicated state information from just the direct predecessor), which only uses information from the direct predecessor, these two parameters are given by

$$C = \begin{bmatrix} 1 & h_0 & 0 & 0 & \dots & & 0 \\ 0 & 1 & 0 & 0 & & & \\ 0 & 0 & 1 & h_0 & & & \\ 0 & 0 & 0 & 1 & & & \\ \vdots & & & & \ddots & & \\ & & & & & 1 & h_0 \\ 0 & & & & & 0 & 1 \end{bmatrix}, \quad x^*_{\text{ref}}(k) = \begin{bmatrix} \hat{x}_{i-1}(k+1) \\ \vdots \\ \hat{x}_{i-1}(k+N) \end{bmatrix} + \begin{bmatrix} -l_{\text{veh}} - d_0 \\ 0 \\ -l_{\text{veh}} - d_0 \\ 0 \\ \vdots \\ -l_{\text{veh}} - d_0 \\ 0 \end{bmatrix} \tag{A-24}$$

where $\hat{x}_{i-1}(k+n)$ is the predicted state of the direct predecessor at step $k+n$, used as reference[2], and d_0 and h_0 are the distance headway and time headway respectively, which make up the desired intervehicle distance according to (3-2).
For configuration II (using state information from the first two predecessors), the matrix C and the vector $x^*_{\text{ref}}(k)$ are given by

$$C = \begin{bmatrix} 1 & q_{s,\text{p1}}h_0 + q_{s,\text{p2}}h_0 & & & & & \\ 0 & 1 & & & & 0 & \\ & & 1 & q_{s,\text{p1}}h_0 + q_{s,\text{p2}}h_0 & & & \\ & & 0 & 1 & & & \\ & & & & \ddots & & \\ & 0 & & & & 1 & q_{s,\text{p1}}h_0 + q_{s,\text{p2}}h_0 \\ & & & & & 0 & 1 \end{bmatrix}, \tag{A-25}$$

$$x^*_{\text{ref}}(k) = \begin{bmatrix} q_{s,\text{p1}}(\hat{s}_{i-1}(k+1) - l_{\text{veh}} - d_0) + q_{s,\text{p2}}(\hat{s}_{i-2}(k+1) - h_0\hat{v}_{i-1}(k+1) - 2l_{\text{veh}} - 2d_0) \\ q_{v,\text{p1}}\hat{v}_{i-1}(k+1) + q_{v,\text{p2}}\hat{v}_{i-2}(k+1) \\ \vdots \\ q_{s,\text{p1}}(\hat{s}_{i-1}(k+N) - l_{\text{veh}} - d_0) + q_{s,\text{p2}}(\hat{s}_{i-2}(k+N) - h_0\hat{v}_{i-1}(k+N) - 2l_{\text{veh}} - 2d_0) \\ q_{v,\text{p1}}\hat{v}_{i-1}(k+N) + q_{v,\text{p2}}\hat{v}_{i-2}(k+N) \end{bmatrix}$$

[2]The predicted states of the predecessors (vehicles $i-1$, $i-2$, and 1, according to which are needed) are calculated as explained in Section 3-3-3. In short, the communicated states are used to predict the future states, assuming the velocity stays constant. For MPC variant 2, these communicated states also comprise a set of future states, up to $k+N-1$, that were the optimal predicted states found in the previous time step.

where $\hat{s}_{i-1}(k+n)$ and $\hat{s}_{i-2}(k+n)$ are the predicted position of respectively the direct and the second predecessor at time step $k+n$, $\hat{v}_{i-1}(k+n)$ and $\hat{v}_{i-2}(k+n)$ are the predicted velocity of respectively the direct and the second predecessor at time step $k+n$, and $q_{s,\mathrm{p1}}, q_{s,\mathrm{p2}}, q_{v,\mathrm{p1}}, q_{s,\mathrm{p2}}$ are weights that are defined in Section 3-3-1.
For configuration III (using state information from the predecessor, and just velocity state information from the leader), the matrices C and $x^*_{\mathrm{ref}}(k)$ are given by

$$C = \begin{bmatrix} 1 & h_0 & 0 & 0 & \dots & & 0 \\ 0 & 1 & 0 & 0 & & & \\ 0 & 0 & 1 & h_0 & & & \\ 0 & 0 & 0 & 1 & & & \\ \vdots & & & & \ddots & & \\ & & & & & 1 & h_0 \\ 0 & & & & & 0 & 1 \end{bmatrix}, \quad x^*_{\mathrm{ref}}(k) = \begin{bmatrix} \hat{s}_{i-1}(k+1) - l_{\mathrm{veh}} - d_0 \\ q_{v,\mathrm{p}}\hat{v}_{i-1}(k+1) + q_{v,\mathrm{l}}\hat{v}_1(k+1) \\ \vdots \\ \hat{s}_{i-1}(k+N) - l_{\mathrm{veh}} - d_0 \\ q_{v,\mathrm{p}}\hat{v}_{i-1}(k+N) + q_{v,\mathrm{l}}\hat{v}_1(k+N) \end{bmatrix} \tag{A-26}$$

where $\hat{v}_1(k+n)$ is the predicted velocity of the leader at time step $k+n$, and $q_{v,\mathrm{p}}$ and $q_{v,\mathrm{l}}$ are weights that are defined in Section 3-3-1.

Then, the following new parameters are defined to simplify the notation:

$$Q^* = \begin{bmatrix} Q & & 0 \\ & \ddots & \\ 0 & & Q \end{bmatrix} \cdot C, \qquad Q^*_{xr}(k) = -\begin{bmatrix} Q & & 0 \\ & \ddots & \\ 0 & & Q \end{bmatrix} \cdot x^*_{\mathrm{ref}}(k),$$

$$R^*_{u\mathrm{p}}(k) = R_{u_{\mathrm{tb}}} \cdot \begin{bmatrix} -u_{\mathrm{tb}}(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad R^*_{j\mathrm{p}}(k) = R_j \cdot \begin{bmatrix} -\delta_1(k-1) - 2\delta_2(k-1) - 4\delta_3(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{A-27a}$$

and

$$R^*_u = R_{u_{\mathrm{tb}}} \cdot \begin{bmatrix} 1 & & & & & & & & & & & & & & & & & & & & & & & \\ -1 & 0 & 0 & 0 & 1 & & & & & & & & & & & & & & & & & & & \\ & & & & -1 & 0 & 0 & 0 & 1 & & & & & & & & & & & & & & & \\ & & & & & & & & & \ddots & & & & & & & & & & & & & & \\ & & & & & & & & & & -1 & 0 & 0 & 0 & 1 & & & & & & & & & \\ & & & & & & & & & & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix},$$

$$R^*_j = R_j \cdot \begin{bmatrix} 0 & 1 & 2 & 4 & 0 & & & & & & & & & & & & & & & & & & & \\ 0 & -1 & -2 & -4 & 0 & 1 & 2 & 4 & & & & & & & & & & & & & & & & \\ & & & & & -1 & -2 & -4 & 0 & 1 & 2 & 4 & & & & & & & & & & & & \\ & & & & & & & & & & & & \ddots & & & & & & & & & & & \\ & & & & & & & & & & & & -1 & -2 & -4 & 0 & 1 & 2 & 4 & & & & & \end{bmatrix} \tag{A-27b}$$

After that, the parameters of (A-27) are substituted into (A-23) to obtain (3-39).

A-6 Combining the model and the performance index into an MILP problem representation

The inequality of (3-42) is a combination of the inequality (3-38), representing the model inequalities, the system limits and the performance constraints in prediction form, and the

inequalities of (3-41), containing new variables ρ_i that are generated as part of a "trick" to make the optimisation problem a linear problem (see (3-40) and (3-41)). When substituting M_{perf} from (3-39) into (3-41), the inequalities of (3-41) become

$$\rho \geq \begin{bmatrix} Q^*\tilde{x}(k) + Q^*_{xr}(k) \\ R^*_u\tilde{u}(k) + R^*_{\text{up}}(k) \\ R^*_j\tilde{u}(k) + R^*_{\text{jp}}(k) \end{bmatrix} \text{ and } \rho \geq -\begin{bmatrix} Q^*\tilde{x}(k) + Q^*_{xr}(k) \\ R^*_u\tilde{u}(k) + R^*_{\text{up}}(k) \\ R^*_j\tilde{u}(k) + R^*_{\text{jp}}(k) \end{bmatrix} \tag{A-28}$$

The inequality (3-42) is built such that the inequalities from (A-28) are now given by its lower $2 \cdot 4N$ lines, and the rest of the lines are the same as (3-38). Hence, the definitions of the new matrices and vector ($E_{5,\rho}$ being a vector) in (3-42) are given by

$$\begin{aligned} E_{1,\rho} &= \begin{bmatrix} E^*_1 \\ 0 \\ R^*_u \\ R^*_j \\ 0 \\ -R^*_u \\ -R^*_j \end{bmatrix}, \quad E_{2,\rho} = \begin{bmatrix} E^*_2 \\ 0 \end{bmatrix}, \quad E_{3,\rho} = \begin{bmatrix} E^*_3 \\ 0 \end{bmatrix}, \\ E_{4,\rho} &= \begin{bmatrix} E^*_4 \\ -Q^* \\ 0 \\ Q^* \\ 0 \end{bmatrix}, \quad E_{5,\rho}(k) = \begin{bmatrix} E^*_5(k) \\ -Q^*_{xr}(k) \\ -R^*_{\text{up}}(k) \\ -R^*_{\text{jp}}(k) \\ Q^*_{xr}(k) \\ R^*_{\text{up}}(k) \\ R^*_{\text{jp}}(k) \end{bmatrix}, \quad E_{6,\rho} = \begin{bmatrix} 0 \\ -I \\ -I \end{bmatrix} \end{aligned} \tag{A-29}$$

where each 0 represents a matrix of only zeros with the occurrences of 0 in the definitions of $E_{1,\rho}$, $E_{4,\rho}$, and $E_{5,\rho}$ all having a hight of $2N$, and the occurrences of 0 in the definitions of $E_{2,\rho}$, $E_{3,\rho}$ both having a hight of $8N$, and the hight of the 0 in $E_{6,\rho}$ equal to the hight of E^*_1, and I here represents an identity matrix of hight $4N$.

The new matrices and vector of (3-43) are given by

$$\begin{aligned} E_{1,\text{subs}} &= E_{1,\rho} - E_{4,\rho}B^*_1 \\ E_{2,\text{subs}} &= E_{2,\rho} - E_{4,\rho}B^*_2 \\ E_{3,\text{subs}} &= E_{3,\rho} - E_{4,\rho}B^*_3 \\ E_{5,\text{subs}}(k) &= E_{4,\rho}A^*x(k) + E_{5,\rho}(k) \end{aligned} \tag{A-30}$$

Rearranging the above gives the MILP problem

$$\min_{y(k)} \; M \cdot y(k), \quad \text{s.t. } E' \cdot y(k) \leq F(k) \tag{A-31}$$

with

$$\begin{aligned} M = [0, \ldots, 0, 0, \ldots, 0, 0, \ldots, 0, 1, \ldots, 1]^{\text{T}}, \quad y = \begin{bmatrix} \tilde{u}(k) \\ \tilde{\delta}(k) \\ \tilde{z}(k) \\ \rho(k) \end{bmatrix}, \\ E' = [E_{1,\text{subs}}, E_{2,\text{subs}}, E_{3,\text{subs}}, E_{6,\rho}], \qquad F(k) = E_{5,\text{subs}}(k) \end{aligned} \tag{A-32}$$

A-7 Simulated annealing

Simulated annealing mimics the annealing of metal. In this process the heated metal is cooled slowly while the metal structure's state is changing. If the cooling rate is sufficiently slow, the state can "escape" from non-optimal structures and reach an optimal crystal-like structure in the end. An optimisation algorithm derived from this process can escape local minima in order to find a more optimal minimum.

In short, at each iteration simulated annealing selects a trial point from the neighbourhood of the current point in the search space. If the objective function value at the trial point is lower than at the current point, the trial point is accepted as the next current point. If the trial point function value is higher, this point is accepted with a probability depending on the current "temperature", decreasing with decreasing temperature. If it is rejected, the current point stays the same for the next iteration. The simulated annealing algorithm as applied in this project looks as follows.

First some functions and parameters are defined. For the decrease in 'temperature' (T) with temperature step (t) a simple cooling schedule is chosen [26], given by

$$T(t) = \frac{T_0}{c^t} \tag{A-33}$$

where t ranges from 0 to t_e, c is the cooling factor calculated based on $T(t_\mathrm{e}) = T_\mathrm{e}$ with T_e given, and T_0 the initial temperature. A neighbourhood function ($F_\mathrm{neigh}(x,t)$) is chosen, which defines how a new trial point is generated from the neighbourhood of the current point (vector $x \in \mathbb{R}^m$ with m the number of search parameters) at each iteration, depending on the current temperature step. The chosen neighbourhood function looks as follows:

$x_\mathrm{try} = F_\mathrm{neigh}(x, \sigma(t))$:
For each dimension i ($\leq m$), find new value $x_{\mathrm{try},i}$ in neighbourhood of x_i:
Draw random value from normal distribution,
with standard deviation $\sigma_i(t)$ and mean x_i;
Then $x_\mathrm{try} = [x_{\mathrm{try},1}, \ldots, x_{\mathrm{try},i}, \ldots, x_{\mathrm{try},m}]$.

where $\sigma(t)$ is a vector of standard deviations for all dimensions at temperature step t. To have a higher probability of finding an optimum, with lower computational cost, following [26, 43, 44, 48] the size of the neighbourhood is chosen to decrease with progressing temperature steps, according to

$$\sigma(t) = \frac{\sigma_0}{c_\sigma^t} \tag{A-34}$$

with $\sigma_0 = \sigma(t = 0)$, and c_σ is determined such that $\sigma_\mathrm{e} = \sigma(t_\mathrm{e})$ has a predetermined value. The number of iterations per temperature step n is set to $n \approx 2.75 t_\mathrm{e}$, where the exact values of n and $t_\mathrm{e} \in \mathbb{N}$ are computed based on the maximum number of allowed function evaluations f_nmax, with $n \cdot (t_\mathrm{e} + 1) \leq f_\mathrm{nmax}$. The final temperature step may have less than n iterations.

Now, the optimisation algorithm looks as follows.

Initialise:
Set $t = 0$, $k = 1$;
Randomly choose a point $x(1)$ from the search space;
Calculate objective function value at $x(1)$: $f(x(1))$;
Remember best value: set initial $f_\mathrm{best} = f(x(1))$, $x_\mathrm{best} := x(1)$;

Iterate:

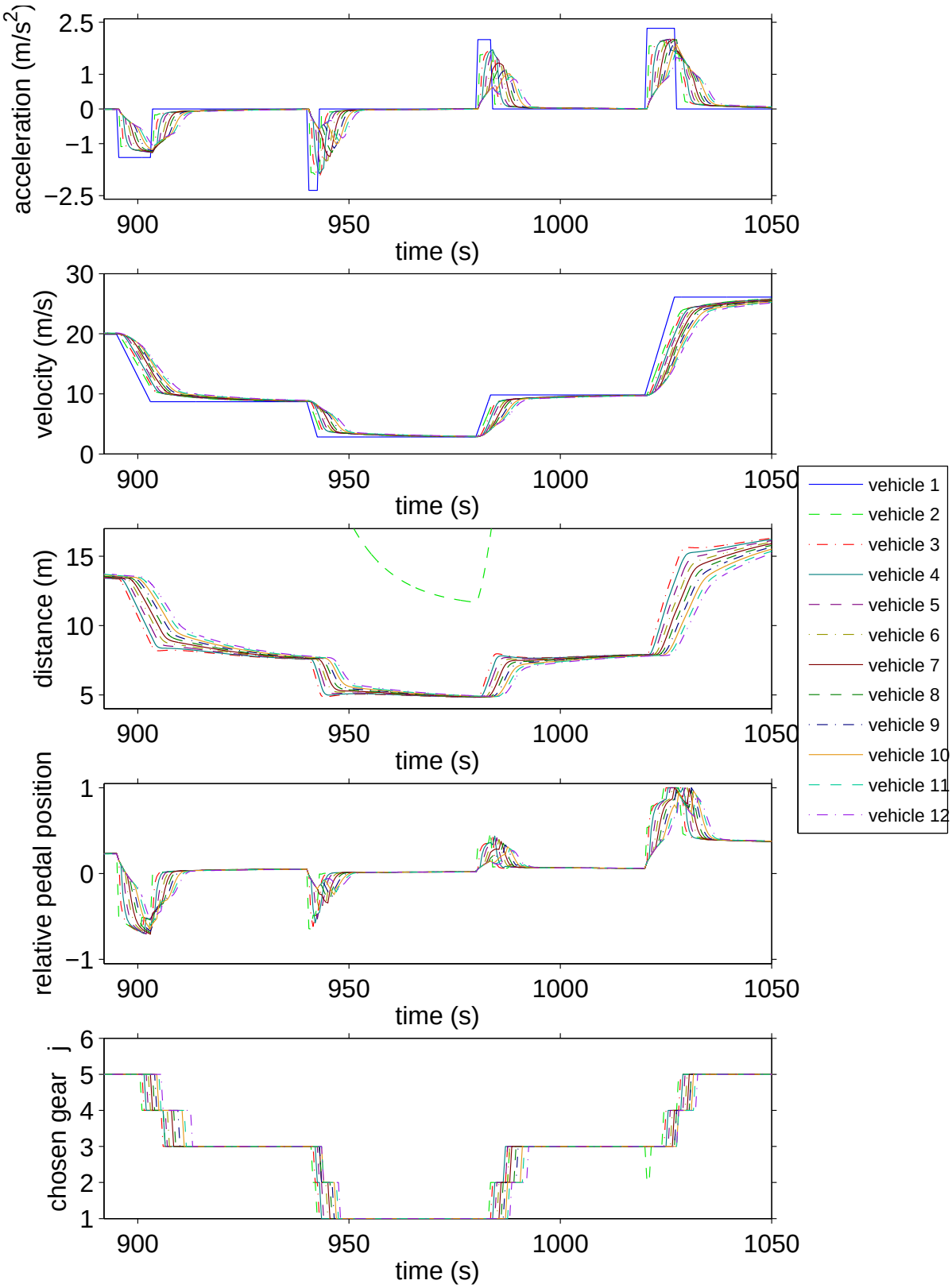
Generate new trial point $x_{\text{try}}(k) = F_{\text{neigh}}(x(k), \sigma(t))$;
Compute $\delta(k) = f(x_{\text{try}}(k)) - f(x(k))$;
Select random number $r(k) \in (0, 1)$;
Accept or reject trial point:
if $r(k) < \exp(-\delta(k)/T(t))$, then
$x(k+1) := x_{\text{try}}(k)$
Check if best value:
if $f(x_{\text{try}}(k)) > f_{\text{best}}$, then $f_{\text{best}} := f(x_{\text{try}}(k))$, $x_{\text{best}} := x_{\text{try}}(k)$
else $x(k+1) := x(k)$;
$k = k + 1$;
Check if cooling step: if n divides k, then $t := t + 1$;
Check stopping criterion: if $k = f_{\text{nmax}}$, then STOP.

Here a "point" means a vector consisting of the tuning parameters each of which having some value (often also called a "configuration"). A second stopping criterion is added which checks whether up to 2.0σ distance from $x(k)$ (since arrived at $x(k)$), at least a predefined number of points are tried. If this is the case and $f(x(k))$ is the lowest value found so far, the search has converged, which means meeting the second stopping criterion. If however it is known to be only a local optimum, because a more optimal point has been found earlier, the search proceeds from that lower point. This has been properly built into the algorithm above. When one of the stopping criteria is met, the algorithm is terminated.

Appendix B

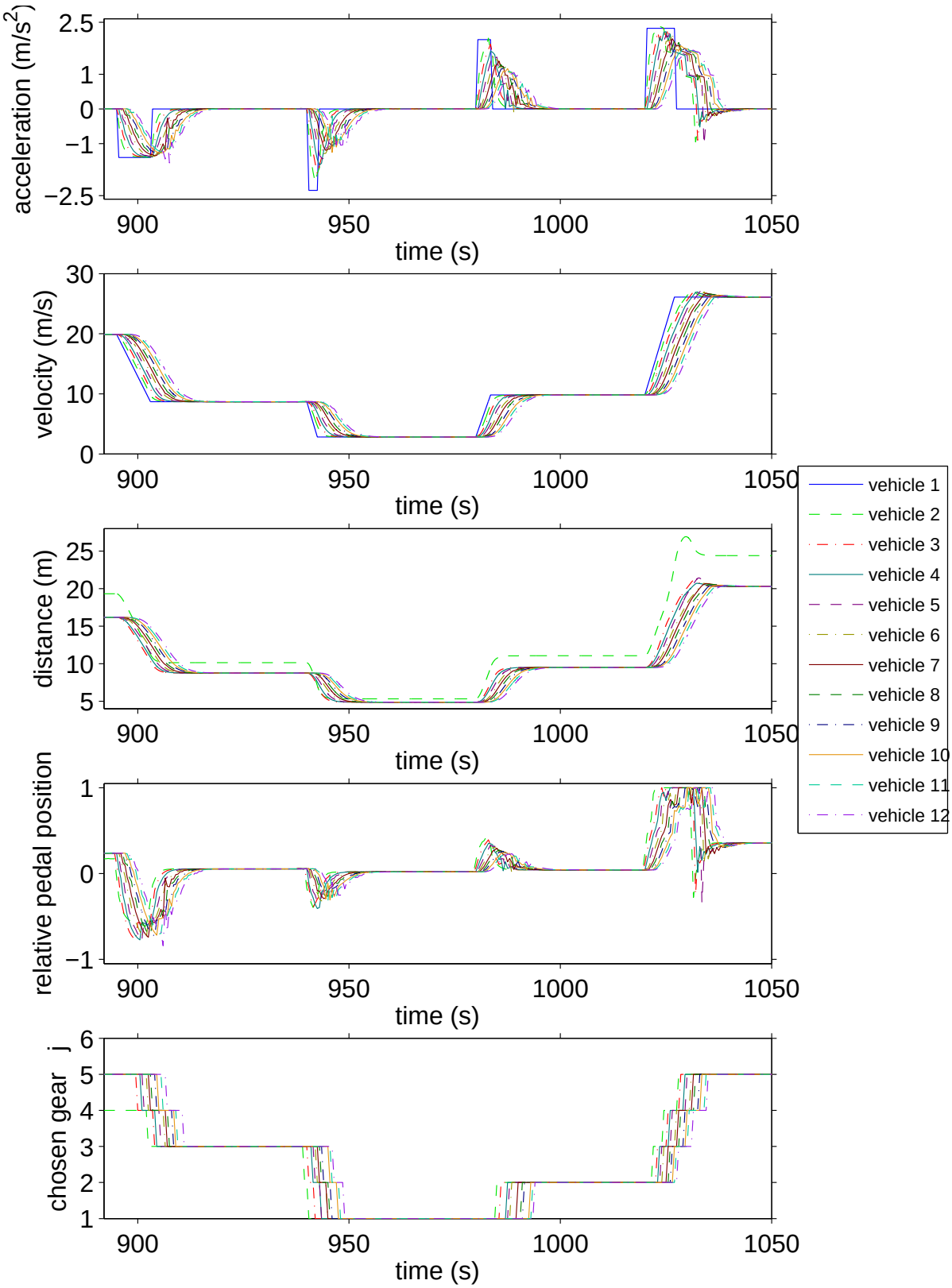
Result plots

Figure B-1 shows parts of the trajectories of states, intervehicle distances, and inputs for the validation scenario (scenario 2) for some of the controllers. This is to show some differences between the controllers— differences that have been described in Chapter 4—at a more global scope, compared with the plots shown in the main text that were more zoomed in.
PID-III, shown in Figure B-1a keeps the shortest intervehicle distances of the three PID controllers. The second vehicle keeps such a big intervehicle distance that most of it is not visible in the plot, because it is zoomed in on the other vehicles. The trajectories of PID-II look roughly similar, but with larger distances, and for PID-I the distances are much more significantly larger, as can be seen in Table 4-5. The trajectories of MPC_1-I are roughly similar to those of MPC_1-III, shown in Figure B-1c, with only slightly less intensive oscillations. It can clearly be seen that MPC_1-II is an improvement compared with the MPC_1 configurations I and III, because it does not show such heavy oscillations. In Figure B-1d the trajectories of MPC_2-II are shown, which are roughly similar to those of the MPC_2 configurations I and III, the main difference being the significantly larger intervehicle distances with configuration I.



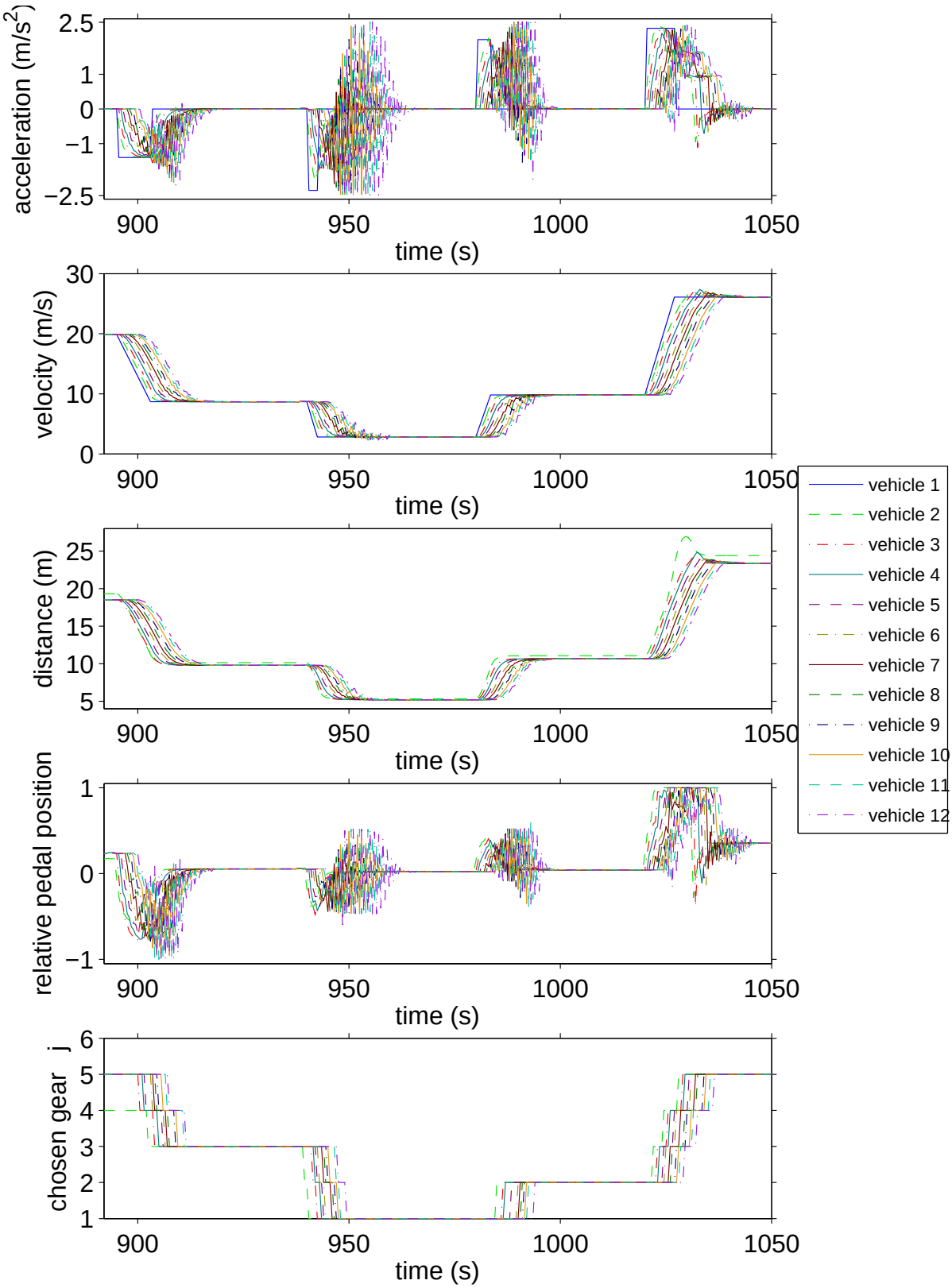
(a) PID-III

Figure B-1: A part of scenario 2.



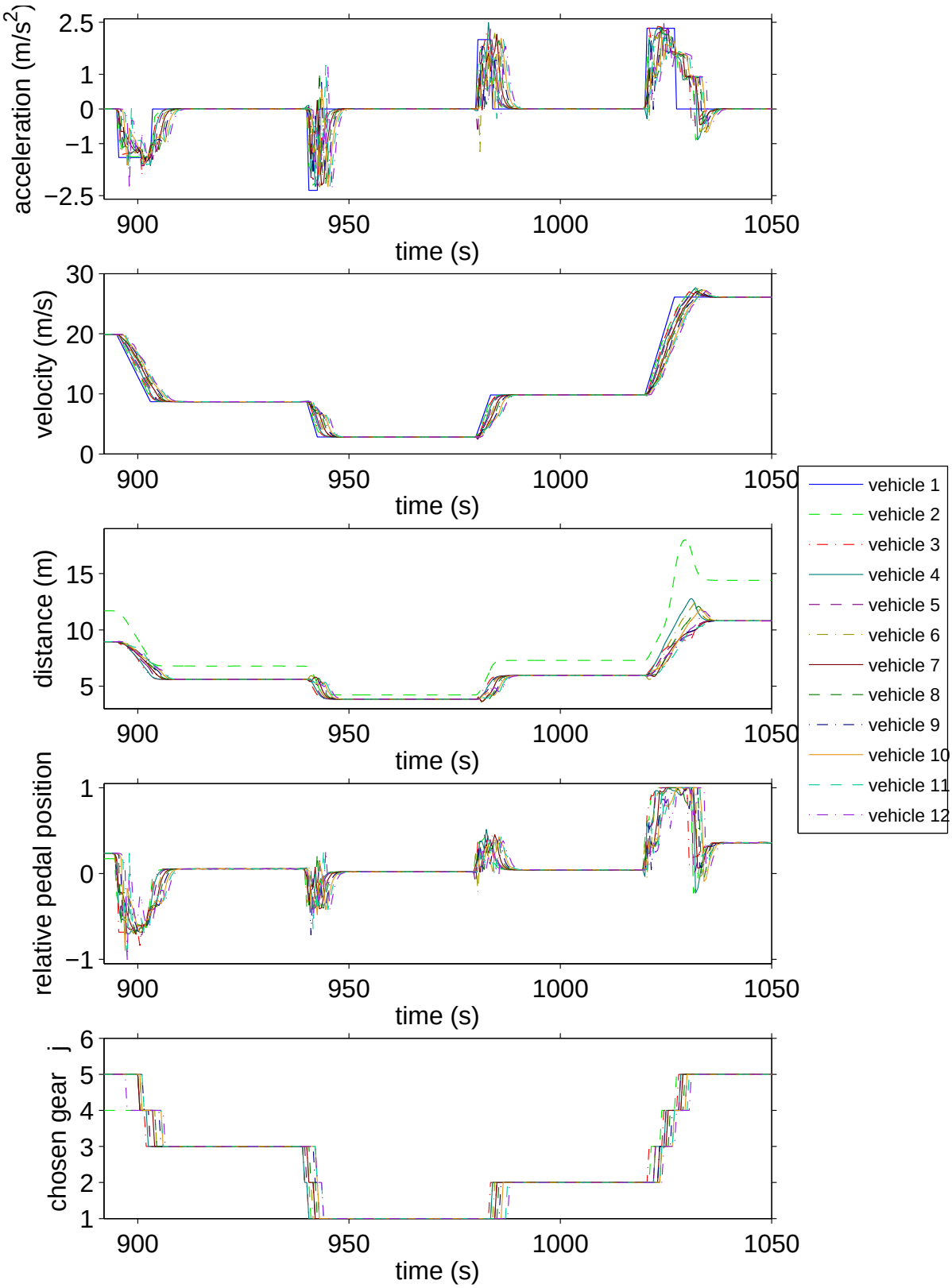
(b) $\mathrm{MPC_1}$-II

Figure B-1: A part of scenario 2.



(c) MPC_1-III

Figure B-1: A part of scenario 2.



(d) $\mathrm{MPC_2}$-II

Figure B-1: A part of scenario 2.

Bibliography

[1] B. van Arem, C.J.G. van Driel, and R. Visser. The impact of cooperative adaptive cruise control on traffic-flow characteristics. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):429–436, 2006.

[2] V.L. Bageshwar, W.L. Garrard, and R. Rajamani. Model predictive control of transitional maneuvers for adaptive cruise control vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 53(5):1573–1585, 2004.

[3] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[4] M. Beukenberg and D. Hummel. Aerodynamics, performance and control of airplanes in formation flight. In *Proceedings of the 17th Congress of the International Council of the Aeronautical Sciences*, pages 1777–1794, Stockholm, Sweden, September 1990.

[5] R. Bishop. *Intelligent Vehicle Technology and Trends*. Artech House, 2005.

[6] A. Bose and P.A. Ioannou. Analysis of traffic flow with mixed manual and semiautomated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 4(4):173–188, 2003.

[7] D. de Bruin, J. Kroon, R. van Klaveren, and M. Nelisse. Design and test of a cooperative adaptive cruise control system. In *Proceedings of the 2004 IEEE Intelligent Vehicles Symposium*, pages 392–396, Parma, Italy, June 2004.

[8] E.F. Camacho and C. Bordons. *Model Predictive Control in the Process Industry*. Springer-Verlag, Berlin, Germany, 1995.

[9] P. Caravani, E. De Santis, F. Graziosi, and E. Panizzi. Communication control and driving assistance to a platoon of vehicles in heavy traffic and scarce visibility. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):448–460, 2006.

[10] C.C. Chien and P. Ioannou. Automatic vehicle following. In *Proceedings of the 1992 American Control Conference*, pages 1748–1752, Chicago, USA, June 1992.

[11] D. Corona and B. De Schutter. Adaptive cruise controller for a SMART car: A comparison benchmark for MPC-PWA control methods. *IEEE Transactions on Control Systems Technology*, 16(2):365–372, 2008.

[12] B. De Schutter, J. Ploeg, L.D. Baskar, G. Naus, and H. Nijmeijer. Hierarchical, intelligent and automatic controls. In A. Eskandarian, editor, *Handbook of Intelligent Vehicles*, chapter 5, pages 81–116. Springer, 2012.

[13] T. van Dijck and G.A.J. van der Heijden. VisionSense: an advanced lateral collision warning system. In *Proceedings of the 2005 IEEE Intelligent Vehicles Symposium*, pages 296–301, Las Vegas, USA, June 2005.

[14] R.W. Eglese. Simulated annealing: A tool for operational research. *Journal of Operational Research*, 46(33):271–281, 1990.

[15] J. Frankel, L. Alvarez, R. Horowitz, and P.Y. Li. Safety oriented maneuvers for IVHS. *Vehicle System Dynamics*, 26(4):271–299, 1996.

[16] W.L. Garrard, R.J. Caudill, A.L. Kornhauser, D. MacKinnon, and S.J. Brown. State-of-the-art of longitudinal control of automated guideway transit vehicles. *High Speed Ground Transportation Journal*, 12(2):35–67, 1978.

[17] O. Gehring and H. Fritz. Practical results of a longitudinal control concept for truck platooning with vehicle to vehicle communication. In *Proceedings of the 1997 IEEE Conference on Intelligent Transportation Systems*, pages 117–122, Boston, USA, November 1997.

[18] A. González-Villaseñor, A.C. Renfrew, and P.J. Brunn. A controller design methodology for close headway spacing strategies for automated vehicles. *International Journal of Control*, 80(2):179–189, 2007.

[19] Q. Gu, X.-Y. Lu, and T. Tang. Energy saving for automatic train control in moving block signaling system. In *Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems*, pages 1305–1310, Washington, USA, October 2011.

[20] D. Gulick, L. Alvarez, and R. Horowitz. Implementation of the regulation layer using SHIFT. Technical Report UCB-ITS-PRR-98-26, Institute of Transportation Studies, University of California, Berkeley, USA, 1998. California PATH Program.

[21] R. Hallouzi, V. Verdult, H. Hellendoorn, P.L.J. Morsink, and J. Ploeg. Communication based longitudinal vehicle control using an extended Kalman filter. In *Proceedings of the 4th IFAC Symposium on Advances in Automotive Control*, Salerno, Italy, April 2004.

[22] R. Hallouzi, V. Verdult, H. Hellendoorn, P.L.J. Morsink, and J. Ploeg. Experimental evaluation of a co-operative driving set-up based on inter-vehicle communication. In *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, July 2004.

[23] K. Holmström, A.O. Göran, and M.M. Edvall. *User's Guide for TOMLAB 7*. Västerås, Sweden, 2009.

[24] P. Ioannou and Z. Xu. Throttle and brake control systems for automatic vehicle following. *Journal of Intelligent Transportation Systems*, 1(4):345–377, 1994.

[25] S. Kumarawadu and T.-T. Lee. On the speed control for automated vehicle operation. In *Proceedings of the 5th World Congress on Intelligent Control and Automation*, pages 2443–2448, Hangzhou, China, June 2004.

[26] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications*. Springer, 1987.

[27] Y. Lv, Z. Zhao, J. Gu, and J. Dong. The research on the gear shift schedules for AMT in HEV. In *Proceedings of the 2009 IEEE Intelligent Vehicles Symposium*, pages 722–729, Xi'an, China, July 2009.

[28] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, England, 2002.

[29] T. Minowa, H. Kimura, N. Ozaki, and M. Ibamoto. Improvement of fuel consumption for a vehicle with an automatic transmission using driven power control with a powertrain model. *JSAE Review*, 17(4):375–380, 1996.

[30] J.E. Naranjo, C. González, R. García, and T. de Pedro. ACC+stop&go maneuvers with throttle and brake fuzzy control. *IEEE Transactions on Intelligent Transportation Systems*, 7(2):213–225, 2006.

[31] G.J.L. Naus, R. Vugts, J. Ploeg, R. Van de Molengraft, and M. Steinbuch. Towards on-the-road implementation of cooperative adaptive cruise control. In *Proceedings of the 16th World Congress & Exhibition on Intelligent Transport Systems and Services*, Stockholm, Sweden, September 2009.

[32] J. Pachl. *Railway Operation and Control*. VTD Rail Publishing, 2002.

[33] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, USA, 2009.

[34] G. Ribichini and E. Frazzoli. Efficient coordination of multiple-aircraft systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 1035–1040, Maui, USA, December 2003.

[35] S. Sakaguchi, I. Sakai, and T. Haga. Application of fuzzy logic to shift scheduling method for automatic transmission. In *Proceedings of the 2nd IEEE International Conference on Fuzzy Systems*, pages 52–58, San Francisco, USA, March 1993.

[36] S.E. Shladover. Longitudinal control of automated guideway transit vehicles within platoons. *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, 100(4):302–310, 1978.

[37] S.E. Shladover. Review of the state of development of advanced vehicle control systems (AVCS). *Vehicle System Dynamics*, 24(6-7):551–595, 1995.

[38] S.E. Shladover. Deployment path analysis for cooperative ITS systems. Technical Report UCB-ITS-PWP-2009-04, Institute of Transportation Studies, University of California, Berkeley, USA, 2009. California PATH Program.

[39] D. Swaroop, J.K Hedrick, C.C. Chien, and P. Ioannou. A comparison of spacing and headway control laws for automatically controlled vehicles. *Vehicle System Dynamics*, 23(8):597–625, 1994.

[40] D. Swaroop, J.K. Hedrick, and S.B. Choi. Direct adaptive longitudinal control of vehicle platoons. *IEEE Transactions on Vehicular Technology*, 50(1):150–161, 2001.

[41] A. Touran, M. Brackstone, and M. McDonald. A collision model for safety evaluation of autonomous intelligent cruise control. *Accident Analysis and Prevention*, 31(5):567–578, 1999.

[42] A. Vahidi and A. Eskandarian. Research advances in intelligent collision avoidance and adaptive cruise control. *IEEE Transactions on Intelligent Transportation Systems*, 4(3):143–153, 2003.

[43] M.T. Vakil-Baghmisheh and A. Navarbaf. A modified very fast simulated annealing algorithm. In *Proceedings of the 2008 IEEE International Symposium on Telecommunications*, pages 61–66, Tehran, Iran, August 2008.

[44] S.R. White. Concepts of scale in simulated annealing. In *Proceedings of the 1984 IEEE International Conference on Computer Design*, pages 261–270, Port Chester, USA, November 1984.

[45] H.P. Williams. *Model Building in Mathematical Programming*. Wiley, third edition, 1993.

[46] D. Yanakiev and I. Kanellakopoulos. A simplified framework for string stability analysis in AHS. In *Proceedings of the 13th IFAC World Congress*, pages 177–182, San Francisco, USA, July 1996.

[47] D. Yanakiev and I. Kanellakopoulos. Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications. *IEEE Transactions on Vehicular Technology*, 47(4):1365–1377, 1998.

[48] X. Yao. Simulated annealing with extended neighbourhood. *International Journal of Computer Mathematics*, 40(3):169–189, 1991.

[49] K. Yi and Y.D. Kwon. An investigation of intelligent cruise control laws for passenger vehicles. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 215(2):159–169, 2001.

[50] Y. Zhang, E.B. Kosmatopoulos, P.A. Ioannou, and C.C. Chien. Autonomous intelligent cruise control using front and back information for tight vehicle following maneuvers. *IEEE Transactions on Vehicular Technology*, 48(1):319–328, 1999.