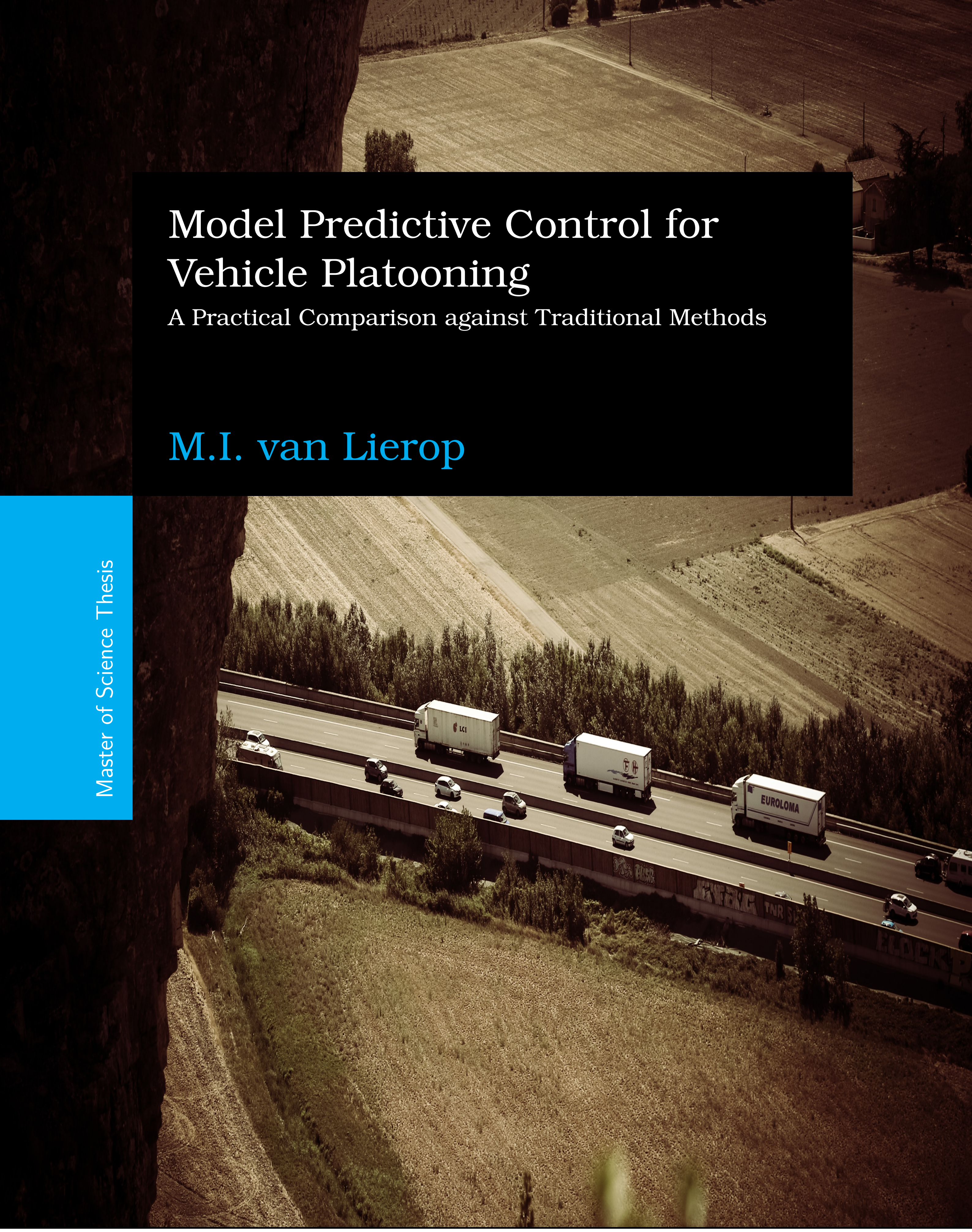


Model Predictive Control for Vehicle Platooning

A Practical Comparison against Traditional Methods

M.I. van Lierop

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Model Predictive Control for Vehicle Platooning

A Practical Comparison against Traditional Methods

Master of Science Thesis

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

M.I. van Lierop

April 13, 2022

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

Model Predictive Control for Vehicle Platooning

by

M.I. van Lierop

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: April 13, 2022

Supervisor(s): ______________________
Dr. R. Ferrari

____
Ir. T. Keijzer

Reader(s): ______________________
Dr. A. Dabiri

Abstract

Due to the increase in traffic, road congestion has gone up. Vehicle platooning is a possible way to increase the capacity of a given road, by decreasing the distance between the vehicles in the platoon. At the moment, the control of vehicle platoons is commonly done using PDD controllers. The advantage of this is that it requires little computational resources. With improvements in computing technology in recent years, the possibility of using a more computationally costly method has opened up. Model Predictive Control (MPC) is such a method, with a couple of important advantages for vehicle platooning. Firstly it is (finite-time) optimal so should lead to smaller deviations from the desired trajectory, thereby decreasing the minimum safe distance between the vehicles. Secondly, it is inherently able to include constraints on the behaviour of the platoon. This is important because it allows for explicitly enforcing the minimum safe distance between the vehicles, thereby preventing collisions.
While many studies have looked into the possibility of using MPC for vehicle platooning before. None of them have made the direct comparison with the current standard e.i. Proportional Double Derivative (PDD) control in a practical setup. This is important as it will give a good indication of both the possible advantages and pitfalls MPC could have in real platoons. To investigate this both a PDD and MPC controller are implemented on a platoon of Radio Controlled (RC)-vehicles and their performance is compared.
To compare the performance five metrics are used on four scenarios. The metrics used are coherence, combined local error, velocity error, energy expenditure and minimum headway. The scenarios are a step up, step down, ramp up and a ramp down.

From the simulation results, it is clear that in an idealised case the MPC outperforms the PDD controller. However, when looking at the lab results this difference is less pronounced. In the lab setting, it is also clear that both controllers struggle to reliably deal with the added uncertainties caused by measurement noise and inconsistent behaviour of the vehicles.
In future research, more emphasis should be placed on the robustness of the controllers used, to mitigate these problems.

Table of Contents

List of Figures

List of Tables

Preface

This document is the culmination of my studies in Systems & Control at Delft University of Technology. The subject of platooning piqued my curiosity because I have always had an interest in different modes of transport. Furthermore, the opportunity to work on a practical application seemed like a fun challenge to bring the theoretical knowledge acquired during my studies to life.

I would like to thank my supervisors Riccardo Ferrari and Twan Keijzer for their guidance and feedback in conducting this thesis. Finally, I would like to thank my friends and family and in particular my parents for their unwavering support in the good times and the bad.

Delft, University of Technology
April 13, 2022

M.I. van Lierop

“He who cannot be a good follower cannot be a good leader.”
— *Aristotle*

Chapter 1

Introduction

In recent decades car use has increased drastically. This has in turn increased road congestion, which causes multiple problems including increased travel times, fuel consumption and car-related accidents. Research has shown that adding or widening roads induces an increase in vehicle kilometres travelled. Therefore it does not decrease road congestion in the long run. [2] So to solve this issue the current roads need to be used more efficiently.
The capacity of a road can be increased by reducing the inter-vehicular distance. In human-driven vehicles, the safe inter-vehicular distance is determined by two factors; the vehicle's braking distance and the driver's reaction time. The first could potentially be reduced by improvements in braking technology, however, the reaction time will always be a limiting factor of human drivers. Automated systems, however, have significantly lower reaction times, so they can realise smaller inter-vehicular distances without compromising on safety. Many cars currently on the market are already fitted with Adaptive Cruise Control (ACC), which uses sensor data to control the inter-vehicular distance. This can lead to increased road capacity of up to 43%. However, even greater increases up to 273% can be reached using Cooperative Adaptive Cruise Control (CACC), which encompasses Vehicle-to-Vehicle Communication (V2V) and cooperative algorithms that take into account the behaviour of the surrounding vehicles. [3, 4]
The increase in car use is also detrimental to the climate. This is compounded by the extra emissions caused by congestion. Apart from fuel savings due to decreased congestion, shorter inter-vehicular distances can also impact a vehicle's fuel consumption directly. The amount of aerodynamic drag a vehicle has to overcome, and thereby the quantity of fuel used, is smaller if it drives in the wake of its preceding vehicle. This is especially significant for trucks, which, due to their larger frontal area, have to use more of their power to overcome aerodynamic drag. Platooning with small inter-vehicular distances can reduce this fuel consumption by up to 10%, and these small distances can only safely be achieved using autonomous driving systems. [5]

1-1 Communication in Networked Control Systems

As mentioned, the most significant improvements in road congestion and fuel savings can only be made using platooning algorithms that are aware of the actions of the surrounding vehicles. This means communication is required between controllers, sensors and actuators that are spatially distant, leading to a Networked Control System (NCS). This type of communication can not always be assumed to be perfect and instantaneous, as communication delay and packet loss can destabilise otherwise stable controllers. [6–8].
Depending on the architecture of the NCS there are two places where these communication faults may happen, either in the communication between the sensor(s) and the controller(s) or between the controller(s) and the actuator(s). Several regional standardisation organisations [9, 10] have developed standards for communication in Intelligent Transportation Systems (ITS), separating the communication protocols based on the various use cases. Depending on the location of the controller with respect to the vehicles, the Dedicated Short Range Communication (DSRC) needed for a platoon control system can be classified as Vehicle-to-Infrastructure Communication (V2I), Vehicle-to-Field Communication (V2F) or V2V.

The NCS architecture describes how subsystems and their sensors, actuators and controllers are connected. In general, this can be divided into two main types; centralised control, where one controller communicates with all sensors and actuators in the system; and distributed control, where the separate subsystems have their own controllers.
The main advantage of a centralised control system is that all sensor information can be used to compute the control inputs for all subsystems. When using an optimal control method, like Model Predictive Control (MPC), this means that global optimality can be achieved [11]. However, since the controller is physically separated from (most of) the sensors and actuators, there is a great reliance on the communication network. This introduces two main issues. Firstly the amount of information transmitted is directly proportional to the number of vehicles in the platoon. This means the size of the platoon could be limited by the communication system or by the computational capacity of the central controller [6, 12]. Secondly, the vehicles are completely dependent on the centralised controller for their inputs, any delays in communication must be accounted for in the design of the controller as they can severely impact the reaction time of the platoon. Furthermore, package loss could even leave a vehicle without any input, causing dangerous situations.

Instead of a centralised controller, it is also possible to have multiple controllers for the same platoon. This is often realised by having a controller in each vehicle in the platoon controlling just that vehicle. This is similar to the already on market ACC, but can be extended with V2V to CACC, achieving better performance [3]. Different communication topologies can be used to determine the level of interconnection within a distributed platoon. This can span every option from all-to-all communication, where each controller has the same amount of information as a centralised controller, to predecessor-following, where each vehicle only receives information from the vehicle immediately in front of it [13].
Since the controller is located in the vehicle itself, the platoon is not dependent on specialised infrastructure or lead vehicles, making it more versatile than centralised control. Furthermore, the control signal doesn't travel over the communication network. Assuming the vehicle also has on-board sensors, only part of the input is subject to possible delay and package loss. [14] However, the lack of centrality of control also means that each controller can only act on the information it can gather with the vehicle's own sensors or is communicated by the

surrounding vehicles, leading to sub-optimal performance [15]. It is also desirable to be able to run the control algorithm on the vehicles' existing hardware, which poses limits on the computational power available. Another issue with distributed control is that for each controller to have access to all information the load on the full communication network scales with N^2 where N is the number of vehicles in the platoon. Therefore, vehicles in distributed platoons usually only communicate with a limited number of surrounding vehicles, thereby decreasing the stability margin of the platoon. [16]

A reality of all (digital) communication protocols is the presence of communication delays, that is the non-zero time between the sending and the receiving of a signal. Also sometimes the signal is not received at all or delayed to the point that it is no longer relevant, also known as packet loss or drop-out. Both these phenomena can be detrimental to the stability and performance of an NCS [17].
Several projects have been conducted in increasing the reliability of DSRC, approaching the problem from the network, data link and physical components [18]. However, from the perspective of controller design, there is little influence on the communication faults. So the problem is combatted by accounting for the inevitability of these faults in the design of the controller.
Since controllers use the current state or output to generate the control inputs for the next time step, the sensor information they receive must be accurate. One method to account for delays in Sensor-to-Controller Communication (S/C) is by adapting the observer. Controllers often work with observers to estimate the states of the system from the outputs. A similar approach can be used to estimate the current state from the last received output message. When the S/C delay is constant, it can be included directly in the estimator equation [19]. Otherwise, it is valuable to time stamp the sensor messages so the delay can be determined once the message arrives at the estimator [7, 20, 21] Another way to model bounded delay is as a Markov chain resulting in a jump linear system. Since the current S/C delay is known by the controller, it is possible to design a set of feedback matrices that switch based on the (discretised) delay [21]. The same approach can also be used in the case of packet loss [22,23].

The key difference between Controller-to-Actuator Communication (C/A) and S/C delay is that the C/A delay is not known by the controller when the control inputs are computed. Therefore, different strategies must be used to deal with its effects. One such strategy is to combat the adverse effects of package loss by increasing the desired headway accordingly [48]. Other methods analyse the stability of the controller to find a bound on the tolerable delay or loss rate. [44] does this by applying optimal Linear-Quadratic- Gaussian (LQG) control, while differentiating between communication protocols with and without acknowledgement of package arrival. In the former case the optimal LQG controller turns out to be linear and an analytical bound can be found on the loss rate to guarantee stability. In the case without acknowledgement of package arrival, the controller is generally non-linear. Another way to analyse NCS stability is to treat them as hybrid systems where the delays are modelled as discreet events acting on a continuous system [30]. However non of these methods actively combat the effects of the C/A delay. While the C/A delay is not known a priori by the controller, it can still be modelled as a Markov chain. This uses the fact that, when always using the most recent information, the delay at time k is related to the delay at k - 1 (which can be measured and sent back to the controller), because it can never have increased by more than one time step. Therefore, the feedback gain can be switched based on the delay at k - 1 [45]. By using the Markov chain model, it is also possible to design an MPC controller that minimises the cost for the upper bound of the worst-case scenario [49]. This is a similar approach to Min-Max Model Predictive Control (MM-MPC) where the inputs are found by minimising the cost of the worst-case taken over multiple models [50]. Another way to combat faults in the communication between the controller and the vehicle actuator, is to introduce a buffer system. The goal of such a buffer is to select and implement a control input that matches the current time step from a sequence of future inputs. This buffer approach works particularly well in conjunction with MPC, because MPC always produces a sequence of future inputs [19, 24–26]. However, it is also possible to obtain such a control input sequence by combining a state estimator with other control methods [27]. Naturally, sending the input sequence for the full prediction horizon for each vehicle can put excessive strain on the communication network. Therefore, [24] uses the same buffer strategy, but only sends a variable horizon based on the delay as measured at the last time step. Thereby it minimises the necessary communication.

1-2 Model Predictive Control

A plethora of controller types have been studied for longitudinal control for (semi)autonomous vehicles, each with its advantages and disadvantages.
The most common, and one of the least computationally intensive controllers to implement is one from the Proportional Integral Derivative (PID)/family (e.i. PID, PD, PDD, etc.). Since the error signal used by the controller is usually the distance error, (double) derivative action is used most commonly, thereby reacting to deviations in velocity and acceleration between the vehicles. [28] and [29] show that it is possible to design a PD controller that is both string and asymptotically stable. However the two major drawbacks of PID control are that it is not an optimal control strategy and that it cannot handle constraints, so adherence to input constraints and output constraints (e.g. a minimal safety distance) can not be guaranteed. To combat the second issue, [30] combines a PID controller with a reference governor that modifies the reference signal received from the preceding vehicle if it leads to a possible violation of the output constraint.

On the contrary, MPC does allow for constraints and it uses dynamic programming methods to find the optimal control sequences within these constraints, leading to safer platoons [31]. Due to the improvements in computational power, MPC has recently seen increased interest. In the field of platooning many studies have looked into the potential pitfalls of MPC and their solutions [25, 32–34].

When using a quadratic cost function and its minimum is contained in the constraints sets, MPC can be proven to be asymptotically stable under nominal conditions if appropriate weights are chosen [11]. However, for real-life applications, nominal conditions can not always be guaranteed. Therefore research has been done into variations on traditional MPC to combat specific limitations.
For distributed control architectures it can prove difficult to guarantee string stability. Therefore, [35] introduces string stability into distributed MPC through the use of constraints. This method does require the controller to know the intended trajectory of the preceding vehicle, which requires extensive V2V communication in platoons with distributed controllers. To get around this [36] uses a controller matching approach to tune the weighting matrices of the

MPC such that its behaviour matches a string stable linear controller when the constraints are inactive.
Also, it has been shown that L-infinity Robust MPC can guarantee that the inter-vehicular distance is kept above a (dynamic) minimum safety distance, for a two car platoon under both modelled and unmodelled disturbances [37].
Furthermore, it is shown that stability can be guaranteed for a linear NCS when using MPC in conjunction with a state-estimator and the buffer algorithm discussed before [19, 24, 26].

Using MPC, the controller needs to run an optimisation algorithm at every time step, which can require a lot of computational power. This can lead to problems if the controller cannot compute the control inputs within the time step. There are several methods of reducing the complexity and therefore the run time of the MPC algorithm.
The chosen vehicle model and constraints have a significant impact on the type of optimisation. However, to reduce computational power, the most important factor is the type of optimisation problem that needs solving. That is whether the problem is linear, quadratic, convex or non-convex. Since the vehicle dynamics are implemented as equality constraints in the MPC framework, the resulting problem can be cast into a convex (or simpler) form if the model is affine [38]. Of course, the main risk here is to oversimplify the model, leading to inaccurate predictions or even instability.
Regardless of the complexity of the model, the complexity of the MPC algorithm can be tuned by changing the prediction horizon or the update frequency. A smaller update frequency allows for more time to solve the optimisation problem, however, it also leads to an increased reaction time. [39] shows that a safe update frequency is related to both the delay in the system and the chosen inter-vehicular distance.
The horizon length is the main determinant in the number of optimisation variables, therefore it has a significant influence on computation times. However, while shorter horizons are more favourable for computation, longer horizons lead to better stability [40].
The most common type of MPC, and the type described here so far, is known as implicit MPC. Implicit MPC solves the optimisation problem on-line at each time step to find the optimal control inputs. Its counterpart is explicit MPC where the optimisation is solved off-line and translated into a (usually) Piecewise Affine (PWA) function of the state. The gains from these segments are then stored in a lookup table [41]. This method is used by [42] for a fuel optimisation problem.

1-3 Research Question

With advancements in embedded technology, both MPC and vehicle platooning have seen increased interest over recent years. So it is only logical that this intersection has also been studied widely. Many studies have been done into how different versions of MPC can be used to create safe and robust vehicle platoons. However, the results of these studies are mostly limited to theory or simulation.

Others have compared an MPC controller with a Proportional Double Derivative (PDD) controller using simulations [43] or in practical tests but limited to ACC [5]. However currently no studies have been done that directly compare an MPC to a PDD controller for a CACC platoon using practical experiments.
It is important to have a quantitative comparison between both methods, because it gives

an objective and reliable view on the potential of MPC to possibility replace PDD as the standard in vehicle control.
While simulation can be a powerful tool for exploring and testing new ideas, this comparison should include practical testing because simulations can never give a full picture of the real-life situation. To make a simulation it is always necessary to make assumptions or generalisations that are not strictly present in the physical world. Therefore simulation results are only an approximation of the real results, often neglecting or simplifying factors like measurement noise and model uncertainties. These discrepancies can have significant effects on the performance of a controller, in the worst case destabilising a system that was stable in simulation. In the case of MPC it is especially important that the controller is tested in a more complex setting than the model used by the controller, because if the model used for testing is the same as the model used by the controller, the predictions are always identical to the outcome, which does not result in a realistic test.

Therefore this study will try to answer the following question:

> How does Model Predictive Control (MPC) perform compared to a conventional (PDD) controller, when used to control a platoon of Radio Controlled (RC) vehicles?

This research question can be split into the following sub-questions:

1. Can MPC be used to control a platoon of RC vehicles?
2. How does this MPC controller perform in simulation?
3. How does the performance in the experimental setup differ from that in simulation?

To answer these questions at their most fundamental level a fairly standard centralised version of MPC is used. This is chosen to provide a benchmark for MPC as a full category of controller and to bring to light the inherent strengths and weaknesses of MPC. This also provides a useful starting point for further research into the different possible variations within MPC. Similarly, RC vehicles are used for this research because they provide an accessible test platform in terms of availability and costs, while still possessing most of the same dynamic properties of full scale vehicles.

Both the simulation results and the MPC heavily rely on models of the experimental setup. Since no complete model exists yet of the setup used, a significant part of this research shall be devoted to acquiring these models.

This report will start with chapter 2 which explains the dynamics of the platoon and the vehicles within it as this is important background to understand before looking at how to control a platoon. This is followed by an explanation of the methodology of this research, consisting of the experimental setup and the metrics used to evaluate the performance of the controllers, in chapter 3. Next, the method and results of the system identification process are described in chapter 4. Now knowing the system, the design considerations of both control methods are explored in chapter 5. This is followed by chapter 6, with the results of both the simulation and lab experiments. Finally, the conclusions of this study and recommendations for further investigation can be found in chapter 7.

Chapter 2

Vehicle Platooning

Before a controller can be designed to control a platoon, it is vital to understand the workings of vehicle platoons. This chapter starts with an explanation of the dynamics of the individual vehicles in section 2-1, followed by the dynamics of the full platoon in section 2-2.

2-1 Vehicle Dynamics

A commonly used model, capturing the longitudinal behaviour of the vehicle, is described by Equation 2-1 [13, 31, 43, 44]:

$$\begin{aligned} ma(t) &= T_e(t)\frac{\eta_T(j)}{r_w} - sgn(v(t))(C_A v^2(t) + \mu mg) \\ \tau(j)\dot{T}_e(t) &= T_{des}(t) - T_e(t) \end{aligned} \tag{2-1}$$

where t is time and p, v and a are the position, velocity and acceleration of the vehicle respectively. η_T and τ are transmission efficiency and the drive train's inertial delay which are both a function of the selected gear j . The actual engine torque is T_e, while the input of the system is the desired engine torque T_{des}. Furthermore, m is the vehicle's mass, r_w the wheel radius, C_A the aerodynamic drag coefficient, μ the rolling resistance coefficient and g the gravitational acceleration.
The model described in Equation 2-1 requires the following assumptions to be made:

- The vehicle's wheels do not experience any slip.
- The vehicle is a symmetric rigid body.
- The vehicle only moves in the longitudinal direction or the motion in other directions does not affect the longitudinal dynamics.
- The vehicle moves on a horizontal plane.

There are several ways to simplify this model further:

1. The dependence on the term $sgn(v(t))$ can be removed under the assumption that the vehicle only moves forwards (in the positive direction of p). This assumption can be made because, in practice, a car will not be using a platooning algorithm when reversing is required.
2. The non-linear aerodynamic drag can be approximated using several methods [43]:
   - A Piecewise Affine (PWA) approximation linearises the drag curve separately for multiple sections of the velocity range. This creates the best approximation of the drag curve, however it does result in a hybrid model, because of the switching between the different sections.
   - A linear least-squares approximation of the drag curve can be used. This is easy to implement because one linearisation is used for the full velocity range, however the results are usually not very accurate.
   - The drag curve can be linearised around a given operating point using its tangent at that point. Compared to the least-squares approximation, this is much more accurate when close to the operating point, but less accurate when further way.
3. The dependence on the gear selection can be neglected by averaging the drive train characteristics for all possible gears [43], or by choosing those that correspond to the gear that will be used most when designing for a specific scenario. Both lead to a fully continuous system instead of a hybrid one.

The model given by Equation 2-1 is both non-linear and defined in continuous time. However, Model Predictive Control (MPC) works with discrete models and to reduce complexity of the the optimisation to a convex problem, the model should be discretised as well as linearised. The resulting model for vehicle i is given by Equation 2-2, where C_{A_l} is the linearised aerodynamic drag coefficient and Δ is the sampling time. The superscript $^+$ denotes the next time step.

$$\underbrace{\begin{bmatrix} p_i^+ \\ v_i^+ \\ T_{e_i}^+ \end{bmatrix}}_{x_i^+} = \underbrace{\begin{bmatrix} 1 & \Delta & 0 \\ 0 & 1-\frac{C_{A_{l_i}}\Delta}{m_i} & \frac{\eta_{T_i}\Delta}{m_i r_{w_i}} \\ 0 & 0 & 1-\frac{\Delta}{\tau_i} \end{bmatrix}}_{A_i} \underbrace{\begin{bmatrix} p_i \\ v_i \\ T_{e_i} \end{bmatrix}}_{x_i} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \frac{\Delta}{\tau_i} \end{bmatrix}}_{B_i} \underbrace{T_{des_i}}_{u_i} + \underbrace{\begin{bmatrix} 0 \\ -\mu_i g \Delta \\ 0 \end{bmatrix}}_{E_i} \tag{2-2}$$

The non-linear dynamics are used to simulate the platoon, while the linear dynamics form the basis of the MPC.

2-2 Platoon Dynamics

When talking about a platoon of N vehicles every vehicle will be denoted by index $i \in \{0, 1, ..., N-1\}$, where 0 is the leader and $N-1$ is the last vehicle in the platoon. Part of a platoon is displayed in Figure 2-1.

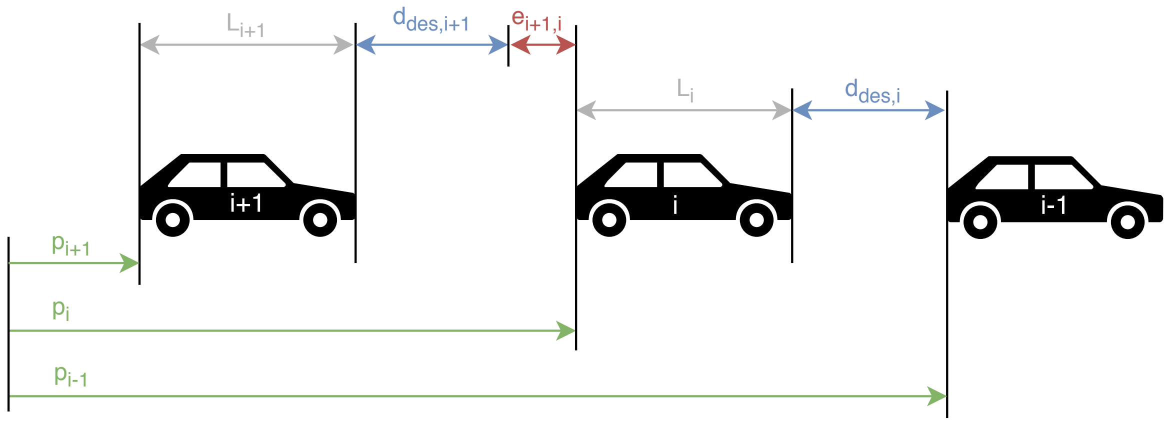


Figure 2-1: Part of a platoon with vehicle i driving at the desired distance from vehicle $i-1$ and vehicle $i+1$ driving too far behind vehicle i.

The objective of the platoon is to have every vehicle driving at the same velocity and maintain a desired inter-vehicular distance (d) with respect to its predecessor. There are three main spacing policies used to determine this desired distance:

- Constant distance (d_0): $d_{des,i} = d_0$ [34, 35, 45]
- Constant time gap (h). Here the desired distance is determined by the velocity of the individual vehicle: $d_{des,i} = v_i h_{des}$ [5]
- Constant headway. This is a combination of constant distance and constant time gap: $d_{des,i} = d_0 + v_i h_{des}$ [25, 28]

The position error (e) and velocity error (f) of vehicle i with respect to vehicle j are given by Equation 2-3 and Equation 2-4 respectively. Here L is the length of the vehicle.

$$e_{i,j}(t) = p_j(t) - p_i(t) - (i-j)(L_i + d_0 + v_i(t)h_{des}) \tag{2-3}$$
$$f_{i,j}(t) = v_j(t) - v_i(t) \tag{2-4}$$

The platoon as a whole can be modelled by stacking the individual vehicle dynamics in Equation 2-2 into the block-diagonal form of Equation 2-5, the errors can then be obtained by choosing the appropriate output equation as shown in Equation 2-6 or by combining these equations into an extended state-space system. Note that the dynamics of the lead vehicle (vehicle 0) are included in the platoon model, since the assumption is made that the input to the lead vehicle will be communicated to the controller.

$$\begin{bmatrix} x_0^+ \\ x_1^+ \\ \vdots \\ x_{N-1}^+ \end{bmatrix} = \begin{bmatrix} A_0 & & & \\ & A_1 & & \\ & & \ddots & \\ & & & A_{N-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} + \begin{bmatrix} B_0 & & & \\ & B_1 & & \\ & & \ddots & \\ & & & B_{N-1} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} + \begin{bmatrix} E_0 \\ E_1 \\ \vdots \\ E_{N-1} \end{bmatrix} \tag{2-5}$$

$$\begin{bmatrix} e_{i,j} \\ f_{i,j} \end{bmatrix} = \underbrace{\begin{bmatrix} -1 & -(i-j)h_{des} & 0 \\ 0 & -1 & 0 \end{bmatrix}}_{C_{i,j}} x_i + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{C_j} x_j + \underbrace{\begin{bmatrix} -(i-j)(L_i + d_0) \\ 0 \end{bmatrix}}_{D_{i,j}} \tag{2-6}$$

Chapter 3

Methodology

Now knowing the workings of vehicles and platoons in general terms, it is time to look at the specifics of the equipment used in this research.
To answer the research question two elements are needed; an experimental test setup and metrics to compare the results. To reliably perform experiments it is vital to understand the test environment and equipment, therefore the experimental setup is explained in section 3-1. Finally, it should be clear how the controllers are evaluated before they are designed, this is discussed in section 3-2.

3-1 Experimental Setup

To evaluate the performance of the controllers in a realistic environment, experiments are performed using Radio Controlled (RC)-vehicles in the Delft Center for Systems and Control (DCSC) robotics laboratory. Figure 3-1 gives a schematic overview of the experimental setup, with the grey boxes representing the hardware components. The setup consists of three main elements. Firstly, a ground station, which is used to control the platoon as well as the communication network. The ground station is discussed in subsection 3-1-1. Secondly, the vehicles that make up the platoon. Their specifications are discussed in subsection 3-1-2. Finally, the Motion Capture System (MoCap), which is used to measure the location of the vehicles, as explained in subsection 3-1-3.

3-1-1 Ground Station

The first component of the experimental setup is the ground station, which is implemented on a Personal Computer (PC). This ground station is used to manage the communication within the network and to run the control algorithms. The ground station is connected to the vehicles using WiFi and to the MoCap using Ethernet.

The ground station manages the communication between the controller(s), vehicle(s) and MoCap using The Robotic Operating System (ROS). ROS is an open-source, meta-operating

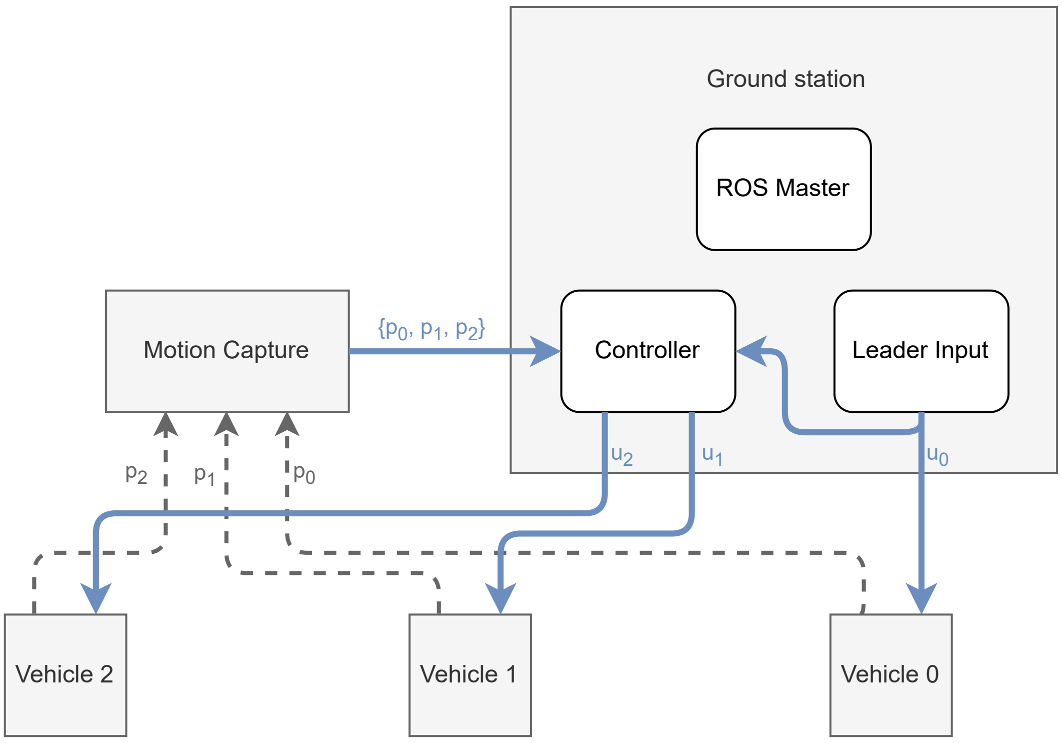


Figure 3-1: Control architecture of the experimental setup for a three-vehicle platoon.

system that runs alongside a traditional operating system. Programs that use ROS are called nodes and different nodes on the same network can communicate with each other through ROS-messages. These messages are organised into topics to which a node can publish messages or subscribe to receive the messages. [46]

The blocks in Figure 3-1 each represent a ROS node and the blue arrows represent the topics with their publisher-subscriber relationships.
The ground station runs the following three ROS nodes[1]:

- ROS Master - This is a node that is the core of any ROS network. Every subsequent node is registered to the Master, which then sets up the necessary communication channels to the existing nodes.
- Leader Input - This node simply feeds a preset velocity profile to the lead vehicle by sending one input value at each time step. Due to the centralised nature of the setup, this input is also received by the controller.
- Controller - This node collects the position data of each vehicle from the MoCap as well as the input to the lead vehicle and applies either the Model Predictive Control (MPC) or Proportional Double Derivative (PDD) control algorithm to this. The obtained control inputs for the following vehicles is then sent to each respective vehicle.

[1]In reality, the ground station runs an extra node per vehicle that forwards each MoCap-message. This is necessary, because there is a bug in MATLAB's ROS-toolbox, that makes MATLAB unable to parse the messages of type `mocap/pose` send by the MATLAB script run on the MoCap-PC. After forwarding these messages the control node is able to receive them.

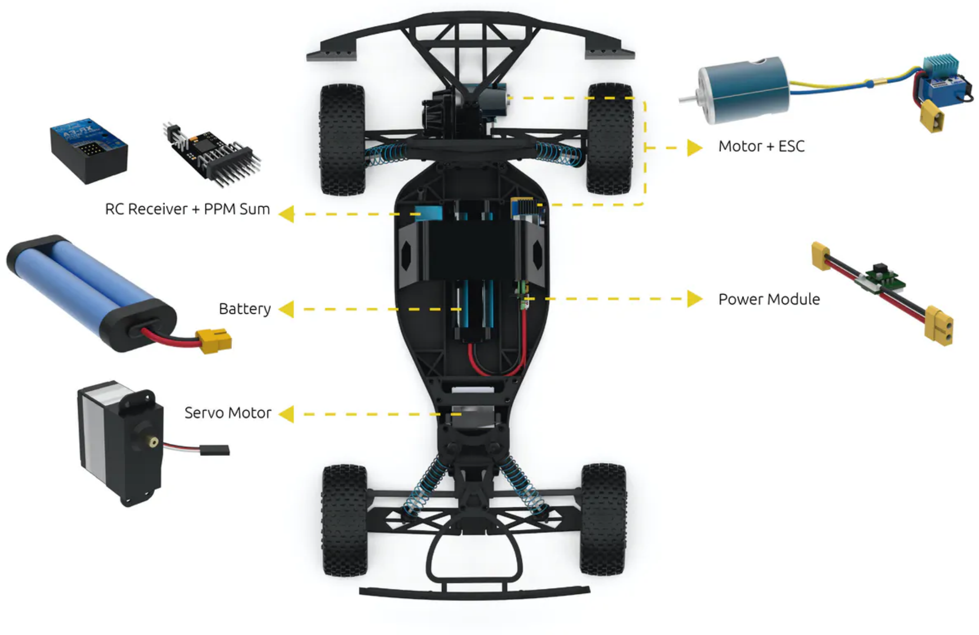


Figure 3-2: The hardware components of the Erle-Rover. [1]

3-1-2 Erle-Rovers

The vehicles used to test the performance of the platoon controllers are RC-rovers produced by Erle-Robotics. The rover and its main hardware components can be seen in Figure 3-2. As shown, the rover has two motors; a servo motor that is responsible for steering the front wheels and a motor with Electronic Speed Controller (ESC) for the rear-wheel drive [1].

The rovers are also equipped with the Erle-Brain 3, a Linux-based embedded computer produced by Erle-Robotics. The Erle-Brain runs the open-source autopilot software ArduPilot, which supports a variety of automated and manual commands. However, to simplify the implementation of custom controllers the Erle-Brain is also equipped with a WiFi receiver. This makes it possible for the rover to receive ROS-messages, that are passed to ArduPilot using the Micro Air Vehicle Communication Protocol (MAVLink). [47] The route the messages take from the ground station to the motor is displayed in Figure 3-3. The ROS messages used for the control of the rover are of the `mavros_msgs/OverrideRCIn` type. These consist of eight uint16 channels of which the first is used for steering and the third for velocity. So the input signals have to be mapped from the physical values, as generated by the controllers, to the corresponding integer values used by the rovers. How this is done is discussed in section 4-1.

3-1-3 Motion Capture System

The experiments will be performed in an area equipped with an OptiTrack Motion Capture System (MoCap), which is used to measure the location of the vehicles.
This system consists of multiple cameras that are used to track reflective markers on the rovers. These cameras send their images to a PC that uses triangulation to compute the

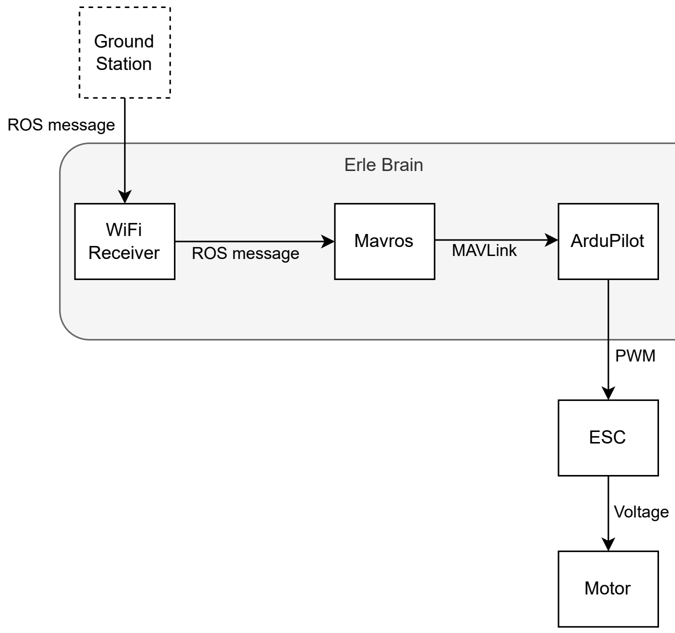


Figure 3-3: The route of a input message from ground station to motor.

position and orientation of each rover in six degrees of freedom [48]. The MoCap PC is also equipped with ROS and uses this to publish the coordinates of the rovers.

Since the area observed by the MoCap is limited, experiments cannot be performed on long straight trajectories. To still allow for longer experiments, they will be performed on a circular trajectory with a constant radius. For the steering, Ackermann steering geometry is assumed. This means the vehicles will have a constant steering angle, γ, as determined by Equation 3-1, where L_{wb} is the length of the wheelbase and r is the radius of the circle.

$$\tan\gamma = \frac{L_{wb}}{r} \tag{3-1}$$

Even though vehicles drive in circles, this research is about the longitudinal behaviour of the platoon. Therefore, the position, p, is determined by the distance travelled by a vehicle instead of by the location in the test area. Ideally, the distance travelled would be determined by the number of laps the vehicle has driven around a constant circle, because then no approximation of the trajectory is needed.
As can be seen in Figure 3-4, in reality, the trajectory of the vehicles is not quite constant, So using the circle to determine the distance travelled by the vehicles would over time lead to discrepancies. Therefore this is calculated using Equation 3-2, this is deemed a valid approximation because the MoCap measures the position of the vehicles 120 times per second. At the maximum test velocity this leads to a deviation from the distance calculated for a perfectly circular trajectory of $6.5 \cdot 10^{-6}\%$ per measurement. The main drawback of this is that the noise is essentially integrated.

$$p^{+} = p + \sqrt{(x^{+} - x)^2 + (y^{+} - y)^2} \tag{3-2}$$

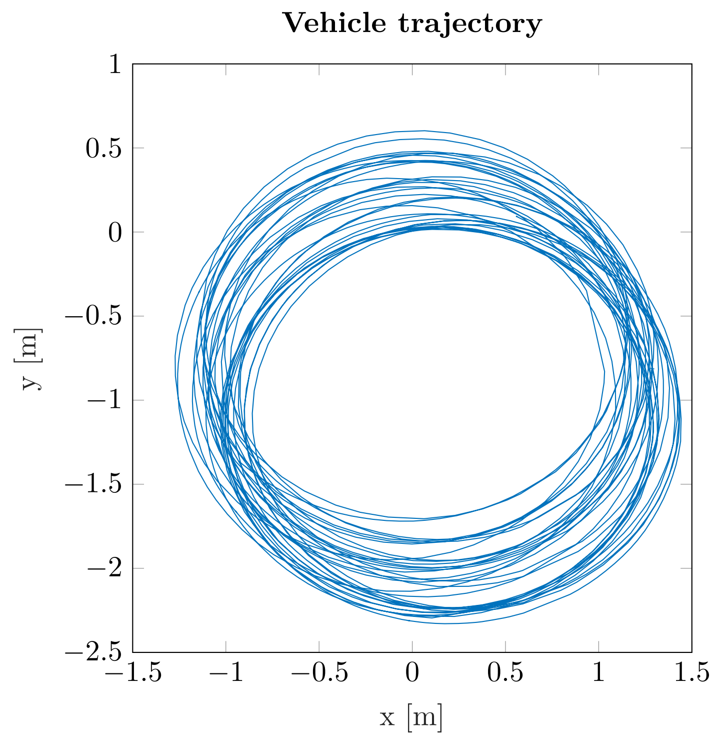


Figure 3-4: Trajectory of a single vehicle over a 5 minute timespan.

3-2 Performance Evaluation

To evaluate the performance of MPC control, two things are needed; a set of performance metrics and a benchmark. The benchmark is provided by implementing a commonly used PDD controller and testing both controllers in the same scenarios. The PDD controller is described in section 5-2. The performance metrics are as follows:

- Coherence (Π_g) - The coherence is a measure of the combined position error of all followers with respect to the leader and can be determined by [49]:

$$\Pi_g = \sum_{i=1}^{N-1} e_{i,0}^2 \tag{3-3}$$

- Combined Local Error (Π_l) - Where the coherence considers the position errors with respect to the leader, the combined local error, given by Equation 3-4, is a measure of the error of each vehicle with respect to its direct predecessor. [49]

$$\Pi_l = \sum_{i=1}^{N-1} e_{i,i-1}^2 \tag{3-4}$$

- Velocity Error - The same formulas used to evaluate the position errors in terms of coherence and combined local error can also be applied to the velocity errors. It should be noted that due to the velocity term in the spacing policy, the velocity and position errors can not both stay at zero during changes in the desired velocity. Therefore, less significance should be placed on this metric when evaluating the platoon in dynamic manoeuvres.

- Energy Expenditure - Both for environmental and cost reasons it is desirable to minimize the energy needed by the platoon. Since the input to the vehicles is a desired velocity, the energy expenditure is approximated using the difference in sequential inputs:

$$E_e = \sum_{i=1}^{N-1} (u_i^+ - u_i)^2 \tag{3-5}$$

- Minimum Headway - One of the goals of platooning is to reduce the distance between vehicles. Therefore it is useful to compare the minimum headway required to prevent collision. This is determined numerically in simulation by decreasing the headway until collision occurs. So the minimum headway is the one used in the last experiment without collision. As explained in section 2-2, the headway can consist of a distance as well as a time gap. However, to simplify this metric the search space is constrained by $d_0 = h_{des}$. Since the vehicles used for testing operate around 1 m/s, both components of the headway will be in the same order of magnitude.

The performance evaluation is done over a number of standardised test scenarios, so each scenario is the same for both controllers and in simulation as well as for the physical setup. For the scenarios, four velocities are used that are equally spaced in the safe velocity range of the vehicles; zero, slow, medium and high. These scenarios, which can be seen in Figure 3-5, are based on common traffic situations:

- Step up - From stationary a step input of medium velocity is applied to simulate a departure scenario (e.g. after a traffic light).
- Step down - Different from the other scenarios, this sudden decrease from high velocity to stationary is not often seen in real life. However, it simulates an emergency stop which is critical for safe participation in traffic.
- Ramp up - Starting at a low velocity the platoon steadily increases its velocity to the maximum operating point. This reflects behaviour at the on ramp of a highway or when speeding back up after a traffic jam.
- Ramp down - Opposite to the increasing ramp, now the platoon steadily loses velocity as could be seen when exiting a highway or approaching am intersection.

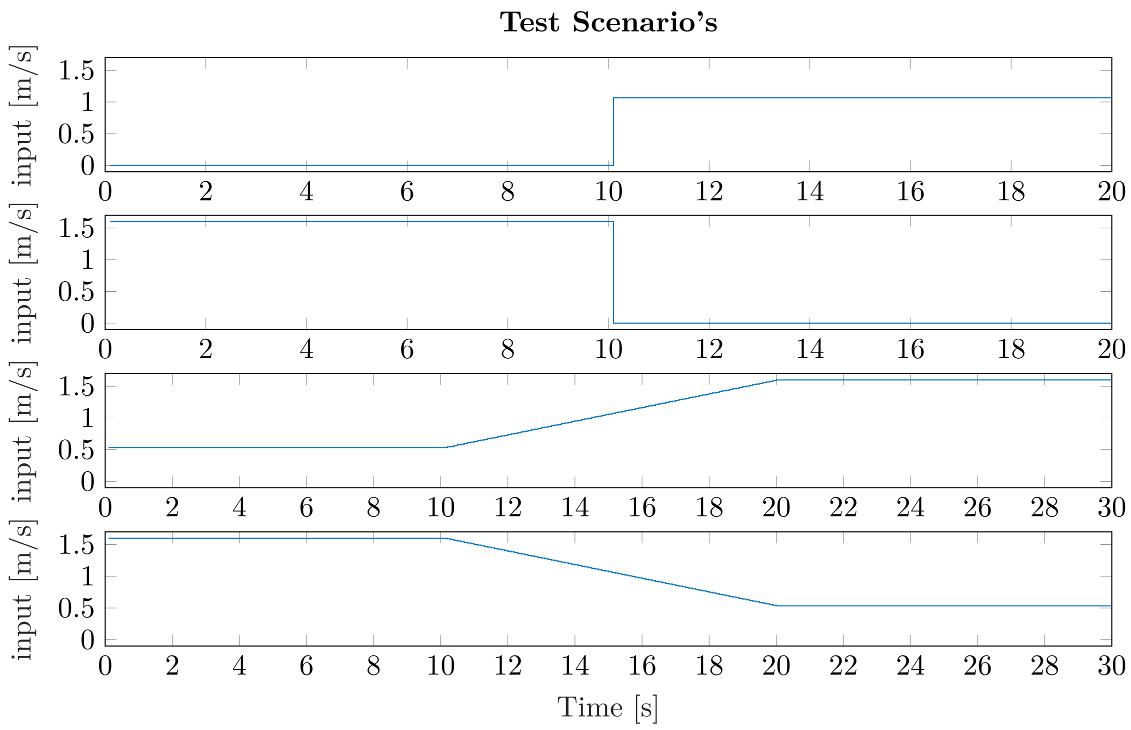


Figure 3-5: Input sequences send to the lead vehicle in each of the standardised test scenarios. From top to bottom; step up, step down, ramp up and ramp down.

Chapter 4

System Identification

Both the simulation results and the Model Predictive Control (MPC) heavily rely on the use of system models. To implement the models for the vehicle dynamics presented in chapter 2, it is necessary to know the parameters of the Erle Rovers. To obtain this information the model must be identified.
The chapter starts with an explanation of the created input mapping in section 4-1, followed by the description of the identification process in section 4-2 and finally the identification results in section 4-3.

4-1 Input Mapping

As mentioned in section 3-1, the rovers are equipped with an Electronic Speed Controller (ESC). Therefore, their longitudinal control input is velocity. However, the vehicle model discussed in chapter 2 uses T_{des} as its input. This difference can be bridged by extending the model with Equation 4-1. Here u is the new input variable, which corresponds to the desired velocity of the rover.

$$T_{des} = mr_w \frac{u - v}{\Delta} \tag{4-1}$$

Furthermore, the input signals to the rovers are uint16 signals. The controllers, however, work with the physical unit of m/s. Therefore, a mapping has to be created to convert between these units.

To create this mapping, firstly a safe test range is established by finding a lower bound at which the rovers are stationary and an upper bound above which the rovers are at risk of leaving their circular trajectory. Within this range a number of experiments are performed in which the rovers are subjected to a staircase style input similar to the one seen in Figure 4-2c. The length of each step is chosen long enough for the rover to settle at a constant velocity. Since it is only possible to measure position, the velocity is obtained using numerical differentiation e.i. by finding the distance travelled in each step and dividing by the step duration. This provides the data points used to map the input values to the real velocity. The data points

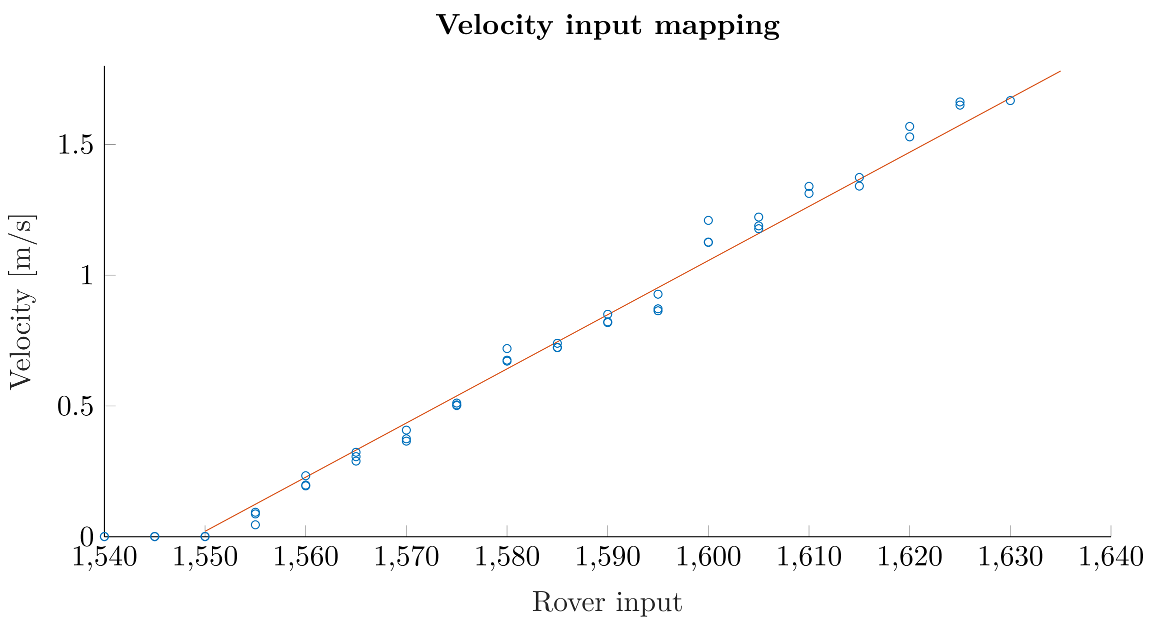


Figure 4-1: The relation between the rover's input values and the corresponding velocity. Here the circles represent measurement points and the line is the mapping used to describe this relation.

and the mapping can be seen in Figure 4-1. The relation between the rover input and the velocity is found to be linear. It is also found that the difference in average velocity for the same input value between the different vehicles is less than 0.5%. Therefore the same mapping is used for all vehicles.

4-2 Identification Procedure

Now the input values have been mapped to their physical units, the input response of a single vehicle can be used to identify its model parameters. The dynamics of the model to be identified can be obtained by combining Equation 2-2 and Equation 4-1, as seen in Equation 4-2. Note that the E matrix from Equation 2-2 is incorporated in the input description to write the linear state-space description in the standard form. This means the gravitational acceleration, g, is added as a constant input. Since position is the only directly measurable state, the output equation of the identification model is given by Equation 4-3.

$$\begin{bmatrix} p^+ \\ v^+ \\ T_e^+ \end{bmatrix} = \begin{bmatrix} 1 & \Delta & 0 \\ 0 & 1 - \frac{C_{A_l}\Delta}{m} & \frac{\eta_T \Delta}{m r_w} \\ 0 & -\frac{m r_w}{\tau} & 1 - \frac{\Delta}{\tau} \end{bmatrix} \begin{bmatrix} p \\ v \\ T_e \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -\mu\Delta \\ \frac{m r_w}{\tau} & 0 \end{bmatrix} \begin{bmatrix} u \\ g \end{bmatrix} \tag{4-2}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \\ T_e \end{bmatrix} \tag{4-3}$$

The parameters in the the model can be divided into three groups:

- Design parameters; Δ has to be chosen so that the discretised model accurately captures the continuous dynamics of the physical vehicle.
- Measurable parameters; m and r_w can be measured directly using scales and callipers respectively.
- Identifiable parameters; η_T, C_{A_l}, μ and τ have to be estimated in the identification process.

To determine the identifiable parameters, MATLAB's linear white-box identification function is used. Therefore a number of experiments are executed to obtain enough input and output data to fit the model. The input signals for these experiments consisted of sinusoidal sequences as well as ramps and step patterns such that a wide spectrum of system behaviour is analysed in the identification process.

4-3 Identification Results

The parameters resulting from the identification process can be found in Table 4-1. The identified values for the engine efficiency and drive train delay are found to be in the expected range. However, the rolling resistance coefficient is identified to be zero, which is obviously not physically possible. Furthermore, the value for the aerodynamic drag coefficient is somewhat higher than initially expected. This can potentially be explained by the fact that both these parameters act on the velocity of the rover, therefore an underestimation of one naturally leads to the overestimation of the other.
These deviations from the expected values are not deemed to be problematic in this situation because, when looking at the position data, the behaviour of the identified model is sufficiently close to that of the physical system. Comparison between these, for different types of input signals, can be seen in Figure 4-2. Figure 4-2c especially, shows that even for a test of more than 50 minutes the model and measured responses stay close. This is supported by a mean normalized root mean squared error of 90% over 19 experiments.

Parameter	Value	Unit
Δ	0.1	s
m	2.036	kg
r_w	$5.6 \cdot 10^{-3}$	m
η_T	0.63	-
C_{A_l}	1.0	kg/s
μ	0	-
τ	0.52	s

Table 4-1: Identified vehicle parameters

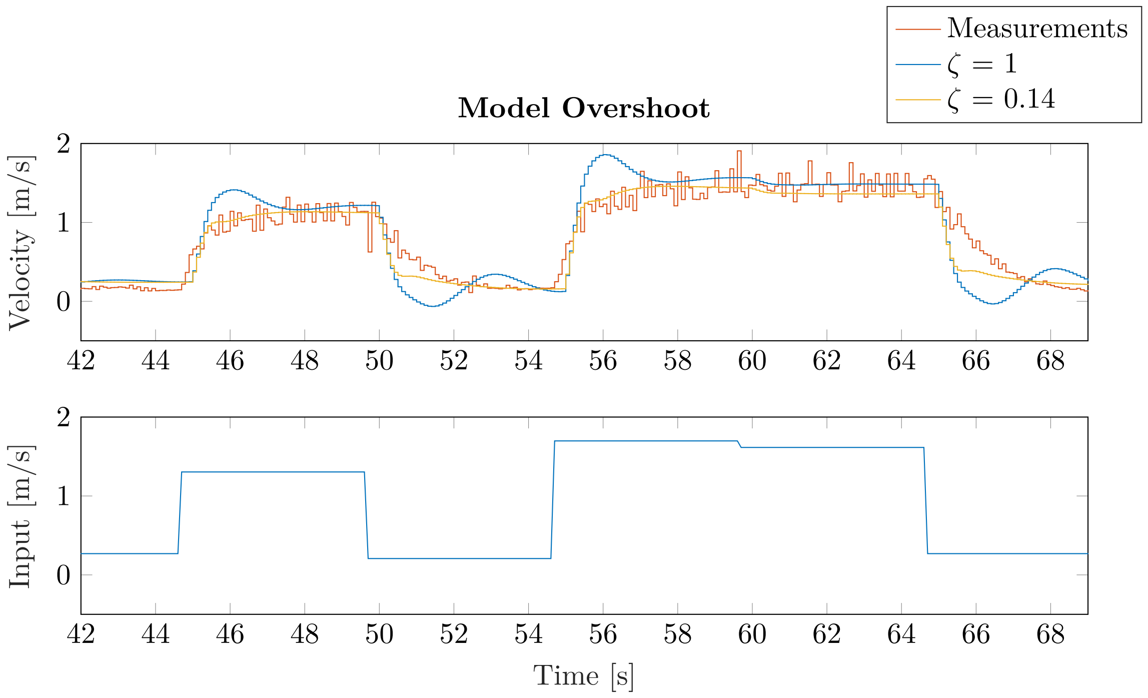
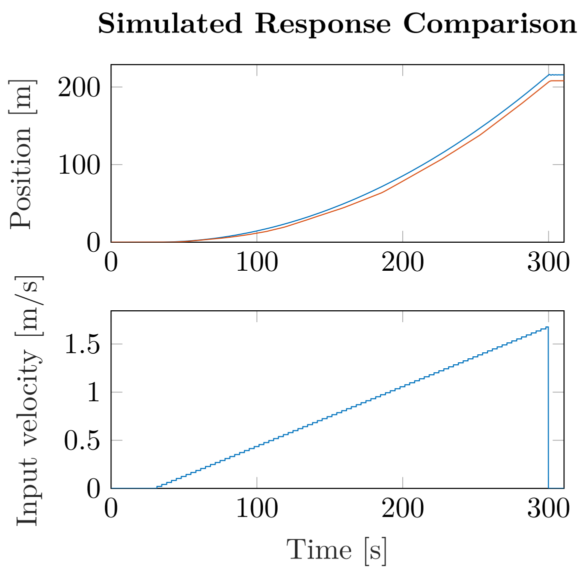


Figure 4-3: Velocity response to a couple of steps for the non-linear model vs the measured data.

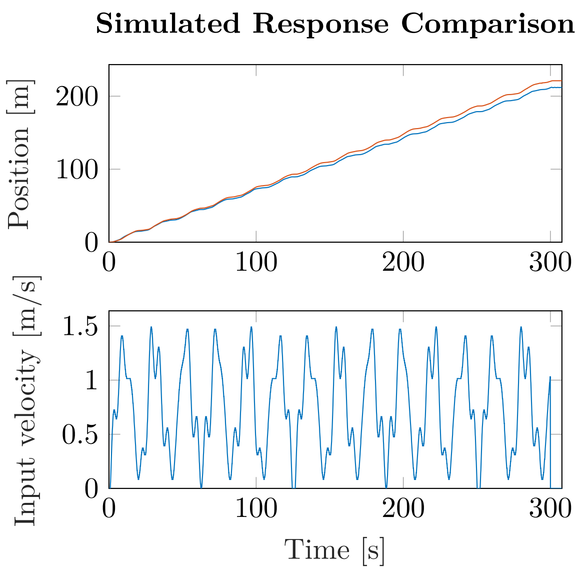
However, an issue occurs when adding the velocity mapping to the non-linear model, as described by Equation 2-1. In Figure 4-3 it can be seen that the model with no correction ($\zeta = 1$), experiences overshoot after every step. The experimental data however has no overshoot at all, instead slowly converging to the new input velocity. This discrepancy is caused by the conversion in Equation 4-1. When a step is applied to the desired velocity, u, the initial large difference between actual and desired velocity leads to a sudden jump in desired torque causing the overshoot in the model. In the Erle Rovers, this behaviour is suppressed by the ESC. To model the ESC, knowledge is needed of the current in different parts of the motor [50]. Since this is not measured in the Erle Rovers, another method has to be used to dampen the jumps in desired torque. Therefore a damping coefficient, ζ, is added to Equation 4-1:

$$T_{des} = \zeta m r_w \frac{u - v}{\Delta} \tag{4-4}$$

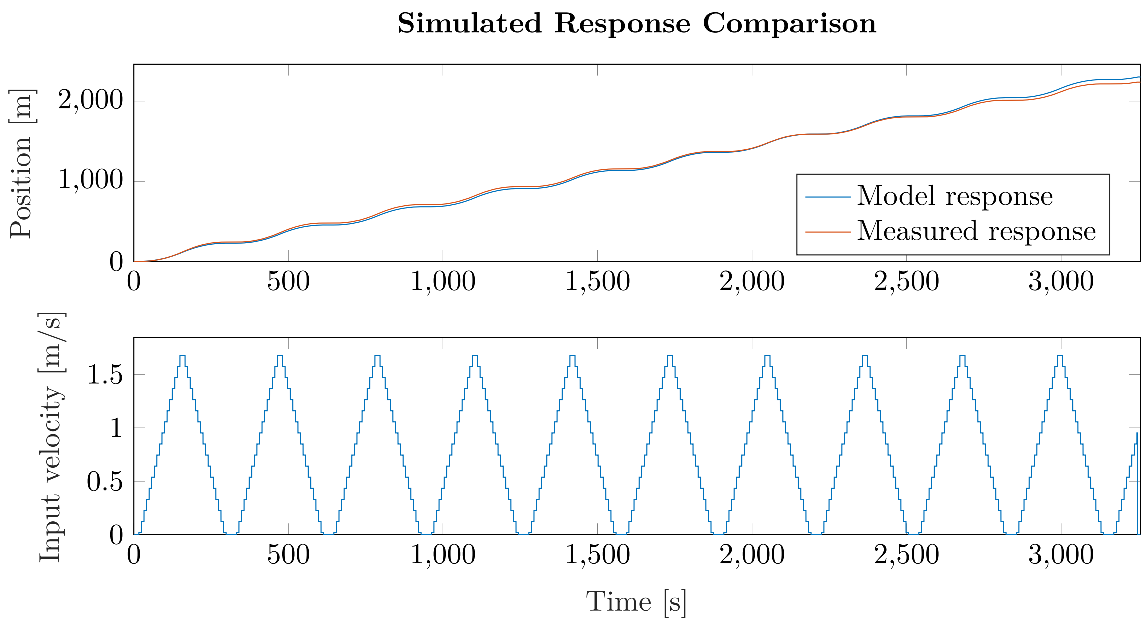
This coefficient is determined by minimizing the difference between the velocity of the non-linear model and the velocity obtained from measurements. Figure 4-3 shows that the new model, with $\zeta = 0.14$, corresponds much better to the experimental setup.



(a) Responses to a ramp input.



(b) Responses to a multi-sinusoidal input.



(c) Responses to a repeated pyramid input.

Figure 4-2: Comparison of the response of the identified model with the response as measured using the lab setup for the same velocity input.

Chapter 5

Controller Design

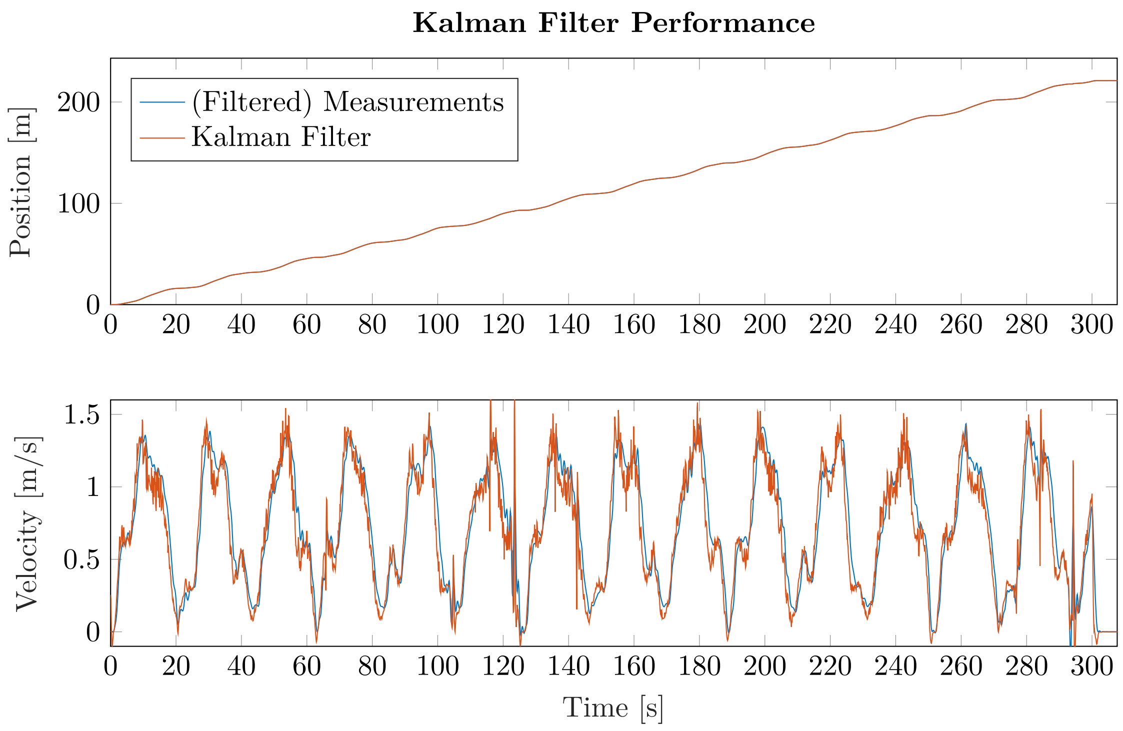
Now the system is identified, the design process for the controllers can be started. This begins with the tuning of the Kalman Filter to obtain the platoon's states. This is done in section 5-1. Then the setup of the Proportional Double Derivative (PDD) and Model Predictive Control (MPC) are discussed in section 5-2 and section 5-3 respectively. Finally, both controllers are tuned in section 5-4.

5-1 Kalman Filter

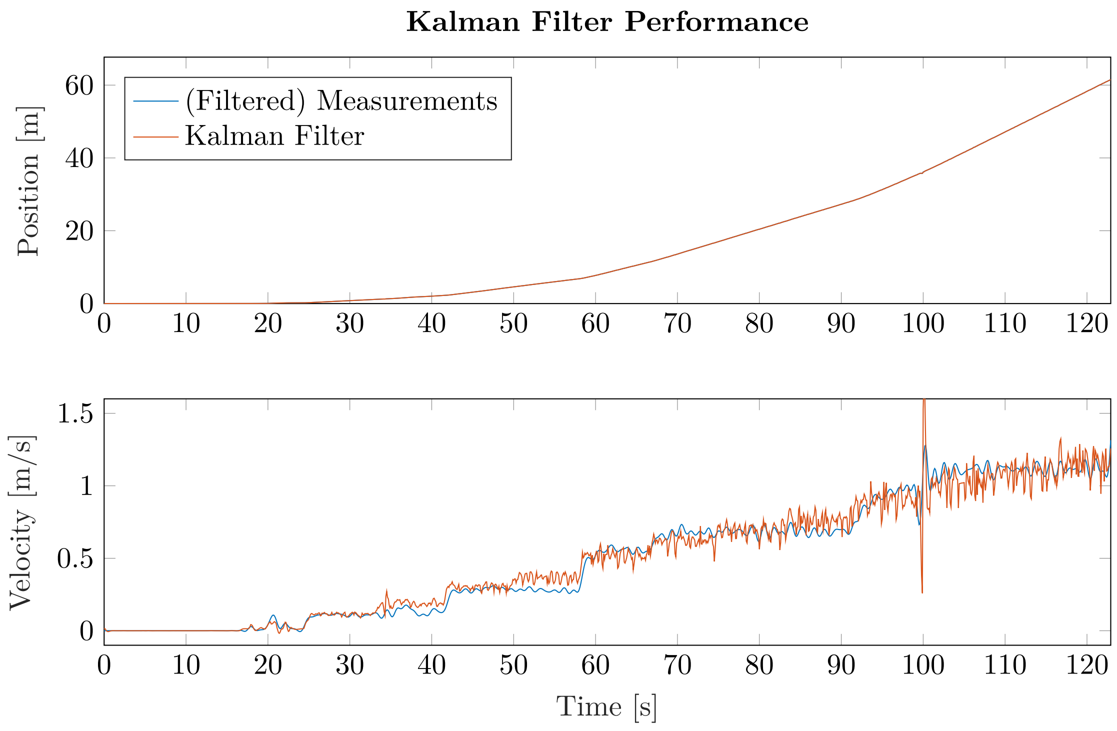
In the experimental setup as described in chapter 3, the only directly measurable outputs are the positions of the vehicles. However, the PDD controller uses the velocities and for MPC it is necessary to know the full states of the system. Therefore, an observer has to be implemented to estimate those states. This is done using a Kalman Filter, because it is able to account for noise.

The Kalman Filter for each vehicle is designed using the vehicle dynamics given by Equation 4-2. The measurements of the Motion Capture System (MoCap) are assumed to be independent for each vehicle. Therefore, the covariance matrix of the measurement noise, R, is diagonal with identical entries ($R = r_n I_N$, where I_N is the identity matrix of dimension N). The covariance of the measurement noise, r_n, is estimated by taking position measurements for a stationary vehicle and calculating the variance of that signal.
Similarly, the process noise covariance matrix, Q, has a block diagonal structure with identical blocks ($Q = I_N \otimes q_n$). However, the process noise covariance for a single vehicle, q_n, cannot be measured, so it is determined by tuning the values of q_n so that the output of the Kalman Filter matches the measured data. A zero-phase derivative filter [51] is used to compare the velocity of the measured data to the Kalman Filter's output.

$$r_n = 6.2 \cdot 10^{-3} \quad , \quad q_n = \begin{bmatrix} 1.5 & 1.5 & 0.03 \\ 1.5 & 750 & 15 \\ 0.03 & 15 & 750 \end{bmatrix} \cdot 10^{-4} \tag{5-1}$$



(a) Comparison for a sinusoidal velocity.



(b) Comparison for a stair-shaped velocity

Figure 5-1: Comparisons of the Kalman filter results to the measurement data. The velocity measurements have been passed to a low-pass filter to remove the measurement noise.

The noise covariances found are given in Equation 5-1. A comparison between the Kalman filter and the (filtered) measurements is shown in Figure 5-1. It can be seen that the position is tracked closely by the Kalman filter, this is of course expected since this is a measurable output. The velocity results also track reasonably well although they are still rather noisy. However, it was found that if the Kalman filter was tuned to filter out more noise it lost too much of the definition around the sharp steps as used in Figure 5-1b.

5-2 Proportional Double Derivative Control Design

As mentioned in chapter 3 conventional Cooperative Adaptive Cruise Control (CACC) is used as a benchmark to evaluate the performance of the MPC controller. For this purpose, the PDD controller designed by [28] is selected.
First, subsection 5-2-1 explains the control law for a single vehicle, then a method to ease implementation for platoons with a centralised control architecture is shown in subsection 5-2-2.

5-2-1 Control Law

The control law for a single vehicle is given in Equation 5-2, here k_p, k_d and k_{dd}, are the proportional, derivative and double derivative gains respectively. These feedback terms are combined with the predecessor's acceleration as a feedforward term. This acceleration is received from the predecessor over the communication network.

$$\dot{a}_i = -\frac{1}{h_{des}}a_i + \frac{1}{h_{des}}\left(k_p e_{i,i-1} + k_d f_{i,i-1} + k_{dd}\ddot{e}_{i,i-1}\right) + \frac{1}{h_{des}}a_{i-1} \tag{5-2}$$

This control law is based on a vehicle with acceleration as its input, since the rovers in the experimental setup take velocity as their input, the result of the control law must be integrated once more before being implemented. For this same reason, the velocity input from the predecessor must be differentiated to obtain the feedforward signal.
Figure 5-2 shows the resulting control structure for a single Predecessor Following (PF) pair. Following [28] $\ddot{e}$ is not used, this means $K = \begin{bmatrix} k_p & k_d \end{bmatrix}$.

5-2-2 Centralised Implementation

Since the experimental setup used in this research relies on a central ground station, the PD controller described in subsection 5-2-1 is implemented centrally as well. This can, of course, be done by running N-1 copies of the same controller. However, this can become hard to manage for larger platoon sizes. Instead, it is also possible to create a single structure containing all copies of the control law. This is done by utilising the differential structure of the control law and converting it into a state-space system. The resulting system is given by Equation 5-3 and Equation 5-4.

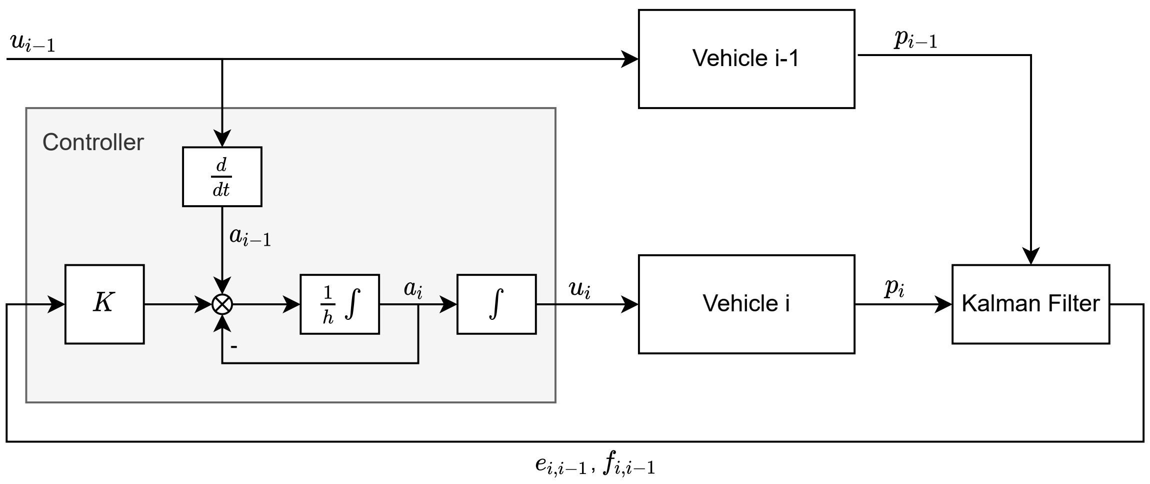


Figure 5-2: Scematic overview of Proportional Derivative (PD) CACC control structure

$$\begin{bmatrix} \dot{u}_1 \\ \dot{a}_1 \\ \dot{u}_2 \\ \dot{a}_2 \\ \vdots \\ \dot{u}_{N-1} \\ \dot{a}_{N-1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & & & & & & \\ 0 & -\frac{1}{h_{des}} & & & & & & \\ & 0 & 0 & 1 & & & & \\ & \frac{1}{h_{des}} & 0 & -\frac{1}{h_{des}} & & & & \\ & & & & \ddots & & & \\ & & & & & 0 & 0 & 1 \\ & & & & & \frac{1}{h_{des}} & 0 & -\frac{1}{h_{des}} \end{bmatrix} \begin{bmatrix} u_1 \\ a_1 \\ u_2 \\ a_2 \\ \vdots \\ u_{N-1} \\ a_{N-1} \end{bmatrix} +$$

$$\begin{bmatrix} 0 & 0 & & & \\ \frac{1}{h_{des}} & \frac{K}{h_{des}} & & & \\ & & 0 & & \\ & & \frac{K}{h_{des}} & & \\ & & & \ddots & \\ & & & & 0 \\ & & & & \frac{K}{h_{des}} \end{bmatrix} \begin{bmatrix} a_0 \\ e_{1,0} \\ f_{1,0} \\ e_{2,1} \\ f_{2,1} \\ \vdots \\ e_{N-1,N-2} \\ f_{N-1,N-2} \end{bmatrix} \tag{5-3}$$

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & & & & & \\ & & 1 & 0 & & & \\ & & & & \ddots & & \\ & & & & & 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ a_1 \\ u_2 \\ a_2 \\ \vdots \\ u_{N-1} \\ a_{N-1} \end{bmatrix} \tag{5-4}$$

5-3 Model Predictive Control Design

The MPC controller used, is based on the standard MPC formulation given in subsection 5-3-1. The behaviour of the MPC controller is determined by three factors, the constraints as described in subsection 5-3-2, the chosen cost function as explained in subsection 5-3-3 and the control and prediction horizon as discussed in subsection 5-3-4.

5-3-1 General Model Predictive Control Formulation

The general formulation for an MPC controller is given by Equation 5-5. This states that at time k the controller computes the input sequence $\boldsymbol{u}_{N_p}(k) = (u(k|k), u(k+1|k), u(k+2|k), \ldots, u(k+N_p-1|k))$ that minimises the cost function in Equation 5-5a over the control horizon with length N_p. This cost function is made up of two parts. Firstly the stage cost (ℓ), which can be a function of the state, the inputs or both. This is used to steer the behaviour of the controller over the control horizon and usually contains the error between the states and a desired reference. The cost on the input is used to conserve resources like energy. The last part of the cost function is the terminal cost (V_f). This is used to take into account the system's future beyond the control horizon. The optimisation is constrained by the system dynamics as given by Equation 5-5b. Finally Equations 5-5c, 5-5d, 5-5e and 5-5f force the solution to stay within the state, input, output and terminal constraint sets $\mathbb{X}$, $\mathbb{U}$, $\mathbb{Y}$ and $\mathbb{X}_f$. It is also possible to add combined constraints (e.g. a function of the state and input) to this framework. [11]
At each time step, the controller performs this optimisation and then implements the optimal input for the first time step only, the rest of the input sequence is discarded. In some circumstances the optimisation problem has no solutions that satisfy all constraints, meaning it is outside the feasible set. In theory, a system that starts within the feasible set, will stay there. However, in practice, unmodelled disturbances can put the system outside this set, in which case the last computed input is repeated. [11]

$$\boldsymbol{u}_{N_c}(k) = \underset{\boldsymbol{u}_{N_c}}{\operatorname{argmin}} \sum_{k=0}^{N_c-1} \ell(x(k), u(k)) + V_f(x(N_c)) \tag{5-5a}$$
$$s.t. \quad x^+ = f(x, u) \tag{5-5b}$$
$$x \in \mathbb{X} \tag{5-5c}$$
$$u \in \mathbb{U} \tag{5-5d}$$
$$y \in \mathbb{Y} \tag{5-5e}$$
$$x(N_p) \in \mathbb{X}_f \tag{5-5f}$$

5-3-2 Constraints

The chosen constraints set on the MPC controller serve two main purposes. The input constraints are used to prevent the controller from giving inputs the vehicles are unable to (safely) handle. The state constraints are used to prevent collisions.

Since the scope of this research only deals with forward motion of the vehicles the input constraints for each vehicle are set by Equation 5-6.

$$0 \leq u_i \leq u_{max} \quad \forall\, i \in 1, ..., N-1 \tag{5-6}$$

The anti-collision constraint as given by:

$$p_i - p_{i-1} < -L \quad \forall\, i \in 1, ..., N-1 \tag{5-7}$$

5-3-3 Cost Function

The stage cost is used to minimise the performance parameters described in section 3-2. The coherence and combined local error for both the position and velocity errors are combined in to this generalised form:

$$\ell_e = \sum_{i=1}^{N-1} \sum_{j=0}^{i-1} w_{i,j} \begin{bmatrix} c_e & c_f \end{bmatrix} \begin{bmatrix} e_{i,j}^2 \\ f_{i,j}^2 \end{bmatrix} \tag{5-8}$$

Here c_e and c_f are used to weigh the position and velocity errors respectively. Then weighting factor, w, can be used to scale the cost in several ways, for example:

- For cost purely on coherence: $w_{i,0} = 1\, \forall\, i$
- For cost purely on local errors: $w_{i,j} = 1\, \forall\, i - j = 1$
- To weight all errors equally: $w_{i,j} = 1 \forall i, j$ This could lead to a more balanced approach. However, the disadvantage of this is that the vehicles further back in the platoon get relatively higher weight because they appear more often in the output vector.
- To mitigate the issue with equally weighted cost, the weight can be scaled by the vehicle number, so: $w_{i,j} = \frac{1}{i}\, \forall\, i, j$

Apart from the position and velocity errors, the energy expenditure is also a metric that needs to be minimised. Since the input given to the vehicles is the desired velocity, the cost can be placed on the change in input. So that cost is given by Equation 5-9, where c_e is the relevant weight.

$$\ell_u = \sum_{i=1}^{N-1} c_u (u_i^+ - u_i)^2 \tag{5-9}$$

The minimum headway is related to the position errors, because smaller position errors overall allow for smaller headways while maintaining safety. Therefore, the minimum headway is the only performance metric not directly weighted. So the full stage cost is given by:

$$\ell = \ell_e + \ell_u \tag{5-10}$$

In addition to the stage cost, the terminal cost also affects the full cost function Equation 5-5a. This cost on the last step in the control horizon is used to include the future of the system beyond the horizon in the optimisation. A higher cost on the errors at this time step can be used to force the platoon's behaviour to converge by leaving room for more deviation off the reference earlier in the, while penalising divergence in the end. Since the same performance indicators are still important at this terminal step, the terminal cost is structured the same as the stage cost on the errors, however, an extra weight w_T is introduced to scale up the significance of the errors at the terminal step. This means the terminal cost is given by:

$$V_f = w_T \, \ell_e \tag{5-11}$$

5-3-4 Control & Prediction Horizon

The last parameter that can be used to tune the behaviour of the MPC controller is the control horizon, N_c. A longer horizon means the optimality of the MPC is valid for longer time intervals, generally leading to solutions closer to the infinite time optimum. However, increasing this, also leads to more optimisation variables, meaning it is computationally costlier. Therefore, for larger platoons, it needs to be checked that the time needed for computation, together with the time needed for the communication from the MoCap to the ground station and that from the ground station to the vehicles, remains shorter than the time step.
One way to still reap some of the benefits of a long horizon, without adding more optimisation parameters, is by adding a prediction horizon (N_p). This means the input sequence remains the same length, however, the cost function is computed over a longer timespan. In this case, the last input in the sequence is repeated until the end of the prediction horizon to evaluate the cost function. An added benefit of this is that the last input must be able to keep the system close to the reference for a longer time so as not to drive the cost up too much.

5-4 Controller Tuning

With the architecture of both controllers defined, it is now time to tune their parameters to achieve the desired behaviour. The tuning is initially performed in simulation, with further tuning in the practical experiments done as needed. A similar approach is used to tune both controllers. Here each tuning parameter is varied one at a time until the best value for that parameter is chosen. This process happens iteratively to account for the interplay of the different parameters. Each controller is evaluated using the performance evaluation as described in section 3-2. The coherence and local position error are used as the main factors in deciding which tuning is best, with the other being used to check for outliers and to settle close contests. Note that all metrics, apart from the minimum headways, are calculated for a headway with $d_0 = 0.5m$ and $h_{des} = 0.5s$.

5-4-1 PDD Tuning

The performance indicators for four possible controllers are shown in Table 5-1. Here it can be seen that when purely looking at both position errors PDD 2 scores the best. However PDD

3 only scores marginally worse on this, while doing significantly better on energy expenditure. Therefore this is the controller that will be used as the benchmark for the MPC.

		Coherence [$\times 10^{-2}$]	Combined Local Error [$\times 10^{-2}$]	Velocity Error [$\times 10^{-2}$]	Energy Expenditure [$\times 10^{-2}$]	Minimum Headway
Step Up	PDD 1	3.3	0.4	0.3	6.4	0.1
	PDD 2	2.2	0.3	0.3	6.9	0.1
	PDD 3	2.3	0.3	0.3	6.1	0.1
	PDD 4	2.8	0.3	0.2	5.5	0.1
Step Down	PDD 1	16.7	2.5	3.6	14.8	0.2
	PDD 2	14.5	1.6	2.5	14.9	0.1
	PDD 3	10.8	1.3	2.0	12.7	0.1
	PDD 4	10.1	1.2	1.8	11.2	0.1
Ramp Up	PDD 1	8.0	1.0	0.2	4.8	0.1
	PDD 2	5.6	0.7	0.2	5.4	0.1
	PDD 3	6.0	0.8	0.2	5.1	0.1
	PDD 4	6.3	0.8	0.2	4.8	0.1
Ramp Down	PDD 1	0.8	0.1	0.1	3.3	0.1
	PDD 2	1.9	0.3	0.05	3.1	0.1
	PDD 3	1.4	0.2	0.05	3.1	0.1
	PDD 4	1.3	0.2	0.06	3.2	0.1

Table 5-1: Comparison of performance metrics for four PDD controllers with a simulated platoon of 5 vehicles. PDD 1: $K = \begin{bmatrix} 0.2 & 0.6 \end{bmatrix}$, PDD 2: $K = \begin{bmatrix} 0.1 & 0.7 \end{bmatrix}$, PDD 3: $K = \begin{bmatrix} 0.1 & 0.6 \end{bmatrix}$, PDD 4: $K = \begin{bmatrix} 0.1 & 0.5 \end{bmatrix}$

5-4-2 MPC Tuning

As described in section 5-3, there are multiple parameters that can be tuned to influence the behaviour of the MPC controller. A number of the different iterations tried can be found in Table 5-2. The preferred MPC controller is deemed to be MPC 3.

		Coherence $[\times 10^{-5}]$	Combined Local Error $[\times 10^{-5}]$	Velocity Error $[\times 10^{-4}]$	Energy Expenditure $[\times 10^{-2}]$	Minimum Headway
Step Up	MPC 1	19.7	13.9	7.2	3.2	0.1
	MPC 2	0.4	8.9	7.2	3.1	0.1
	MPC 3	1.8	3.5	7.2	3.1	0.1
	MPC 4	2.3	3.1	7.2	3.1	0.1
	MPC 5	2.0	3.9	7.2	3.1	0.1
	MPC 6	2.1	3.3	7.2	3.1	0.1
Step Down	MPC 1	44.5	31.5	16.0	4.8	0.1
	MPC 2	0.9	20.1	16.0	4.6	0.1
	MPC 3	4.1	7.9	16.0	4.6	0.1
	MPC 4	5.2	7.0	16.0	4.6	0.2
	MPC 5	5.6	9.4	16.0	4.6	0.1
	MPC 6	4.7	7.4	16.0	4.6	0.1
Ramp Up	MPC 1	12.0	8.5	4.4	2.4	0.1
	MPC 2	0.2	5.4	4.4	2.4	0.1
	MPC 3	1.1	2.1	4.4	2.4	0.1
	MPC 4	1.4	1.9	4.4	2.4	0.1
	MPC 5	1.2	2.4	4.4	2.4	0.1
	MPC 6	1.3	2.0	4.4	2.4	0.1
Ramp Down	MPC 1	8.7	6.8	3.2	2.0	0.1
	MPC 2	0.2	4.2	3.1	2.0	0.1
	MPC 3	0.8	1.5	3.0	2.0	0.1
	MPC 4	1.0	1.4	3.0	2.0	0.1
	MPC 5	0.8	1.7	3.0	2.0	0.1
	MPC 6	0.9	1.4	3.0	2.0	0.1

Table 5-2: Comparison of performance metrics a number of MPC controllers.
MPC 1: $c_e = \begin{bmatrix}1 & 1\end{bmatrix}, w_{i,0} = 1 \forall i, w_T = 1, c_u = 1, N_p = 10, N_c = 10$
MPC 2: $c_e = \begin{bmatrix}12 & 5\end{bmatrix}, w_{i,0} = 1 \forall i, w_T = 1, c_u = 1, N_p = 10, N_c = 10$
MPC 3: $c_e = \begin{bmatrix}12 & 5\end{bmatrix}, w_{i,j} = \frac{1}{i} \forall\, i, j, w_T = 1, c_u = 1, N_p = 10, N_c = 10$
MPC 4: $c_e = \begin{bmatrix}12 & 5\end{bmatrix}, w_{i,j} = \frac{1}{i} \forall\, i, j, w_T = 2, c_u = 16, N_p = 10, N_c = 10$
MPC 5: $c_e = \begin{bmatrix}12 & 5\end{bmatrix}, w_{i,j} = \frac{1}{i} \forall\, i, j, w_T = 1, c_u = 1, N_p = 16, N_c = 8$
MPC 6: $c_e = \begin{bmatrix}12 & 5\end{bmatrix}, w_{i,j} = \frac{1}{i} \forall\, i, j, w_T = 1, c_u = 1, N_p = 12, N_c = 10$

Chapter 6

Results

To answer the research question posed it is essential that the controllers designed in chapter 5 are actually tested. This chapter will show and discuss the results of these tests, first in simulation in section 6-1, followed by the physical tests in section 6-2.

6-1 Simulation Results

In section 5-4, some of the simulation results were already shown to demonstrate the tuning results. However, in this section the results will be further analysed by comparing the figures and metrics for the two tuned controllers. This is done using the metrics and scenarios as defined in section 3-2. The system that is simulated consists of a platoon with the dynamics as defined in Equation 2-1, combined with Equation 4-1. Just like the lab setup, it only gives the position as an output, so the Kalman Filter is used to estimate the velocity for the controllers to use.
From Table 6-1, it is clear that the Model Predictive Control (MPC) controller scores better on every criterion. It should also be noted that the minimum headway is always at least 0.1 second because that is the sampling time of the system.

Looking at the platoon behaviour in more detail in Figures 6-1, 6-2, **??** and 6-4, the following things stand out:

- For all scenarios almost all errors are around a factor 2-5 smaller for the MPC than for the Proportional Double Derivative (PDD) controller. This trend is broken by the velocity error in both ramp scenarios, which are roughly equal in magnitude for both controllers. This is because during changes in velocity the desired headway also changes. Therefore, it is not possible to keep both the position and velocity errors near zero.
- In the step scenarios in Figures 6-1 and 6-2, a small amount of overshoot can be seen in the velocity of the lead vehicle. This means that the damping (ζ) placed on the transformation from velocity to torque is not quite low enough to eliminate steps of this magnitude.

- In most scenarios, the local error increases for vehicles further back in the platoon. This means the platoon is not string stable. String stability is a desirable characteristic because it means the platoon can grow further without increased risk of collision.
  An exception can be found in the MPC platoon during the step up. Here the largest local error is made by vehicle 2. This still does not strictly mean the platoon is string stable, because then the error should decrease for each vehicle further down the platoon.

- Looking at the inputs during the step down, it is clear that the MPC inputs are never below zero. This behaviour is enforced by the input constraint placed on the controller. The PDD controller is not constrained in such a way and therefore freely gives negative inputs leading the vehicles to drive backwards for part of the settling time. In real traffic situations a limiter should be placed on the PDD controller as well to prevent this dangerous behaviour, even if that increases the braking distance.

- During both ramp scenarios, the maximum errors can be found at the end of the ramp. This shows that, since the vehicles have no knowledge of the trajectory of the leader they can not catch up to the changing velocity.

		Coherence	Combined Local Error	Velocity Error	Energy Expenditure	Minimum Headway
Step Up	MPC	$1.8 \cdot 10^{-5}$	$3.5 \cdot 10^{-5}$	$7.2 \cdot 10^{-4}$	$3.1 \cdot 10^{-2}$	0.1
	PDD	$2.3 \cdot 10^{-2}$	$0.3 \cdot 10^{-2}$	$0.3 \cdot 10^{-2}$	$6.1 \cdot 10^{-2}$	0.1
Step Down	MPC	$4.1 \cdot 10^{-5}$	$7.9 \cdot 10^{-5}$	$16.0 \cdot 10^{-4}$	$4.6 \cdot 10^{-2}$	0.1
	PDD	$10.8 \cdot 10^{-2}$	$1.3 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$	$12.7 \cdot 10^{-2}$	0.1
Ramp Up	MPC	$1.1 \cdot 10^{-5}$	$2.1 \cdot 10^{-5}$	$4.4 \cdot 10^{-4}$	$2.4 \cdot 10^{-2}$	0.1
	PDD	$6.0 \cdot 10^{-2}$	$0.8 \cdot 10^{-2}$	$0.2 \cdot 10^{-2}$	$5.1 \cdot 10^{-2}$	0.1
Ramp Down	MPC	$0.8 \cdot 10^{-5}$	$1.5 \cdot 10^{-5}$	$3.0 \cdot 10^{-4}$	$2.0 \cdot 10^{-2}$	0.1
	PDD	$1.4 \cdot 10^{-2}$	$0.2 \cdot 10^{-2}$	$0.05 \cdot 10^{-2}$	$3.1 \cdot 10^{-2}$	0.1

Table 6-1: Comparison of performance metrics for the best PDD and MPC controllers with a simulated platoon of 5 vehicles.

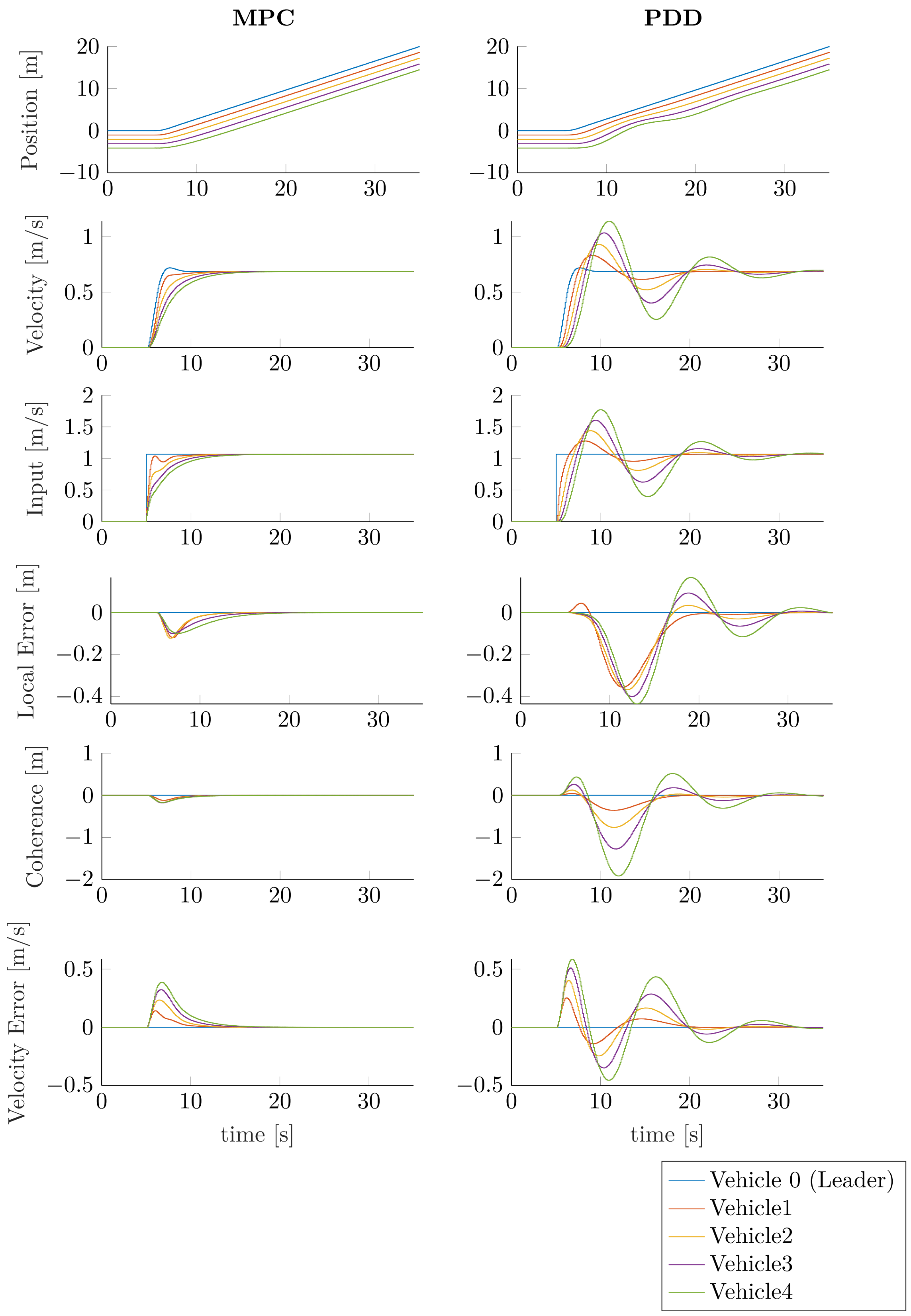


Figure 6-1: Platoon behaviour for both controllers during a step up scenario.

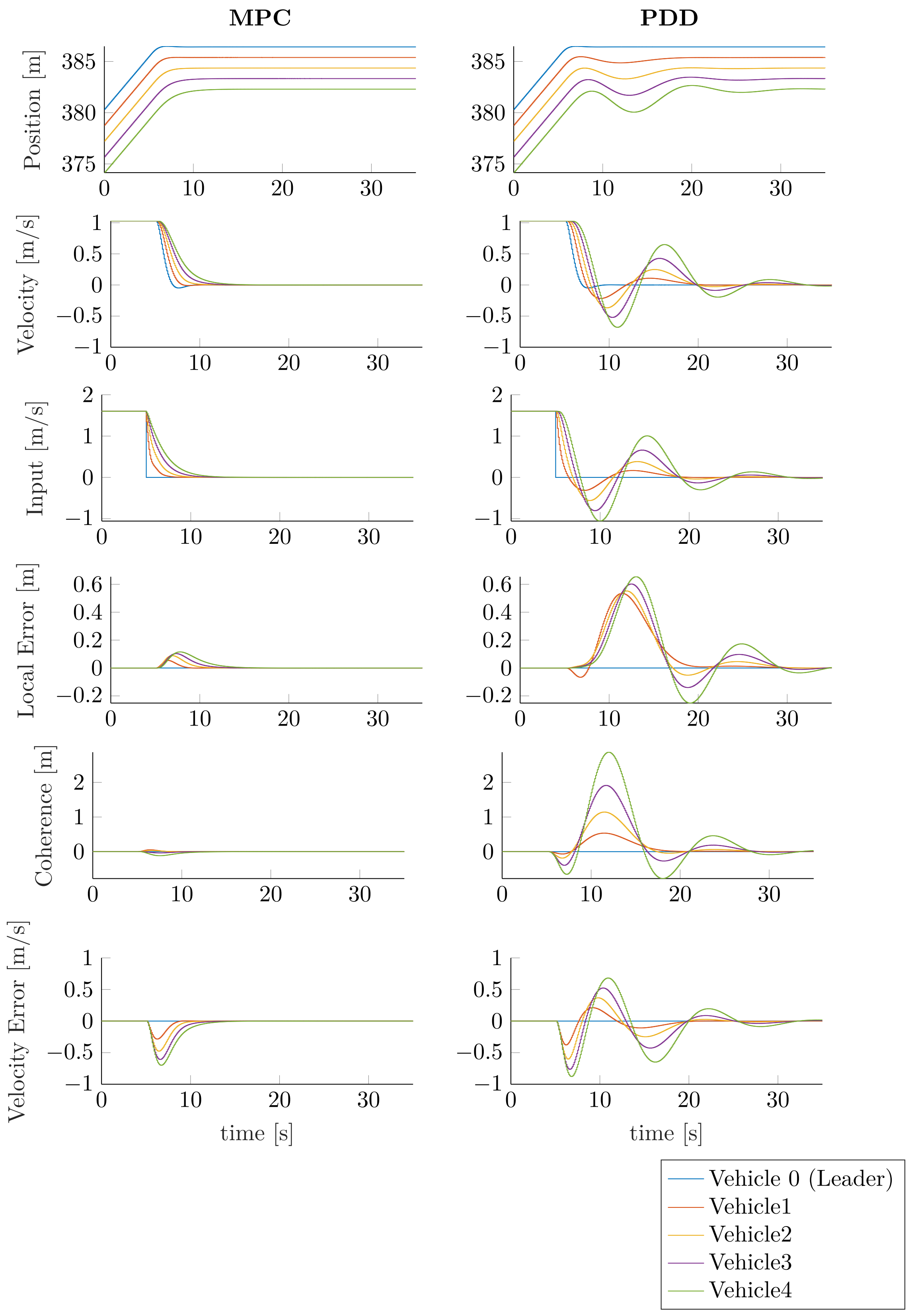


Figure 6-2: Platoon behaviour for both controllers during a step down scenario.

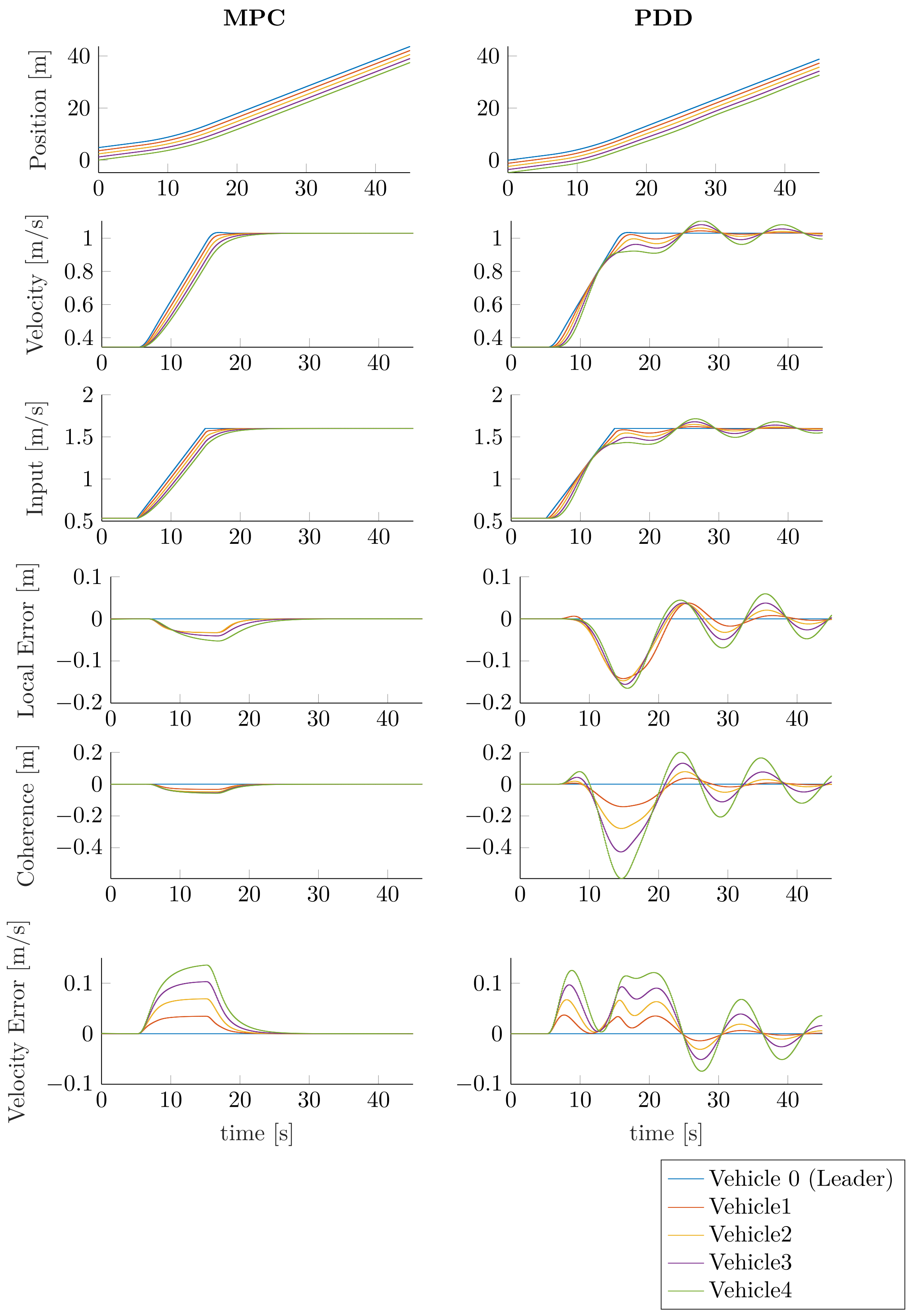


Figure 6-3: Platoon behaviour for both controllers during a ramp up scenario.

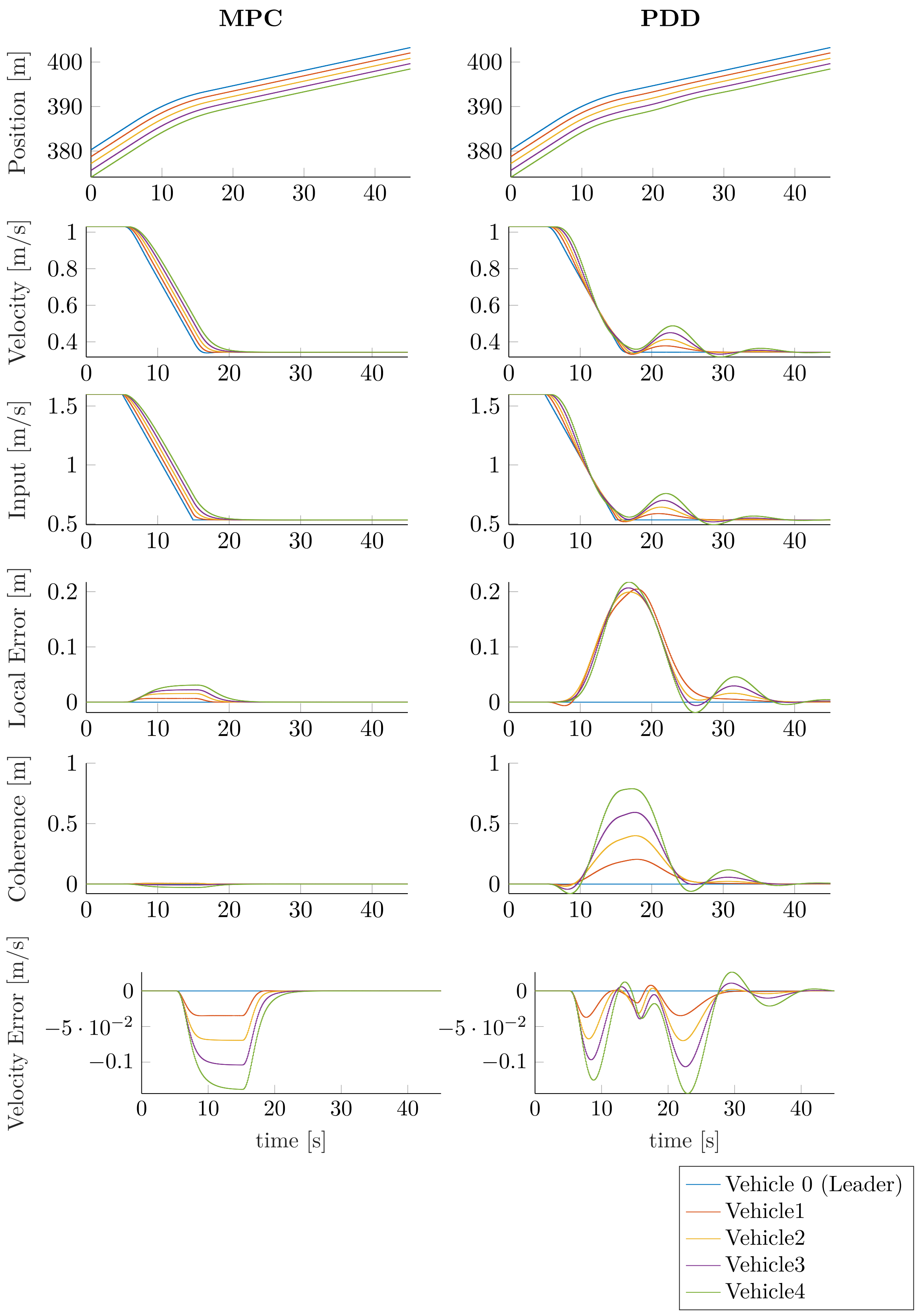


Figure 6-4: Platoon behaviour for both controllers during a ramp down scenario.

6-2 Lab Results

With both controllers tuned in simulation, they are ready to be tested in the lab environment. The differences between the simulation and lab results and ways to mitigate any practical issues are discussed in subsection 6-2-1. The comparison between both controllers is finally made in subsection 6-2-2.
It should be noted that due to time constraints the tuning of the controllers as used in the lab are slightly different from the best ones found in the tuning. The controllers used in the lab are a PDD with $K = \begin{bmatrix} 0.2 & 0.7 \end{bmatrix}$ and an MPC with prediction horizon = 16, control horizon = 8, error costs = [12 5], rate of input cost = 16, terminal cost factor = 1, cost type = scaled.

6-2-1 Discrepancies from Simulation

As expected the simulation model does not perfectly emulate reality, so the platoon performance is also different. The most notable difference are:

- Measurement noise - It is in line with expectations that any physical measurements are subjected to a degree of measurement noise and the Kalman filter helps the controllers to handle that. However, due to the way the position is determined from the Motion Capture System (MoCap) data (see Equation 3-2) this noise is essentially integrated, therefore accumulating over time. This does not cause much of an issue when all vehicles are driving because the noise is similar for each vehicle. However, when the vehicles are stationary for a longer period, the accumulative measurement noise can make the control system believe the vehicles are in fact in motion. This can cause following vehicles to start driving even though their predecessor is in fact stationary. An example of this can be seen in Figure 6-5. Another issue with the measurements is that in rare cases the MoCap system has a faulty measurement. One such case can be seen in Figure 6-6. Vehicle 1 seems to travel almost 2 meters in a single time step and due to the assumption that the vehicles can only move forward, the correction in the next measurement is seen as another leap of 2 meters. The controller response accordingly and forces vehicle 2 to keep up with what it believes vehicle 1 is doing, thereby causing a collision between the two that can not be seen from the data because it shows the incorrect position of vehicle 1. Naturally, results with errors like those seen here are excluded from the results of this research, however, they do need to be considered for truly safe vehicle platooning.
  A way to prevent both these issues is to equip the vehicles with velocity sensors so this data can be used to validate the MoCap data. However, that is outside the scope of this research.

- Velocity variations - As explained in section 3-1 the Erle Rovers are equipped with an Electronic Speed Controller (ESC), which is meant to make the vehicle drive the velocity corresponding to the given input signal. This ESC adds an extra layer to the overall control architecture of the platoon that cannot be altered and was not included in the vehicle model. Figure 6-7 shows the response of a single vehicle to a step followed by a constant input. It can clearly be seen that the velocity of the vehicle is not constant but rather that it oscillates in the region near the input value. This means that both

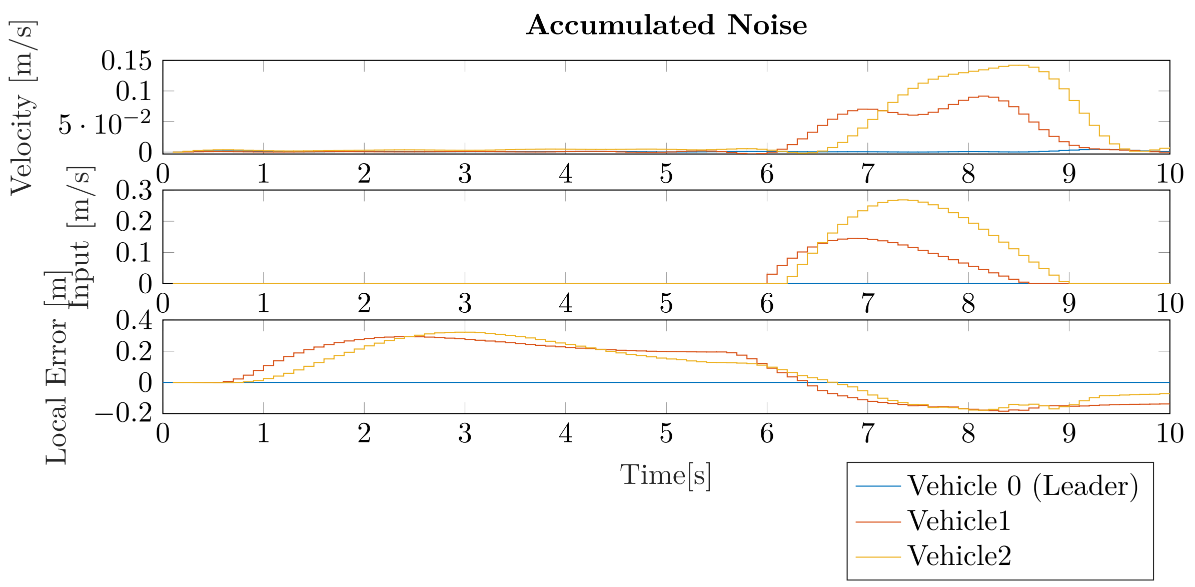


Figure 6-5: Following vehicles are made to to drive due to an accumulation of measurement noise even though the lead vehicle is stationary.

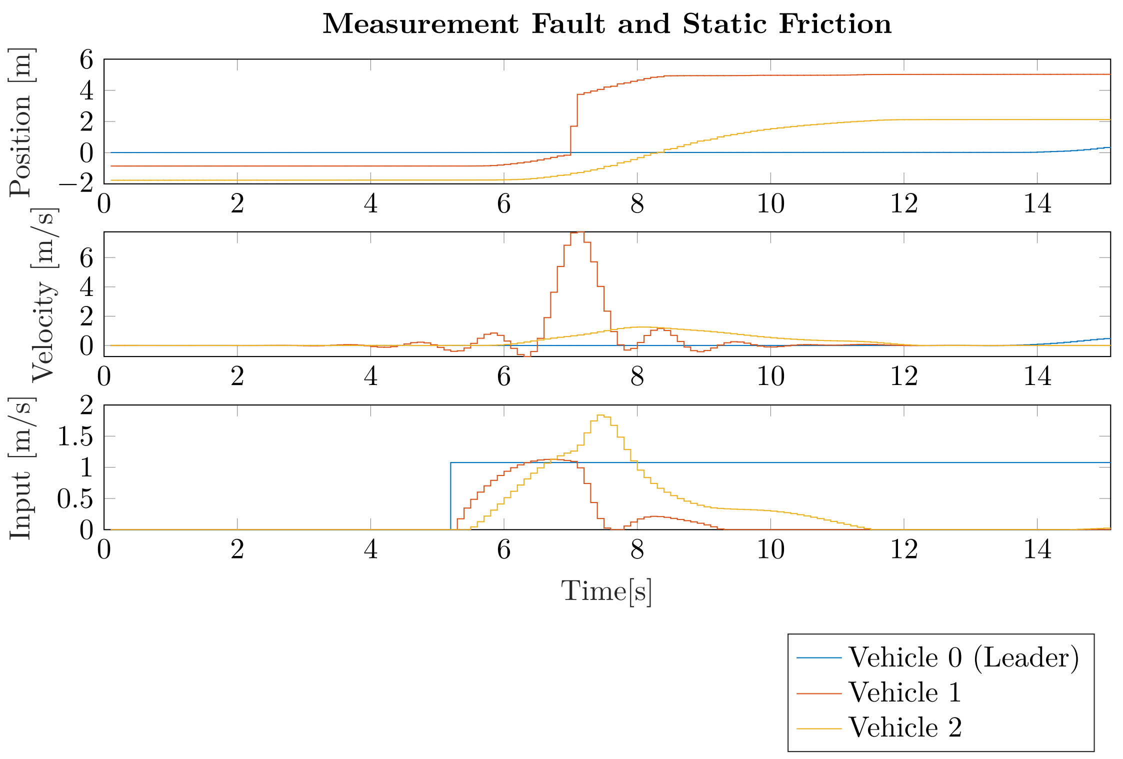


Figure 6-6: Section of the response to a step input, clearly showing a measurement error for vehicle 1 and a delayed response for vehicle 0.

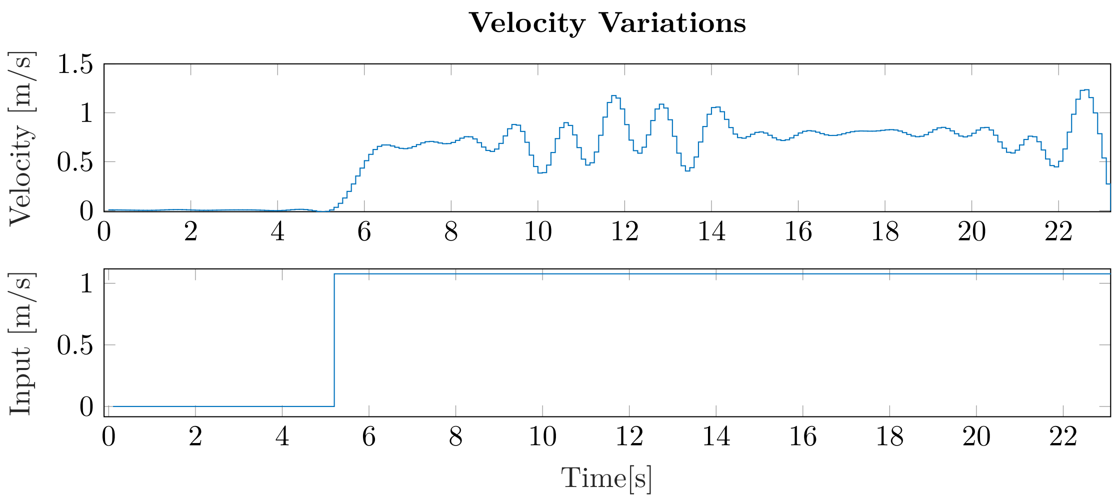


Figure 6-7: Response of the lead vehicle to a step followed by a constant input.

the feedforward component of the PDD controller, as well as the behaviour predicted by the model used by the MPC will never give a truly accurate picture of the velocity of the preceding vehicles.

- Drive train issue - Apart from the clear measurement error, Figure 6-6 also shows that vehicle 0 does not respond to the step input for almost 10 seconds. During the experiments it regularly occurred that, while it was audible that the motor was spinning, this vehicle did not actually move. This happened mostly at lower velocities and could usually be mitigated by giving the vehicle the slightest manual push to overcome the friction in the drive train. This friction was not constant and usually got worse over consecutive experiments. Due to this variable and unpredictable nature issues like this can not be included in a fixed model.
- Initial Errors - In all simulations the initial conditions are set up so that all errors are zero. The same is done for the step up scenario for the lab test. However, since all other scenarios start with a non-zero initial velocity and due to the issues already described it is not possible to realise these same initial conditions for them. In itself that poses little problem since the controllers should be able to cope with small deviations. However, it does pose an issue for the comparison of both controllers because the initial conditions will never be truly equal for both. It also means that both decreasing scenarios can only be tested if the controller can reliably perform at least one of the increasing scenarios.

The inconsistencies caused by both the drive train issues and the ESC can cause the PDD controller to destabilise, like in Figure 6-8. This behaviour is forced by the feedforward component of the PDD controller used, which acts purely on the input signal to the preceding vehicle without regard for whether that desired velocity is actually realised. To dampen this effect the derivative component of the PDD gain should be increased, because this acts on the actual velocities of the vehicles. It is found that returning to the original gain of $K = \begin{bmatrix} 0.2 & 0.7 \end{bmatrix}$ improves the stability of the platoon under these velocity inconsistencies. Furthermore, it is found that the gain should not be increased further because it can increase the acceleration of the following vehicles to the point where their overshoot causes collisions.

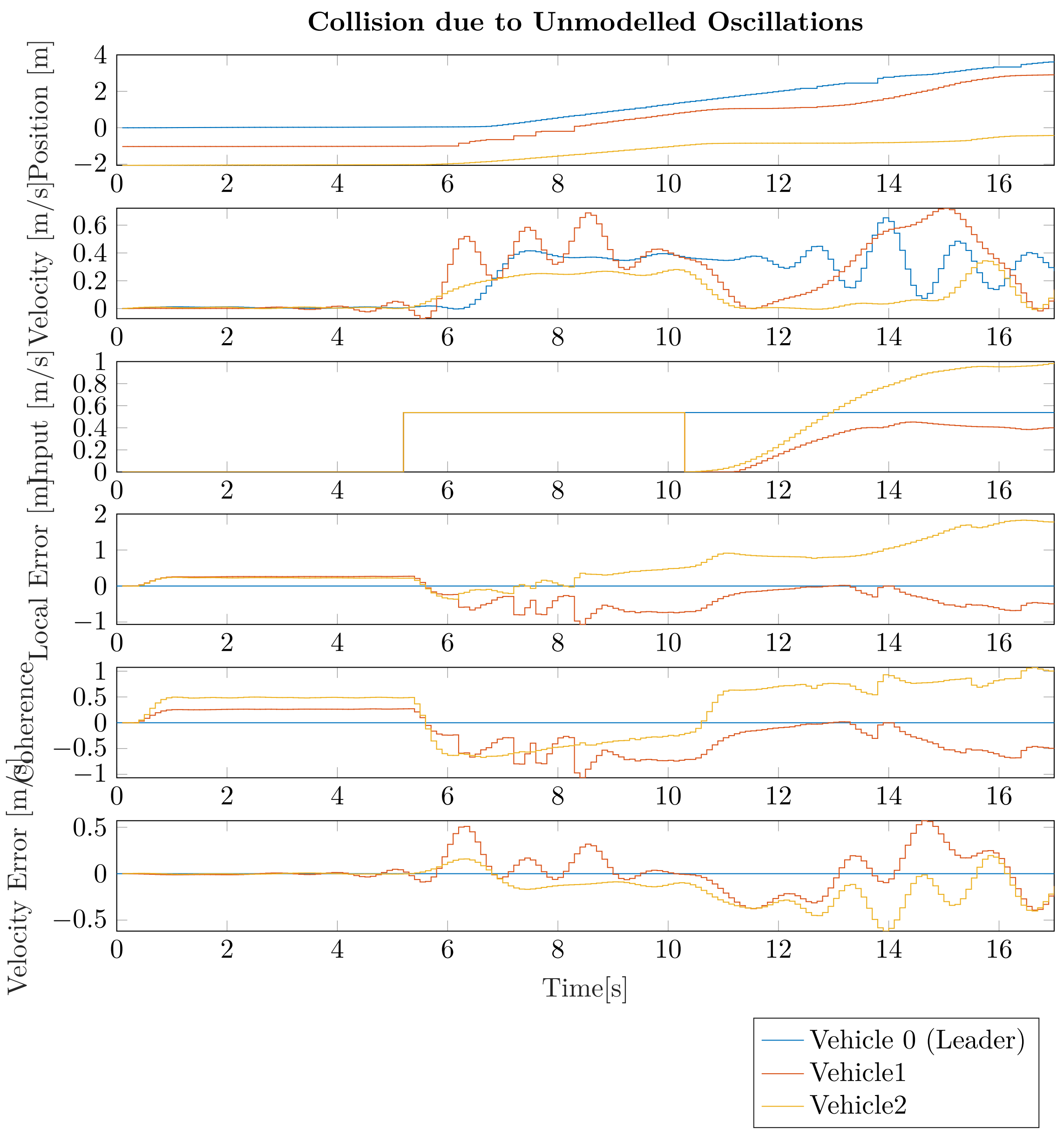


Figure 6-8: The oscillations caused by the ESC result in a collision between vehicle 1 & 0 after 16 seconds.

Apart from the issues arising from the discrepancies between the model and reality, the main issue with the practical implementation of MPC is that it can not be guaranteed that the platoon will always stay within the feasible set. The default behaviour when the optimisation is infeasible is to repeat the last control input. In some cases, the situation resolves itself and the platoon re-enters the feasible set, however, it is also possible that the repeated input drives the system further outside it.
One way to prevent infeasible optimisations is to add slack variables to (some of) the constraints. This softens the constraint by allowing the optimisation to break the constraint, but it adds a high cost to this violation to prevent it from happening in most cases. However, in this case there is only one group of constraints that can be softened, namely the upper bound on the inputs. The anti-collision constraints can not be softened because that directly jeopardises the safety of the platoon and the lower bound on the inputs cannot be violated, because negative velocities would violate the assumptions on which the model was built.

It should be noted that because of the sizeable discrepancies between both, the performance metrics of the lab experiments can not be directly compared to those of the simulations.

6-2-2 Controller Comparison

With the differences between reality and simulation discussed, the lab results can now more carefully be assessed. Firstly each scenario will be assessed qualitatively and a quantitative comparison based on the performance metrics will follow at the end of this section.

Starting with the step up scenario, the responses for which can be found in Figure 6-9 and Figure 6-10 for PDD and MPC respectively. Firstly it must be noted that the lead vehicle experience notably more drive train issues in the PDD scenario, as can be seen from the delay between the step input and the corresponding increase in velocity. Even so, the most notable difference between both controllers is the degree of overshoot. The PDD controller has a considerable amount of overshoot, to the point that both followers subsequently have to correct for it by slowing to velocities well below that of the leader. Conversely, in the MPC platoon both followers undershoot the leaders step input and settle a lot quicker in its neighbourhood. The delay in lead vehicle response does contribute to this overshoot though because the behaviour expected by the feedforward component of the PDD is not consistent with the lead vehicle's actual behaviour.
Another aspect to notice is that both local errors in the PDD scenario almost consistently have the same sign, leading to an accumulative element to the coherence. This could lead to problems for larger platoons, where the length of the full platoon could drastically increase in such a scenario. This is not seen in the MPC platoon, where both local errors tend to have opposite signs. This is probably caused by the fact that in the MPC case the coherence is explicitly included in the cost function.

Unfortunately no truly successful completions of the step down scenario were performed by the PDD controller. This in itself is a big drawback for the PDD controller because this scenario is designed to mimic an emergency stop and this should be able to be performed reliably for save platoon control. The step down scenario in Figure 6-11 did not result in a collision itself, but it did start with vehicle 0 and 1 already touching due to a measurement issue while getting up to speed.
So while this comparison should be taken with a grain of salt, it must be noted that the PDD

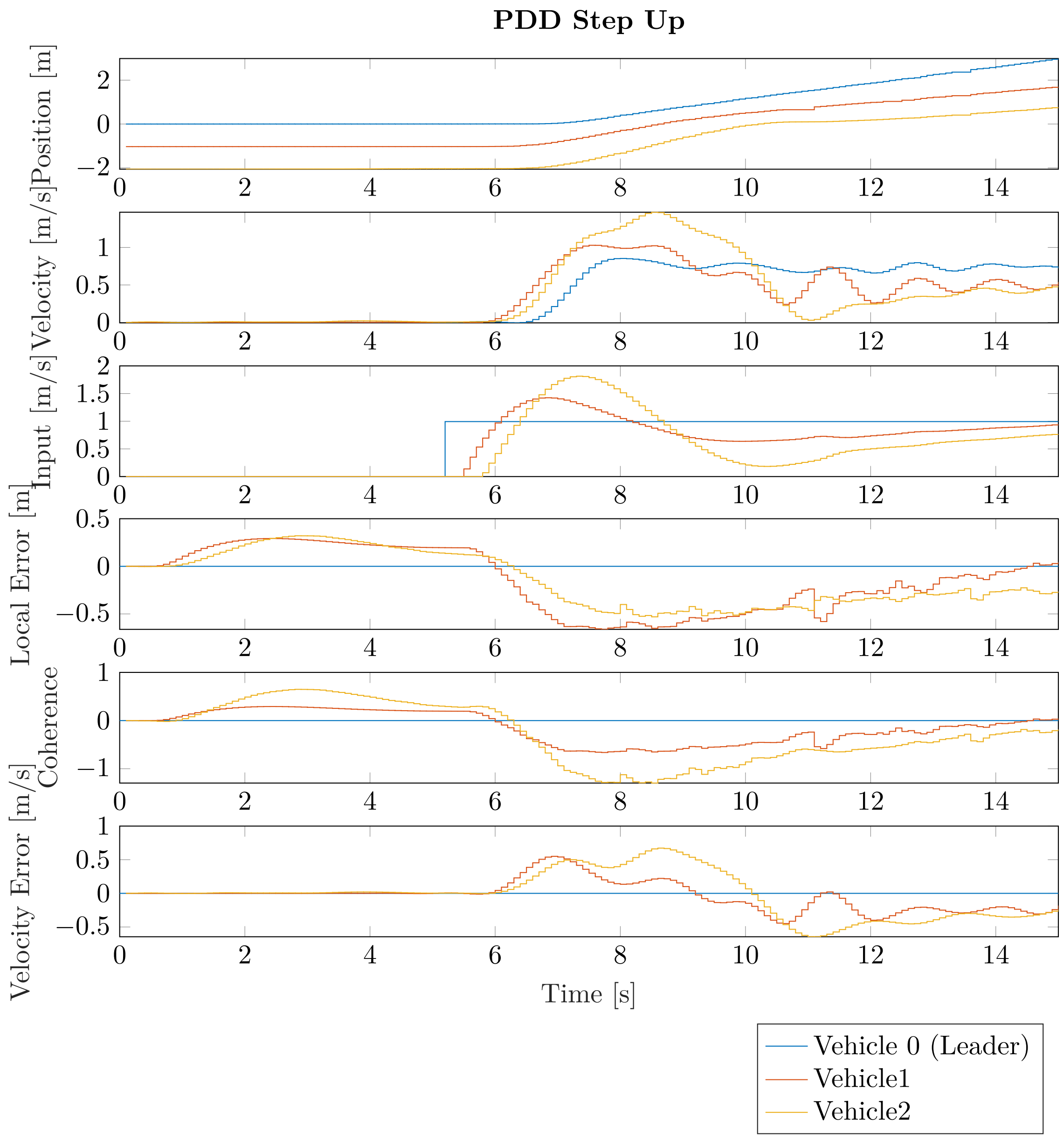


Figure 6-9: Response of an PDD controlled 3 vehicle platoon to an upward step.

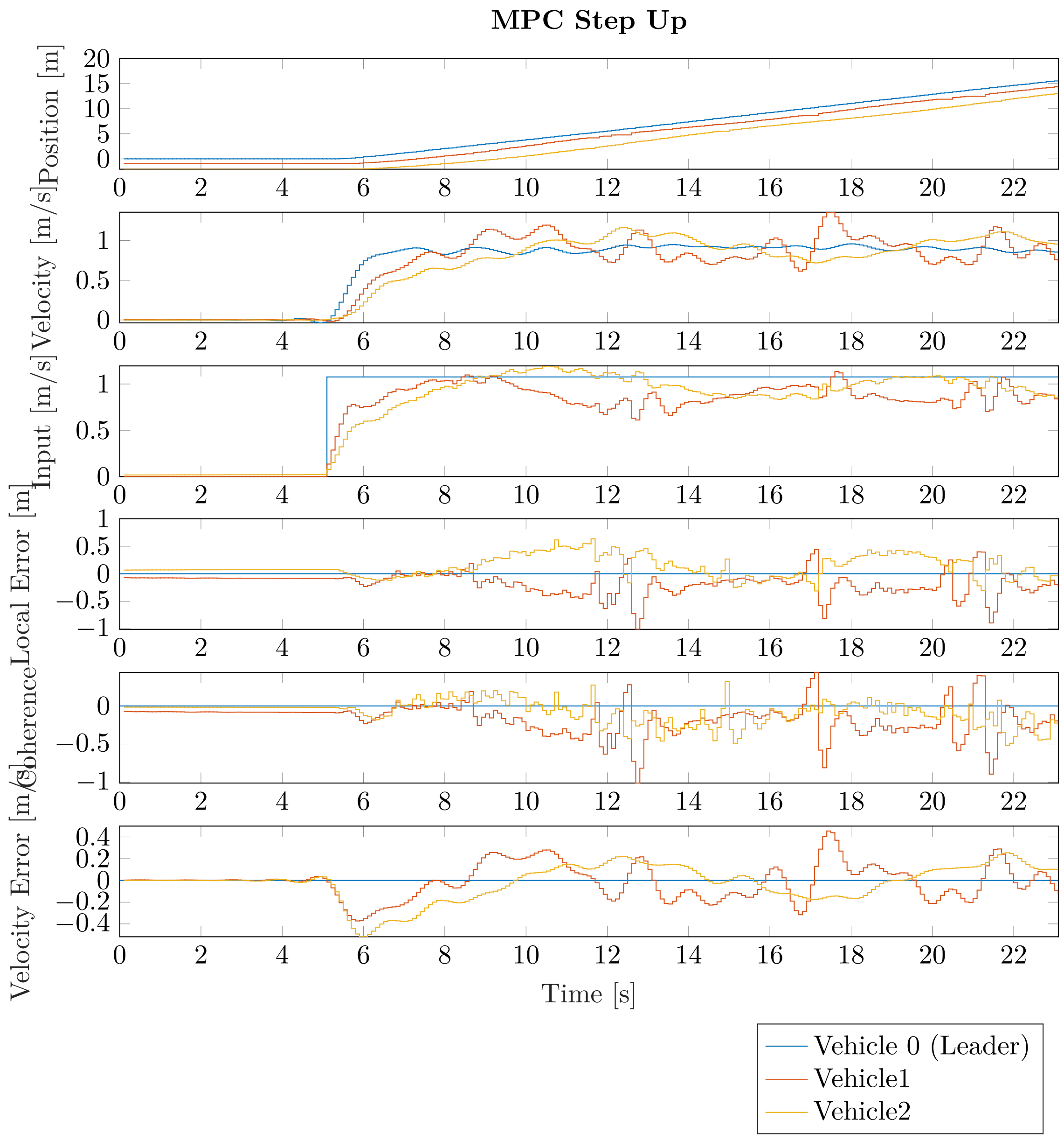


Figure 6-10: Response of an MPC controlled 3 vehicle platoon to an upward step.

controlled platoon is actually faster in stopping all vehicles than the MPC platoon seen in Figure 6-12. However, the MPC still succeeded in stopping all vehicles without collisions and looking at the local error in both cases, vehicle 2 did not come closer to collision than vehicle 1, which is encouraging for use with larger platoons.

For the ramp up scenario, it must first be noted that it was adapted from the scenario as used in simulation. This was done, because it was found that a fairer comparison could be drawn when excluding the leading section at low velocity, because at this low velocity the drive-train issues are extra pronounced, meaning the platoon does not get a stable point to settle on, thereby leading to incomparable starting points. Instead, ramps starting from zero were used.
Even so, the leading vehicle was clearly slower than the input in the PDD scenario, as seen in Figure 6-13. This led vehicle 1 to slowly decrease the distance to its predecessor and while this did not lead to a collision, it does not score well on the performance metrics. The MPC, as seen in Figure 6-14, had less of a discrepancy to deal with and did not display such behaviour. The most interesting aspect noticeable for both controllers is that the errors are largest after the ramp has ended. This can partly be explained by the fact that this is later in the experiments, so the controllers have more accumulated errors to possibly over-correct for. However, another factor is that the change in acceleration has to be compensated for.

Due to a large number of issues with the test setup, no ramp down scenario was completed by the MPC controller. Also, while the PDD controller technically did complete the scenario without collision it is clear from the results in Figure 6-15. That this case was also plagued by issues, since the velocity of the lead vehicle in no way corresponds to its input signal. Therefore this scenario is not considered for comparison at the moment.

After this qualitative analysis, it is time to look at the performance metrics to get a more objective and direct comparison between the two controllers. These metrics can be found in Table 6-2.
The minimum headway is not evaluated in the practical setup because the discrepancies as described in subsection 6-2-1 make the results too unreliable to definitively say whether a collision happened due to an issue with the setup or insufficient headway.

While, as expected from the simulation results, the MPC controller performs better in some metrics, there are a few notable exceptions.
Firstly the energy expenditure is lower for the PDD across all scenarios. This can be explained by two things. Firstly, the PDD control law is based on changes in desired acceleration. So even with large changes in the error states it takes some time for this change in acceleration to translate into a change in velocity. The MPC is not bounded in such a way, therefore allowing for larger jumps in input velocity. Secondly, during the tuning of the MPC controller relatively little significance was placed on the tiny differences in energy expenditure achieved by altering the input cost.
Secondly, the velocity error is smaller too for the PDD controller. This can largely be attributed by the increased importance given to velocity in the tuning of the PDD. This was necessary partly to prevent overshoot oscillations caused by directly steering on position errors and partly to compensate for the large reliance on the input to the predecessor that was often problematic in the lab tests. Conversely, the MPC has a large cost on position error that does not cause the same overshoot because the future is already accounted for in the optimisation problem.

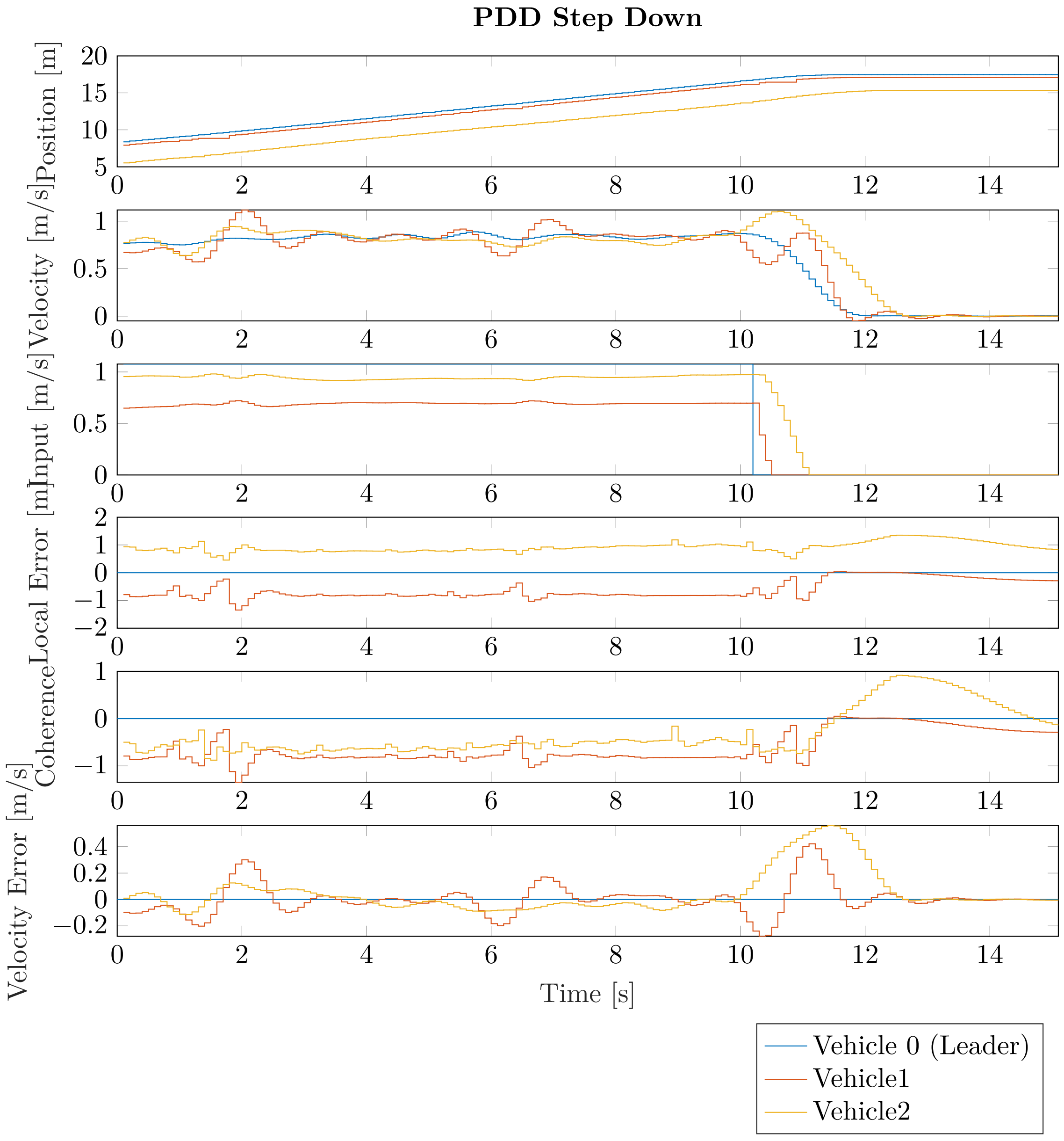


Figure 6-11: Response of an PDD controlled 3 vehicle platoon to an downward step.

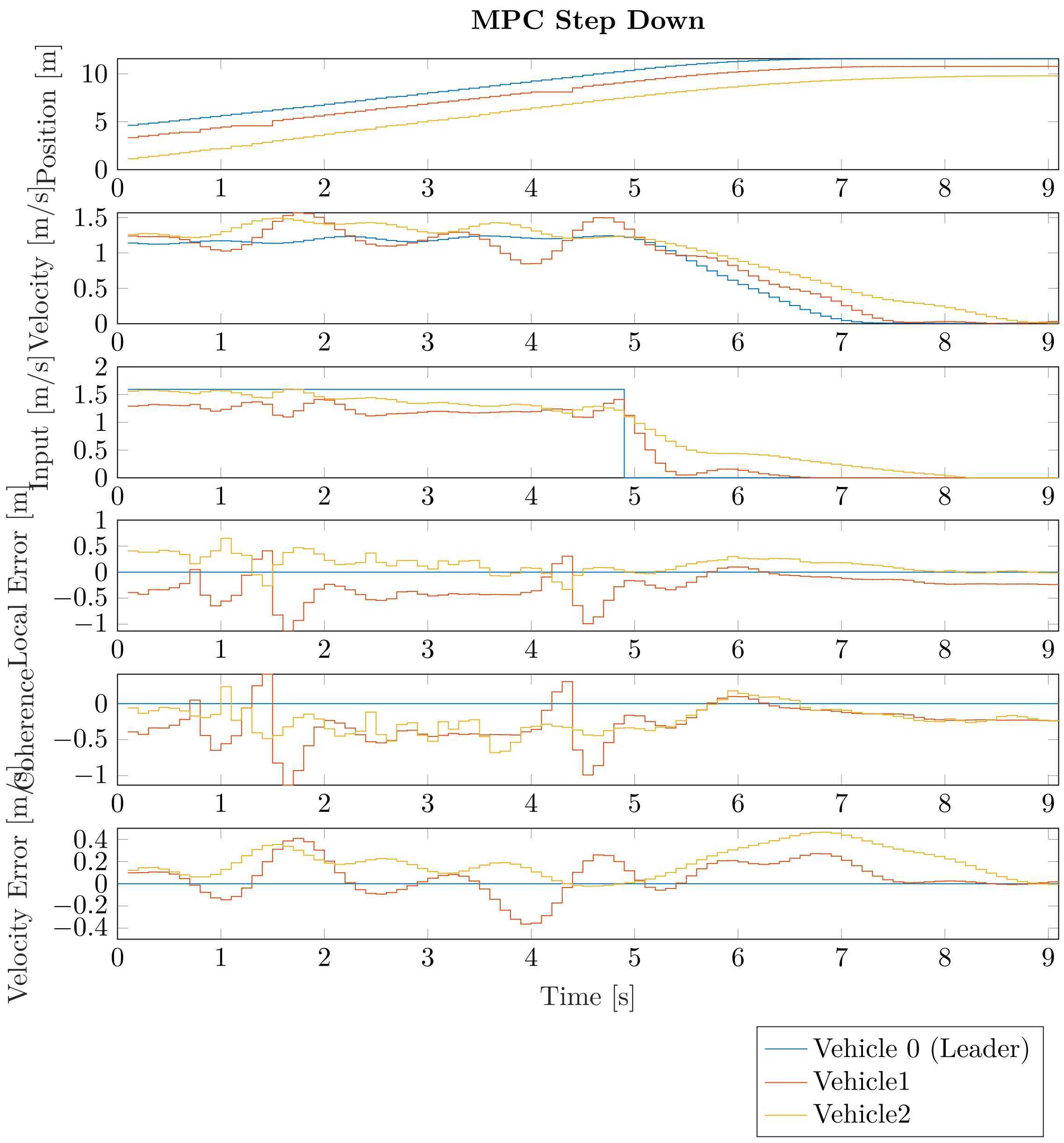


Figure 6-12: Response of an MPC controlled 3 vehicle platoon to an downward step.

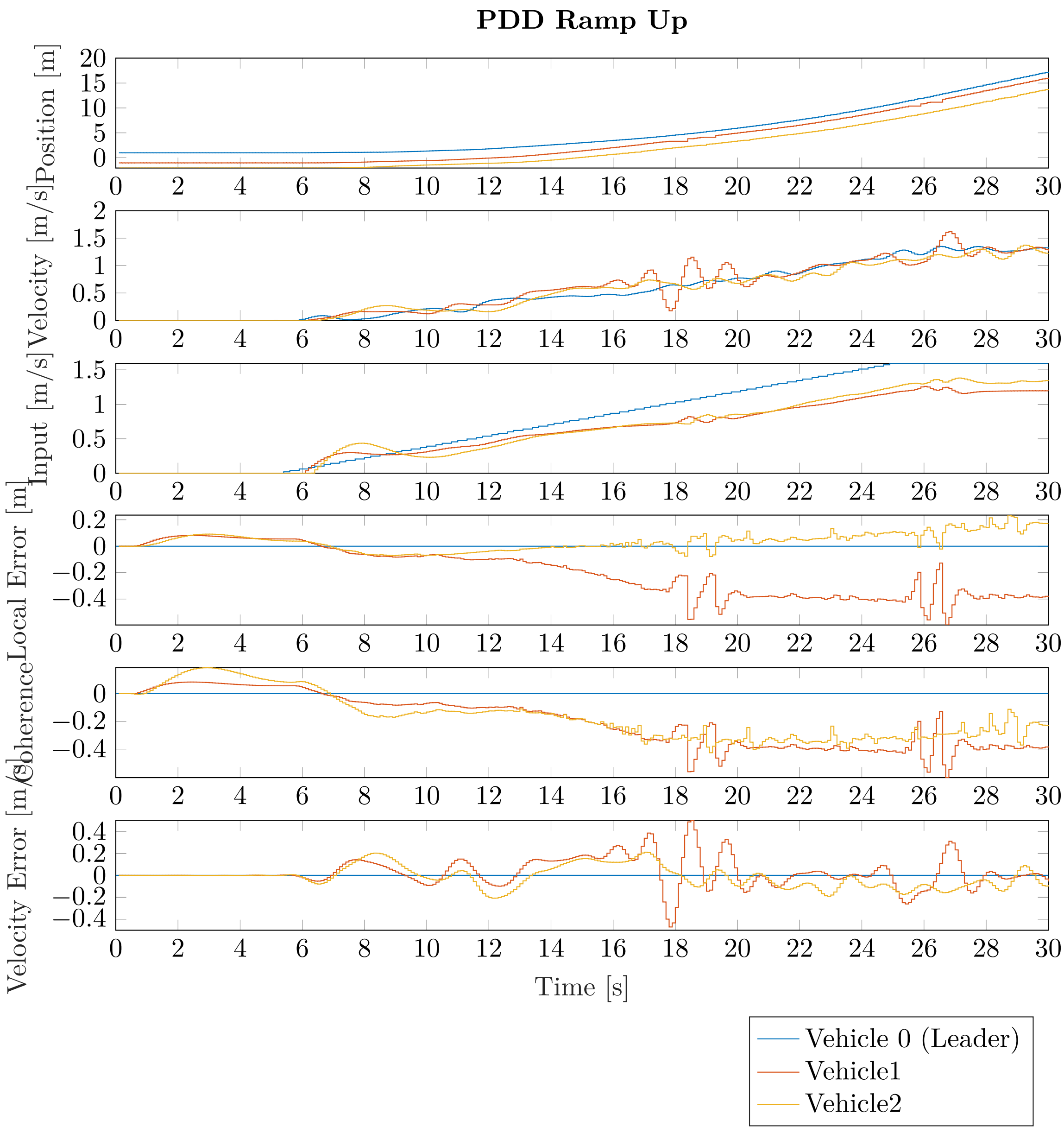


Figure 6-13: Response of a PDD controlled 3 vehicle platoon to an upward ramp.

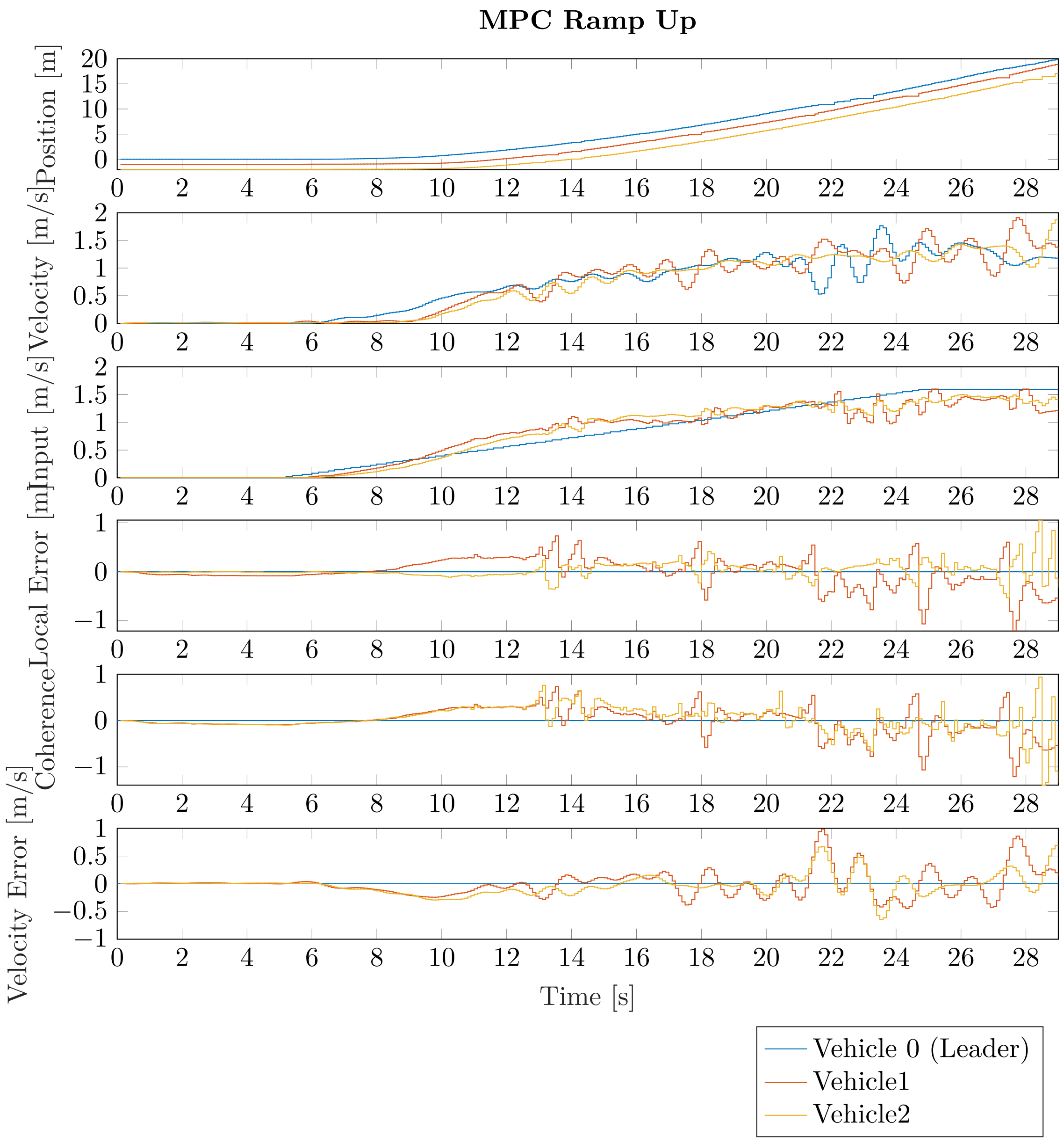


Figure 6-14: Response of an MPC controlled 3 vehicle platoon to an upward ramp.

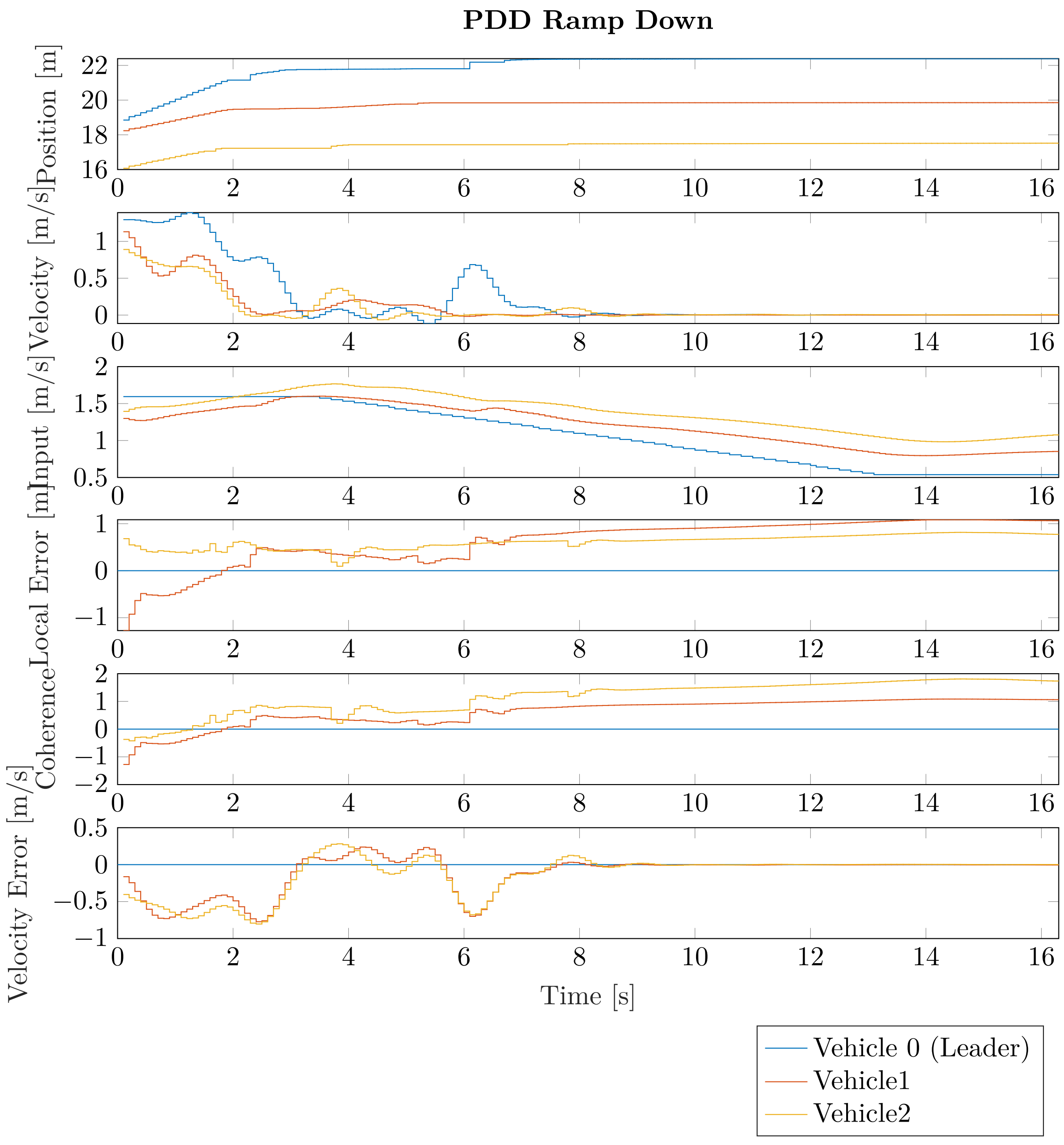


Figure 6-15: Response of an PDD controlled 3 vehicle platoon to an upward ramp.

When considering the magnitude of the performance metrics it can still be concluded that the MPC performs better in the practical setting. However, the difference is much less significant than in simulation.

		Coherence	Combined Local Error	Velocity Error	Energy Expendi-ture
Step Up	MPC	0.050	0.061	0.026	0.12
	PDD	0.28	0.11	0.020	0.05
Step Down	MPC	0.11	0.096	0.040	0.28
	PDD[1]	0.84	1.11	0.020	0.18
Ramp Up	MPC	0.080	0.054	0.045	0.18
	PDD	0.80	0.49	0.014	0.096
Ramp Down	MPC	–	–	–	–
	PDD[2]	1.15	0.51	0.084	0.096

Table 6-2: Comparison of performance metrics for the PDD and MPC controller with an experimental platoon of 3 vehicles.
[1] Scenario started after a minor collision occurred, so vehicle 0 and 1 were touching at the beginning.
[2] Completed without collision, but with abnormal behaviour of the lead vehicle.

Chapter 7

Conclusion

In this study a qualitative as well as quantitative comparison is made between the current standard for vehicle platooning, namely Proportional Double Derivative (PDD) control, and Model Predictive Control (MPC). This is done to provide an objective view on the potential of MPC to replace PDD as the standard for platoon control as well as the potential pitfalls of the method. This is done by answering the following question:

> How does Model Predictive Control (MPC) perform compared to a conventional (PDD) controller, when used to control a platoon of Radio Controlled (RC) vehicles?

With the following sub-questions:

1. Can MPC be used to control a platoon of RC vehicles?
2. How does this MPC controller perform in simulation?
3. How does the performance in the experimental setup differ from that in simulation?

The answer to the first sub-question is that it is definitely possible to use MPC to control a platoon of RC vehicles. However, the basic form of MPC used in this research is not able to reliably handle the faults and uncertainties found in the experimental setup used. These uncertainties lead to test results that are not fully repeatable and the need to perform the same test scenarios multiple times to even achieve a collision-free result. Although the same can be said for the PDD controller used for reference.

To compare the performance of the controllers, four scenarios are used to evaluate five metrics. These scenarios include a step response, an emergency stop and both an upward and downwards ramp. The performance metrics include the errors within the platoon, in terms of position with respect to the leader and position and velocity with respect to the vehicle's predecessor. Furthermore, it includes a measure for the energy used by the platoon and in

simulation the minimum required headway is evaluated.
From the simulation results, it is clear that the MPC controller vastly outperforms the PDD controller on almost all metrics. Clearly showing that in an idealised case, the optimality of MPC leads to better results.

However, the performance in the practical setup has some significant differences from simulation. Naturally, small discrepancies due to measurement noise and unmodelled behaviour are expected. However, it is found that these differences are larger than foreseen, largely due to issues with the test setup, like variable problems with the drive train making the affected vehicle stall sometimes at lower velocities. As well as faulty measurements of the Motion Capture System (MoCap) resulting in inaccurate velocity estimations.
Under these circumstances, the performance of both controllers is unreliable, but even in the test that did deliver good results the difference between both controllers is much less stark than in simulation, with the MPC controller scoring much better on the position errors but losing to the PDD controller in input rate and velocity errors.

The limitations of this research can mainly be found in two areas; the equipment and the the control strategies used.

The control strategies are purposefully chosen to be fairly standard versions of their respective classes. This is done to give a comparison that is reflective of a class of controllers rather than one specific variation on it. The drawback of this is that while these variations work well in simulation, they are not necessarily the ones that are best suited to the challenges posed by the experimental setup. The unpredictable and variable nature of the setup could probably better be dealt with using control methods made with robustness in mind. An adaptive element to the vehicle model, for example, might be used to combat the issues with the static friction.

The limitations posed by the equipment are again twofold. Firstly, there are the already discussed issues around the reliability of the vehicles and measuring equipment. This could probably be at least partly remedied by adding more sensors to the setup. An onboard velocity sensor could for example be used to validate the MoCap data and combat the oscillations caused by the Electronic Speed Controller (ESC).
Secondly, the physical tests in this research are limited to only three vehicles. This limits the value to these results as only inferences could be made to the extension into larger platoons. These larger platoons are much more sensitive to an issue like the computation speed of the control algorithm, which is a known weakness of MPC. Similarly, the existence of string stability is not very relevant for a platoon of such a small size and therefore requires further research.

Concluding, further research is necessary in two directions. Firstly, the comparison between these basic PDD and MPC controllers should be done on a larger scale and on a more stable setup. Secondly, different methods should be explored to compare different MPC variations to deal with the inherent inconsistencies of a platform like the one used in this research.

Bibliography

[1] Erle Robotics Team, "Pi0Rover: A Smart Rover with the Pi Zero,". *Available at* https://www.raspberrypi.org/forums/viewtopic.php?t=185688 (*Visited on* 2020-11-18) 2017.

[2] G. Duranton and M. A. Turner, "The Fundamental Law of Road Congestion: Evidence from US Cities," *American Economic Review*, vol. 101, pp. 2616–2652, oct 2011.

[3] P. Tientrakool, Y.-C. C. Ho, and N. F. Maxemchuk, "Highway Capacity Benefits from Using Vehicle-to-Vehicle Communication and Sensors for Collision Avoidance," in *2011 IEEE Vehicular Technology Conference (VTC Fall)*, pp. 1–5, IEEE, sep 2011.

[4] 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, pp. 429–436, dec 2006.

[5] V. Turri, B. Besselink, and K. H. Johansson, "Cooperative Look-Ahead Control for Fuel-Efficient and Safe Heavy-Duty Vehicle Platooning," *IEEE Transactions on Control Systems Technology*, vol. 25, pp. 12–28, jan 2017.

[6] M. S. Mahmoud, *Control and Estimation Methods over Communication Networks.* Cham: Springer International Publishing, 2014.

[7] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *IEEE Control Systems*, vol. 21, pp. 84–99, feb 2001.

[8] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A Survey of Recent Results in Networked Control Systems," *Proceedings of the IEEE*, vol. 95, pp. 138–162, jan 2007.

[9] Unisted States Department of Transportation, "Architecture Reference for Cooperative and Intelligent Transportation.". *Available at* http://local.iteris.com/arc-it/index.html (*Visited on* 2020-01-31)

[10] ETSI, "ETSI - Automotive Intelligent Transport | Intelligent Transport Systems (ITS).". *Available at* https://www.etsi.org/technologies/automotive-intelligent-transport (*Visited on* 2020-01-31)

[11] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design.* Nob Hill Publishing, second ed., 2017.

[12] T. Willke, P. Tientrakool, and N. Maxemchuk, "A survey of inter-vehicle communication protocols and their applications," *IEEE Communications Surveys Tutorials*, vol. 11, no. 2, pp. 3–20, 2009.

[13] S. E. Li, Y. Zheng, K. Li, Y. Wu, J. K. Hedrick, F. Gao, and H. Zhang, "Dynamical Modeling and Distributed Control of Connected and Automated Vehicles: Challenges and Opportunities," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 3, pp. 46–58, 2017.

[14] R. R. Negenborn and J. M. Maestre, "Distributed model predictive control: An overview and roadmap of future research opportunities," *IEEE Control Systems*, vol. 34, no. 4, pp. 87–97, 2014.

[15] O. Khorsand, A. Alam, and A. Gattami, "Optimal Distributed Controller Synthesis for Chain Structures - Applications to Vehicle Formations," in *Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics*, vol. 1, pp. 218–223, SciTePress - Science and and Technology Publications, 2012.

[16] H. Hao, P. Barooah, and P. G. Mehta, "Stability Margin Scaling Laws for Distributed Formation Control as a Function of Network Structure," *IEEE Transactions on Automatic Control*, vol. 56, pp. 923–929, apr 2011.

[17] Xiangheng Liu, A. Goldsmith, S. Mahal, and J. Hedrick, "Effects of communication delay on string stability in vehicle platoons," in *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings*, pp. 625–630, IEEE, 2001.

[18] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions," *IEEE Communications Surveys Tutorials*, vol. 13, no. 4, pp. 584–616, 2011.

[19] G. P. Liu, J. X. Mu, D. Rees, and S. C. Chai, "Design and stability analysis of networked control systems with random communication time delay using the modified MPC," *International Journal of Control*, vol. 79, pp. 288–297, apr 2006.

[20] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, and S. Sastry, "An LQG Optimal Linear Controller for Control Systems with Packet Losses," in *Proceedings of the 44th IEEE Conference on Decision and Control*, vol. 2005, pp. 458–463, IEEE, 2005.

[21] L. Zhang, Y. Shi, T. Chen, and B. Huang, "A new method for stabilization of networked control systems with random delays," *IEEE Transactions on Automatic Control*, vol. 50, pp. 1177–1181, aug 2005.

[22] P. Seiler and R. Sengupta, "Analysis of communication losses in vehicle control problems," in *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, vol. 2, pp. 1491–1496 vol.2, IEEE, 2001.

[23] J. Wu and T. Chen, "Design of Networked Control Systems With Packet Dropouts," *IEEE Transactions on Automatic Control*, vol. 52, pp. 1314–1319, jul 2007.

[24] P. L. Tang and C. W. de Silva, "Compensation for transmission delays in an ethernet-based control network using variable-horizon predictive control," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 4, pp. 707–718, 2006.

[25] E. Van Nunen, J. Verhaegh, E. Silvas, E. Semsar-Kazerooni, and N. van de Wouw, "Robust model predictive cooperative adaptive cruise control subject to V2V impairments," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-March, pp. 1–8, IEEE, oct 2018.

[26] P. L. Tang and C. W. De Silva, "Stability validation of a constrained model predictive networked control system with future input buffering," *International Journal of Control*, vol. 80, pp. 1954–1970, dec 2007.

[27] Y. Yang, Y. Wang, and S.-H. Yang, "Design of a networked control system with random transmission delay and uncertain process parameters," *International Journal of Systems Science*, vol. 39, pp. 1065–1074, nov 2008.

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

[29] 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, pp. 296–305, feb 2014.

[30] B. Sakhdari and N. L. Azad, "A Distributed Reference Governor Approach to Ecological Cooperative Adaptive Cruise Control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, pp. 1496–1507, may 2018.

[31] J.-Q. Wang, S. E. Li, Y. Zheng, and X.-Y. Lu, "Longitudinal collision mitigation via coordinated braking of multiple vehicles using model predictive control," *Integrated Computer-Aided Engineering*, vol. 22, pp. 171–185, feb 2015.

[32] S. Wu, S. Yu, D. He, and Y. Sun, "Model predictive control of vehicle platoon with ranged-limited sensors," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 141–145, IEEE, nov 2016.

[33] N. Chen, M. Wang, T. Alkim, and B. van Arem, "A Robust Longitudinal Control Strategy of Platoons under Model Uncertainties and Time Delays," *Journal of Advanced Transportation*, vol. 2018, pp. 1–13, 2018.

[34] J. Dold and O. Stursberg, "Distributed predictive control of communicating and platooning vehicles," in *Proceedings of the 48h IEEE Conference on Decision and Control*

(CDC) held jointly with 2009 28th Chinese Control Conference, pp. 561–566, IEEE, dec 2009.

[35] 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, pp. 620–633, mar 2012.

[36] R. Kianfar, P. Falcone, and J. Fredriksson, "A control matching model predictive control approach to string stable vehicle platooning," *Control Engineering Practice*, vol. 45, pp. 163–173, dec 2015.

[37] C. Massera Filho, M. H. Terra, and D. F. Wolf, "Safe Optimization of Highway Traffic With Robust Model Predictive Control-Based Cooperative Adaptive Cruise Control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 3193–3203, nov 2017.

[38] B. De Schutter and T. Van den Boom, *Optimization in Systems and Control*. Delft Center for Systems and Control, 2018.

[39] S. Oncu, J. Ploeg, N. van de Wouw, and H. Nijmeijer, "Cooperative adaptive cruise control: Network-aware analysis of string stability," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 1527–1537, aug 2014.

[40] E. F. Camacho and C. Bordons, *Model Predictive control*. Advanced Textbooks in Control and Signal Processing, London: Springer London, 2007.

[41] A. Alessio and A. Bemporad, "A Survey on Explicit Model Predictive Control," in *Lecture Notes in Control and Information Sciences, vol 384.*, pp. 345–369, Springer-Verlag Berlin Heidelberg, 2009.

[42] C. Zhai, Y. Liu, and F. Luo, "A Switched Control Strategy of Heterogeneous Vehicle Platoon for Multiple Objectives With State Constraints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, pp. 1883–1896, may 2019.

[43] D. Corona and B. De Schutter, "Adaptive Cruise Control for a SMART Car: A Comparison Benchmark for MPC-PWA Control Methods," *IEEE Transactions on Control Systems Technology*, vol. 16, pp. 365–372, mar 2008.

[44] Y. Zheng, S. Eben Li, J. Wang, D. Cao, and K. Li, "Stability and Scalability of Homogeneous Vehicular Platoon: Study on the Influence of Information Flow Topologies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 14–26, jan 2016.

[45] Y. Zheng, S. E. Li, K. Li, F. Borrelli, and J. K. Hedrick, "Distributed Model Predictive Control for Heterogeneous Vehicle Platoons under Unidirectional Topologies," *IEEE Transactions on Control Systems Technology*, vol. 25, pp. 899–910, may 2017.

[46] J. M. O'Kane, *A gentle introduction to ROS*. No. 2.1.3, 2016.

[47] Erle Robotics, "Erle Robotics Documentation.". *Available at* http://docs.erlerobotics.com/ (*Visited on* 2019-08-05)

[48] OptiTrack, “OptiTrack for Robotics.”. *Available at* https://optitrack.com/applications/robotics/ (*Visited on* 2020-11-18)

[49] F. Lin, M. Fardad, and M. R. Jovanovic, “Optimal Control of Vehicular Formations With Nearest Neighbor Interactions,” *IEEE Transactions on Automatic Control*, vol. 57, pp. 2203–2218, sep 2012.

[50] M. M. Rivera and L. B. Gutierrez, “Dynamic model of an aircraft propulsion system with electric motor and propeller,” in *2021 IEEE 5th Colombian Conference on Automatic Control (CCAC)*, pp. 157–162, IEEE, oct 2021.

[51] S. K. Mitra, *Digital Signal Processing: A Computer Based Approach.* New York, New York, USA: McGraw-Hill, 1998.

Glossary

List of Acronyms

ACC	Adaptive Cruise Control
C/A	Controller-to-Actuator Communication
CACC	Cooperative Adaptive Cruise Control
DCSC	Delft Center for Systems and Control
DSRC	Dedicated Short Range Communication
ESC	Electronic Speed Controller
ITS	Intelligent Transportation Systems
MAVLink	Micro Air Vehicle Communication Protocol
MoCap	Motion Capture System
MPC	Model Predictive Control
NCS	Networked Control System
PC	Personal Computer
PD	Proportional Derivative
PDD	Proportional Double Derivative
PF	Predecessor Following
PID	Proportional Integral Derivative
PWA	Piecewise Affine
RC	Radio Controlled
ROS	The Robotic Operating System
S/C	Sensor-to-Controller Communication
V2F	Vehicle-to-Field Communication
V2I	Vehicle-to-Infrastructure Communication
V2V	Vehicle-to-Vehicle Communication

List of Symbols

Δ	Sampling time
η_T	Transmission efficiency
γ	Steering Angle
μ	Coefficient of rolling resistance
Π_g	Coherence
Π_l	Combined Local Error
τ	Delay
ζ	Input Damping Coefficient
a	Acceleration
C_A	Coefficient of aerodynamic drag
c_e	Input Rate Cost
c_e	Position Error Cost
c_f	Velocity Error Cost
d	Inter-vehicular distance
e	Position Error
f	Velocity Error
g	Gravitational acceleration
h	Inter-vehicular time gap
I_N	Identity matrix of dimension N
j	Selected gear
K	P(I)D(D) Gain in Vector Form
k	Discreet time step
k_d	Derivative Gain
k_p	Proportional Gain
k_{dd}	Double Derivative Gain
L	Vehicle Length
L_{wb}	Wheelbase length
m	Mass
N	Number of vehicles in the platoon
N_c	Control Horizon
N_p	Horizon length
N_p	Prediction Horizon
p	Position
Q	Process noise covariance matrix
q_n	Process noise covariance for a single vehicle
R	Measurement noise covariance matrix
r	Trajectory Radius
r_n	Measurement noise covariance

r_w	Wheel radius
t	Time
T_e	Engine torque
v	Velocity
V_f	Terminal cost
w	Weighting Factor
w_T	Terminal Weight
$_0$	Constant
$_{des}$	Desired
$_{i,j}$	Vehicle i with respect to vehicle j
$_i$	Vehicle index relative to leader
$_l$	Linearised
$^+$	Next time step
ℓ	Stage cost
$\mathbb{U}$	Input constraint set
$\mathbb{X}$	State constraint set
$\mathbb{X}_f$	Terminal constraint set
$\mathbb{Y}$	Output constraint set
$\boldsymbol{u}_N$	Input sequence of length N

Index