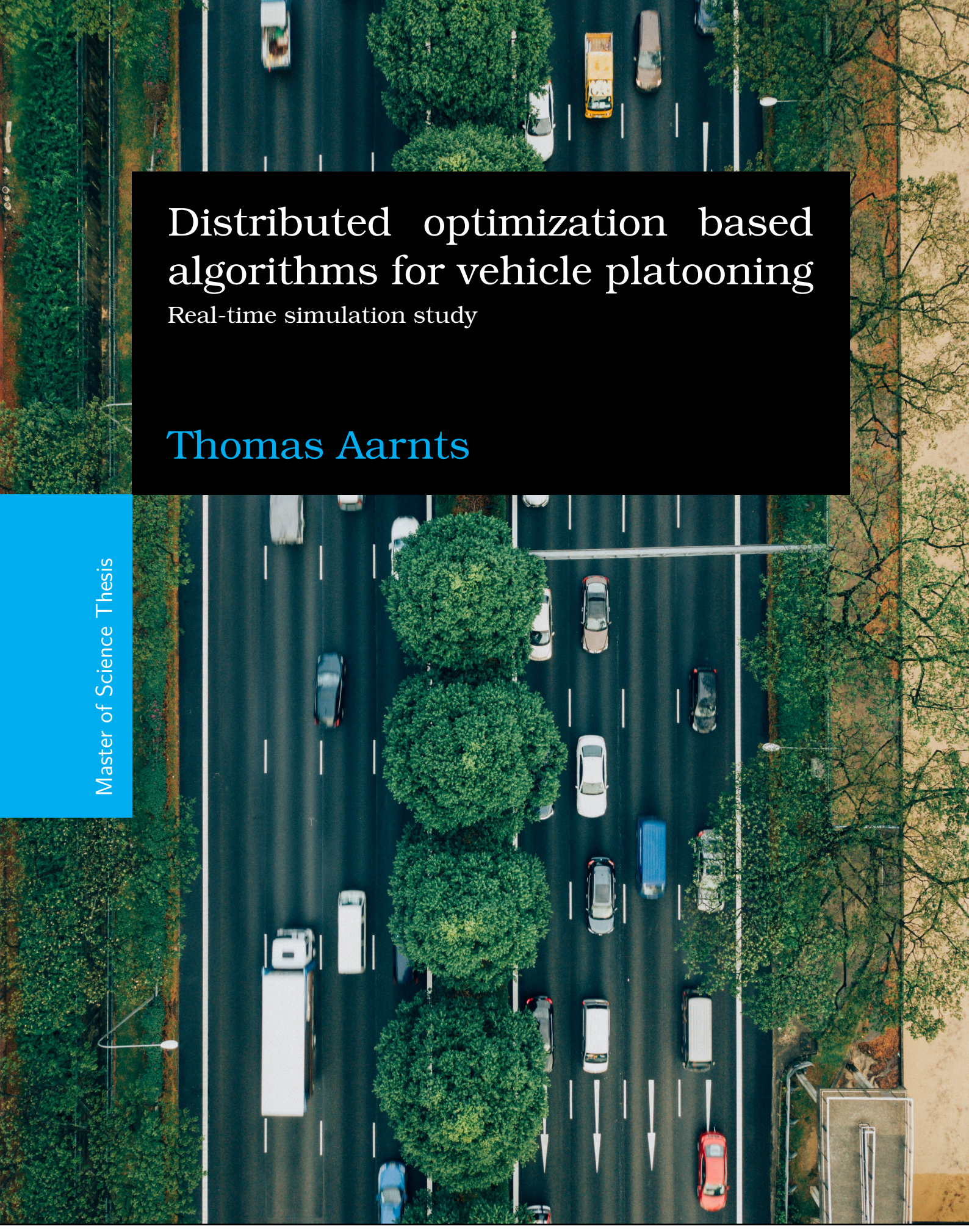

Distributed optimization based algorithms for vehicle platooning

Real-time simulation study

Thomas Aarnts

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Distributed optimization based algorithms for vehicle platooning

Real-time simulation study

Master of Science Thesis

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

Thomas Aarnts

February 11, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Distributed optimization based algorithms for vehicle platooning

by

Thomas Aarnts

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: February 11, 2019

Supervisor(s):

Dr. Manuel Mazo Jr.

Dr. Anton V. Proskurnikov

Reader(s):

Dr.ir. Meng Wang

Dr.ir. Tamas Keviczky

Abstract

The rise of vehicle usage causes roads to reach capacity limits. When capacity is reached, traffic jams and accidents are more likely to occur. By raising the efficiency of traffic, the capacity of roads can benefit without the need to expand infrastructure. Automating longitudinal controls of vehicles and creating vehicle platoons show potential to raise traffic efficiency.

A platoon consists of multiple vehicles travelling closely behind each other. Automating the longitudinal controls shows promise to decrease distances between these vehicles, while maintaining safety and raising traffic throughput. This smaller distance is below that of conventional human control, thus improving traffic density.

Most modern cars already have the option of being equipped with longitudinal control. The most advanced being Adaptive Cruise Control (ACC) which can either hold a predefined velocity or adapt that velocity to maintain distance to a preceding vehicle. However, due to the limitation of using only on-board sensors, these systems do not guarantee safety or increase traffic throughput.

Research is being done on expanding ACC with Vehicle to Vehicle (V2V) communication creating Cooperative Adaptive Cruise Control (CACC). The addition of communication provides more precise data of neighbouring vehicles than conventional on-board sensing can provide and allows for vehicles to communicate intent. Optimization control for CACC systems has been researched, but it rarely incorporates passengers comfort or the future intent of the preceding vehicles.

The goal of this thesis is to implement a decentralized optimization algorithm on a real-time simulated platoon of three vehicles, which takes into consideration safety, propagation of errors through the platoon, and comfort. These three aspects are subsequently implemented and evaluated on a real-time simulation platform in a distributed fashion.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I want to thank my supervisors Dr. Anton Proskurnikov and Dr. Manuel Mazo Jr. for their guidance and help throughout my master thesis. In addition, I would like to thank the members of Manuel's research group for their contribution to the process.

Finally, I would like to extend my gratitude to my direct colleagues in the lab for the day to day atmosphere and to my family and friends who supported me during the entirety of my studies.

Delft, University of Technology — Thomas Aarnts
February 11, 2019

The research was supported by NWO Domain TTW, The Netherlands, under the project TTW#13712 "From Individual Automated Vehicles to Cooperative Traffic Management - predicting the benefits of automated driving through on-road human behavior assessment and traffic flow models" (IAVTRM).

"There are no big problems, there are just a lot of little problems."
— *Henry Ford*

Chapter 1

Introduction

This chapter introduces the subject of decentralized optimization based vehicle platooning. After this introduction, the benefits of vehicle platooning are explained followed by current industry implementations and research already performed. Finally, the goal of the thesis is introduced together with an overview of the report structure.

1-1 Vehicle platooning

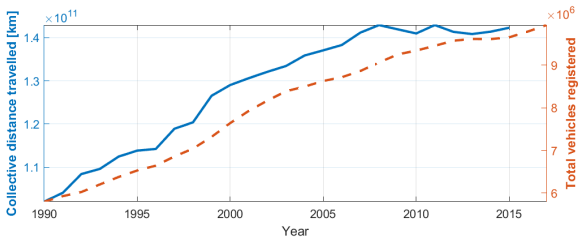

Figure 1-1: Distance travelled (blue solid) and vehicles registered (red dashed) in the NL, 1990 to 2016 [1]

Last decades traffic volume and distance travelled were on the rise, Figure 1-1 displays this trend in the Netherlands. This rise causes roads to reach their traffic capacity, resulting in more congestion. More congestion leads to a rise in travel-time and pollution, causing damage to both the economy and population health of nations. A solution is to raise road capacity by decreasing the distance between vehicles. Unfortunately, human control is insufficient to maintain safety at smaller inter-vehicle distances due to a long reaction time. Automating the longitudinal control of vehicles offers a solution to this issue, which then allows for a decrease in inter-vehicle distances.

A platoon consists of multiple vehicles travelling behind each other in single file, with each vehicle holding a predetermined distance to a predecessor. When all vehicles have automated controls, human error is removed and the platoon as a whole becomes automated.

1-2 Automated longitudinal control

The automation of longitudinal controls brings multiple advantages. Firstly, safety could benefit from the decrease of system reaction time, the removal of human error, and by the implementation of controllers which mitigate error propagation. Secondly, fuel efficiency can be improved by limiting the throttle action of the controllers, reducing air drag by maintaining small inter-vehicle distances, and the prevention of traffic jams. Lastly, an increase in comfort can be realised by limiting the acceleration and jerk the passengers are subjected to and by removing the need to drive manually.

Initial research into longitudinal control dates back to halfway through the 20th century [2–4]. This is followed by the erection of multiple programs to research autonomous vehicles, of which longitudinal control is a large contributor. Noteworthy research programs are PATH[1](1986) [5] at the University of California, DRIVE[2](1989) and PROMETHEUS[3](1986) [6,7] within the European Union, the university contest for cooperative driving GCDC[4] [8], and the Dutch program DAVI[5] [9] which is a collaboration for autonomous road vehicles between Dutch universities and the government.

Currently, the automotive industry already has implemented multiple forms of longitudinal automation in production vehicles. A regular implemented method is the basic Cruise Control (CC) which controls throttle to maintain velocity. However, this method only holds velocity and does not adapt to other vehicles or objects. More recently, an extension to CC has been introduced to production vehicles, usually referred to as Adaptive Cruise Control (ACC). ACC has the added ability to maintain distance to a predecessor, which is detected by use of on-board sensors such as radar or imaging.

An issue with current implemented ACC technologies is the deterioration of stability when more vehicles are added. As a result, there is no attenuation of disturbances to the rear of the platoon [10–12] which poses a threat to safety and traffic efficiency. Furthermore, inter-vehicle distances maintained by ACC are similar to human control, thus not resulting in a significant increase in road capacity [13].

A possible solution is to implement inter-vehicle communication to exchange data, thus adding data from other vehicles to the controller. The availability of on-board data from other vehicles is an improvement over remotely sensed data which has more inaccuracies, especially for higher-order data. In addition to exchanging vehicle states, communication also allows for the platoon to exchange additional information concerning the abilities of vehicles or specific commands. Both these aspects would create a scenario with more certainty of the surrounding vehicles, allowing the controller to operate within tighter margins thus decreasing the inter-vehicle distance.

Addition of communication to ACC is referred to as Cooperative Adaptive Cruise Control (CACC). Current literature shows promise that CACC is a viable solution to raise traffic throughput. Some experimental implementations have been verified in research [14–16]. Figure 1-2 shows a simple visualization of the multiple control methodologies.

[1] Program on Advanced Technology for the Highway
[2] Dedicated Road Infrastructure for Vehicle Safety in Europe
[3] PROgramme for a European Traffic with Highest Efficiency and Unprecedented Safety
[4] Grand Cooperative Driving Challenge
[5] Dutch Automotive Vehicle Initiative

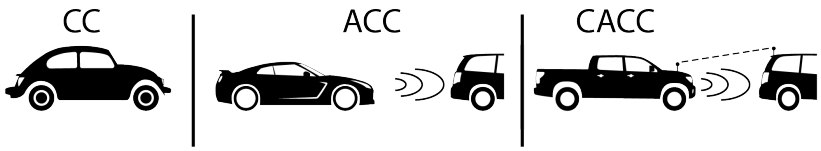

Figure 1-2: Visualization of common methods of longitudinal automation [17]

1-3 Decentralized optimization control

The control problem of an automated platoon knows a multitude of constraints. These concern limits on individual vehicles and the inter-vehicle values within a platoon. Model Predictive Control (MPC), also known as Receding Horizon Control (RHC), applies repeated optimization of planned trajectories to determine control inputs at each time step. With the application of optimization comes the benefit of explicitly subjecting the solution and resulting states to constraints.

However, MPC does require a substantial amount of computation and applying MPC in a centralized manner on large platoons results in multiple issues. The control problem and required communication both grow with each additional vehicle, with this growth comes the issue of a growing physical distance to the centralized controller. This increase in data volume and physical distance creates the risk of communication delay or dropouts, which in turn poses a threat to the stability and safety of a platoon [18]. Decentralizing the controller to individual vehicles would lessen loads and delays concerning communication. An added benefit of decentralization is the logistical advantage from splitting the computational effort required, adding redundancy and modularity to the system.

1-4 Thesis goal

Research into automated decentralized platooning shows promising results, both theoretical and experimental. Suggested by Dunbar et al [19, 20] is to apply MPC in a decentralized manner with the addition of communicating the planned future trajectories upstream within the platoon. These algorithms promise platoon stability and are successfully evaluated on a second order simulation. However, there is no notion in [19, 20] of an extension beyond this simulation.

The goal of this thesis is to research and evaluate an optimization based controller for vehicle platooning which implements future intent of the preceding vehicle based on [19, 20]. The optimization problem is to be constrained by the stability criteria suggested in [19, 20], constraints concerning local stability and constraints to better passenger comfort.

For evaluation, the controller is implemented on multiple real-time vehicle simulators, creating a more realistic simulation. Due to the real-time distributed implementation, the controller is extended to handle communication and timing while having limited computational resources available. The aim is to evaluate the performance of the controller based on a set of standard manoeuvres while maintaining the stability which [19, 20] has proven in theory.

1-5 Report outline

The report starts with setting the mathematical framework used in this report in chapter 2. This is followed by Chapter 3 in which the applied algorithms are presented. After which, in Chapter 4, the used simulation setup is introduced together with the method of implementation. Chapter 5 presents the experiments executed and discusses the results of the implemented algorithms. Once the results are presented, Chapter 6 presents further discussion of the results. Finally, at the end of the report a conclusion is presented in combination with suggestions for future work in Chapter 7.

At the end of the report, the reader can find additional information in the appendixes, the bibliography displaying used sources, and a nomenclature presenting used acronyms and symbols.

Chapter 2

Preliminaries

This chapter defines the preliminaries of the thesis. The mathematical notation is introduced, followed by the definition of a vehicle, a platoon, and the notation of state trajectories. In the end, a formulation of the thesis problem is presented. All variables and constants are quantified by SI units, and data size is denoted in Bytes (B).

An overview of all symbols and acronyms is presented in the nomenclature, located at the end of the report.

2-1 Mathematical preliminaries

This section gives the basis of the mathematical framework used throughout the thesis.

Basic notation In this thesis, a basic notation to denote numbers, vectors, and matrices is applied. When defining vectors or matrices square brackets are used around the values. Both matrices and vectors are depicted in bold, with matrices capitalized. '$\top$' is applied in superscript for transposing matrices and vectors.

Variables and parameters are denoted either as real numbers $\mathbb{R}$ or integers $\mathbb{Z}$. Both domains are unbounded unless a superscript denoting a bound is applied. Hence,$\mathbb{Z}^{\geq 0}$ denotes a set of non-negative integers.

Vector norms Norms are used to quantify vector qualities. A norm is denoted using double vertical lines $||\mathbf{x}||_p$ with $\mathbf{x}$ being an example column vector of arbitrary length and $p \in \mathbb{Z}^{\geq 1}$ denotes which norm is used. For this research two norms were used, the 2 norm and the infinity norm. Equation (2-1) presents both norms.

$$||\mathbf{x}||_2 = (\mathbf{x}^\top \mathbf{x})^{1/2}, \quad ||\mathbf{x}||_\infty = \max_{\forall i} \{|x_i|\} \tag{2-1}$$

2-2 System definitions

This section defines a vehicle platoon and the individual vehicles within. First, the individual vehicle is defined together with the vehicle model. Then, the platoon and the internal interactions are defined. Figure 2-1 depicts a platoon with most variables visualized.

2-2-1 Vehicle

Before the platoon is defined the parameters and variables of a single vehicle are set. Each vehicle is defined as a single mass with three states and an applied force as input, all of which are in the set $\mathbb{R}$. Vehicle position is denoted by $x_i(t)$, which is measured from the front of the vehicle to a common origin behind the platoon in line with the travelling direction. Velocity of a vehicle is denoted by $v_i(t)$ and acceleration by $a_i(t)$. The input to the system is defined as a longitudinal force $u_i(t)$ which is directly applied to the centre of mass.

To describe the dynamic behaviour of a vehicle multiple constants are defined. Mass is denoted as $m_i \in \mathbb{R}^{>0}$, overall vehicle length as $L_i \in \mathbb{R}^{>0}$, and the aerodynamic drag constant as $c_{d,i} \in \mathbb{R}^{\geq 0}$. The aerodynamic drag constant is used to model the aerodynamic drag force by the formula shown in Equation (2-2), which is a common method for modelling drag [21].

$$F_{drag,i}(t) = c_{d,i} \cdot v_i(t)^2 \tag{2-2}$$

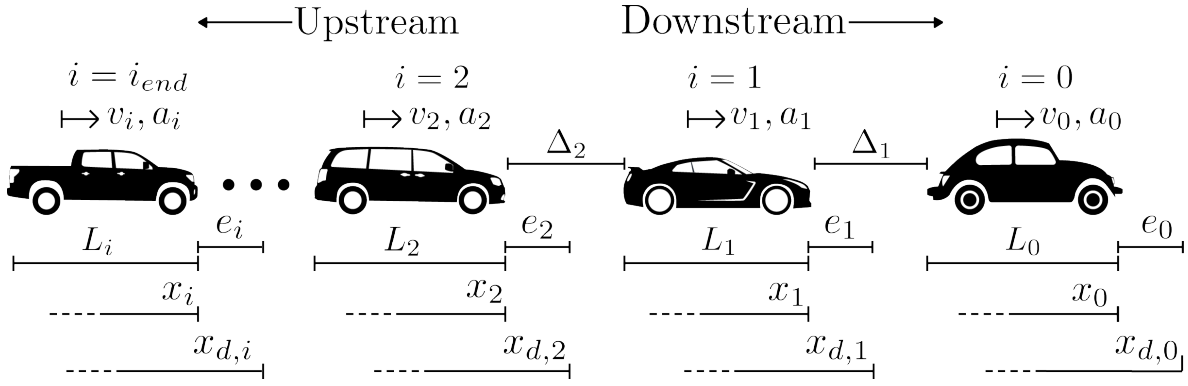

Figure 2-1: Schematic depicting platoon and vehicle states [17]

Vehicle model For Model Predictive Control (MPC) the controller needs to predict the resulting states of a candidate input sequence. A mathematical vehicle model is defined which allows for modelling these results. The vehicle model is based on a double integrator model extend with a non-linear aerodynamic drag component.

Equation (2-3) shows the continuous models definition, in which the velocity of the vehicle is a direct integration of the acceleration. Acceleration, in turn, is a function of input and aerodynamic drag. Note that the model becomes linear when the air drag component is removed, i.e., $c_{d,i} = 0$.

$$\begin{bmatrix} \dot{x}_i(t) \\ \dot{v}_i(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_i(t) \\ v_i(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m_i} \end{bmatrix} u_i(t) - \begin{bmatrix} 0 \\ \frac{c_{d,i}}{m_i} \end{bmatrix} v_i(t)^2 \tag{2-3}$$

2-2-2 Platoon

A platoon is defined as multiple vehicles driving behind each other in a single string. For this research only longitudinal issues are taken into account. Thus, the platoon is assumed to drive in a straight line without any cornering. Furthermore, it is assumed there are no obstacles or elevation differences ahead of the platoon.

Additional inter-vehicle variables and parameters arise when a platoon is defined. To start vehicles are numbered with an index number $i \in [0, 1, \cdots i_{end}]$ with $i = 0$ being the lead vehicle and $i = i_{end}$ the last. All states, inputs, and constants regarding a single vehicle are referred to using the corresponding index number as a subscript, e.g. $x_i(t)$.

Next is inter-vehicle distance, which is measured from a car's front bumper to the rear bumper of the preceding car. This value, denoted by $\Delta_i(t) \in \mathbb{R}$, is shown in Equation (2-4). Note that this definition denotes the distance between vehicle i and $i-1$ thus in front of the vehicle i.

$$\Delta_i(t) = x_{i-1}(t) - x_i(t) - L_{i-1} \tag{2-4}$$

d_{sep} denotes the desired separation distance and is defined by the desired distance between the front bumpers of vehicles. For simplicity d_{sep} is constant and equal for all vehicles. As a result, the desired inter-vehicle distance $\Delta_{d,i}$ becomes dependent on the length of the preceding vehicle, this is seen in Equation (2-5).

$$\Delta_{d,i} = d_{sep} - L_{i-1} \tag{2-5}$$

Two directions in the platoon are used when discussing the flow of data or events. Upstream depicts the direction opposite to the travel direction towards the initial position of the platoon. In other words upstream is the direction from the lead vehicle $i = 0$ to the last vehicle $i = i_{end}$. In similar fashion downstream is the opposite direction of upstream.

Finally desired states and resulting errors are defined. The desired velocity $v_d \in \mathbb{R}^{\geq 0}$ and the corresponding initialization time $t_0 \in \mathbb{R}^{\geq 0}$ are determined by the lead vehicle. Both are then used in combination with vehicle index, initial leader position, and desired separation to determine the desired position $x_{d,i}(t)$ for each vehicle. Equation (2-6) shows the formula for the desired position.

$$x_{d,i}(t) = x_0(t_0) + (t - t_0)v_d - i \cdot d_{sep} \tag{2-6}$$

The difference between vehicle states and desired states define the two error states of the vehicle. Both errors are grouped in an error vector $\mathbf{z}_i(t)$, with the first entry being the position error $e_i(t)$ and the second the velocity error $\dot{e}_i(t)$. The definition of the vector and its elements are shown in Equation (2-7).

$$\mathbf{z}_i(t) = \begin{bmatrix} e_i(t) \\ \dot{e}_i(t) \end{bmatrix} = \begin{bmatrix} x_{d,i}(t) - x_i(t) \\ v_d(t) - v_i(t) \end{bmatrix} \tag{2-7}$$

2-2-3 Timing and trajectories

Additional definitions are required for the timing and trajectories of the control algorithm. The actual method of implementation is further discussed in Chapter 4.

The samples are numbered by $k \in \mathbb{Z}^{\geq 0}$ and have a constant duration of $\delta \in \mathbb{R}^{>0}$s. Initial time of each sample is defined by t_k with t_0 describing the initialization time of the current desired velocity. Equation (2-8) shows the construction of t_k. When the lead vehicle decides to change the desired velocity, the lead vehicle is to set a time when this change goes into effect. t_{plan} denotes this time, and is to be in the future i.e. $t_{plan} > t$.

Platoon control is realised with MPC which plans trajectories for a fixed horizon, the amount of samples within this horizon is denoted by $N \in \mathbb{Z}^{\geq 1}$. Corresponding is the lookahead time $T \in \mathbb{R}^{>0}$ which is shown in Equation (2-9).

$$t_k = t_0 + k\delta \tag{2-8}$$

$$T = N\delta \tag{2-9}$$

Two types of trajectories are used, predicted and planned. Both trajectories have a length of N samples. Equation (2-10) shows both notations applied to the error vector. Note that the notation is also applied to other values than the error vector.

Predicted trajectories are noted with a superscript p and are used as variable candidate trajectories of the optimizer. These trajectories are to be optimized within bounds.

Planned trajectories, denoted with a hat ($\hat{}$), are the solutions of the previous optimization. These are static within the optimization, are used to define the bounds of the optimization problem, and are communicated from vehicle to vehicle.

$$\begin{aligned} \mathbf{z}_i^p(:|t_k) \quad &: \text{Predicted error trajectory} \\ \hat{\mathbf{z}}_i(:|t_k) \quad &: \text{Planned error trajectory} \end{aligned} \tag{2-10}$$

Trajectories are updated at each sample. For the explanation of the procedures it is necessary to distinguish what time a trajectory begins and which entries are used. To accomplish this a dedicated time notation is introduced, which is noted by two variables separated by a vertical line between brackets, e.g. $(\tau|t_k)$. In this notation, the value on the right denotes the initial time of the sequence and the value on the left the individual instances of that sequence. Hence, τ is a single value or vector with length $\leq N$. All entries of which are within the complete sample range $\{t_k, \cdots, t_{k+N-1}\}$. When the entire sequence is used a double dot is applied instead of a vector as seen in Equation (2-10).

Lastly the objective function is denoted by $J_{i,t_k}(\cdot)$, the contents of which are explained in Section 3-1-1.

2-3 Problem formulation

Current literature depicts no notion of an experimentally implemented decentralized platoon controller with the utilization of predecessor trajectories. However, the paper from Dunbar and Caveney [19] depicts a Cooperative Adaptive Cruise Control (CACC) controller which in theory promises stability and safety if specific constraints are satisfied. According to literature, this controller has only been simulated off-line on a simple heterogeneous platoon model.

With no data on the performance of this controller on a more realistic platoon, a logical next step is an implementation on a real-time simulator which models a more realistic platoon. This more realistic simulation requires the controller to be executed in real-time, an actual communication structure, and is to overcome timing issues.

The goal for this research is to design and implement a controller on a realistic platoon simulation. This controller is to satisfy three notions: safety, platoon stability, and comfort. These notions are written below and further formalized as actual constraints in the MPC framework in Section 3-2.

General Safety and individual stability Prevention of collisions is an essential task of a vehicle controller. Three definitions are used to promise safety.

Firstly, vehicle stability: all vehicles are to stabilize the position error $e_i(t)$ and velocity error $\dot{e}_i(t)$ both to zero with time progressing. Equation (2-11) shows this definition.

Secondly, collision prevention: the inter-vehicle distance between all vehicles is greater than zero at all time. In other words, vehicles are not to make contact.

Finally, no reversing: velocity is to be larger than or equal to zero at all times for all vehicles. Equation (2-12) shows the last two definitions.

$$\forall i: \qquad \lim_{t\to\infty} e_i(t) = 0, \qquad \lim_{t\to\infty} \dot{e}_i(t) = 0 \tag{2-11}$$

$$\forall i, t: \qquad \Delta_i(t) > 0, \qquad v_i(t) \geq 0 \tag{2-12}$$

Platoon stability Platoon stability entails the notion that disturbances do not amplify while propagating through the platoon. This notion prevents traffic jams or situations in which vehicles are not able to maintain safety constraints due to physical limits. The mathematical definition taken from [19] sets an upper limit on absolute vehicle position errors for all future time, the upper limit knows two types.

The first, less stringent type, is defined as leader-follower string stability, in which all vehicles have a lower maximum absolute position error than the lead vehicle as seen in Equation (2-13).

Second is a more stringent type is that for each vehicle the upper limit is set by the maximum error of the preceding vehicle, this is called predecessor-follower stability and the definition is noted in Equation (2-14). Note that predecessor-follower string stability also satisfies leader-follower string stability.

$$\forall i > 0 : \max_{t\geq t_0} |e_i(t)| < \max_{t\geq t_0} |e_0(t)| \tag{2-13}$$

$$\forall i > 0 : \max_{t\geq t_0} |e_i(t)| < \max_{t\geq t_0} |e_{i-1}(t)| \tag{2-14}$$

Comfort An important aspect of a controllers performance when considering CACC is passenger comfort. Quantification of comfort is based on ISO standard 22179 [22] which limits acceleration and jerk of a vehicle depending on the velocity. Equations (2-15) and (2-16) show the piecewise definitions for the acceleration limits and Figure 2-2 shows a visualization of both the acceleration and jerk limits.

$$a_{max}(v) = \begin{cases} 4, & \text{if } v < 5 \\ 4 - 2\frac{v-5}{15}, & \text{if } 5 \leq v \leq 20 \\ 2, & \text{if } v > 20 \end{cases} \tag{2-15}$$

$$a_{min}(v) = \begin{cases} -5, & \text{if } v < 5 \\ -5 + 1.5\frac{v-5}{15}, & \text{if } 5 \leq v \leq 20 \\ -3.5, & \text{if } v > 20 \end{cases} \tag{2-16}$$

The limits for jerk are similar but defined symmetric with respect to zero. The definitions are shown in Equations (2-17) and (2-18).

$$j_{max}(v) = \begin{cases} 5, & \text{if } v < 5 \\ 5 - 2.5\frac{v-5}{15}, & \text{if } 5 \leq v \leq 20 \\ 2.5, & \text{if } v > 20 \end{cases} \tag{2-17}$$

$$j_{min}(v) = -j_{max} \tag{2-18}$$

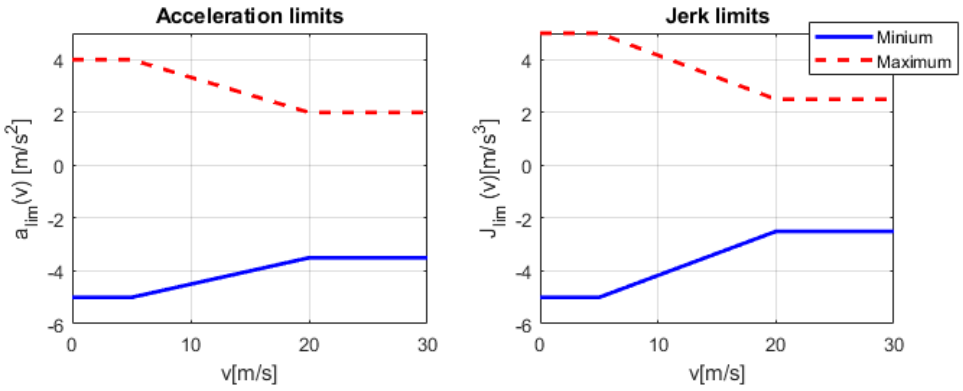

Figure 2-2: Acceleration and jerk constraints, ISO 22179 [22]

Chapter 3

CACC Algorithm

This chapter explains the CACC algorithm and the three different variants deployed. First, the algorithm and the objective function are explained, after which three sets of constraints are presented defining the three different controllers.

3-1 Dunbar's algorithm

This section explains the algorithm of Dunbar and the objective function. The algorithm applies Model Predictive Control (MPC) which repeatedly solves an optimization problem. This problem consists of an objective function to be minimized within a specific set of constraints.

The objective function penalizes multiple aspects, most noteworthy is the one containing the error trajectory of the preceding vehicle $\mathbf{z}_{i-1}(:|t_k)$. Implementation of this trajectory creates the possibility to optimize for future inter-vehicle values and subject these to constraints.

3-1-1 Objective function

At the heart of the optimization problem lies the objective function, which returns a single non-negative output that is to be minimized by the optimization algorithm. The objective function is a function of the predicted input trajectory $\mathbf{u}_i^p(:|t_k)$, the predicted error trajectory $\mathbf{z}_i^p(:|t_{k+1})$, the planned error trajectory $\hat{\mathbf{z}}_i(:|t_{k+1})$, and the received planned error trajectory of the preceding vehicle $\hat{\mathbf{z}}_{i-1}(:|t_k)$.

$$J_{i,t_k}(\mathbf{u}_i^p(:|t_k), \mathbf{z}_i^p(:|t_{k+1}), \hat{\mathbf{z}}_i(:|t_{k+1}), \hat{\mathbf{z}}_{i-1}(:|t_{k+1})) = \sum_{n=t_k+1}^{t_{k+N}} \Big(\underbrace{||\mathbf{Q}_i \mathbf{z}_i^p(n|t_{k+1})||_2}_{\text{Error}} + \underbrace{||\mathbf{F}_i(\mathbf{z}_i^p(n|t_{k+1}) - \hat{\mathbf{z}}_i(n|t_{k+1}))||_2}_{\text{Move supression}} + \underbrace{||\mathbf{G}_i(\mathbf{z}_i^p(n|t_{k+1}) - \hat{\mathbf{z}}_{i-1}(n|t_{k+1}))||_2}_{\text{Relative error}} + \underbrace{||R_i u_i^p(n-1|t_k)||_2}_{\text{Input effort}} \Big) \tag{3-1}$$

Equation (3-1) shows the content of the objective function. The function is a sum of four distinct aspects over the entire prediction horizon. Each aspect has a weight applied by one of the dedicated weighting matrices, $\mathbf{Q}_i, \mathbf{F}_i, \mathbf{G}_i \in \mathbb{R}^{(2\times 2)}$ and $R_i \in \mathbb{R}$. The time index of the objective function t_k denotes the time of the first input of the planned input sequence. Error sequences in the objective functions are one sample ahead in time due to the error functions being a result of the input sequence. All sequences span the entire prediction horizon of N samples.

The first aspect concerns 'Error' which contains the predicted error vector $\mathbf{z}_i^p(:|t_{k+1})$ as described in Equation (2-7). This aspect concerns the deviation of vehicle states with respect to the desired position and velocity, and is weighted by matrix $\mathbf{Q}_i$.

The second aspect is 'Move suppression', weighted by matrix $\mathbf{F}_i$, which is a penalization of deviation between the predicted error trajectory $\mathbf{z}_i^p(:|t_{k+1})$ and the previously planned error trajectory $\hat{\mathbf{z}}_i(:|t_{k+1})$.

The third aspect is the 'Relative error' and is defined by the difference in error vectors between the predicted error of the host vehicle $\mathbf{z}_i^p(:|t_{k+1})$ and the planned error of the preceding vehicle $\hat{\mathbf{z}}_{i-1}(:|t_{k+1})$. The result is weighted by $\mathbf{G}_i$. Expanding the error vectors, seen in Equation (3-2), shows that the first term directly penalizes deviation of the inter-vehicle distance from the desired value. The second term penalizes the relative velocity between vehicles.

The final aspect is the 'Input effort', weighted by R_i, which penalizes the use of input. In the case of vehicle platooning this is a penalty on acceleration commands.

$$\mathbf{z}_i(t) - \mathbf{z}_{i-1}(t) = \begin{bmatrix} \Delta_i(t) - \Delta_{d,i} \\ v_{i-1}(t) - v_i(t) \end{bmatrix} \tag{3-2}$$

The objective function, while a function of 4 trajectories, is only optimized by the predicted input sequence $\mathbf{u}_i^p(:|t_k)$. This is the result of the planned trajectories being constant during an optimization, and the predicted error trajectory $\mathbf{z}_i^p(:|t_{k+1})$ being a function of the input trajectory and other static entries. Equation (3-3) shows a simplification of this function where $H(\cdot)$ denotes the evaluation of the vehicle model. Planned trajectories are from the previous sample and are time-shifted to match, the procedure of which is further elaborated in Section 4-2-1.

$$\mathbf{z}_i^p(t_i|t_{k+1}) = H(x(t_k), v(t_k), x_d(t_k), v_d, \mathbf{u}(\{t_k \cdots t_{i-1}\}|t_k)) \tag{3-3}$$

3-1-2 Constraints

The algorithm depicted in [19] sets a multitude of constraints. When satisfied, the resulting trajectories satisfy the string stability criteria. There are two sets of constraints. The first is applied when a change in desired velocity occurs and the second set is applied for all remaining optimizations. The constraints are quite extensive and strict. Due to implementation, constraints grow tighter over time and with each added vehicle. Section 3-2-2 presents these constraints.

3-2 Deployed algorithms

This section elaborates on the algorithms deployed to the simulator. Each algorithm utilizes the same controller framework but differs in the applied constraints. All constraints are implemented using the structure of $G_i(\cdot) < 0$, in which $G_i(\cdot)$ is an arbitrary function depending on vehicle (error) states. Thus, all constraints are to be defined as being implementable as one-sided constraints.

3-2-1 Safety

The first algorithm applies constraints to the optimization problem which aims to satisfy the general safety notion as presented in Equations (2-11) and (2-12). This requires constraints which prevent vehicle contact, reversing, and individual instability.

The first constraint, depicted in Equation (3-4), concerns a terminal inequality constraint and forces the final error states to be close to zero This enforces the error to converge and is a common method to stabilize MPC control [23]. The second constraint limits a vehicle's velocity to remain positive at all times and is shown in Equation (3-5). Finally the last constraint, depicted in Equation (3-6), forces the inter-vehicle distance $\Delta_i(t)$ to remain positive for the coming horizon.

$$\forall i: \quad |\mathbf{z}_i^p(t_{k+N}|t_{k+1})| < [0.001, 0.001]^\top \tag{3-4}$$

$$\forall i: \quad v_i^p(:|t_{k+1}) \geq 0 \tag{3-5}$$

$$\forall i \geq 1: \quad e_i^p(:|t_{k+1}) - \hat{e}_{i-1}(:|t_{k+1}) > L_{i-1} - d_{sep} \tag{3-6}$$

The constraints are all enforced during normal operation, except for initialization when the last constraint of Equation (3-6) lacks the initial trajectories from other vehicles. Thus, instead of using the predecessor error trajectory that of the leader is used. It is preferred to use the leader's trajectory instead of disabling the constraint. This preference is a result of the assumption that a preceding vehicle would be more likely to have an error trajectory similar to that of the leader than having no error at all. Equation (3-7) shows the constraint during initialization.

$$\forall i \geq 1: \quad e_i^p(:|t_{k+1}) - \hat{e}_0(:|t_{k+1}) > L_{i-1} - d_{sep} \tag{3-7}$$

3-2-2 Platoon stability

The second algorithm enforces the platoon stability constraints set by [19] as depicted in Equation (2-14). There are two dedicated constraints. One constraint is applied during initialization only. The other is applied for the remaining samples until a next desired velocity change. Both constraints set limits on the predicted position error of the vehicle $e_i^p(:|t_k)$.

Initialization constraint For initialization, a dedicated constraint for all vehicles except the leader is created. This is depicted in Equation (3-8) and is implemented as an equivalent single-sided constraint seen in Equation (3-9). The constraint sets upper and lower bounds onto the predicted error trajectory of all following vehicles, based upon the initial planned error trajectory of the lead vehicle $\hat{e}_0(:|t_0)$. The bounds are parametrized by γ and ξ, both $\in (0,1)$.

$$\forall i \geq 1: \quad (1-\xi_i)\gamma_i|\hat{e}_0(:|t_1)| \leq |e_i^p(:|t_1)| \leq (1+\xi_i)\gamma_i|\hat{e}_0(:|t_1)| \tag{3-8}$$

$$\forall i \geq 1: \quad ||e_i^p(:|t_1)| - \gamma_i|\hat{e}_0(:|t_1)|| \leq \xi_i\gamma_i|\hat{e}_0(:|t_1)| \tag{3-9}$$

By combining the constraints of multiple vehicles, the bounds present a relation between vehicles depicted in Equation (3-10). This relation subsequently results in predecessor-follower stability for the initial trajectory when the combined parameter β_i satisfies $\in (0,1)$. For constructing the relation the upper bound on e_i and lower bound of e_{i-1} were used.

$$\begin{aligned} &\forall i \geq 0 : |e_i^p(:|t_1)| \leq \beta_i|e_{i-1}^p(:|t_1)| \\ &\beta_i = \begin{cases} (1+\xi_1)\gamma_1, & \text{if } i = 1 \\ ((1+\xi_i)/(1-\xi_{i-1}))(\gamma_i/\gamma_{i-1}), & \text{if } i > 1 \end{cases} \end{aligned} \tag{3-10}$$

Regular constraint Once the platoon is initialized, different constraints are enforced for the remainder of the operation. The constraints limit a vehicle's maximum absolute difference between the predicted and planned position error trajectory. This has to be smaller or equal than the maximum error of the coming sample of either the vehicle itself or its predecessor, as required for stability proofs in [**?**]. There are two exceptions: one for the lead vehicle, which has no predecessor, and one for the last vehicle, which is not required to be bound by its own error. Equation (3-11) shows the three constraints for the platoon. The constraint is parametrized by a predetermined sample dependent parameter $\epsilon_{i,k} \in (0,1)$.

$$\begin{aligned} i = 0: \; &||e_0^p(:|t_{k+1}) - \hat{e}_0(:|t_{k+1})||_\infty \leq \epsilon_{0,k} \, ||e_0^p([t_{k+1}, t_{k+2}]|t_{k+1})||_\infty \\ 1 \leq i < i_{end}: \; &||e_i^p(:|t_{k+1}) - \hat{e}_i(:|t_{k+1})||_\infty \leq \epsilon_{i,k} \, \min\{||\hat{e}_{i-1}([t_{k+1}, t_{k+2}]|t_{k+1})||_\infty, \\ &\qquad\qquad ||e_i^p([t_{k+1}, t_{k+2}]|t_{k+1})||_\infty\} \\ i = i_{end}: \; &||e_i^p(:|t_{k+1}) - \hat{e}_i(:|t_{k+1})||_\infty \leq \epsilon_{i,k} \, ||\hat{e}_{i-1}([t_{k+1}, t_{k+2}]|t_{k+1})||_\infty \end{aligned} \tag{3-11}$$

Predecessor-follower stability, as defined in Equation (2-14), is satisfied with the successful application of both initial and regular constraints and the parameter definitions β_i and $\epsilon_{i,k}$ satisfy the constraint in Equation (3-12).

$$\forall i \geq 1: \quad \beta_i + \beta_i \sum_{k=1}^{\infty} \epsilon_{i-1,k} + \sum_{k=1}^{\infty} \epsilon_{i,k}(1 + \epsilon_{i-1,k}) < 1 \tag{3-12}$$

Parameter choice As seen in the previous paragraphs, there are multiple parameters to be defined. The selection of all constraint parameters are to satisfy Equation (3-12) to guarantee predecessor-follower stability. [19] Sets a formulation on the choice of parameters to meet the stability constraints for the entire platoon. First, the sample instance dependent parameter $\epsilon_{i,k}$ is defined as $\epsilon_{i,k} = (\xi_i)^k$, this simplifies the number of parameters to tune to two. Then, by applying this new definition to Equation (3-12) a new constraint emerges as the sums can be redefined. The new equation is depicted in Equation (3-13).

$$\forall i \geq 1: \quad \frac{\beta_i}{1 - \xi_{i-1}} + \frac{1}{1 - \xi_i} + \frac{1}{1 - \xi_{i-1} \cdot \xi_i} < 3 \tag{3-13}$$

Constraint parameters are to be set in order from $i = 0$ to $i = i_{end}$ due to the dependency of the parameter constraint on preceding vehicles. First, a ξ_0 is chosen for the lead vehicle, after which β_1, ξ_1 are determined satisfying Equation 3-13. These parameters are then used to define γ_i using Equation 3-10. This process is repeated for all following vehicles.

3-2-3 Comfort

The last algorithm aims to satisfy the comfort constraints as set by Equations (2-15) to (2-18). The bounds are set for all samples in the prediction horizon. For acceleration, the bound can be directly applied to the result of the vehicle model.

However, Jerk by definition is the derivative of acceleration with respect to time. Due to the application of direct acceleration control, the evaluation of jerk results in a large impulse at the start of each control sample when changing the acceleration. Limiting the amplitude of these pulses would result in an undesired decrease in allowed acceleration change.

As a workaround, instead of limiting jerk the change in acceleration is limited. This change is noted by $da(t)$. While not true to the actual goal, this does allow setting a limit on the average jerk per sample.

The maximum absolute value of $da(t)$ was set to that of the original objective multiplied by the sample time δ. This would satisfy the jerk goal on average over a sample.

$$\forall i, \tau \in \{t_k, t_{k+N-1}\}: \qquad a_i(\tau|t_k) < \begin{cases} 4, & \text{if } v(\tau) < 5 \\ 4 - 2\frac{v(\tau)-5}{15}, & \text{if } 5 \leq v(\tau) \leq 20 \\ 2, & \text{if } v(\tau) > 20 \end{cases} \tag{3-14}$$

$$\forall i, \tau \in \{t_k, t_{k+N-1}\}: \qquad -a_i(\tau|t_k) < \begin{cases} 5, & \text{if } v(\tau) < 5 \\ 5 - 1.5\frac{v(\tau)-5}{15}, & \text{if } 5 \leq v(\tau) \leq 20 \\ 3.5, & \text{if } v(\tau) > 20 \end{cases} \tag{3-15}$$

$$\forall i, \tau \in \{t_k, t_{k+N-1}\}: \qquad |da(\tau|t_k)| < \begin{cases} 5\delta, & \text{if } v(\tau) < 5 \\ (5 - 2.5\frac{v(\tau)-5}{15})\delta, & \text{if } 5 \leq v(\tau) \leq 20 \\ 2.5\delta, & \text{if } v(\tau) > 20 \end{cases} \tag{3-16}$$

Chapter 4

Real-time simulator implementation

This chapter explains the real-time simulation platform used for this thesis and how the controller is implemented. To start, the vehicle simulation platform 'Dynacar' is introduced, followed by the hardware and software used to compute the simulation. Next, the controller framework and operational sequence of the controllers are set. Finally, communication between the separate systems and their controllers is explained.

4-1 Dynacar real-time vehicle simulator

For vehicle simulation, the real-time vehicle simulator Dynacar by Tecnalia is used [24]. Dynacar is designed for testing a wide range of vehicle systems within a real-time simulation. The simulation can be visualized, logged, tuned, and allows for control with a gaming steering wheel. The simulation includes tire dynamics, aerodynamics, inertia of moving parts, and suspension modelling. Figure 4-1 shows the visualization of a single Dynacar vehicle.

Evaluation of the model is subject to a predefined environment. The user can set this environment to a scenario of choice, such as a racing circuit or a city block. In this study, the environment is set to an open plain because only longitudinal issues are taken into account.

Dynacar is designed in 'LabVIEW RT' and is executed real-time at a sample frequency of 1000 Hertz dedicated hardware. By using multiple simulators simultaneously, a simulated platoon is constructed. All simulated vehicles are subjected to the same vehicle parameters, thus creating a homogeneous platoon. The constructed platoon can communicate between vehicles and all be visualized simultaneously in the same environment.

Besides communication, vehicles do not influence each other. Aerodynamics only depend on a vehicle's own states and there is no implementation of collision dynamics. Thus, there is no notion of collisions, drag reduction or turbulence due to other vehicles.

The version of Dynacar used does not have a drive-train implemented but instead applies a direct torque input to all wheels. This direct torque can either be a braking or an acceleration torque. For this research, only the brake and forward acceleration are implemented. Thus, the vehicle is not able to reverse via input commands.

4-1-1 Hardware

The hardware of the simulator setup consists of two main parts, a central host-pc and three real-time computation systems.

The host-pc initializes the real-time systems via the Ethernet interface and monitors resulting data. The received data can be used to visualize the vehicles in a game environment (Unity [25]). While visualizing the simulation, a USB connected gaming steering wheel can be used to control the simulator in real-time.

Dedicated real-time hardware from National instruments is used to perform the required computation. The hardware used consists of three identical PXIe-8880 controllers. All three real-time computation systems (further referred to as PXI) run deployed models from the host-pc. By deploying single vehicle models to dedicated pieces of hardware, the simulations are evaluated independently of each other. A single unpopulated PXI chassis, as used in this research, is shown in Figure 4-2.

4-1-2 Deployment and control

Deployment is realised with software from National Instruments, 'Veristand 2014'. Veristand deploys compiled models to the hardware, initializes the simulations, monitors outputs, and allows for sending commands.

The models consist of compiled C code, compiled with Microsoft Visual Studio 2008. On a single PXI system, multiple models can be deployed. Separate models are used for the vehicle simulation and the controller. Within a single PXI system, these models can interact with each other via internal ports. Thus, a single PXI system simulates a single vehicle and controller.

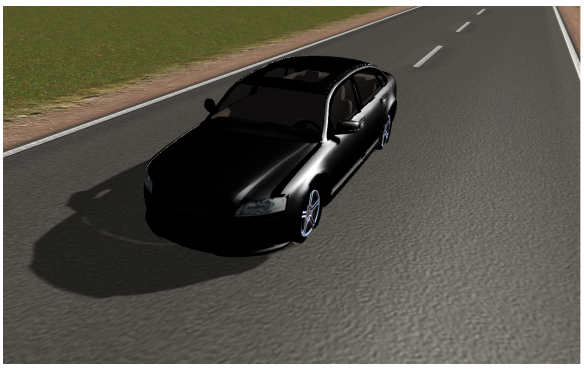

Figure 4-1: Dynacar render

Figure 4-2: NI PXIe-1071 chassis [26]

4-2 Controller framework

A controller framework is constructed to enable direct programming of control algorithms. The controller is deployed on the same PXI as the vehicle it controls.

Two modules are used, both running on independent processor cores next to the Dynacar vehicle model. The first module manages the timing, communication, triggers optimization, and sends the control input to the vehicle. This module is designated as 'Puppetmaster'. The second module, named 'Optimizer', solves the optimization problem as issued by the Puppetmaster. All vehicles are subjected to identical code but differ in controller parameters.

Each vehicle has specific parameters defining index number, optimization weights, constraint parameters, and settings for the Optimizer. The index number is used to determine the desired position trajectory as previously written in Equation (2-6). Optimization weights, which define the objective function, are tuned per vehicle and remain constant. The same holds for the constraint parameters which define the constraints. Optimizer settings control the operation of the Optimizer, such as the number of iterations or stopping criteria.

For all vehicles and all experiments, the sample time of the controller was set to 0.5s This sample time was based on the original simulation done in [19].

4-2-1 Modules

This section details the specifics of each control module, Figure 4-5 displays an overview of the data flows and control structure.

Puppetmaster This module triggers specific actions based on the system clock and has an update rate of 100 Hertz. At each update instance, the module reads vehicle states and monitors the Optimizer for results. When a new sample starts, this module issues a new optimization problem to the Optimizer and transmits the previously planned torque input to the simulator. Next, via the host-pc, the module sends computed planned trajectories of the host vehicle and receives the planned trajectories from the preceding vehicle, if applicable it also receives initialization data from the lead vehicle.

Optimizer For the intensive computation of creating an optimal input trajectory, a dedicated code module was created. By separating the optimization, a dedicated process is created operating independently of the other models. The goal is to generate an input sequence $\hat{\mathbf{u}}_i(:|t_{k+1})$ which minimizes the objective function as seen in Equation (3-1), while satisfying predefined constraints. The identified vehicle model presented non-linear behaviour when applying large input signals. Hence, the entries in the input sequence were limited. The limit was set to $|\hat{\mathbf{u}}_i(:|t_{k+1})| < 1000\text{N}\ \forall i$

The Optimizer module follows the same procedure for all samples, except when initializing. During initialization, specific constraints are applied and weights on unavailable information are set to zero. The unavailable information concerns that of previous calculations, which are determined with the use of an old desired velocity and thus do not apply to the new one.

Optimization starts when commanded by the Puppetmaster module. This command is issued at the beginning of a new sample and has a run time of 80% of the sample time. For the optimization, current vehicle states $x_i(t_k), v_i(t_k)$ together with the previous calculated and received assumed trajectories $\hat{\mathbf{u}}_i(:|t_k)$, $\hat{\mathbf{z}}_i(:|t_{k+1})$, $\hat{\mathbf{z}}_{i-1}(:|t_{k+1})$ are required. Note that the input sequence precedes the error sequences by a single sample, this is a result of the error sequences being a result of the input sequence.

First, the initial states $x_i(t_k)$,$v_i(t_k)$ and first planned input signal $\hat{u}_i(t_k|t_k)$ are used to determine the states at time t_{k+1}. This step is required to estimate the initial states at the start of the prediction horizon. Next, error trajectories are shifted by a single sample to match the next prediction horizon. The first entry of the old sequence is unused and all remaining are cast into the new shifted sequence. The unknown last sample of the prediction horizon is set to zero. This value is based on the assumption that the final value of the error vector converges to zero. This procedure is shown in Equation (4-1) and is applied to both $\hat{\mathbf{z}}_i(:|t_{k+1})$ and $\hat{\mathbf{z}}_{i-1}(:|t_{k+1})$.

$$\begin{aligned} \hat{\mathbf{z}}_i([t_{k+2}, t_{k+N}]|t_{k+2}) &= \hat{\mathbf{z}}_i([t_{k+2}, t_{k+N}|t_{k+1}) \\ \hat{z}_i(t_{k+N+1}|t_{k+2}) &= 0 \end{aligned} \tag{4-1}$$

For the planned input sequence the last value is set to zero as well. While this is not the expected value of that instance, the input sequence is only used as an initial estimate and not used as a variable in further computation. Equation (4-2) shows the time shift for the input.

$$\begin{aligned} \hat{\mathbf{u}}_i([t_{k+1}, t_{k+N-1}]|t_{k+1}) &= \hat{\mathbf{u}}_i([t_{k+1}, t_{k+N-1}|t_k) \\ \hat{u}_i(t_{k+N}|t_{k+1}) &= 0 \end{aligned} \tag{4-2}$$

With trajectories matching the coming prediction horizon, the solver is started and creates an input trajectory $\hat{\mathbf{u}}_i(:|t_{k+1})$ to minimizes the objective function. The solver iterates though a multitude of candidate input trajectories $\mathbf{u}_i^p(:|t_{k+1})$. At each iteration the input trajectory is used to evaluate the vehicle's error trajectory $\mathbf{z}_i^p(:|t_{k+2})$ using the vehicle model. Then, using the 2 candidate trajectories and the 2 static error trajectories $\hat{\mathbf{z}}_i(:|t_{k+2})$,$\hat{\mathbf{z}}_{i-1}(:|t_{k+2})$, the objective function is evaluated. The resulting value is then returned to the solver which creates a new candidate input trajectory for the next iteration.

The minimization is performed by the COBYLA solver by Powell [27] as implemented in the open source toolbox NLOpt [28]. This solver iterates on the optimization problem by creating linear approximations of the objective and constraint functions. These approximations are solved and the solutions are evaluated on the original problem. The results are then used as a starting point for the next iteration. The optimization main criterium to stop optimizing was the maximum amount of iterations. This value was tuned experimentally for the Optimizer to compute within the sample time of 0.5s.

Once the Optimizer is done, the new resulting trajectories, $\hat{\mathbf{u}}_i(:|t_{k+1})$, $\hat{\mathbf{z}}_i(:|t_{k+2})$, $\hat{\mathbf{z}}_{i-1}(:|t_{k+2})$, are returned to the Puppetmaster module for transmission and application.

4-2-2 Operational sequence

There are two main sequences when the controller is operating. For the majority of the runtime, the 'Normal' sequence is used. However, when the leader changes desired velocity a single 'Initialization' sequence is triggered.

Weights of the objective function stay constant during regular operation but can differ between vehicles. During an initial optimization the planned error trajectory, of both the host and preceding vehicle, becomes invalid for the new desired velocity. Thus, for the initial optimization, these weights are set to zero, neglecting the invalid values. The lead vehicle, which has no predecessor, has the weights on relative error set to zero.

Normal sequence During normal operation, four events occur within a sample as depicted in Figure 4-3. The sequence illustrated starts at t_k and ends at t_{k+1}. Before the start of a sample, it is assumed all planned data from the previous step are available. These data consists of the planned input trajectory $\hat{\mathbf{u}}_i(:|t_k)$, the planned error trajectory of host vehicle $\hat{\mathbf{z}}_i(:|t_{k+1})$, and predecessor's planned error trajectory $\hat{\mathbf{z}}_{i-1}(:|t_{k+1})$.

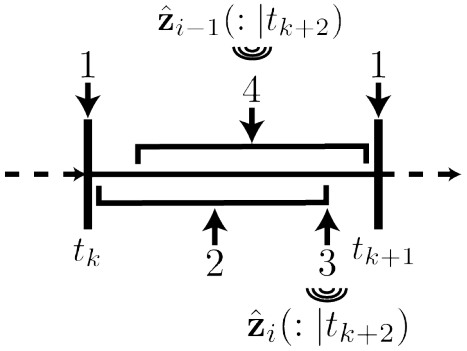

Figure 4-3: Sample procedure

(1) At the start of a sample, the Puppetmaster applies the previously planned input $\hat{u}_i(t_k|t_k)$ to the vehicle. This input signal is held constant for the duration of the sample. In the same instance, the Optimizer is provided with the required trajectories calculated in the previous sample, $\hat{\mathbf{u}}_i(:|t_k), \hat{\mathbf{z}}_i(:|t_{k+1}), \hat{\mathbf{z}}_{i-1}(:|t_{k+1})$, and the current system states $x_i(t), v_i(t)$.

(2) Upon receiving new information, the Optimizer solves the optimization problem with objective function $J_{i,t_{k+1}}(\cdot)$. This results in a new planned input trajectory $\hat{\mathbf{u}}_i(:|t_{k+1})$ and the resulting error trajectory $\hat{\mathbf{z}}_i(:|t_{k+2})$.

(3) Once the Optimizer finishes, the results are stored for use in the next sample and the planned error trajectory $\hat{\mathbf{z}}_i(:|t_{k+2})$ is transmitted to the first following vehicle upstream $(i+1)$.

(4) During the sample, the planned error trajectory $\hat{\mathbf{z}}_{i-1}(:|t_{k+2})$ of the preceding vehicle downstream is received by the Puppetmaster module.

With all new data available, the system is ready to repeat this procedure for the following sample. All vehicles follow this procedure, except for the last vehicle $i = i_{end}$ and the lead vehicle $i = 0$ who respectively do not send and receive error trajectories.

Initialization sequence For an initial sequence, a dedicated procedure is in effect. This procedure is triggered by a change of desired velocity from the lead vehicle. The procedure is illustrated in Figure 4-4. When a vehicle is to optimize the initial trajectories spanning $(:|t_{plan})$ specific initial constraints are in effect. The same is true for weights on aspects requiring previously calculated data, which are set to zero. The diminishing of weights is required due to the fact that the data available is not yet optimized for the new desired velocity but rather to the previous one.

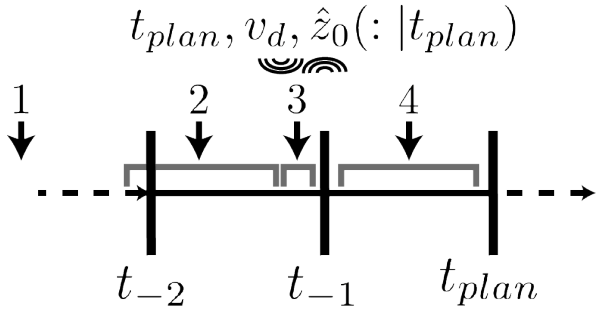

Figure 4-4: Initialization procedure

(1) At a certain time, the leader plans a new velocity v_d and time to implement t_{plan}, whereby t_{plan} is at least three samples ahead, i.e. $t_{plan} > t + 3\delta$. It is assumed that the entire platoon is stable, thus $\forall i : \mathbf{z}_i(t_k) = 0$ for $t_k = t_{-1}$ and $t_k = t_{plan}$.

(2) With the new desired data set, the leader starts an optimization sequence directly after the optimization of t_{-3} finishes, as opposed to waiting for the sample to end. The goal is to create the initial sequences $\hat{\mathbf{u}}_0(:|t_{plan})$, $\hat{\mathbf{z}}_0(:|t_{plan+1})$. These trajectories are found by solving the optimization problem with objective function $J_{0,t_0}(\cdot)$. Vehicles upstream of the leader still resume regular operation.

(3) Between the Optimizer finishing and time t_{-1}, the initialization data from the leader, $t_{plan}, \hat{\mathbf{z}}_0(:|t_{plan+1}), v_d$, is transmitted to all upstream vehicles. This data is consequently received before t_{-1} by all upstream vehicles.

(4) With all followers aware of the change and having received the trajectory of the leader, all create a planned input $\hat{\mathbf{u}}_i(:|t_{plan})$ and error trajectory $\hat{\mathbf{z}}_i(:|t_{plan+1})$ for the initial sample. This is realised by solving the optimization problem with objective function $J_{i,t_0}(\cdot)$, constraint to the specific initialization constraints.

After (**4**) regular operation is resumed.

For $t < t_{plan}$, all vehicles apply planned inputs calculated to match the previous desired velocity. However, the lead vehicle misses one calculation for the input during $t \in (t_{-1}, t_{plan}]$. This missing input is taken from the previous calculation, thus $\hat{u}_i(t_{-1}|t_{-2})$ is applied.

4-3 Communication

Implementation of Cooperative Adaptive Cruise Control (CACC) requires a communication structure. This section elaborates on the method employed to communicate between the physical pieces of hardware.

4-3-1 Structure

All communication transfers via the data structure 'double' [29] which uses 8 bytes of data. With a precision up to 15 significant digits, the data structure is more than sufficient for the application of vehicle platooning.

For the simulation platform, data is transferred via Ethernet cable. All communication between vehicle simulators is relayed through the host-pc using LabVIEW. The host-pc runs a dedicated 'visual Instrument' (VI) in LabVIEW 2014, which reads, logs, and relays the outputs of each PXI. Figure 4-5 shows a graphical depiction of the communication structure of all data flows with an expansion of the internal structure of vehicle i.

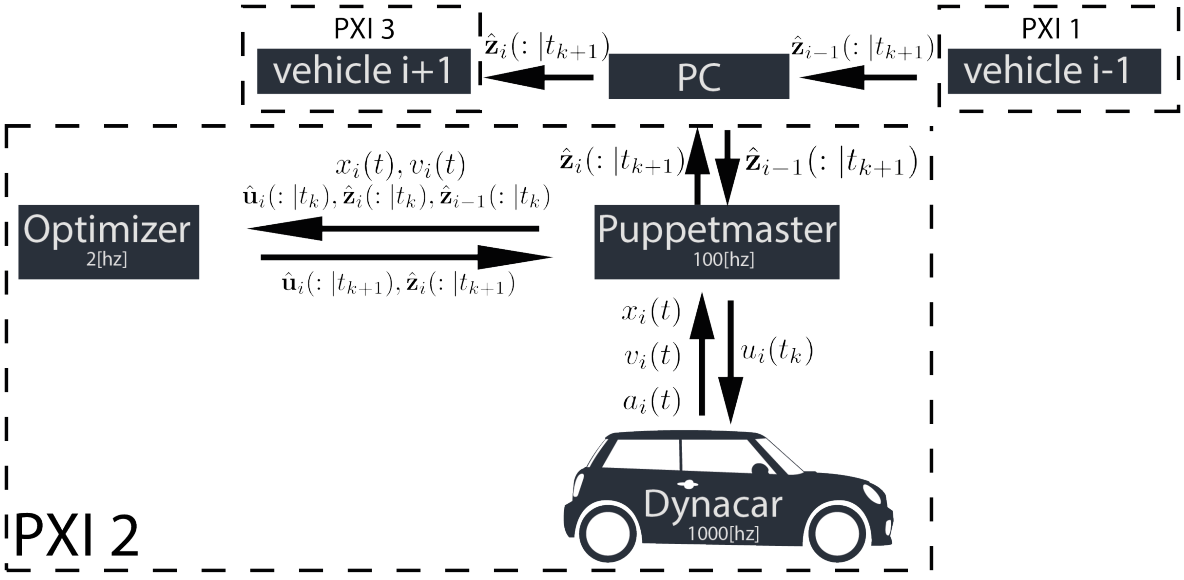

Figure 4-5: Communication structure

4-3-2 Latency, bandwidth, and failure handling

The introduction of communication raises the issues of limits concerning data latency and throughput.

Latency describes the travel time of data. This travel time creates a delay between vehicles concerning communication. For the structure employed experiments, which relayed the PXI clock signal, showed a round trip time of either 60ms or 77ms with a mean of 66ms. Under the assumption of symmetry between transmitting and receiving, a single transmission has a latency of 33 ms. This amount of delay fits within the transmission window created by the operational sequences. This window consists of 20% of the sample time, in the case of the controller sample time of 0.5s this leaves about 70ms for the transmission.

Bandwidth, defined by the amount of data per second, also requires analysis. During normal operation, each vehicle only communicates the planned error trajectory $\hat{\mathbf{z}}_i(:|t_k)$ once per sample. These datasets contain the two error vectors each consisting of 25 'doubles'. With a double spanning 8B of data, a single transmission of a sample contains 400B of data. This data has to be transferred in the time window between the Optimizer finishing and the end of the current sample. Under the assumption that the Optimizer operates as expected and neglecting the delay, this would leave 0.1s for transmission of the data. The window of 0.1s containing 400B of data would require a minimum bandwidth of 4kB/s. Taking into account the worst case measured one-way communication delay of 35ms, the requirement would rise to almost 6.2kB/s. This minimal required bandwidth is well below the capabilities of communication via modern computer networks.

For single clients, the transmission requirements pose no limit on extending the system. However, the central host-pc will need to relay more data proportional to the number of simulators running. With the used setup of three vehicles, the total amount of data to relay concerns two sets, hence requiring the host-pc to transfer data at 8kB/s.

During the initialization operation, a different communication protocol is applied. In this scenario, communication occurs at the start of a sample and is received shortly after. This behaviour is outside of the window of regular trajectory communication. Data communicated consist again of the planned error vectors, containing a total of 50 values, with the addition of the new desired velocity and planned time to start. Thus a total of 52 doubles are to be communicated. This data is transmitted in a larger window of 0.2s due to the early optimization of the leader. Thus using the same calculations as for the regular behaviour, the minimum transmission rate required would be 2.08kB/s.

Communication failure handling If communication is not received, either by failure or delay, the controller is required to continue operation. As a failover procedure, all vehicles retain the last successfully received data until new data is received. Thus if a transmission is not received, the controller can time-shift the last known trajectory to match using Equation (4-2). This process can be repeated until the prediction horizon of the last received transmission is reached. From this point, the entire original received trajectory has passed and there is no remaining knowledge of the other vehicles.

During experiments, there was no loss of communication detected.

4-4 Vehicle model

For evaluation of the objective function, future error trajectories are required. Prediction of these trajectories is done by evaluation of a candidate input sequence to a vehicle model. The vehicle model returns resulting position and velocity trajectories, which in turn are used to calculate the error trajectories.

The vehicle model is based on a double integrator model of which the parameters were identified by the use of an identification experiment. Equation (2-3) shows the model and Appendix A elaborates on the identification experiment. The identification was done with the input limited to an absolute value of 1000N, this is due to non-linearities occurring at higher values. The controller was subjected to the same input constraints as well.

The resulting parameters are shown in Table A-1 and are identical for all vehicles and algorithms. Note that the vehicle simulator accepts wheel torque and the model applies direct force. As a result, the identified parameters are scaled to the simulator value.

With the continuous model set and identified, it is implemented in the optimization module employing integration. To integrate the model, Euler's method was applied with a step size of $\frac{0.5}{30}$s thus iterating 30 times per prediction sample.

Chapter 5

Experiments and results

This chapter introduces the experiments implemented, after which the results are presented. To illustrate these results a selection of graphs is shown. All remaining graphs are can be found in Appendix B.

5-1 Experiment setup

Each algorithm as described in Section 3-2 is subjected to five experiments ran in a single simulation. Before an experiment is started, the desired velocity is set to the initial value. Enough time is taken into account to allow the platoon to stabilize. The first three experiments concern a change in desired velocity and the last two a disturbance on system operation.

All vehicle simulators are set to be identical and controlled by individual controllers, which differ only in tuning weights and vehicle index. The weighting matrices are only tuned by their diagonal entries, setting the non-diagonal entries to zero. This results in penalizing the different variables independently. The simulation environment is set to a large open plain without any obstacle or change in elevation. Visualization of the vehicles and manual steering control are disabled for the experiments.

5-1-1 Changing velocity

These three experiments demonstrate how the system responds to a change in desired velocity. All are based on preceding research by Krzesinski [30], which is based on [31]. At each change of desired velocity the initialization procedure is triggered as described in Section 4-2-2. Thus, no upstream vehicle has prior knowledge of the velocity change.

For the graphs depicting single experiments, the time axis was adjusted such that at $t = 0$s the first velocity change occurs, thus setting $t_0 = 0$. All experiments, together with an overview of the full experiment, can be seen in Figure 5-1.

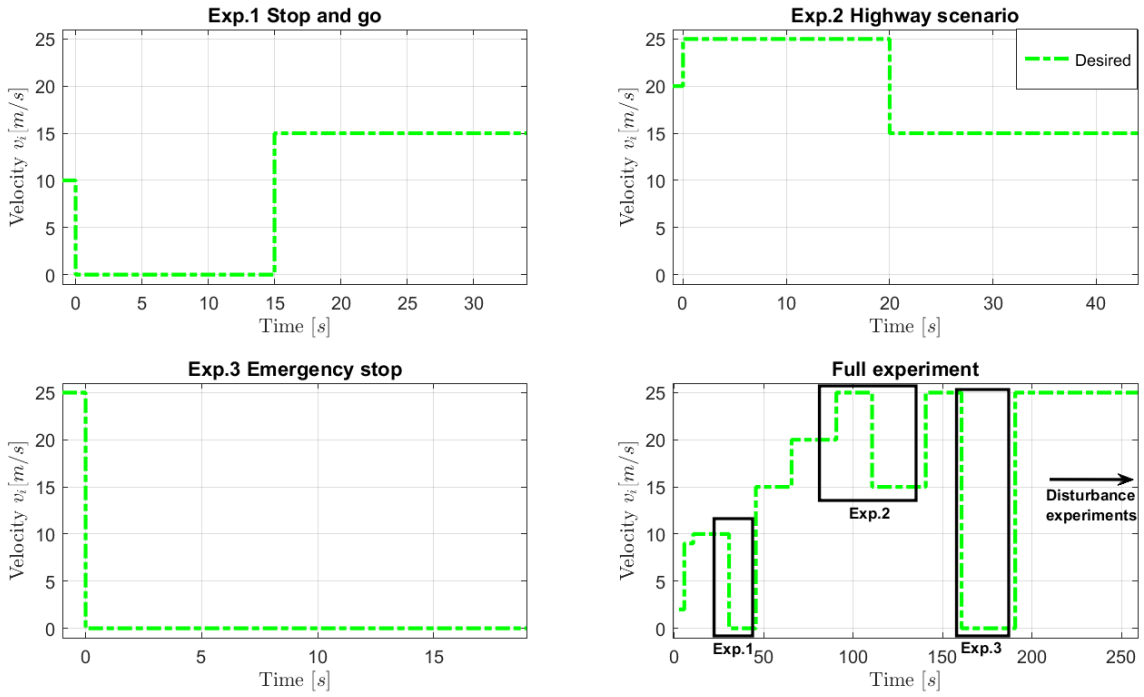

Figure 5-1: Desired velocity profiles for experiments. Time axis shifted such that $t = 0$ at initial velocity change for Exp. 1, 2, and 3.

Stop and go The first experiment is defined as 'Stop and go', which requires the platoon to make a full stop after which it is to accelerate. The challenge lies in how the platoon responds to a standstill. Equation (5-1) shows the desired velocity profile applied.

$$v_d(t) = \begin{cases} 10, & \text{if } t < t_0 \\ 0, & \text{if } t_0 \leq t < t_0 + 15 \\ 15, & \text{if } 15 \leq t \end{cases} \tag{5-1}$$

Highway scenario The second experiment aims to emulate a common occurrence with regular highway driving. This experiment starts at a low highway velocity, followed by a switch to a higher velocity. After 20 seconds, the desired velocity is set lower than the initial value. The primary interest is to see how the platoon stabilizes and how string stability is affected. Equation (5-2) shows the desired velocity profile applied.

$$v_d(t) = \begin{cases} 20, & \text{if } t < t_0 \\ 25, & \text{if } t_0 \leq t < t_0 + 20 \\ 15, & \text{if } 20 \leq t \end{cases} \tag{5-2}$$

Emergency stop The last experiment that involves a change of desired velocity is the 'Emergency stop'. This scenario has the desired velocity go to standstill from a high velocity. Equation (5-2) shows the desired velocity profile applied.

$$v_d(t) = \begin{cases} 25, & \text{if } t < t_0 \\ 0, & \text{if } t_0 \leq t \end{cases} \tag{5-3}$$

5-1-2 Handling disturbances

This section depicts the remaining two experiments where the desired velocity is held constant at $v_d = 25[m/s]$ and the system is subjected to 2 disturbance scenarios. The experiments are executed after the velocity experiments.

Force disturbance The first disturbance scenario is an applied disturbance force to the lead vehicle unknown to the controller. The Puppetmaster applies this force by subtracting the planned disturbance from the applied input. The force is set to a brake force of 600N, constantly applied for 3s. This is 60% of the brake force that can be applied by the controller. The controller has no notion of this disturbance other than the states changing. Thus, the force cannot be accounted for in the prediction.

Communication drop out The last experiment concerns how the platoon responds when communication fails. To simulate this, the leader stops communicating upstream for 1.5s. This results in the second vehicle resorting to communication failover.

5-2 Results

This section presents the results of the experiments. Three subsections are presented each discussing how the algorithms performed. Plotted results are presented in Appendix B. All experiments used the same objective function weights, which were tuned on the unconstrained experiment. Table 5-1 presents the applied weights.

The results plotted show data directly from the vehicle simulator, not from the controller. Thus, these results are not from the simplified vehicle model. All simulated sensor data is from the vehicle's centre of mass, situated in the passenger cabin. Due to update instances of the different simulator not being synchronised, the data shown is interpolated onto a common time vector. This vector is constructed by combining the three individual time vectors.

5-2-1 Unconstrained tuning

For tuning of the objective weights and to create a base of results for comparison, a set of initial experiments were performed. The weights were tuned for the platoon to present stable and non-colliding behaviour. The best results were seen by giving vehicles lower constraints than that of the predecessor. Except for the weights on position error and velocity, which were held identical and were set higher respectively. Table 5-1 presents the used objective weights.

Because the results of this experiment were similar to that of the safety algorithm, graphs of the results were omitted from the appendix. Two graphs relevant for comparison are presented below in Figure 5-2 and 5-3.

The results presented stable non-colliding behaviour. Interesting behaviour is seen in the acceleration plot when coming to a standstill. The acceleration shows large oscillation around zero. It is assumed that these oscillations are a result of braking and suspension dynamics present in the simulator.

Table 5-1: Objective function weights

Par.	Value	Par.	Value	Par.	Value
$Q_0^{0,0}$	100	$Q_1^{0,0}$	100	$Q_2^{0,0}$	100
$Q_0^{1,1}$	150	$Q_1^{1,1}$	200	$Q_2^{1,1}$	250
$G_0^{0,0}$	-	$G_1^{0,0}$	100	$G_2^{0,0}$	50
$G_0^{1,1}$	-	$G_1^{1,1}$	100	$G_2^{1,1}$	50
$F_0^{0,0}$	200	$F_1^{0,0}$	100	$F_2^{0,0}$	50
$F_0^{1,1}$	200	$F_1^{1,1}$	100	$F_2^{1,1}$	50
R_0	1	R_1	0.2	R_2	0.1

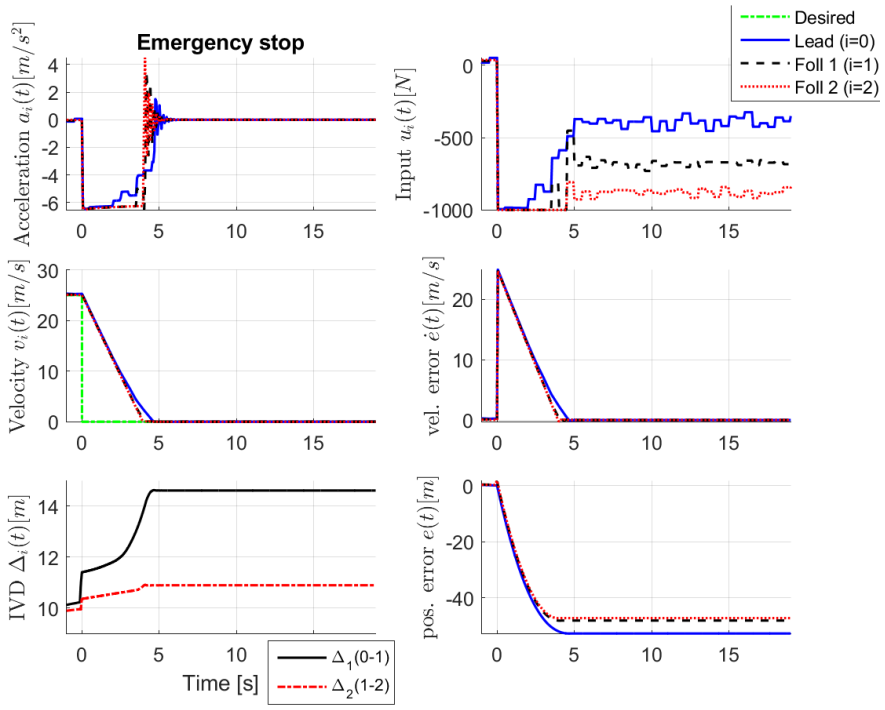

Figure 5-2: Unconstrained 'Emergency stop'. Note the braking behaviour and the rising inter-vehicle distance.

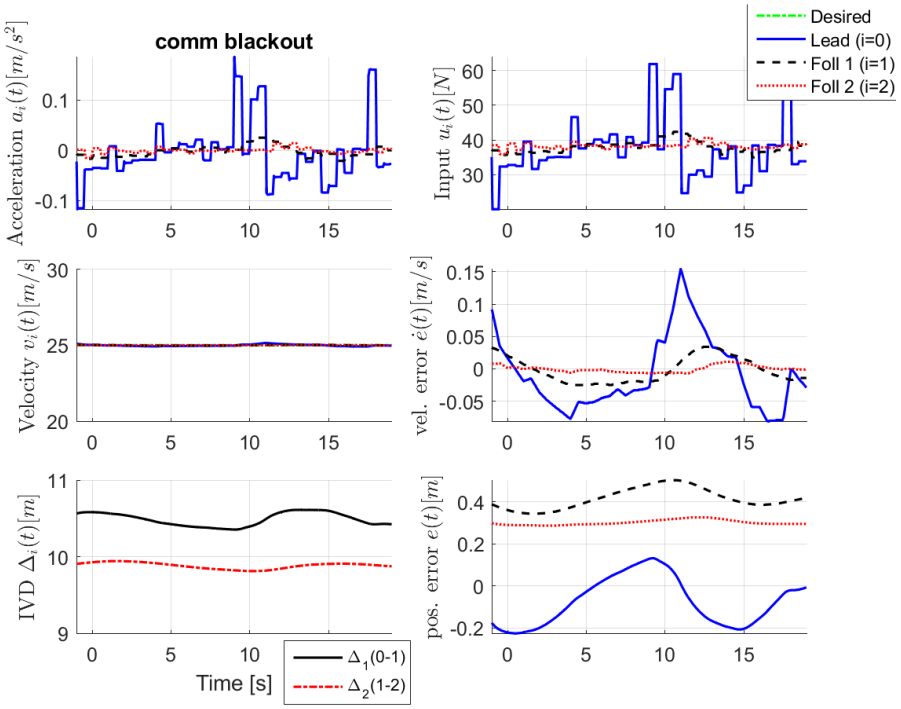

Figure 5-3: Unconstrained 'communication blackout'. Note that behaviour of 'Foll 1' is similar to that of 'Lead', suggesting lack of communication has little influence.

5-2-2 Safety algorithm

The safety algorithm was first implemented with the constraints depicted in Section 3-2-1. These constraints aim to satisfy the notion of 'general safety and stability' as presented in Section 2-3.

With implementation of constraints, the platoon was subjected to the experiments. For experiments without a standstill, the platoon presented proper behaviour similar to the unconstrained experiment. However, when the platoon was required to slow down to a stop an issue appeared. The platoon did not hold the desired velocity and predecessor-follower stability was lost. This behaviour is shown in Figure 5-4, especially in the bottom two graphs.

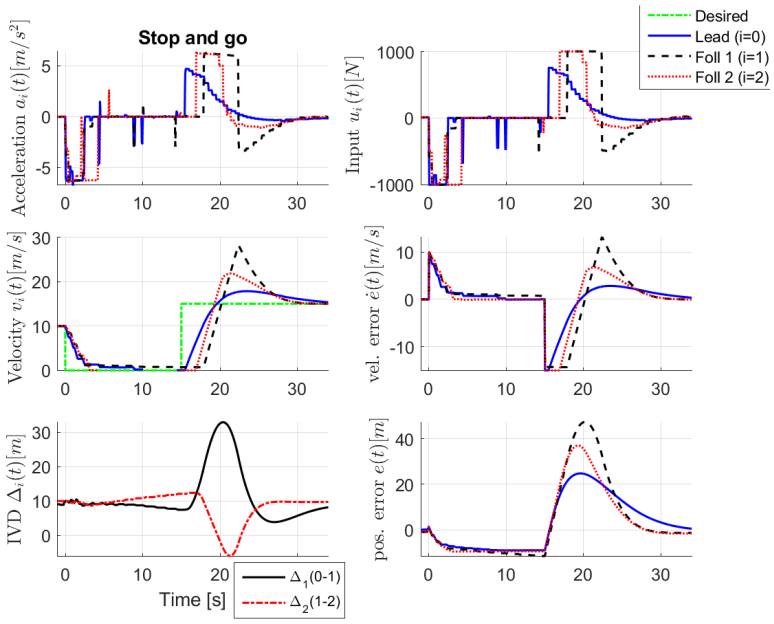

Figure 5-4: Safety algorithm 'Stop and go', unsatisfactory behaviour. Inter-vehicle distance shows movement while standstill is desired and position error violates platoon stability.

To mitigate this behaviour the standstill prevention constraint was neglected when the desired velocity was below 1m/s. This alteration effectively removes the constraint. The removal allowed for the controller to plan trajectories with negative velocity. However, due to the method of implementation, the controller is only able to apply brakes and cannot reverse. The issue and the alteration are further discussed in 6-1-1.

Planned trajectories by the controller showed to satisfy the implemented constraints for all experiments except the 'force disturbance' experiment. During which, the inter-vehicle distance between the lead vehicle ($i = 0$) and the first following vehicle ($i = 1$) showed negative values. While the preceding vehicle did react, the reaction was insufficient. This is a deterioration compared to the unconstrained base experiment. While the 'no collision' constraint is clearly violated in the results, the following vehicle did not anticipate the collision. This is a result of the preceding vehicle not being aware of the disturbance, communicating a more beneficial trajectory compared to reality.

When comparing the results to the unconstrained experiment, the safety constraints showed a slight improvement of inter-vehicle distance. The 'emergency stop' experiments, seen in Figure 5-2 and B-3, show this improvement when compared.

5-2-3 Platoon stability algorithm

Experiments with platoon stability constraints, as defined in Section 3-2-2, presented issues in satisfying the optimization constraints. The choice of parameters was made within the constraint depicted in Equation 3-13. Different attempts were made to find parameters that would enable the Optimizer to find solutions, but there was no success. Also, the complexity caused computational limits to be reached requiring lowering of the number of allowed iterations for the Optimizer. The failure of finding a solution within the constraints is due to infeasibility, which is further described in Section 6-1-2.

While the solutions did not satisfy the constraints, they did approximate them. The controller implemented input signals with partially satisfying results. Most experiments presented results which prevented collisions and maintained predecessor-follower stability. The results and used parameters can be found in Appendix B-2.

Velocity experiments showed less deviation in inter-vehicle distance between the last two vehicles, in comparison to other algorithms. This behaviour is a result of Optimizer solutions approaching the constraints, which results in the last two vehicles having similar trajectories. However, there was a violation of platoon stability as defined in Equation (2-11) for the 'stop and go' experiment as seen in Figure B-6. This was caused by a change in planned trajectory, presumably caused by Optimizer issues.

For the disturbance experiments, the trajectories were similar to other algorithms. The 'force disturbance' experiment showed a collision presence, as seen in Figure B-4. However, the size of inter-vehicle distance violation was an improvement over other algorithms.

5-2-4 Comfort algorithm

Comfort algorithm experiments were done with the constraints depicted by Section 3-2-3. These constraints were implemented on the acceleration results of the vehicle model, and the change of that acceleration. Appendix B-3 presents the results

The best results were presented in the 'highway scenario', depicted in Figure B-12. Both comfort and jerk constraints were mostly satisfied, both on the vehicle simulation as in the planned trajectories. In addition, string stability was maintained for the experiment. When comparing this result to other algorithms the performance of the maximum error is similar, but the time to error convergence is longer. Figure 5-5 present the violation count of the 'Highway scenario'.

All results showed a deterioration in rise time towards the desired velocity compared to other experiments, which is to be expected when a vehicle's performance is limited. In addition, some experiments showed abrupt changes in acceleration and deceleration. These abrupt changes resulted in the violation of platoon stability. For example, the 'Stop and go' scenario depicted in Figure B-11 showed the last vehicle to lower its acceleration, causing it to lag behind on the platoon. A further consequence is the more jagged inter-vehicle plots due to two vehicles executing abrupt changes.

In all experiments, acceleration commands were found to be jagged and the results do not seem intuitive. One would expect when restricted, that the applied inputs would result in the vehicle operating at its acceleration limits. However, the inputs from the controller do not. In addition, at large velocity jumps which are required to transition between experiments, the vehicles trajectories did not manage to stabilize. This resulted in the 'emergency stop' having a non zero error at time t_0. It is suspected that the Optimizer results can be improved. This is further discussed in Section 6-2

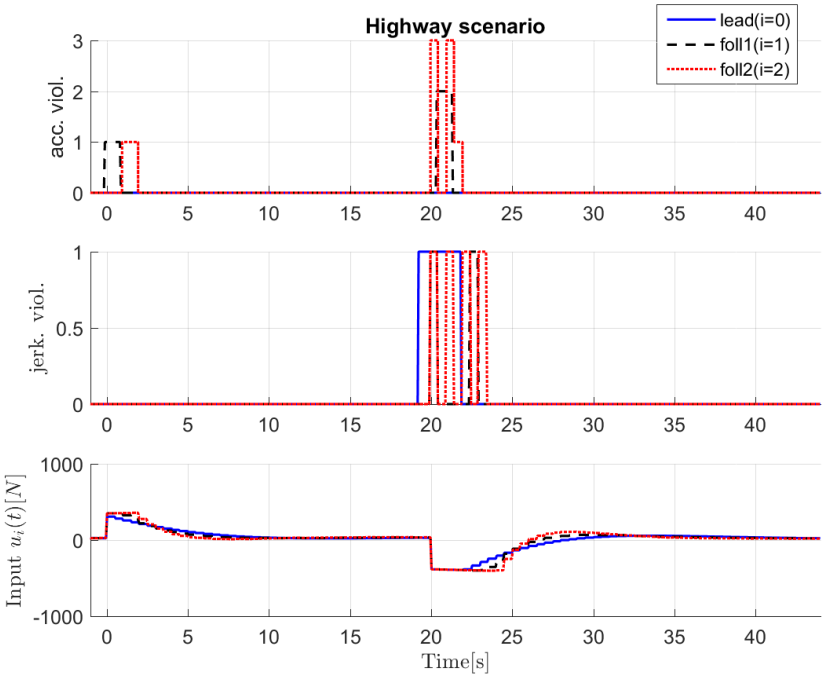

Figure 5-5: Comfort algorithm, planned violation count, and planned input. Note that violations occur at changes in desired velocity and are more frequent for upstream vehicles.

Chapter 6

Discussion

6-1 Constraint feasibility issues

Results from the experiments showed that the application of constraints caused issues with the Optimizer calculating a solution. Two constraint sets are examined further in this section, the no reversal constraint from the safety algorithm and the constraints for platoon stability.

6-1-1 Vehicle reversing

In the original safety experiment (first paragraph of Section 5-2-2), vehicle reversing was constraint by a lower limit of zero on velocity. This constraint is depicted in Equation (3-5). When applied, the vehicle would not remain stationary when the desired velocity was set to zero.

The direct cause was the controller sending positive input signals while at a standstill. This caused the vehicle to move. These positive inputs were generated by the Optimizer returning sub-optimal sequences, which were a result of the Optimizer not finding a solution that satisfied the constraints.

Two constraints directly conflict when the desired velocity is set equal to, or beyond a vehicle velocity limit at the initialization sequence. These concern the lower limit on velocity and the final value constraint on position error, as seen in Equation (3-4) and (3-5) respectively. The conflict becomes clear when evaluating the velocity error from Equation (2-7) and the position error at the first sample after a desired velocity change, i.e. at time $t = t_1$. For analysis, the conflict is described for two general cases. The first a lower limit on velocity $v_i(t) \leq v_{min}$, the second an upper limit on velocity $v_i(t) \geq v_{max}$.

Evaluation of the velocity error definition shows that velocity error $\dot{e}_i(t)$ becomes either upper or lower bounded by zero when the velocity is constrained, and the desired velocity coincides or surpasses that boundary. Equation (6-1) depicts the resulting bounds for both upper and lower limited velocity.

$$\begin{aligned} v_i(t) \geq v_{min} \quad \wedge \quad v_d \leq v_{min} \quad &\implies \dot{e}_i(t) \leq 0 \quad ,\forall t > t_0 \\ v_i(t) \leq v_{max} \quad \wedge \quad v_d \geq v_{max} \quad &\implies \dot{e}_i(t) \geq 0 \quad ,\forall t > t_0 \end{aligned} \tag{6-1}$$

Next, the evaluation of the first position error at $t = t_1$. Both the vehicle simulator and controller apply force control. For the controller the amount of force is bounded, this results in the inability to track discrete velocity changes exactly. This inability, the limit on velocity, and desired velocity on or over this velocity limit cause an error at $t = t_1$. The sign of this error coincides with the direction of the desired velocity change. Equation (6-2) presents the logic.

$$\begin{aligned} v_i(t) \geq v_{min} \quad \wedge \quad v_i(t_0) > v_d \quad \wedge \quad v_d \leq v_{min} \implies e_i(t_1) < 0 \\ v_i(t) \leq v_{max} \quad \wedge \quad v_i(t_0) < v_d \quad \wedge \quad v_d \geq v_{max} \implies e_i(t_1) > 0 \end{aligned} \tag{6-2}$$

By combining Equations (6-1) and (6-2) a constraint on the final position error is constructed, as seen in Equation 6-3. This constraint is in direct conflict with the final value constraint on position error, depicted in Equation (3-4). Thus the application of a velocity constraint, in combination with a desired velocity on or over that constraint, prevents any solution satisfying the final position error constraint.

$$\begin{aligned} e_i(t_1) < 0 \quad \wedge \quad \dot{e}_i(t) \leq 0 \ ,\forall t > t_0 \quad &\implies e_i(t) < 0 \quad \forall t > t_1 \\ e_i(t_1) > 0 \quad \wedge \quad \dot{e}_i(t) \geq 0 \ ,\forall t > t_0 \quad &\implies e_i(t) > 0 \quad \forall t > t_1 \end{aligned} \tag{6-3}$$

The infeasibility in the experiment was solved by removing the desired velocity constraint. An alternative would be to introduce a limit on the desired velocity or prevent large discrete steps. The limit would be a subset of the velocity constraints such that the velocity error $\dot{e}_i(t)$ can change signs. Prevention of large discrete steps would allow a vehicle to move to a velocity close to a boundary, before coinciding. This would still require some introduction of slack on the final constraint due to an unavoidable small error.

Another solution would be to alter the controller to not follow a desired position $x_{d,i}(t)$. Only creating constraints on relative variables, not dependent on a constant absolute position. However, for the controller framework implemented this is difficult to realize. Whereas the only data from other vehicles available are the relative error trajectories. And as such the derived constraints are dependent on the initial position of the lead vehicle to cancel out between vehicles.

6-1-2 Platoon stability

For the 'Platoon stability' experiment the Optimizer was unable to find a solution which satisfied the constraints. The initialization constraint was violated for most prediction horizon samples on all following vehicles. Whereas for the regular constraint, the only results within bounds were by the lead vehicle after a change in desired velocity. The following vehicles never satisfied the constraint. However, they did approach it. Figure 6-1 shows in the top graph the regular constraint value for each sample, in the middle the number of samples violating the initialization constraint, and at the bottom the amount of missed communications.

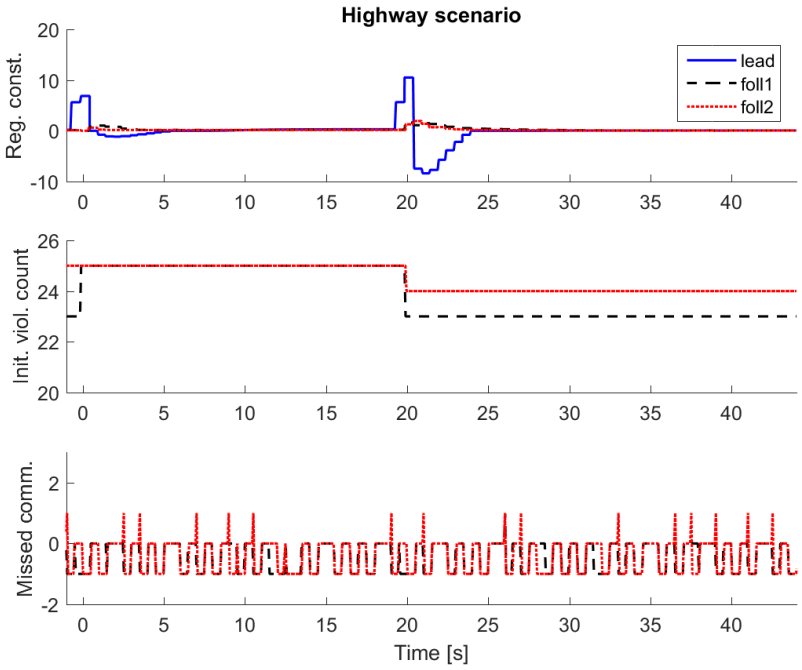

Figure 6-1: Platoon algorithm constraint violation count. Note the regular constraint is only satisfied (negative) for the 'Lead' vehicle and the initial constraints are violated for most samples.

As mentioned the constraints for platoon stability are strict and thus resemble an equality constraint due to the nature of the parametrization. For the initialization constraint, the planned error trajectory of all following vehicles is to be within a band of the initial planned trajectory of the lead vehicle. The structure itself does not necessarily become infeasible, but due to the parametrization that band becomes very narrow and is scaled to be smaller than that of the leader. Thus, following vehicles are to plan an initial trajectory closely matching that of the leader but a multitude smaller. This trajectory is infeasible due to the limit on vehicle inputs, and no solutions were found for most samples.

For the regular constraint, the issue is similar. Parameter $\epsilon_{i,k}$, which scales the upper bound on deviation of a planned trajectory, is defined as $\epsilon_{i,k} = \epsilon_i^k \in (0, 1)$. This was done to simplify the required conditions for the stability proofs, as presented in [19].

The simpler definition, combined with the tight parametrization, resulted in the value rapidly approaching zero with each passing sample. As a result, the inequality constraint in Equation (3-11) effectively becomes an equality constraint, as seen in Equation (6-4). This equality constraint sets all vehicles absolute maximum trajectory deviation to be virtually zero.

$$\begin{aligned} ||e_i^p(:|t_k) - \hat{e}_i(:|t_k)||_\infty - \epsilon_{i,k} \min\left\{||\hat{e}_{i-1}([t_k, t_{k+1}]|t_k)||_\infty, ||e_i^p([t_k, t_{k+1}]|t_k)||_\infty\right\} &\leq 0 \\ ||e_i^p(:|t_k) - \hat{e}_i(:|t_k)||_\infty &\lessapprox 0 \\ e_i^p(:|t_k) &\approx \hat{e}_i(:|t_k) \end{aligned} \tag{6-4}$$

This requirement eventually results in a constraint limiting all errors to zero. This is caused by the requirement to maintain the previously calculated trajectory, which is time-shifted each sample and appended with a zero, as was presented in Section 4-2-1.

The method of implementing these constraints is too tight to allow for any error. This makes practical implementation infeasible. A solution is not straightforward. Loosening constraints by altering parameters, would result in breaking the constraints on the parameters themselves, which are required to hold platoon stability. An alternative would be that a vehicle could plan an error trajectory with lower absolute values to that of a predecessor directly. However, the requirement of downstream error attenuation would still require growing input actions downstream. This would put a limit on the number of vehicles, and the issue of model mismatch is not solved.

Another approach would be initializing a common trajectory for all vehicles when a velocity change occurs. This would require a more centralized setup were the computed trajectory needs to fulfil all vehicles needs and capabilities.

6-2 Optimizer performance

In the experiments, the optimized inputs did not seem to be optimal. This mainly appeared in the comfort experiments seen in Section B-3. In those experiments, it can be seen that the acceleration seems to operate well within the limits while other aspects were worsening. Large frequent jumps are also seen in the results for the 'Emergency stop', as depicted in Figure B-13.

To research Optimizer performance, the comfort experiment was repeated with simpler jerk constraints. First, the jerk constraint was simplified to be constant ($|\Delta a(t)| < \frac{5}{2} m/s^2$). This resulted in the platoon to succeed in the transition movements between experiments. However, acceleration was still not applied on its limits and the changes were still abrupt. Figure 6-2 presents the results.

Then, the experiment was repeated without a jerk constraint. This new experiment showed vehicles operating at their acceleration limits. Due to equal comfort constraints for all vehicles, vehicle trajectories were close to identical. The results can be seen in Figure 6-3.

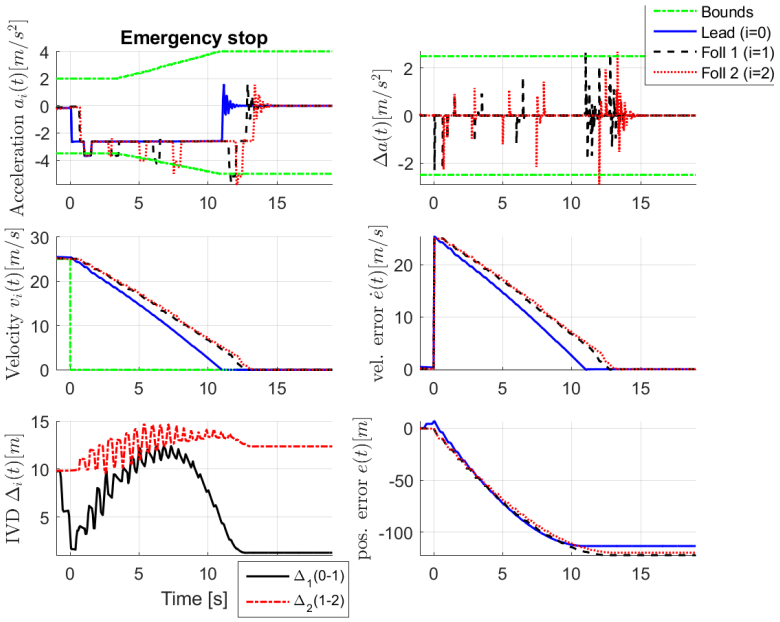

Figure 6-2: Comfort algorithm 'Emergency stop', simplified jerk constraint. An improvement over the full constraint

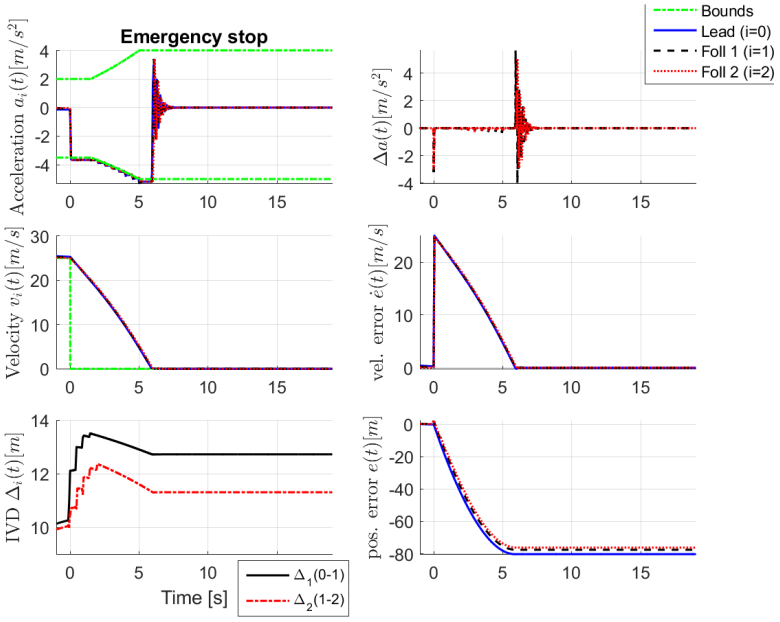

Figure 6-3: Comfort algorithm 'Emergency stop', no jerk constraint. Note acceleration operating on the limits and vehicles matching trajectories.

When reviewing the repeated comfort experiment, the Optimizer seems to perform poorly when extensive constraints are applied. For assessing the performance of the Optimizer, the comfort 'emergency stop' experiment was repeated once more with the original jerk constraints. This additional experiment excessively logged the variables transmitted to the Optimizer. This additional data allows for repeating the optimization off-line.

The logging was done on the second vehicle ($i = 1$). The experiment results for this vehicle are plotted in Figure 6-4. Within the new results, a single point was chosen highlighted by a star. This point in the graph coincides with the applied input which resulted from the initial optimization. All data was logged, and the experiment was repeated off-line with the sole purpose to optimize this situation without time constraints.

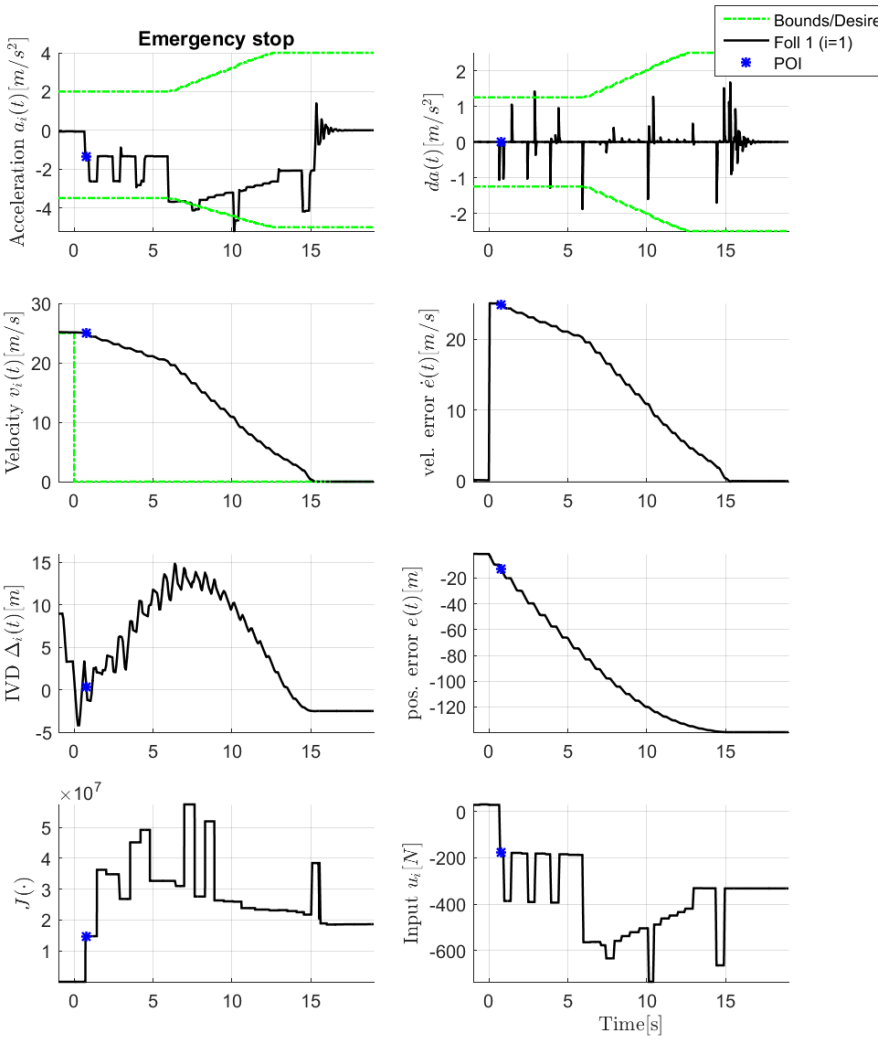

Figure 6-4: Comfort algorithm 'Emergency stop', follower 1. The blue star notes the used experiment data for the off-line evaluation

All recorded inputs were input to the original Optimizer code with both a raised maximum iteration count and time limit. During optimization, the minimal value of the subjective function of each 75 iterations was logged. The resulting plot is displayed in Figure 6-5.

This plot shows that the found optimal value is calculated within the first 75 samples, all iterations after which show a higher value. The values do vary slightly, yet there is no trend towards zero. It can be seen that the Optimizer does not improve the early solution with more iterations. This suggests that the implemented optimization algorithm does not suit the optimization problem. The non-constant and non-linear constraints are a probable cause for this issue.

When the experiment was repeated for the simplified jerk constraint and the removed jerk constraints the plot showed the same behaviour.

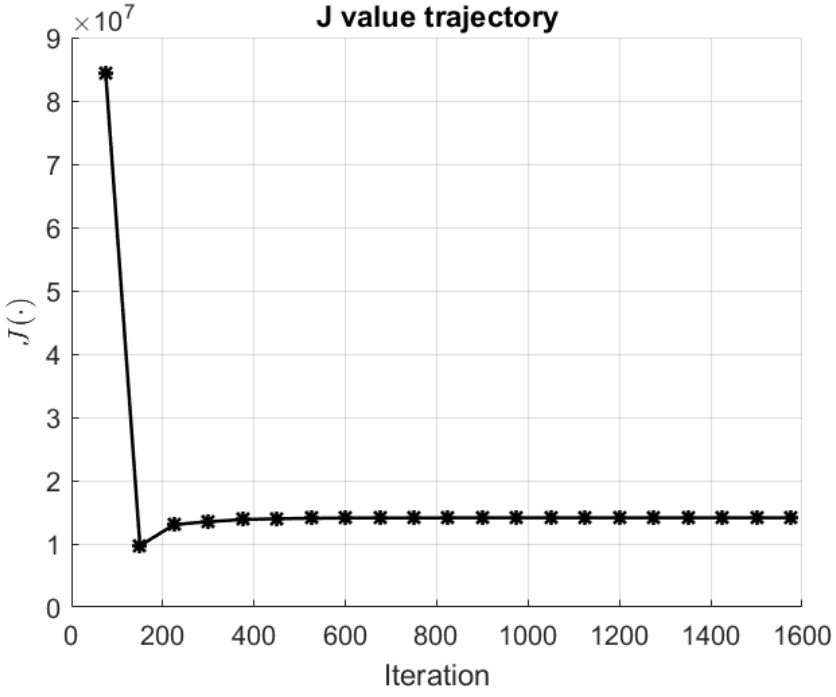

Figure 6-5: Objective function value over optimization iterations. Note the found minimal value is calculated in the first set of iterations.

Chapter 7

Conclusion and Future work

7-1 Conclusion

In this thesis, an algorithm for vehicle platooning applying Model Predictive Control (MPC) with the addition of predecessor trajectory information was implemented and evaluated. The MPC controller was subjected to three separate types of constraints concerning safety, predecessor-follower stability, and comfort. All of these constraints were experimentally evaluated on a predefined set of desired velocity trajectories.

The implementation was successfully realised on three independent real-time simulators in a decentralised fashion. This required a dedicated timing and communication structure to function. When tuned correctly and unconstrained, successful implementation was realised in real-time. Results from the experiments showed the ability to track the desired velocity trajectories while satisfying safety and stability requirements. The same holds when the system was subjected to constraints concerning vehicle safety.

With the application of constraints which guarantee predecessor-follower stability, feasibility becomes an issue. These tight constraints are easily violated when a vehicle is not modelled very accurately. This model mismatch causes small initial position and velocity errors, which violate these tight constraints. As a result, there is no feasible solution within these constraints when initial errors occur and the controller does not maintain stability and safety.

Application of comfort constraints did show promise to better comfort, but once implemented solutions found to the optimization problem showed undesired behaviour. The applied input sequence by the controller caused a violation of constraints and degradation of convergence time. Further analysis showed that the utilized optimization algorithm does not perform as expected, and did not improve on initial found solutions.

In summary, vehicle platooning utilising MPC with the addition of predecessor trajectory information is implementable in real-time. However, the strict constraints required for the guarantee of predecessor-follower stability are hard to satisfy.

7-2 Future work

This section presents suggestions for future work on the subject of vehicle platooning. First, improvement of work presented in this thesis is suggested, this is then followed by possible extensions of platooning research.

7-2-1 Improvement of presented work

The application of MPC with the addition of predecessor trajectories shows promise to better platoon behaviour. However, tight constraints, optimization issues, and model mismatch are big concerns for implementation. Finding solutions to tight constraint problems was proven to be difficult, infeasible even for some combinations. Design of new constraints with potential slack or dedicated event handling could be an improvement for finding control actions satisfying constraints. The applied optimization algorithm performed below expectation and did not result in satisfying control inputs. Development or implementation of a more suitable optimization algorithm will benefit results.

7-2-2 Extension of platoon research

A suggested extension to platoon research is to improve on the simulation scenarios and to solve practicalities. Addition of drive train dynamics, vehicle cornering, sensor disturbance, and application onto a heterogeneous platoon all would increase simulation realism. Another suggestion is research concerning practical issues for real-world implementation. Actual hardware development, procedures for platoon creation/extension/dismemberment, failover procedures, and human acceptance are of interest.

Appendix A

Model identification

This appendix shows the graphs and method of the identification experiment. The goal of which is to identify the parameters of the vehicle model depicted in Equation (2-3).

The identification experiment was designed within the input range of $|u(t)| < 1000$ N $\forall(t)$, inputs outside this range resulted in large non-linearities. A sequence of inputs was set which would result in a velocity range of $(0, 40]$m/s, this is the range to be expected in normal behaviour. Standstill was excluded due to the change of modelling required at that instance. The identification was done using the Matlab identification toolbox on data gathered from the simulator. For fitting the model, a non-linear least square method was employed.

Table A-1 presents the identified parameters. The verification of the identified model to the data showed a match of 91.6%, measured as a normalized RMS error between the data of the simulator and a simulation of the identified model. The mass is low due to the mismatch between the mathematical model defining input as a direct force, whereas Dynacar applies the input as a direct torque to the wheels. This difference is scaled by a constant.

Table A-1: Non-linear model identified parameters

Par.	Value
m[kg]	165.8265
c_d[N/m/s]	0.0482

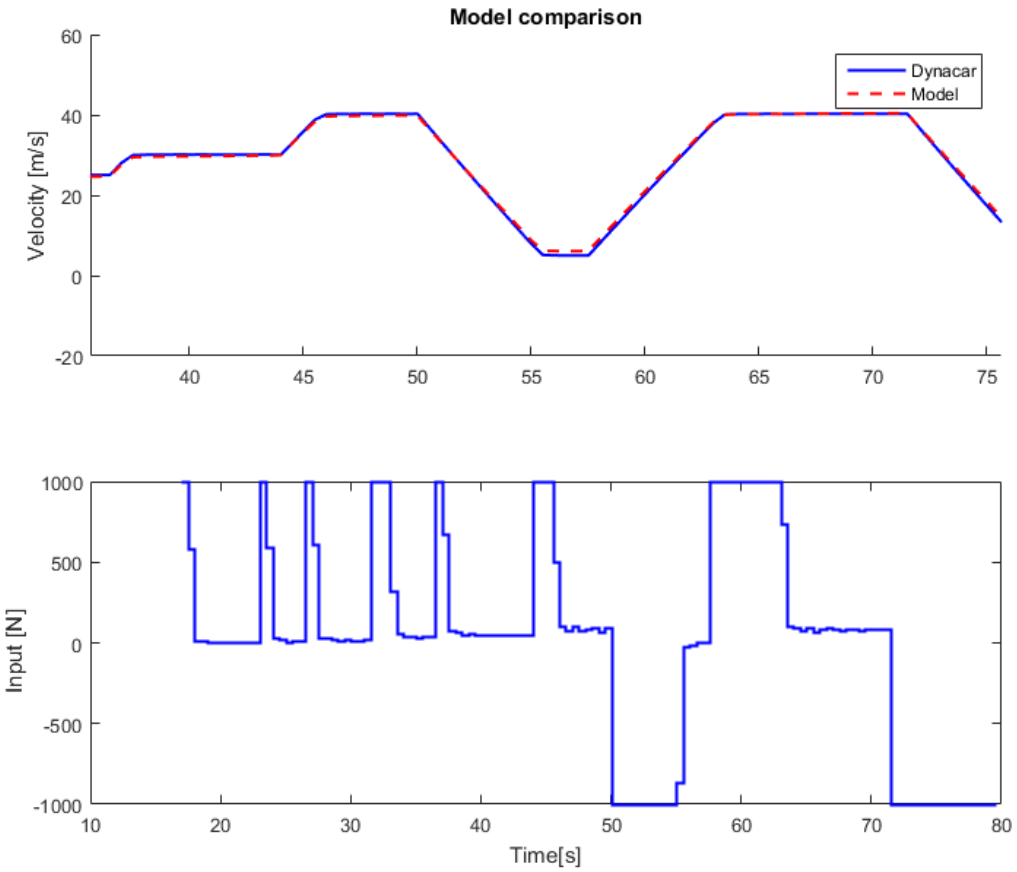

Figure A-1: Non-linear model identification experiment

Appendix B

Simulation results

B-1 Safety algorithm

Algorithm is applied as depicted in Section 3-2-1, with the objective weights as seen in Table 5-1.

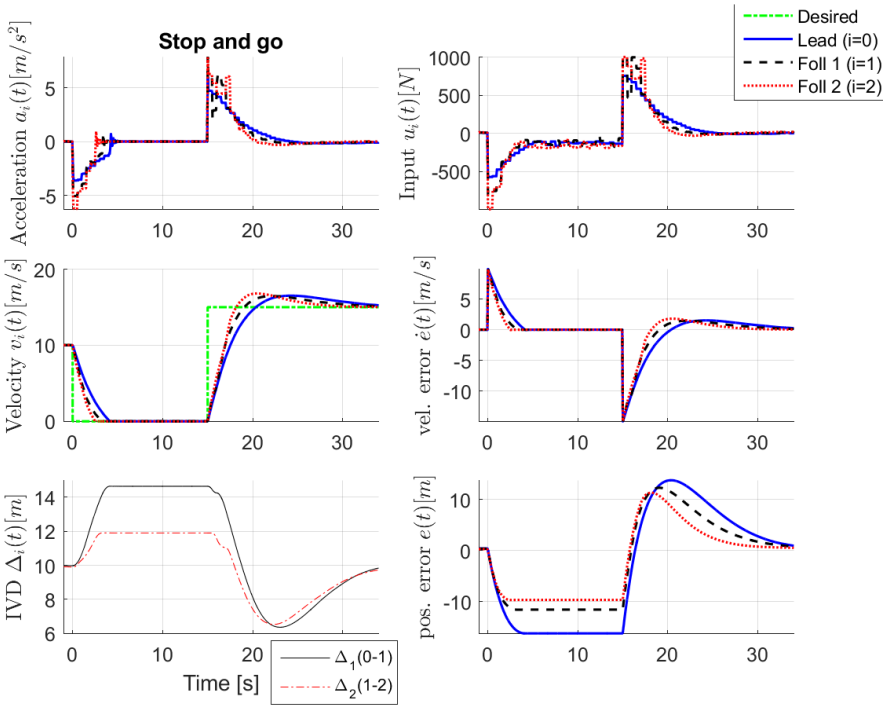

Figure B-1: Safety algorithm 'Stop and go'

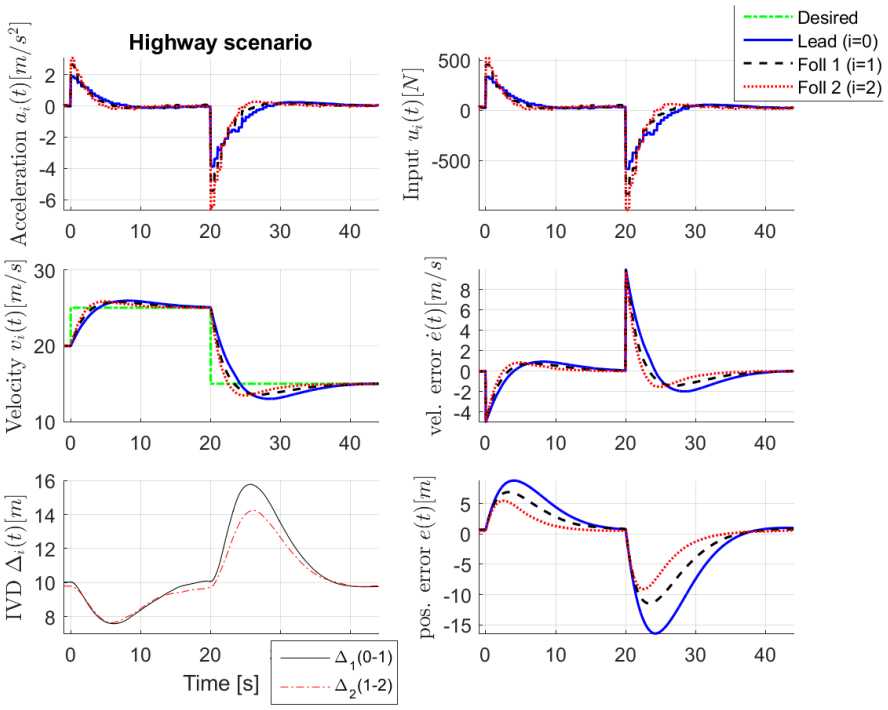

Figure B-2: Safety algorithm 'Highway scenario'

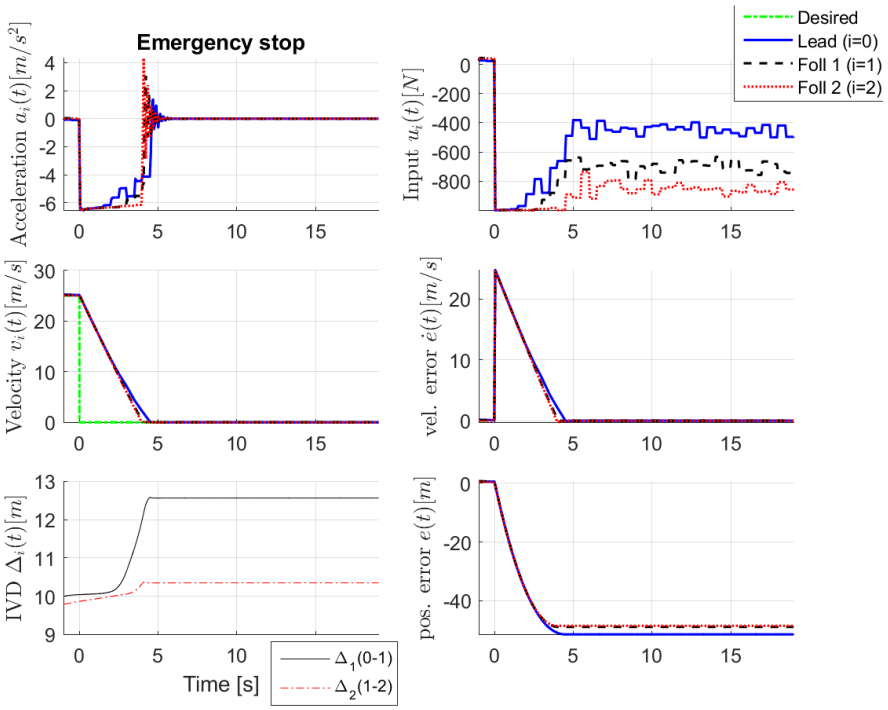

Figure B-3: Safety algorithm 'Emergency stop'

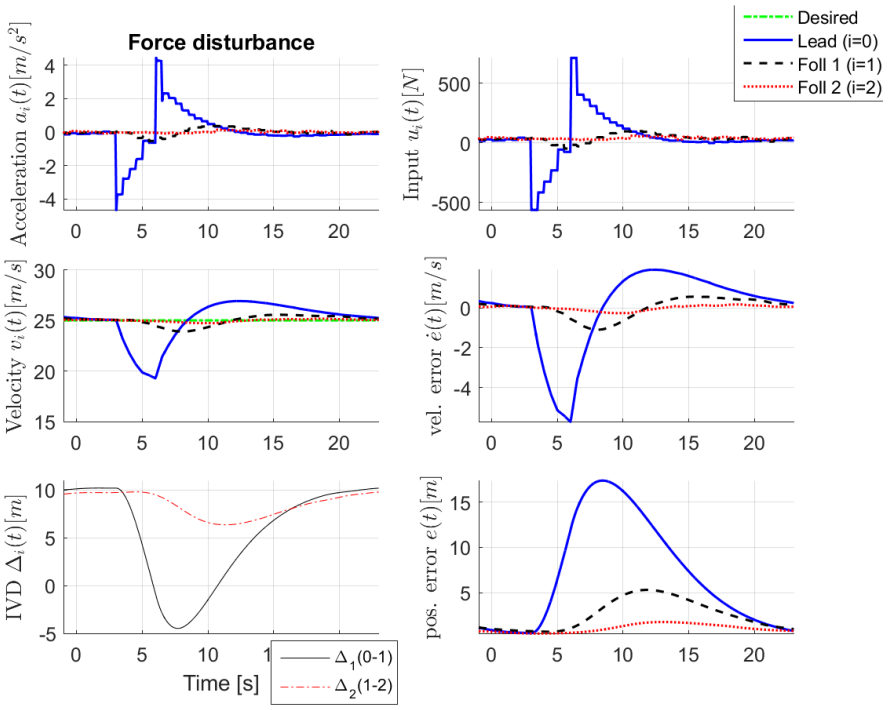

Figure B-4: Safety algorithm 'Force disturbance'

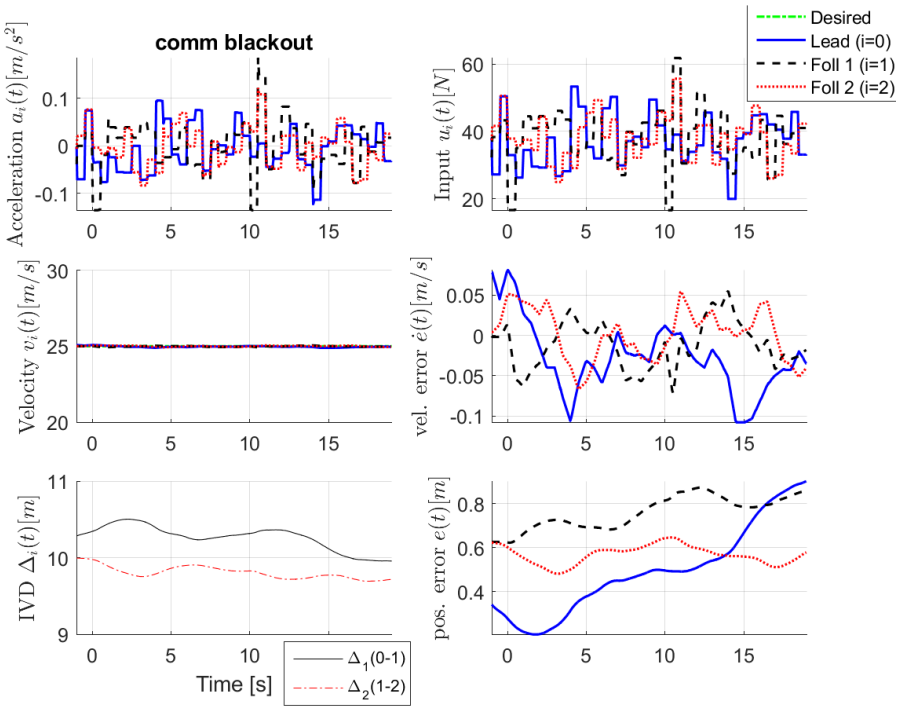

Figure B-5: Safety algorithm 'Communication blackout'

B-2 Platoon stability algorithm

This appendix shows the experiment results of the platoon stability algorithm, as written in Section 3-2-2. Table B-1 depicts the used constraint parameters, and for the objective function identical weights were used as for the safety algorithm as depicted in Table 5-1

Table B-1: Platoon stability constraint parameters

i	ξ_i	γ_i	β_i
0	0.8	0	1
1	0.1	0.1455	0.16
2	0.05	0.1047	0.84

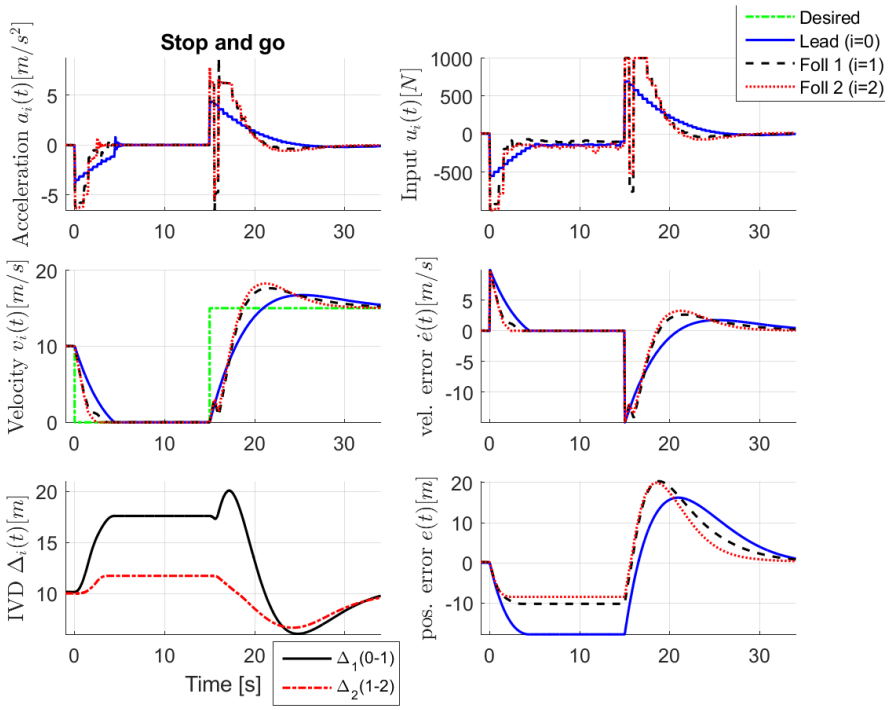

Figure B-6: Platoon stability algorithm 'Stop and go'

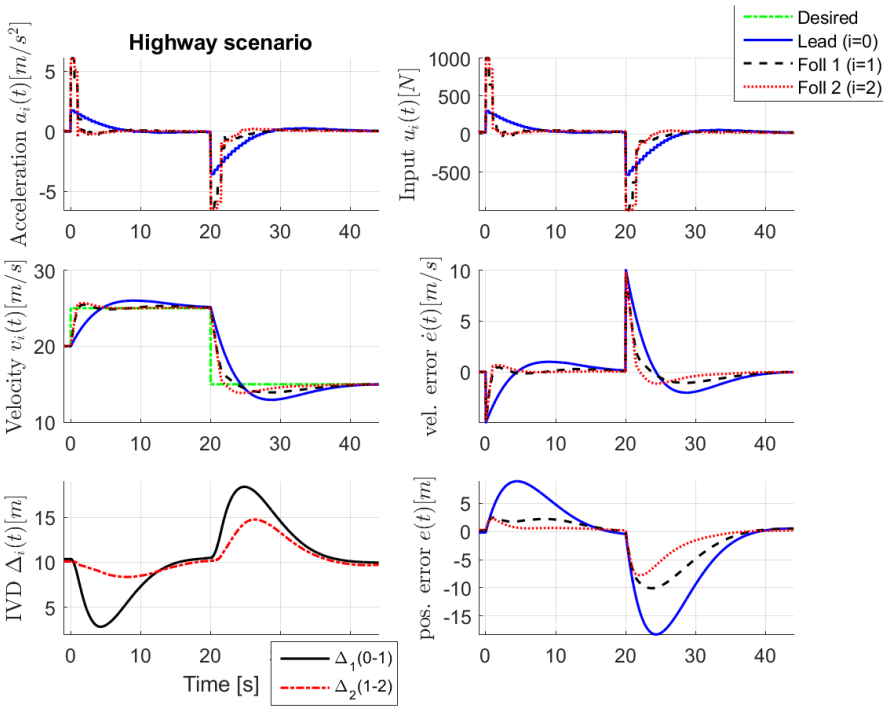

Figure B-7: Platoon stability algorithm 'Highway scenario'

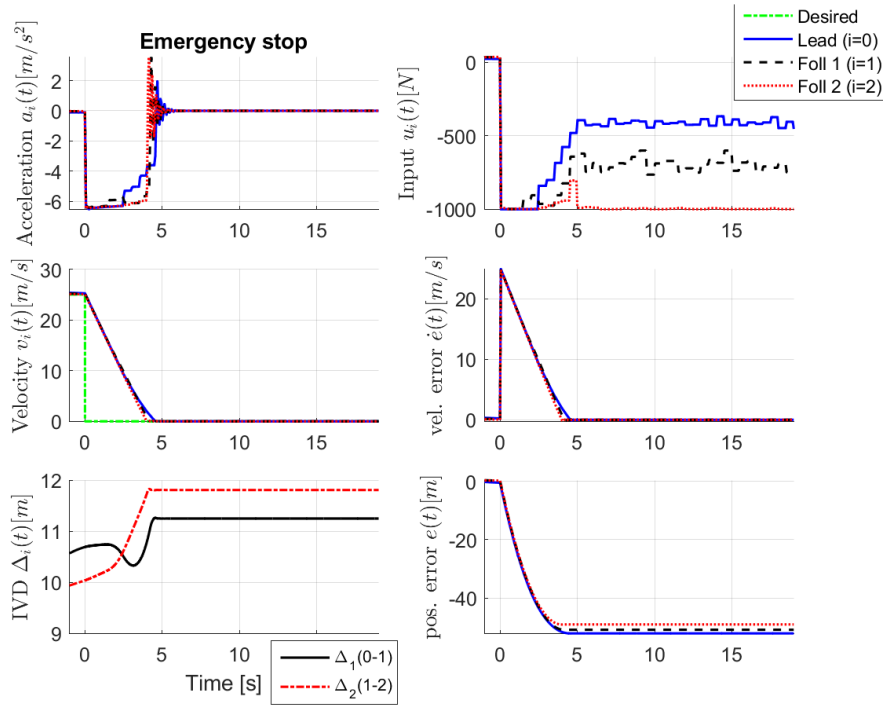

Figure B-8: Platoon stability algorithm 'Emergency stop'

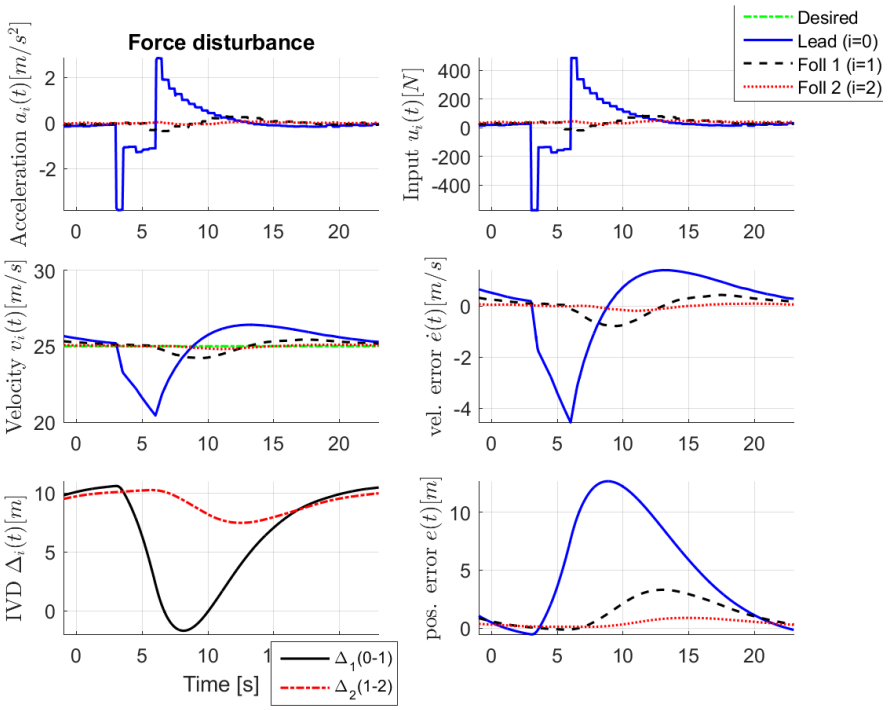

Figure B-9: Platoon stability algorithm 'Force disturbance'

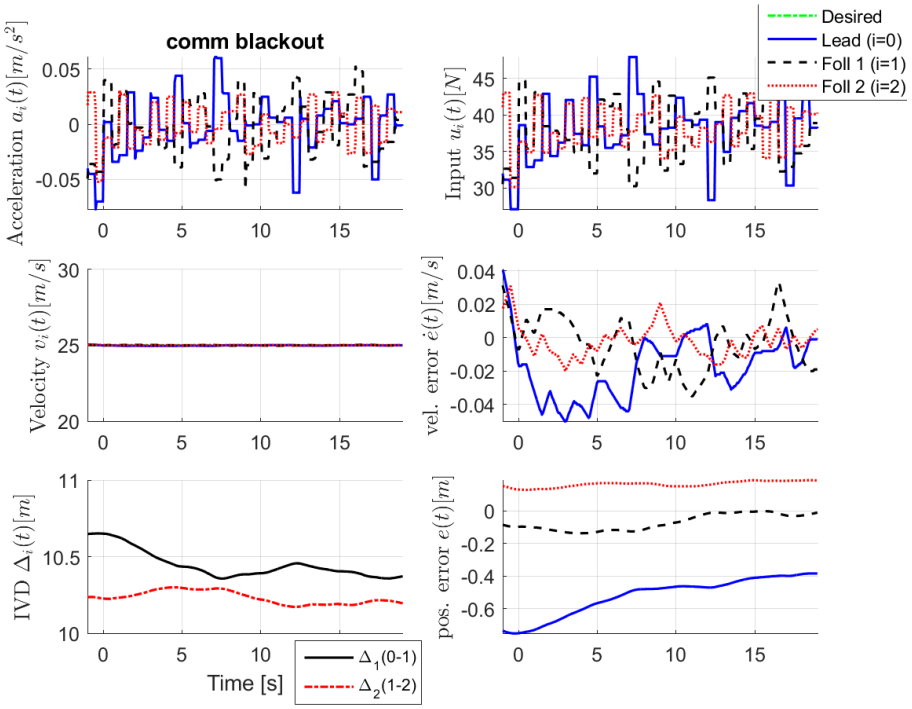

Figure B-10: Platoon stability algorithm 'Communication blackout'

B-3 Platoon comfort algorithm

This appendix present the results from the comfort algorithm as depicted in Section 3-2-3. Identical objective function weights were used as for the safety algorithm as depicted in Table 5-1.

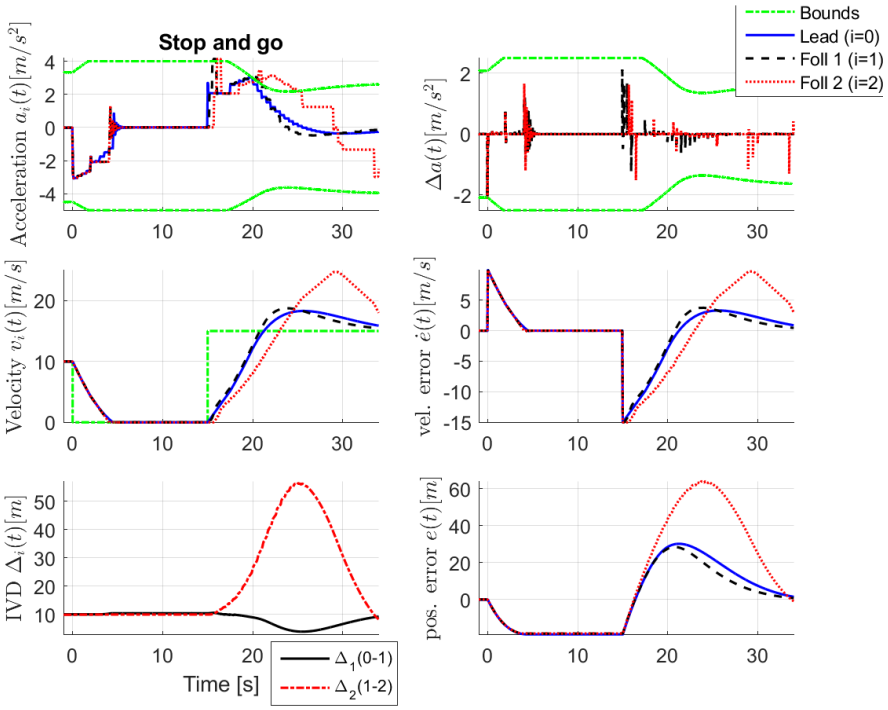

Figure B-11: Comfort algorithm 'Stop and go'

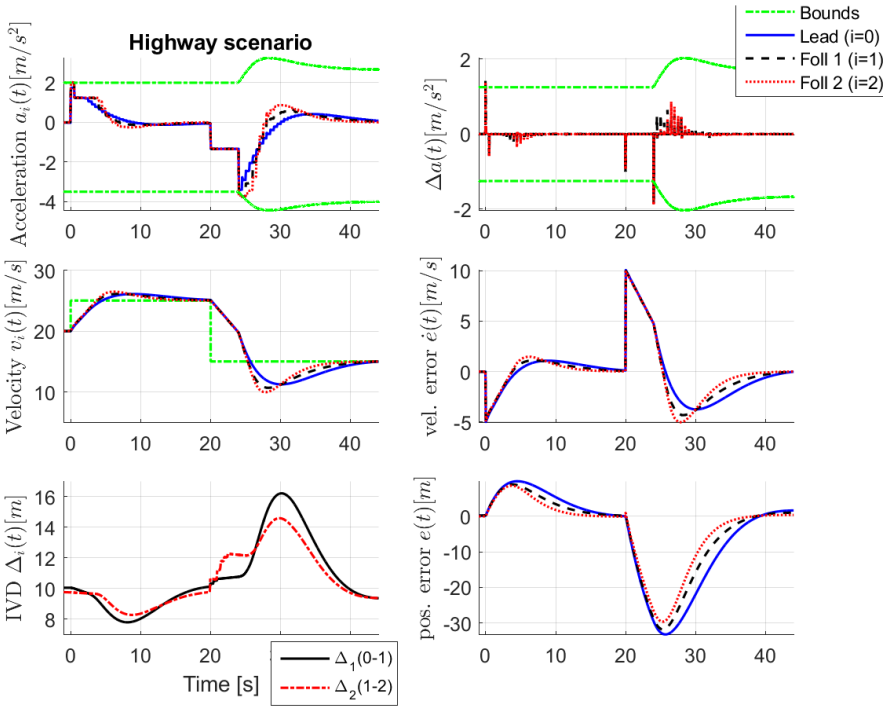

Figure B-12: Comfort algorithm 'Highway scenario'

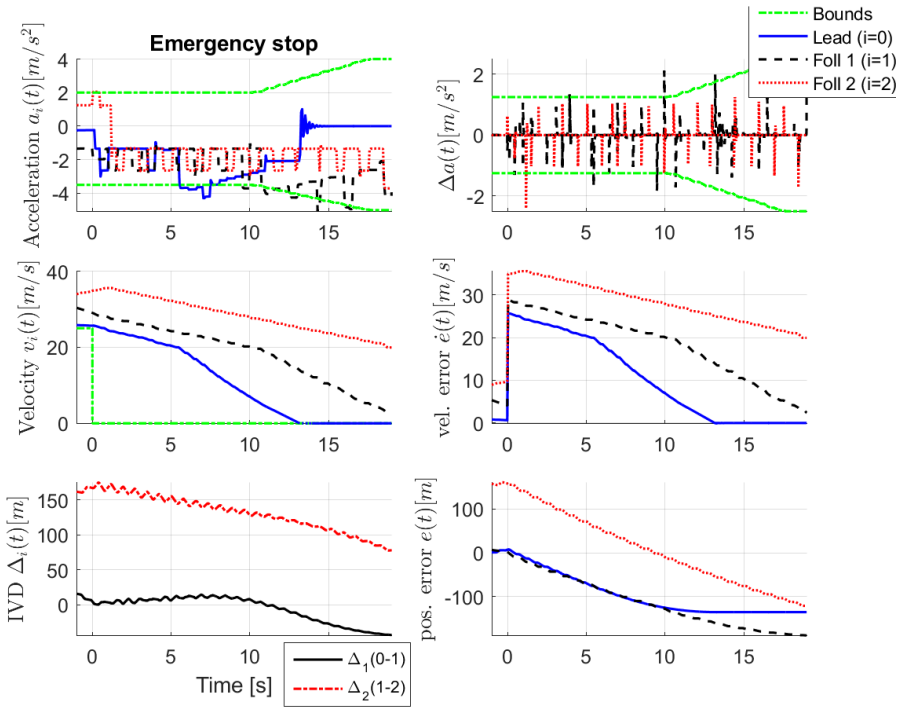

Figure B-13: Comfort algorithm 'Emergency stop'

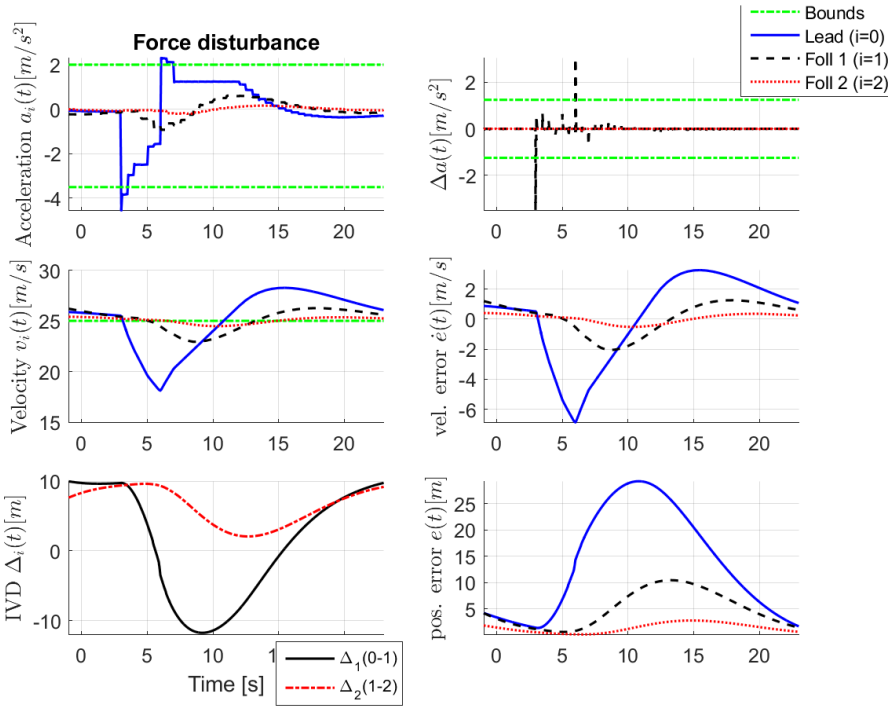

Figure B-14: Comfort algorithm 'Force disturbance'

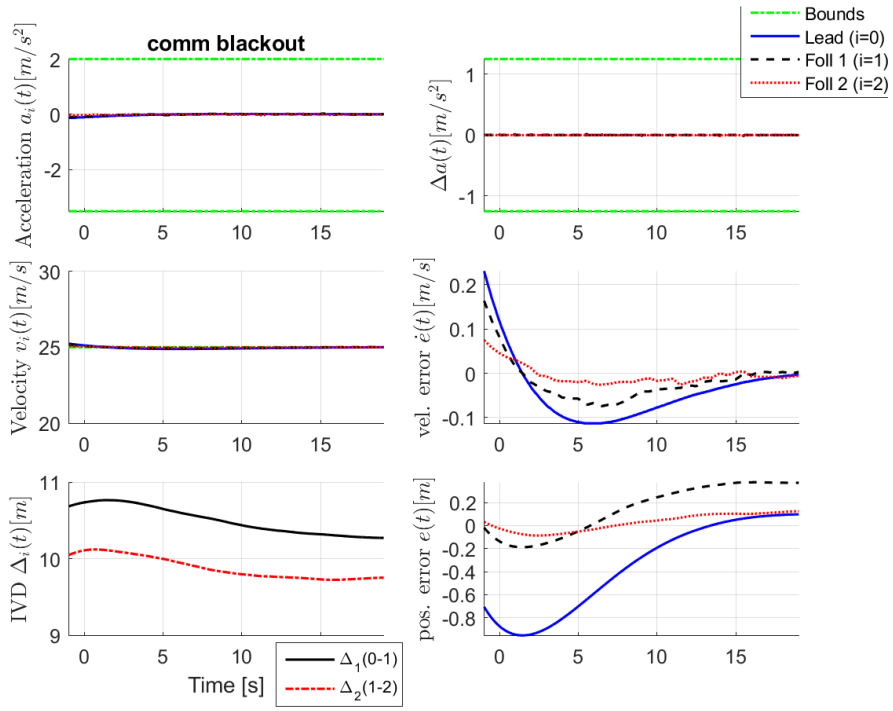

Figure B-15: Comfort algorithm 'Communication blackout'

Bibliography

[1] CBS, "Motorvoertuigenpark; inwoners, type, regio, 1 januari. and,verkeersprestaties motorvoertuigen; kilometers, voertuigsoort, grondgebied," 2017. http://statline.cbs.nl; accessed 12-October-2017;dutch source.

[2] J. G. Bender and R. E. Fenton, "A study of automatic car following," *Ieee Transactions on Vehicular Technology*, vol. Vt18, no. 3, p. 134, 1969.

[3] S. E. Shladover, "Review of the state of development of advanced vehicle control-systems (avcs)," *Vehicle System Dynamics*, vol. 24, no. 6-7, pp. 551–595, 1995.

[4] L. Ws and M. Athans, "On the optimal error regulation of a string of moving vehicles," vol. 11, pp. 355–361, 1966.

[5] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, W. B. Zhang, D. H. Mcmahon, P. Huei, S. Sheikholeslam, and N. Mckeown, "Automatic vehicle control developments in the path program," *Ieee Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 114–130, 1991.

[6] I. Catling and B. Mcqueen, "Road transport informatics in europe - major programs and demonstrations," *Ieee Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 132–140, 1991.

[7] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *Ieee Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 143–153, 2003.

[8] GCDC, "Grand cooperative driving challenge," 2017. http://www.gcdc.net/nl/; accessed 12-February-2018.

[9] Connekt, "Dutch automotive vehicle initiative," 2017. http://davi.connekt.nl/; accessed 30-January-2018.

[10] G. J. L. Naus, R. P. A. Vugts, J. Ploeg, M. J. G. van de Molengraft, and M. Steinbuch, "String-stable cacc design and experimental validation: A frequency-domain approach," *Ieee Transactions on Vehicular Technology*, vol. 59, no. 9, pp. 4268–4279, 2010.

[11] L. E. Peppard, "String stability of relative-motion pid vehicle control-systems," *Ieee Transactions on Automatic Control*, vol. Ac19, no. 5, pp. 579–581, 1974.

[12] T. Stanger, L. del Re, and Ieee, "A model predictive cooperative adaptive cruise control approach," *2013 American Control Conference*, pp. 1374–1379, 2013.

[13] S. E. Shladover, D. Y. Su, and X. Y. Lu, "Impacts of cooperative adaptive cruise control on freeway traffic flow," *Transportation Research Record*, no. 2324, pp. 63–70, 2012.

[14] V. Milanes, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative adaptive cruise control in real traffic situations," *Ieee Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 296–305, 2014.

[15] J. Ploeg, B. T. M. Scheepers, E. van Nunen, N. van de Wouw, H. Nijmeijer, and Ieee, "Design and experimental evaluation of cooperative adaptive cruise control," *2011 14th International Ieee Conference on Intelligent Transportation Systems*, pp. 260–265, 2011.

[16] 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*, vol. 7, no. 4, pp. 429–436, 2006.

[17] Freepik, "Assortment of fantastic car silhouettes," 2017. https://www.freepik.com/ ;accessed 12-October-2017;used as base for diagram.

[18] X. H. Liu, A. Goldsmith, S. S. Mahal, J. K. Hedrick, Ieee, and Ieee, "Effects of communication delay on string stability in vehicle platoons," *2001 Ieee Intelligent Transportation Systems - Proceedings*, pp. 625–630, 2001.

[19] W. B. Dunbar and D. S. Caveney, "Distributed receding horizon control of vehicle platoons: Stability and string stability," *Ieee Transactions on Automatic Control*, vol. 57, no. 3, pp. 620–633, 2012.

[20] W. B. Dunbar and R. M. Murray, "Distributed receding horizon control for multi-vehicle formation stabilization," *Automatica*, vol. 42, no. 4, pp. 549–558, 2006.

[21] R. Rajamani, *Vehicle Dynamics and Control, Second Edition*, pp. 1–496. Mechanical Engineering Series, New York: Springer, 2012.

[22] ISO, "Iso 22179:2009 intelligent transport systems," 2009.

[23] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[24] Tecnalia, "Dynacar," 2019. http://www.dynacar.es/en/home.php; 30-January-2018.

[25] Unity, "Unity 3d gaming engine," 2019. https://unity3d.com/; 30-January-2018.

[26] N. Instruments, "Pxie-1071," 2018. http://www.ni.com/nl-nl/support/model.pxie-1071.html ;accessed 13-December-2018.

[27] M. J. D. Powell, "A direct search optimization method that models the objective and constraint functions by lineat interpolation," *Advances in Optimization and Numerical Analysis*, vol. 275, pp. 51–67, 1993.

[28] S. G. Johnson, "The nlopt nonlinear-optimization package," 2014. http://ab-initio.mit.edu/nlopt ;accessed 29-october-2018.

[29] IEEE, "Ieee standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, 2008.

[30] P. Krzesinski, *Decentralized CACC controllers for platoons of heterogeneous vehicles with uncertain dynamics*. Master of science, 2017.

[31] F. A. Mullakkal-Babu, M. Wang, B. v. Arem, and R. Happee, "Design and analysis of full range adaptive cruise control with integrated collision a voidance strategy," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 308–315.

Glossary

Acronyms

CC	Cruise Control
ACC	Adaptive Cruise Control
CACC	Cooperative Adaptive Cruise Control
V2V	Vehicle to Vehicle
MPC	Model Predictive Control
RHC	Receding Horizon Control
COBYLA	Constrained Optimization BY Linear Approximations, optimization algorithm by M. J. D. Powell

Nomenclature

Greek symbols

β_i, γ_i, ξ_i	Constraint parameters	$[-]$
δ	Sample time	$[s]$
$\Delta_i(t)$	Inter-vehicle distance	$[m]$
$\Delta_{d,i}$	Desired inter-vehicle distance	$[m]$
$\epsilon_{i,k}$	Update dependent constraint parameters	$[-]$

Roman symbols

$\mathbf{Q}_i, \mathbf{F}_i, \mathbf{G}_i, R_i$	Weighting matrices	$[-]$
$a_i(t)$	Vehicle acceleration	$[m/s^2]$
$a_{min}(v), a_{max}(v)$	Acceleration limits	$[m/s^2]$
$c_{d,i}$	Vehicle drag coefficient	$[N/m/s]$
d_{sep}	Desired seperation	$[m]$
$da(t)$	Acceleration change	$[m/s^2]$
$e_i(t)$	Position error	$[m]$
$F_{drag,i}(t)$	Drag force	$[N]$
$J_{i,t_k}(\cdot)$	Objective function	$[-]$
$j_{min}(v), j_{max}(v)$	Jerk limits	$[m/s^3]$
L_i	Vehicle length	$[m]$
m_i	Vehicle mass	$[kg]$
N	Sample horizon	$[-]$
T	Horizon time	$[s]$
t_0	Initialization time	$[s]$
t_k	Sample time	$[s]$
$u_i(t)$	Vehicle input	$[N]$
$v_i(t)$	Vehicle velocity	$[m/s]$
$v_{d,i}(t)$	Desired vehicle velocity	$[m/s]$
v_{min}, v_{max}	Velocity limits	$[m/s]$
$x_i(t)$	Vehicle absolute position	$[m]$
$x_{d,i}(t)$	Desired position	$[m]$
$\mathbf{z}_i(t)$	Error vector	$[m, m/s]^\top$

Subscripts

i	Vehicle index	$[-]$
i_{end}	Last vehicle in platoon	$[-]$
k	Sample count	$[-]$
t_{plan}	Future time of new v_d	$[s]$

Superscripts

$\hat{}$	Planned trajectory	[–]
^{p}	Predicted trajectory	[–]

Sets

$\mathbb{R}$	Real set	[–]
$\mathbb{Z}$	Integer set	[–]