Optimal control for decentralized platooning

ACHOUR AMAZOUZ

Degree project in
Automatic Control
Master Thesis,
Stockholm, Sweden 2013

XR-EE-RT 2013:005
Acknowledgements

With this thesis, I complete my electrical engineering education begun at Supélec, France and continued at KTH, Royal institute of Technology, Sweden. This work has been carried out at the Automatic Control Department, KTH in close collaboration with the pre-development department of Intelligent Transport Systems (REPI), Scania CV AB. I want to express my gratitude to Ali Madadi, Ali Reyhanogullari, Carl-Johan Elm, Magnus Almroth, Maxime Baudette and Olivier Goubet who on a daily basis dispensed, humour, wise advise and great team atmosphere. Great friends made the master thesis a great experience. I want to express my further gratitude to Jonas Mårtensson, my supervisor at the Automatic Control Department for all the advice and guidance and for giving me the opportunity to face this first motivating engineering challenge in the best conditions. I also want to thank Marriette Annergren from the Automatic Control Department, Assad Alam and Henrik Pettersson from REPI for their valuable time, irreplaceable knowledge and for the examples of professionals they give.
Abstract

The idea of autonomous vehicles and automated highway systems is no new concept to the automotive industry. The potential benefits of such a technology are numerous. The platooning approach would imply energy economy through air drag reduction, but also reduced traffic congestion and increased safety. The question of longitudinal control in a platoon configuration is central, the main concern being relative to safety. In this thesis, different classical control approaches will be compared and applied to the platooning problem. Among these approaches, one was tested in November 2012 in a demonstration which involved three teams and multiple vehicles from different Swedish universities.

Constrained optimal control comes with the prospect of increased safety and better handling of some characteristics of physical systems. The main negative impact of this constraint handling lies in its computational complexity. Numerical problems were encountered and described with the use of MPC. Proportional-Integral and Linear Quadratic controllers were retained and applied to the tracking problem in the context of vehicle platooning. These methods will be compared in a simulation environment.
# Contents

1 Introduction .................................................. 7  
   1.1 Platooning ................................................. 7  
   1.2 Context of the CoAct / Grand Cooperative Driving Challenge 9  
       1.2.1 System structure .................................. 10  
       1.2.2 String stability .................................. 11  
   1.3 Objective .................................................. 12  

2 Modelling .................................................... 15  
   2.1 Problem description ..................................... 15  
   2.2 Physical Modelling ....................................... 17  
       2.2.1 Power train dynamics .............................. 17  
       2.2.2 Longitudinal dynamics ............................ 20  
       2.2.3 Discussion on the physical modelling ............... 20  
       2.2.4 Change of modelling paradigm ..................... 21  
       2.2.5 Piece-wise affine systems ......................... 22  
   2.3 System Identification .................................... 23  
       2.3.1 Context ............................................ 23  
       2.3.2 Method choice .................................... 24  
       2.3.3 Adaptation of the models ......................... 25  

3 Controller design .............................................. 29  
   3.1 Introduction ............................................. 29  
   3.2 Linear Quadratic Regulation .............................. 29  
       3.2.1 Preliminaries ...................................... 29  
       3.2.2 Tracking problem .................................. 31  
       3.2.3 Solving the optimisation problem ................. 33  
       3.2.4 PWA approach ..................................... 37  
   3.3 Model Predictive Control ................................ 37  
       3.3.1 Introduction ...................................... 37  
       3.3.2 Optimisation problem formulation .................. 38  
       3.3.3 Tracking problem with linear plant ............... 40  
   3.4 Proportional-Integral Controller ....................... 44  
   3.5 Platooning strategies .................................... 46  
       3.5.1 Average platoon speed ............................. 46  
       3.5.2 Most restrictive control strategy ................. 47  
       3.5.3 Speed convergence strategy ....................... 48  
       3.5.4 Considerations on overtaking manoeuvres .......... 49  
       3.5.5 Platoon filtering .................................. 49
## 4 Implementation

4.1 Hardware and software resources ........................................ 51
4.2 Interface system / controller ........................................... 52
4.3 Safety measures .......................................................... 54
4.4 Model Predictive Control implementation ............................. 54
  4.4.1 Introduction of slack variables ................................... 54
  4.4.2 Computational considerations .................................... 56
  4.4.3 PWA and non-linear plants ....................................... 61

## 5 Analyses

5.1 Controllers’ Tuning ....................................................... 64
5.2 Platooning schemes ..................................................... 67
5.3 Experimental conclusion ................................................. 83

## 6 Conclusion


Chapter 1

Introduction

The work of this thesis is part of a team effort aiming at designing a system to be used in concrete experimentations. In this chapter, we introduce a system, describe the motivation of the study conducted and state the objectives of the thesis.

1.1 Platooning

The concept of heavy duty vehicle platoon can be seen as the transposition of the concept of train to the classical road network. The motivations for such a transposition are of different natures. Decreasing the fuel consumption of commercial vehicles is to an extent achieved with every new generation of combustion engines. Unfortunately, given the current state of the art in that field, the one percent of spared litre of fuel is now the consequence of years of development and optimisation. However, the air-drag reduction obtained by maintaining several vehicles at close distance in a platoon could induce fuel savings of roughly 5 to 8% according to [1], which represents an important motivation for logistics companies whose work load is continuously increasing. The running costs will be highly effected by such a result.

Moreover, the emergence of active safety systems such as brake assistance, electronic stability control and automatic cruise control can be seen as big steps towards complete automation of vehicles. Removing the possibility of human errors could help increasing security, reducing traffic congestion and represents an exciting and complex engineering challenge. The concept of platooning is of strategic importance in this prospect of autonomous vehicles.

The Grand Cooperative Driving Challenge (GCDC) was created with this outlook on autonomous vehicles. GCDC is an initiative born from the Dutch Organization for Applied Scientific Research (TNO), aiming at organising periodical contests focused on cooperative driving. As stated by Egbert-Jan Sol, chief technical officer of TNO [9], cooperative driving involves a:

"paradigm shift from a car receiving information only to a car communicating"

This paradigm shift requires formalisations and tests of communication mechanisms. The GCDC offers a framework to the different teams taking
part to the challenge. Its ambition is to involve and motivate the interest of
industrial and governmental authorities to create a synergy and to accelerate
the development of cooperative driving, as stated in [10]. In 2011 KTH,
the Royal Institute of Technology, participated in the first occurrence of the
GCDC with the team name SCOOP. This represented the first opportunity
for the team to deal with the specifications of the challenge. The SCOOP
project is a joint effort between Scania CV AB and KTH. A heavy-duty ve-
dicle appropriately equipped for the prototyping computer architecture used
in the project is made periodically available to the project for conducting
experimentations.

The next occurrence of GCDC will introduce the handling of overtaking.
An overtaking is decomposed in sub-manoeuvres such as creating a gap,
changing lane or changing the tracking target. Those modification to the
GCDC imply great changes in the platoon logic and the packet structure
for vehicle to vehicle communication (V2V). This aspect of the platooning
problem is studied and developed in another master thesis related to the
SCOOP team. In 2012, the software and hardware architecture of the system
evolved. These modifications of the system described in [4] implied the
need for an adaptation of the controller used in 2011. This presented an
opportunity to explore different controller possibilities than the one used in
2011.

Different scenarios were prepared to incrementally test every step of an
overtaking and then finally perform autonomously the entire manoeuvre.
The concept of tracked-vehicle mentioned in the previous paragraph is im-
portant in the context of platooning. All the operations discussed in this
report will be described from the controller’s point of view. We will call
ego-vehicle, the vehicle in which the controller is installed. The control
scheme is decentralised in the framework of GCDC. While every vehicle in
the platoon broadcasts information concerning its speed, acceleration and
GPS position, the controller considers the vehicle directly ahead of the ego-
vehicle as tracked-vehicle. We consider that in a platoon, it is only possible
to regulate in a consistent way the distance to one vehicle at the time. The
situation is different in the case of centralised control. The tracked-vehicle
is a particular vehicle of the platoon whose acceleration, speed and position
are directly used to compute the control. The other vehicle’s information is
used in the context of platooning strategy. In this report different platoon-
ing strategies are evaluated. We call platooning strategy a scheme allowing
to handle the data coming from vehicles in the platoon different than the
tracked-vehicle. The fundamental platooning problem is considered as being
the tracking of one vehicle. The information from the other vehicle is used
to improve the global behaviour of the platoon. The changes in speed and
acceleration happen first to the leading vehicles in the platoon and take time
to be transmitted to the tracked-vehicle.

In a plain platooning scenario where the vehicles are only driving in a
platoon formation, the tracked-vehicle should be the vehicle directly ahead
of the ego-vehicle. While taking over, the situation becomes different and
the tracked vehicle might be at the head of the platoon. Similarly, certain
situations exist where the tracking is done with regards to a vehicle placed
two positions ahead in the platoon. That is for example the case when the
vehicle just ahead of the ego-vehicle is about to leave the platoon to perform
an overtaking. These switches in the tracked-vehicle are performed by the platoon logic. The platoon logic needs to send the pertinent information to the controller depending on the situation.

This thesis is one of five theses contributing to the preparation for the first milestone towards possible future GCDC challenges. This milestone took place during November 2012 under the name CoAct 2012, as a testing event involving Linköping University, Chalmers University of Technology, and KTH. Among the five theses related to the CoAct, one is dedicated to building computer algorithms allowing handling of manoeuvres [6], the result of this work will be referred to as platoon logic in the following. A second thesis focused on the system architecture and the implementation of communication protocols between different components of the system [4]. The third one aimed at building a simulation environment allowing pretesting of control and logic algorithms [5]. The two last ones were closely related and dealt with the control algorithms themselves [7].

1.2 Context of the CoAct / Grand Cooperative Driving Challenge

The GCDC defines in its *Rules and Technology document* [10] the whole environment allowing Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications and also how the performances of the control are to be assessed and the set of tasks to be tested during the challenge.

The GCDC currently focuses on longitudinal autonomous control and requires a constant steering action from a driver also responsible for emergency brake use. The term infrastructure refers to all elements in the environment potentially communicating with the vehicles in the platoon. Those elements can for example be, traffic lights or speed limitation signs.

Two kinds of scenarios were considered in 2011. The first one corresponded to an urban set-up where the platoon was initially separated into two parts and in which the second part started at stand-still and had to catch up with the first one. The second scenario was a highway set-up. It consisted of standard platooning with a leader following a certain acceleration profile. The quality of every controller was assessed with a system of points. Points were attributed to the vehicles in the platoon depending on how quickly the operations were executed. To collect a maximum number of points, a vehicle should at all time maintain a distance as close as possible to a security distance to the vehicle just ahead.

Many constraints were formulated on the system, some of them are of great importance to the controller. The speed was for example constrained to be positive and with a maximum of 80 km/h. The upper and lower bounds for the acceleration were respectively 2 and $-4 \text{ m/s}^2$ but every vehicle had to manage to reach accelerations from $1.5$ to $-3.5 \text{ m/s}^2$. The objective of the controller is then to ensure tracking of one of several vehicles taking into account their speeds, accelerations and relative distance while observing the constraints expressed here above. According to the GCDC specifications, the relative distance in between two following vehicles would have to be bigger than $d_{safety} = d_0 + h\nu_{lead}$ where $d_0$ is the minimal distance at stand still, $h$ is the minimal required time headway with respect to its predecessor.
and \( v_{\text{leader}} \) is the speed of the tracked-vehicle.

The GCDC requires all the vehicles taking part to the challenge to broadcast dynamic vehicle information, such as position, heading, acceleration and speed. The quality of the information transmitted is assessed by a jury. The system is characterised by a certain degree of redundancy in the way we compute the different outputs. The vehicle speed is both measured by a tachometer in the vehicle and computed by the GPS component. An estimator/filter takes care of fusing the data and filtering them. The information given to the controller will not require any filtering and will be directly used to generate the control.

CoAct 2012, was the first opportunity to test overtaking manoeuvres in a platooning configuration. No performance criterion were used on the contrary to the GCDC, the demonstration being a first opportunity to test and tune the system and perform debugging.

1.2.1 System structure

The structure of the system is roughly presented in Figure 1.1. The Figure represents the system as functionally seen by the controller. It omits a great number of components not directly related to the controller. For more details on the system architecture we refer the reader to the master thesis [4].

Different hardware is symbolised by different colors. The blocks in red correspond to the ego-vehicle. The block in green represents the whole platoon. From the ego-vehicle, we mainly receive information on the current velocity and acceleration as measured by embedded sensors. Every single vehicle of the platoon broadcasts via wireless communications, its own position, speed and acceleration. The components in purple and blue represent devices added to the ego-vehicle to perform the control of our system. They do not normally belong to the Scania vehicles and correspond to easily maintainable prototyping hardware. The blue blocks correspond to the embedded computer on which the major functionalities of the system are implemented, the purple blocks represent components of our system external to the embedded computer. Dynamic information about the ego-vehicle are measured in the GPS block and routed to an estimator. The estimator is not considered as part of this work. Its task is to fuse and filter the data coming from different sources to improve the quality of the speed and acceleration measurement. The measurement given to the controller is considered as perfectly filtered and well-conditioned for the control application. The focus of this study is represented by the controller block of the Figure 1.1. The controller takes as input the state of the ego-vehicle, constituted by its speed and acceleration, and an array of platoon information, containing among others, relative position, speed and velocity of vehicles in the platoon. This platoon array is created by the Logic Component which sorts and rearranges the data from the different vehicles. The data is received and reordered based on the GPS coordinates and headings. The logic acts like a coordination system whose task is to orchestrate the good evolution of manoeuvres. The Logic also decides the values for the distances to be held between the ego-vehicle and different other vehicles of the platoon. The inter-vehicular distance is meant to evolve in the manoeuvres when, for
instance, gaps are opened or closed.

Figure 1.1: Simplified system structure. The controller receives the ego-vehicle state from the estimator and the state of other vehicle from the logic. It outputs speed and acceleration references to the Controller Area Network (CAN) communication bus of the vehicle.

The controller generates two kinds of reference signals to be sent to the vehicle through CAN transmission. These references are the input to two subsystems, cruise controller and brake management system, allowing speed and acceleration regulation.

1.2.2 String stability

The goal of the GCDC is to create a framework allowing future actors in the market of autonomous vehicles to interact with each other. It aims at defining adequate protocols and communication standards. The GCDC framework then considers the platoon from a decentralised point of view. Every vehicle in the platoon will have its own way to generate its control based on the same information. A stability problem could emerge from this. We define string stability as the capacity the vehicles have to damp out oscillations in the acceleration profile of vehicles ahead of them. The GCDC settled on the use of a criterion to ensure string stability of the platoon. The criterion formalizes the requirement on the acceleration profile of the platoon’s vehicles. In [10], the criterion is

\[ \left\| \frac{A_m(j\omega)}{A_n(j\omega)} \right\|_{H_\infty} \leq 1, \] (1.1)

where \( A_m \) and \( A_n \) are the acceleration of the lead vehicle and acceleration of the ego-vehicle respectively. The notation \( H_\infty \) is a reference to the supremum of the transfer function’s module

\[ \|H_a(j\omega)\|_{H_\infty} = \max_\omega \|H_a(j\omega)\| \leq 1, \]

where \( H_a \) is the transfer function from the acceleration of the platoon leader to the acceleration of the ego-vehicle. The frequency domain criterion (1.1) enforces the requirement of damping acceleration oscillations in the platoon. The transfer function described here corresponds to a multiple-input multiple-output (MIMO) system having as input, speed, acceleration and
position of several vehicles. This system is highly non-linear. The non-linearities come from the vehicles themselves but also from the structure of the communication system containing delays and packet losses. We can also note that this system contains all the vehicles situated in between the leading vehicle and the current ego-vehicle with their respective controllers.

The global system, as a chain of vehicles, is a complex and heterogeneous system whose description is strenuous. The determination of the transfer function $H_p$ is thus difficult. Even so, the principle of string stability remains essential, and needs to be considered at least from an experimental point of view. We will come back to this in Section 4.3.

1.3 Objective

The objective of this master thesis is twofold. The first pragmatic goal is to contribute to the CoAct event by proposing and implementing a controller adapted to the set of tasks given and assisting the other members of the team towards completion of the tasks. The second goal is to evaluate the use of Model Predictive Control (MPC) for the same application. MPC is a model-based optimal controller. The team SCOOP, to which this work is related, used a simpler proportional-integral (PI) controller in 2011 when taking part in the GCDC. The decision was taken at the beginning of this thesis, to design and compare different sorts of controllers such that MPC and Linear Quadratic (LQ) control. An improvement in the system’s performance is expected with MPC and LQ. No MPC controller was ultimately successfully designed. An adapted version of the controller used in 2011 was eventually implemented after being rewritten and made compatible to the new implementation of the system.

The difficulties encountered in the implementation come from the degree of sophistication of the MPC which has as consequence high computational complexity and numerical difficulties. Several variants of LQ controllers were however designed. They will be compared to different implementations of PI controllers in a simulation environment. These controllers were not designed in time for the CoAct 2012. MPC and LQ control are related in the sense that they formulate a similar optimisation problem to determine the control to be generated. Similar formulations of controllers based on both methods will be presented in Chapter 3. The use of MPC is motivated by the need to take into account rigid constraints on the systems speed, velocity and position as formulated by the GCDC. The MPC control needed for such a problem is known to be computationally tough. The added computational complexity brought by constraints handling became a problem in the implementation attempted during the master thesis.

In the second chapter of this thesis, we start with the analysis of the vehicle to be controlled and the determination of a model adapted to the different control schemes to be used. Three different kinds of controllers, namely PI, LQ and MPC, are described and adapted to the platooning problem. Their concrete mathematical derivation is described in the third chapter and their implementation is mentioned in what constitutes the fourth chapter. The control problem is first studied in the simplified context of a two-vehicle platoon. The other vehicles of the platoon are then included in the control scheme based on platooning strategies. Multiple platooning strategies exist.
The final chapter of this thesis is dedicated to simulation and comparison of the different control schemes described. The platooning strategies used are named average speed strategy in the case of PI controllers, speed convergence strategy and most restrictive control strategy in the case of LQ controllers.
Chapter 2

Modelling

Control rely on a mathematical representation of systems to be regulated. In this chapter, we study the physical characteristics of the vehicle to be controlled and propose a simple representation of the system.

2.1 Problem description

As a rule, before studying the design of a controller for a given application, modelling of the systems to be dealt with should be carefully studied. This is even more important in the case of MPC and LQ Control which rely internally on a given representation of the system to be controlled, as described in the Section 3.3 and Section 3.2. We expect a model to be descriptive, i.e. to grasp the main features of the system, while remaining simple. Model complexity indeed induces control complexity.

In order to efficiently analyse the system, build a controller and simulate the controlled system, three different kinds of models are necessary. The analysis needs a thorough and accurate model highlighting the different properties of the system. Control building requires a simplification of the analysis model. The control model is derived from the analysis model through linearisation and order reduction. Reducing the model complexity has for consequence a loss in the quality of the system representation, a compromise needs to be found between complexity and accuracy of the model. MPC is known to be computationally intensive. The complexity of its computation is related to the control-model’s complexity. To maximise the computational efficiency of MPC, the control-model will be taken as simple as possible. The simulation model should lie in a middle ground in between the control model and analysis model. The simulation model is used to test and judge the behaviour of the controller. In a first step the controller is tested using control oriented model. The test shows how the controller would behave in an ideal world. The simulation model used in Chapter 5 is based on physical modelling and matches the model presented in the current section.

The engine thrust could be controlled in may ways, for example by controlling the fuel injection directly or by giving a torque request to the engine management system. The vehicle provided by Scania CV AB is equipped with a cruise-control and a braking system taking as inputs a velocity and an acceleration reference, respectively. These two controllers represent an
interface to the vehicle. They handle the management of the throttle and brakes cylinder, two mechanical components of the vehicle. The implementation of these controllers is complex and unfortunately not known. The cruise control and braking system can be considered as a speed-regulation system and an acceleration-regulation system. However, here the terms of speed and acceleration references do not exactly correspond to the common conceptions of reference as encountered with classic linear systems. We consider our vehicle as an association of two subsystems acting on the same physical entity through different means, but whose simultaneous use is not allowed. The speed-regulation system is doing its best for the actual speed to converge to the value set as speed reference with a sole action on the engine throttle. The acceleration-regulation system performs the same operation relatively to an acceleration reference but only for negative values of the acceleration and only by acting on the brakes’ pressure. To some extent, these controllers conserve the way the manually operated truck would be driven. The human driver spends most of his time using the gas pedal to control his speed and now and then pushes on the brake pedal if a sudden decrease in speed is required.

It is important to notice that accelerating a truck takes much more time than retarding it through the brakes. The system resulting from the concatenation of our two subsystems will intrinsically be characterised by this fact. The controller might embrace the way the system is divided into two subsystems and model the system as a hybrid system. The switching between acceleration regulation and speed regulation introduces a discrete state to the vehicle model. This implies use of piece-wise affine (PWA) modelling and is discussed in the next Section 2.2. A second possibility is to add an interface between the platoon controller that we want to design and the two subsystems already present in the truck. This gives a layered structure to the vehicle control scheme as represented in Figure 2.1 and is the solution implemented in the actual system.

The analysis model is derived from physical modelling. It consist of a study of the mechanics of the vehicle. A thorough model of the system also considers the electronics between controllers and actuators or between the different elements of the physical implementation of the controller which
may induce packet losses, delays, quantization and re-sampling of the signals. As described in Chapter 4, the controller to be implemented is run on an embedded computer connected to the vehicle components through a CAN bus. For instance the delay induced by this communication is perceived as being close to one second when breaking.

PreScan [25], a simulation tool for the development and validation of Advanced Driver Assistance Systems and active safety systems, was studied and used to build a simulation environment by one of the team SCOOP member as described in [5]. The simulation environment built with PreScan is set to perform the testing of a real platoon settings. PreScan allows the use of sensors such as radars and modelling of wireless inter-vehicular communication. This environment allows facing more realistic situations than a purely Simulink based dynamical model. This complex and exhaustive simulation environment was used for the simulation carried out in Chapter 5.

2.2 Physical Modelling

In this section, we study a physical non-linear model of the Scania vehicle to be controlled. The model will provide us with a deeper understanding of the vehicle. The model will provide us with a deeper understanding of the system and will help designing a control-oriented model. The analysis model is also derived to make sure that the simulation model included in PreScan matches the system studied. We consider a strictly longitudinal model as lateral control is not yet considered in this work.

2.2.1 Power train dynamics

The combustion engine of a vehicle generates torque according to a specific non-linear characteristic maximum torque generated on the engine shaft / angular velocity of the engine shaft \(^1\). The characteristic depends on the thermodynamics specification of the engine and can be approximated by a polynomial. We call the polynomial \(P_{max\tau}\) and define the maximal torque produced on the engine shaft for the current engine angular velocity \(\omega_e\) as

\[\tau_{e,max} = P_{max\tau}(\omega_e).\]

An example of such a characteristic is shown in Figure 2.2 taken from [21].

Figure 2.2 represents in red the maximal torque available at a given instant depending on the current engine angular velocity. This is equivalently described by the maximum power produced as a function of the angular velocity at the engine’s shaft. The torque actually applied to the transmission chain is itself obtained as a fraction of the maximal torque for the current engine angular velocity. The fraction is a function of the throttle \(\Theta\) (accelerator pedal position). We assume here that the available torque depends proportionally on the throttle position, that is,

\[\tau_e = \Theta P_{max\tau}(\omega_e).\]

Following [1], the general structure of a vehicle’s drive-line is given in Figure 2.3. The torque produced at the engine’s shaft is transferred to the

\(^1\)or equivalently a maximum power / angular Velocity
wheels after transiting through a clutch, a gear-box, and different shafts and gears taking part in the transmission of the energy. The clutch is considered as ideal, in other words the whole energy from the engine is transferred to the gear-box when the clutch is engaged. Every other element of this transmission chain transfers the energy they receive with a certain efficiency. We denote the efficiency $\eta$, a real positive number less than 1. Let $E_w$ and $E_e$ be the energy at the wheel axle and engine shaft, respectively. Energy losses exist due to friction. They depend on the geometries and the materials of the gears. The friction decreases the energy available at the output of the drive-line,

$$E_w = \eta E_e,$$  \hspace{1cm} (2.1)

with

$$E_w = \tau_w \omega_w, \quad E_e = \tau_e \omega_e.$$  \hspace{1cm} (2.2)

The transfer of the energy is done with adaptation of the angular veloci-
ties. The angular velocity at the wheel axle $\omega_w$ is proportional to the engine shaft’s velocity $\omega_e$. The ratio is $\nu$ and is function of the gearbox ratio, that is,

$$\frac{\omega_e}{\omega_w} = \nu.$$  \hfill (2.3)

Equation 2.1, Equation 2.2 and Equation 2.3 give the relation of torques as

$$\frac{\tau_e}{\tau_w} = \frac{1}{\eta \nu}. \hfill$$

As formulated by Newton’s second law, the angular velocity of the engine as function of the engine torque $T_e$ and the resistive torque $T_r$, coming from the transmission chain, can be written as

$$J_{eq} \dot{\omega}_e = T_e - T_r, \hfill$$

where $J_{eq}$ is the equivalent inertia of the complete transmission chain (including the wheels, all the gears and axles of the chain), at the engine’s axle. The resistive torque $T_r$ consists of the resistive torque at the wheels transported to the engines shaft (using the multiplying factors $\nu$ and $\eta$ given earlier). It also includes the contribution from the brakes and the reaction to the engine’s torque exerted through adherence by the ground to the wheels. The longitudinal component of this adherence is the driving force of the vehicle, it will be denoted $F_w$. Let $r_w$ be the radius of the wheels, $\tau_b$ the breaking torque, the relations described in this paragraph can be summarized into one equation,

$$\dot{\omega}_e = tP_{max}(\omega_e) - \frac{\eta}{\nu}(\tau_b + r_w F_w). \hfill (2.4)$$

If we now consider the longitudinal speed of the vehicle, $v$, we have

$$v = r_w \omega_w = \frac{r_w}{\nu} \omega_e.$$

Let us consider the polynomial $\tilde{P}_{max}$, obtained via variable change from $P_{max}$, such that

$$\tilde{P}_{max}(v) = P_{max}(\omega_e).$$

Equation 2.4 can then be rewritten

$$J_{eq} \frac{\nu}{r_w} \dot{v} = t\tilde{P}_{max}(v) - \frac{\eta}{\nu}(\tau_b + r_w F_w).$$

The driving force of the vehicle is then

$$F_w = t \frac{\nu}{r_w \eta} \tilde{P}_{max}(v) - \tau_b - J_{eq} \frac{\nu^2}{\eta r_w^2} \dot{v}. \hfill (2.5)$$

\footnote{If a mass of inertia $J_i$ rotates at speed $\nu \omega_e$ in the transmission chain, it contributes up to the quantity $J_i \nu^2$ to the equivalent torque computed at the engine shaft. This is obtained through conservation of the chain’s kinetic energy, $J_{eq} \omega_e^2 = \sum J_i \omega_i^2$.}
2.2.2 Longitudinal dynamics

In the context of the GCDC, we consider the roads to be completely flat and we neglect any effect of gravity on the vehicle. The forces exerted to the vehicle consist of the air-drag, the longitudinal resultant of force applied by the ground on the driving wheels, and the rolling resistance. The final physical model of the vehicle is then obtained by applying Newton’s second law of motion,

\[ m \dot{v} = F_w - F_{\text{drag}} - F_{\text{roll}}, \]

where \( m \) is the mass of the vehicle, \( F_{\text{drag}} \) is the air-drag and \( F_{\text{roll}} \) the rolling resistance.

If we write \( M = m + J_{\text{eq}} \nu^2 \), we get

\[ M \dot{v} = t \nu \eta \tilde{P}_{\text{max}}(v) - \frac{1}{r_w} \tau_b - F_{\text{drag}} - F_{\text{roll}}. \]

The air-drag applied to a vehicle running at longitudinal speed \( v \) can, according to [1], be written as

\[ F_{\text{drag}} = \frac{1}{2} c_D(d) A_a \rho_a v^2, \]

where \( A_a \) is the maximum cross-sectional area of the vehicle, \( \rho_a \) is the air density, \( d \) is the distance to the vehicle precisely ahead of the ego-vehicle and \( c_D(d) \) is the air-drag coefficient. The coefficient is close to 0 if \( d \) gets small and converges to a given value \( c_D \) when \( d \) becomes larger. The rolling resistance results from friction at the contact wheels / asphalt. Still according to [1], if we let \( c_r \) be a constant rolling resistance coefficient and call the gravitational constant \( g \), we have

\[ F_{\text{roll}} = c_r mg. \]

The pressure force resulting from the pressure applied to the brakes is yet another non-linear function. We approximate the braking force with a polynomial called \( Q \), which takes as argument the brakes pressure \( p \), that is

\[ F_b = Q(p), \]

where \( F_b \) represents the braking force.

The system’s differential equation of motion becomes

\[ M \dot{v} = t \nu \frac{\eta}{r_w} \tilde{P}_{\text{max}}(v) - \frac{Q(p)}{r_w} - \frac{1}{2} c_D(d) A_a \rho_a v^2 - c_r mg. \] (2.6)

This equation summarizes the whole dynamic of the vehicle. It considers as input the parameters \( t, b \) and \( d \) representing throttle, braking and distance to the vehicle just ahead. The output is then given in terms of speed \( v \).

2.2.3 Discussion on the physical modelling

The differential Equation 2.6 takes three parameters as inputs: the brake pressure, the engine throttle and the distance to the closest vehicle in front of the ego-vehicle. The value of the first two parameters can be set freely by the controller. However, the last one depends on the behaviour of the
leading vehicle and can to some extent be considered as a perturbation. This inter-vehicular distance is important in the tracking-problem which is central in the platooning application. In a platoon, the ego-vehicle needs to keep the distance to the vehicle just in front as close as possible to a reference value \( d_{\text{ref}} \), guaranteeing safety. We assume \( d \) to be constant and equal to the desired \( d_{\text{ref}} \) to simplify the model and overlook the perturbation.

In addition, the model does not include two important components. The cruise control which was designed as a Proportional-Integral (PI) controller characterised by varying-parameter\(^3\) and the braking systems. The braking system consist of a feed-forward control of multiple braking systems. Among them are a classic friction brake and an engine break turning the engine into an air compression device by closing the exhaust pipe. The multiplicity of the braking devices make the concept of brake pressure introduced in physical modelling above a little abusive. One of the biggest problems of the physical modelling lies in the identification of the polynomials and unknown parameters. The simulation model also include a cruise-control taken from the PreScan software. This cruise-control has the task to make the vehicle speed converge to the value given by the controller. The actual vehicle works in that fashion. The model includes various parameters which need to be identified or estimated. The air drag and the polynomials characterizing the braking and thermodynamics of the engine make the differential equation non-linear. It was not possible in the scope of this thesis to proceed to the identification of the parameters. The model used in the PreScan simulation environment is based on a physical model of a vehicle. The different elements presented here are included in the simulation models proposed by PreScan. PreScan also includes a switching logic for the gear ratio. The different non-linear elements of the drive-line are included as look-up tables. The degree of complexity of the PreScan model is comparable to the non-linear model introduced in the previous paragraph.

PreScan offers a range of ready made models. The values of the parameters given in these models do not match the specifications of the truck we will be building our system on. However, with these ready made models, we have a panel of potential simulation systems which allow us to test the robustness of our controllers.

### 2.2.4 Change of modelling paradigm

In the context of this study, we decided to step back and consider the system from another point of view. The decision was made to adopt black-box system identification, trying to get a simple representation of our system. Due to the non-linearities of the physical plant and the unknown components of the system, we are not able to have an accurate physical model of the vehicle. Yet we assume that a controller based on a simple linear model will behave well on non-linear plants in the context of the experiment to be conducted in the coAct and GCDC. The physical model depicted in this chapter is only used for testing purposes. The assumption that a simple linear system will be sufficient to control the system is rough. Before applying it to the actual system, we need to apply the same method to the simulation system

---

\(^3\)The P and I parameters of the cruise control are chosen in a look up table depending on the magnitude of the current speed-discrepancy and the gear engaged.
taken from PreScan. The PreScan model based on physical modelling will be simplified using black-box system identification and the control method that we want to apply to real truck can be tested. The whole procedure is illustrated in Figure 2.4.

The testing methodology exposed in Figure 2.4 suggests to apply physical modelling to build a complex non-linear model which will constitute a fictive system. Both the actual system and the fictive system will go through black-box identification. The simulation on the fictive system are done in Simulink with PreScan and are used to validate the method. The results shown in Chapter 5 are from the simulation realized in PreScan. This environment helps tuning the controllers and studying their respective advantages and drawbacks before application to the actual system.

An important result of this master thesis is that the computational complexity of the main type of controller investigated in this work, namely MPC, did not allow use of complex plants. The linear plants of restricted order used in this work can be perceived as a disappointment. This led us to consider that the computational efficiency of MPC and the power of modern days embedded computer represent the bottleneck of such control schemes.

2.2.5 Piece-wise affine systems

As mentioned earlier, the system may be considered as an hybrid system considering the discrete nature of the gearbox ratio but also the duality of the control based sometimes on velocity regulation, with an action on the throttle, and sometimes on acceleration regulation, with an action on the brake pressure. The two actions cannot be applied at the same time which means that \( t \) and \( p \) in Equation 2.6 cannot have a non-zero value simultaneously. This removes the multiple input aspect of the system we can see described by Equation 2.6. It is in the end more the case of a switching dynamics following the two equations

\[
M \dot{v} = \begin{cases} 
\frac{\nu}{r_0} \theta \tilde{P}_a ax(v) - \frac{1}{2} c_D(d) A_a \rho_a v^2 - c_r mg, & \text{if not braking.} \\
- \frac{Q(p)}{r_o} - \frac{1}{2} c_D(d) A_a \rho_a v^2 - c_r mg, & \text{otherwise.}
\end{cases} \tag{2.7}
\]
The concept of PWA models needs to be introduced. As described in [20], a PWA system is defined by an affine state space representation similar to the classic representation of a linear system. Mathematically speaking, we partition the input-state space into multiple polyhedra in which one of the affine representations is valid

\[
\begin{align*}
    x[k+1] &= A_i x[k] + B_i u[k] + f_i, \\
    y[k] &= C_i x[k] + D_i u[k] + g_i, \\
    \text{for } (x[k], u[k]) &\in P_i,
\end{align*}
\] (2.8)

where \(x, y, u, A_i, B_i, C_i\) and \(D_i\) follow the classic nomenclature of linear systems, the subscript denotes the index of the polyhedron corresponding to the matrices considered. The parameters \(f_i\) and \(g_i\) are real entries vectors of adapted dimensions. LQ and MPC are model based control strategies. Only MPC is technically compatible with PWA representations of the system. The complexity added is then considerable. Computational complexity is a main concern in the case of MPC. In this work we decided to attempt to simplify the control method through simplification of the control model and using an explicit formulation of MPC. Such a formulation would allow use of PWA representation of the system. Unfortunately, no success was found in the experimentations as mentioned in Section 3.3.

### 2.3 System Identification

#### 2.3.1 Context

We presented in the previous section the need to take some distance from physical modelling in order to generate a simple representation of the system and bypass the need for identifying particular physical phenomena at the engine level for example. As expressed in the introduction, some of the controllers evaluated in this thesis, and notably the MPC sort, suffer from complexity problems and are often considered to be only adapted to low sampling-time applications. This fact can be tempered by simplifying the model. Besides, frequency-domain methods such as PI-control also require knowledge of the model to be tuned in a sensible way. PI-control and LQ-control do not present the inconvenience of MPC-control relatively to computational complexity. However, both these methods are also concerned by the non-linearities and the lack of information in the model still doesn’t help linearising it.

In black-box identification the system is merely perceived in terms of its transfer characteristics between inputs and outputs, without taking a look to its inner mechanisms. Our point here, is that trying to perform physical modelling of the vehicle with poor approximations of the system internals, will not be much better than trying to drive the system, around some pre-defined operational conditions and trying to reproduce, by and large, its behaviour. This approach tries to stick to tangible facts from experiments only. When using physical modelling, as we simplify the model, its consistency reduces, and as a consequence, even if the model is closer to reality from a physical point of view, behavioural differences can be bigger.
2.3.2 Method choice

System identification is a large sub-field of automatic control. Many different methods allowing black-box identification are available. The truck to be identified does not allow the use of all kinds of excitations. Giving the cruise control a speed reference that is white noise would be much more interesting than a simple step from a system identification point of view but the process of identification becomes more complicated. The complexity of MPC requires the system representation to be as simple as possible. We decided to describe the system controlled through the cruise control by step responses in the speed reference for different given set points of the vehicle speed and different magnitudes of the steps. An example of step as measured with the vehicle is given in Figure 2.5. A least mean square method was then used on this curve to identify the plant to a linear model of given order. The System Identification Toolbox [24] was used to tune the parameters of the linear plant to obtain the best fit of the step response of the identified model and the provided step response. Three different models were derived around three different set-points in speed. It was decided, as the controllers will regulate through small increment in the speed reference, to study steps of magnitude one around the set-points 5 m/s, 10 m/s and 19 m/s. The three step responses from the three linear systems, that we will call sys56, sys1011 and sys1920 in the following, are represented in Figure 2.6.

![Step response of identified models](image)

Figure 2.5: Step response in velocity measured in actual vehicle, used to identify a linear system via Least Mean Square method.

The three models given here are characterised by different rise times.
The controllers detailed in the next chapters of this report need to rely on one unique linear representation of the system. The three linear models showed here will be tested and the model offering the best performances will be retained.

### 2.3.3 Adaptation of the models

The speed regulated system is always characterised by a unit gain that is there is no steady-state error in the speed regulation. Using the method of step response identification given here above, the plant is reduced to a set of really simple linear systems supposed to capture a rough description of the system’s behaviour.

In Figure 2.1 (p.16) an interface controller was introduced to the system. This was required for two main reasons. The interface controller allows driving of the vehicle through an unique control variable $v_{ref}$, speed refer-
ence provided to the cruise control. Taking into account the braking system would involve using PWA formulation. Since braking only occurs a minor portion of the time, we decided to disregard it in the system representation. The objective of the interface controller is to apply cruise control and braking seamlessly through the use of only one control variable. The interface controller takes into consideration the fact that the engine itself can provide a retardation down to -0.5 m.s\(^{-2}\). The interface controller is placed between the platooning controller, aim of this study, and the sub-controllers already embedded in the truck. To rule the switching between the two subsystems and make the platoon controller handle only speed references, a predictive scheme was adopted and is pictured in Figure 2.7.

The interface controller is built on the following principle: It receives as input the current speed and acceleration of the vehicle and the speed reference computed by the platoon controller. It estimates the acceleration obtained after a given number of samples \(N\), by holding the same value of the speed reference. Then, the minimum value of the acceleration over those samples is compared to a threshold value \(a_{\text{switch}}\) a predefined value representing the maximum retardation obtainable with the engine.

Figure 2.7: Switching predictive scheme for braking system, where \(a_{\text{ref}}\) and \(v_{\text{ref}}\) are the acceleration and speed reference given to the braking system and cruise control, \(v\) and \(a\) are the velocity and acceleration of the ego-vehicle, \(X\) the observed state of the ego-vehicle, \(\hat{a}\) predicted acceleration, \(a_{\text{switch}}\) a predefined value representing the maximum retardation obtainable with the engine.

The interface controller is built on the following principle: It receives as input the current speed and acceleration of the vehicle and the speed reference computed by the platoon controller. It estimates the acceleration obtained after a given number of samples \(N\), by holding the same value of the speed reference. Then, the minimum value of the acceleration over those samples is compared to a threshold value \(a_{\text{switch}}\) = -0.5 m.s\(^{-2}\). This threshold marks the transition to the braking system. Figure 2.7 formalises this under the mixed shape of a flowchart and a block diagram. By overlooking the dynamic of the brakes in the model of the vehicle, we allow the system to consider braking and accelerating with the same dynamics. When the control computed leads the vehicle into accelerations low enough to require
the brakes, the interface controller makes use of them. The problem lies in the fact that the brakes have a time constant far lower than the vehicle controlled through speed regulation. This is a problem if we consider the existence of the state observer between the vehicle estimator and the platoon controller. Such an observer will be needed in case the states of the model used for control are different than the measured variables (velocity and acceleration), which is typically the case if the model order is higher than 2. We assume that a model of the vehicle can predict the future acceleration of the vehicle based on the current speed and acceleration and on the speed reference required by the controller. Another simpler way of deciding when to switch to the braking system is represented in Figure 2.8. In this paradigm, we assume that the speed reference is reached in a given time $T_{\text{settle}}$. We use the braking system only if the mean value of the acceleration over the time needed to reach the speed reference is lower than $a_{\text{switch}}$. The three controllers reported better results with different switching methods. These two switching methods contain certain parameters which values effect the behaviour of the truck regarding braking. The number of samples $N$ of the first method, and the $T_{\text{settle}}$ of the second are two parameters necessary to tune in order to guarantee smooth braking.

The observer problem In this chapter we described a way to obtain a simple model of the system supposed to help performing a decent control in the particular context of the GCDC. We assumed that we could identify a black box model of the vehicle through best fit to measured step responses with a model of chosen degree. Yet we discovered after first testing and experimentation that not any model could be used efficiently. For one thing, the variables measured in the actual vehicle are its speed and its acceleration. To identify the system with a model of order higher than 2 would then imply the use of a state observer. This proved to be quite problematic. When a switch from the braking system to the speed regulation is realized, the quick dynamics of the braking system perturbs the state observation. The perturbation of the observation resulted in deterioration on the control which became oscillatory after braking. This led us to abandon the use of plants of order greater than two. With a model of second degree, it is possible to find a realization of our model having the speed and acceleration of our vehicle as states. This then comes handy for the observation part which then is summarized to copying the values of the speed and acceleration in the state vector.
The problem is then to search appropriate matrices $A$, $B$ and $C$ with appropriate dimension, knowing that the system under this simple model should be represented by two states, two outputs and one input. The choice of the two poles $p_1$ and $p_2$ given by approximation of the step responses implies the eigenvalues of $A$ are $p_1$ and $p_2$. We want to have the speed and acceleration of the vehicle as states of the model. The speed and acceleration are also output of the system, thus

$$C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

the first output being the speed and the second, the acceleration.

One of the main characteristics of the cruise-control is to have gain one from speed reference to speed output, after application of the final value theorem on the corresponding transfer function, this corresponds to

$$[1 \ 0] (-A)^{-1}B = 1. \tag{2.9}$$

Finally, the acceleration being the derivative of the velocity, gives

$$A = \begin{pmatrix} 1 & T_s \\ a_1 & a_2 \end{pmatrix} \quad B = \begin{pmatrix} 0 \\ b \end{pmatrix}. $$

The condition on the poles are

$$a_2 = p_1 + p_2 - 1, \quad a_1 = \frac{p_1 + p_2 - p_1 p_2 - 1}{T_s}, $$

The condition on the static gain from Equation 2.9 is solved as:

$$b = \frac{-T_s b}{a_2 - T_s a_1} $$

The second order model obtained, simplifies the implementation of the controller removing the need for an observer. The vehicle model is on the other hand rudimentary. The two poles identified just give some insight on the vehicle’s behaviour. They can be seen as tuning parameters. They will influence the controllers’ quality. The simulations discussed in Chapter 5 show that the approach does not limit the quality of the control in a critical way. The control schemes can be reliable and be tuned to display good performances even based on such simple models.
Chapter 3

Controller design

This chapter describes the design of three different classes of controllers and introduces the concept of platooning strategies. Model Predictive Control and Linear Quadratic Control are first described as similarities in their formulations exist. A Proportional Integral controller, similar to a reference implementation used in 2011, is then described as an alternative to the first two optimal controller. Three platooning strategies entitled Average Platooning Speed Strategy, Most Restrictive Control Strategy and Speed Convergence Strategy; are detailed in the last part of this chapter.

3.1 Introduction

In this chapter we describe the different kinds of controllers to be evaluated. We present a formulation of these controllers adapted to the tracking problem. The tracking problem consist of the regulation of the distance between a tracked-vehicle and the ego-vehicle. This problem is a fundamental sub-problem of platooning. The main objective of this thesis was to evaluate the possibilities of using MPC, this objective was not fulfilled. Different kinds of controllers were considered to present alternatives. The MPC implementation was already proven difficult in the case of Josefin Kemppainen’s work [3], she highlighted in her report the computational complexity making the operation impossible. While we decided to try other formulations, less computationally intensive using simpler system modelling, computational problems were still unsolved. The following details the problems encountered with MPC and explores different control strategies to be used instead of MPC.

3.2 Linear Quadratic Regulation

3.2.1 Preliminaries

As explained in the Introduction, this work focuses on the control and not the estimation and filtering of the vehicle’s output. We briefly present the basic concepts of LQ control in its simplest form and in the next section, we adapt it to the tracking problem encountered with vehicle platooning.

The system to be controlled is described by a linear discrete-time state-
where $\Phi$ is the state transition matrix, $\Gamma$ is the input matrix and $C$ is the output matrix. The ego-vehicle measures its speed and acceleration which represent the state of the vehicle as identified in Section 2.3. In the context of platooning we also need to consider the inter-vehicular distance, measured via GPS position comparisons but taken as the integral of the relative speed between the two concerned vehicles. The state of the system given in Equation 3.1 will in the tracking problem be augmented by the distance tracked-vehicle/ego-vehicle. In this paragraph we only consider an abstraction of the problem in order to present how LQ control works. The problem of LQ control is expressed as finding, for every time instant $k$, the optimal control $u[k]$ minimizing an objective function $J$. In the case of infinite time-horizon the cost function is defined as

$$ J(x[\cdot], u[\cdot]) = \sum_{i=k}^{\infty} (x[i]^T Q x[i] + u[i]^T R u[i]) .$$

The optimal control is according to LQ given at the current time $k$ via solving the minimization:

$$ \min_{u[\cdot]} J(x[\cdot], u[\cdot]) \\ \text{s.t. } x[i+1] = \Phi x[i] + \Gamma u[i] \forall i > k,$$

where $x[\cdot]$ and $u[\cdot]$ represent the state and input to the system studied for all instants in the prediction horizon.

The objective function $J$ is characterized by the positive definite weight matrix $R$ on the input, and the symmetric positive semi-definite weight $Q$ on the state. The matrices effect the control performances and will be chosen in a tuning step. The solution to the optimisation problem in Equation 3.3 is then expressed as a constant linear state-feedback

$$ u[k] = -K_x x[k],$$

where the feedback gain $K_x$ is expressed as

$$ K_x = (R + \Gamma^T P \Gamma)^{-1} \Gamma^T P \Phi,$$

where $P$ is the solution to the discrete algebraic Riccati equation [8]

$$ P = Q + \Phi^T \left( P - P \Gamma \left( R + \Gamma^T P \Gamma \right)^{-1} \Gamma^T P \right) \Phi.$$

In case the time-horizon is finite, the objective function limits its scope to the prediction horizon of $N$ time-steps where $N$ is a positive integer and is augmented by a term representing the final cost (characterized by a weight on the final state $S_f$),

$$ J(x[\cdot], u[\cdot]) = x[k+N]^T S_f x[k+N] + \sum_{i=k}^{k+N-1} (x[i]^T Q x[i] + u[i]^T R u[i]) $$

$$ (3.4)$$

30
In this context the solution to the minimisation of the cost-function is given by a time-dependent state-feedback

\[ u[k] = -K_x[k]x[k], \]

where the feedback gain \( K_x \) is expressed as

\[ K_x[k] = (R + \Gamma^T P[k]\Gamma)^{-1}\Gamma^T P[k] \Phi, \]

where \( P \) is the solution to the recursive Riccati equation \[8\]

\[ P[k] = Q + \Phi^T \left( P[k+1] - P[k+1] \Gamma (R + \Gamma^T P[k+1] \Gamma)^{-1} \Gamma^T P[k+1] \right) \Phi. \]

The solution to the finite time horizon problem is more complex than for the infinite prediction-horizon case even if an explicit solution to the recursive Riccati equation can sometimes be found. In the Section 3.2.3, both the methods will be studied to picture the influence of the prediction-horizon tuning. The choice of a prediction-horizon also is a problem in case of MPC which will be presented in Section 3.3. It is also useful to note that the infinite time horizon problem gives stability guarantees not provided by the finite time horizon problem.

### 3.2.2 Tracking problem

The control problem introduced in the preliminary Section 3.2.1 only considers control of a given linear plant based on weights put on its states and input. The problem of platooning is first studied as a two-vehicle platoon only. In Section 3.5, we study how to apply to the two-vehicle problem, the LQ method presented in the previous paragraph. The state vector of the ego-vehicle, alone is \( x_{ego} \) and is expressed in the case of the simple 2-state model.

\[ x_{ego} = \begin{pmatrix} a_{ego} \\ v_{ego} \end{pmatrix}. \]

With the interface controller introduced earlier, it accepts as input \( v_{ref} \), a velocity reference given to the cruise control. We now consider two vehicles labelled ego and tracked vehicle, respectively. The control problem consists of determining an optimal control to apply to the ego-vehicle depending on the behaviour of the tracked-vehicle. In the tracking problem, we then need to extend the state vector, including the relative distance to the tracked-vehicle. In the platooning problem, at steady state, the ego-vehicle should have the same speed and acceleration as the tracked-vehicle and should maintain a desired distance to it. Let \( r \) be a reference vector containing the variables to be tracked by the ego-vehicle. A typical reference vector includes acceleration and speed of the tracked-vehicle and the desired distance to be nominally set in between the two vehicles, that is

\[ r = \begin{pmatrix} a_{lead} \\ v_{lead} \\ d_{lead} \end{pmatrix}. \]

We define \( e \) as the tracking error vector,

\[ e = \begin{pmatrix} \Delta a \\ \Delta v \\ \Delta p \end{pmatrix}, \quad (3.5) \]
with

\[ \Delta_a = a_{\text{lead}} - a_{\text{ego}}, \]
\[ \Delta_v = v_{\text{lead}} - v_{\text{ego}}, \]
\[ \epsilon_p = p_{\text{lead}} - p_{\text{ego}}, \]
\[ \Delta_p = \epsilon_p - \Delta_a. \]  

(3.6)

The reference vector \( r \) modifies the LQ formulation from the previous section. The reference vector represents the variables to be tracked. To take into account the reference, in the present section we will come back to the derivation of the LQ control based on the Pontryagin Minimum Principle \(^1\).

The cost function of the LQ controller is here defined as a penalisation of the error vector. We let \( Q \) be a diagonal weight matrix and \( R \) a scalar input weight, then

\[
J(x(.), u(.), r(.)) = \frac{1}{2}(x[k+N]^T S_f x[k+N] + \sum_{i=k}^{k+N} \epsilon[i]^T Q e[i] + u[i] R u[i])
\]

\[ = \Psi(x[k+N]) + \sum_{i=k}^{k+N} f_0(k, x(.), u(.), r(.)). \]

(3.7)

In the case of an infinite time horizon problem, the same expression is used with \( N = \infty \) and \( \Psi(x[k+N]) = 0 \).

The definition of the error vector given in Equation 3.5 includes the term \( \epsilon_p \) defined as inter-vehicular distance. The term represents the integral of the relative velocity of the tracked-vehicle and can be included in the dynamics of the ego-vehicle which was defined in Section 2.3. To do so, the reference vector is used as a measured disturbance to the system. We then introduce a matrix \( G \) which represents the impact of the reference vector on the state dynamics.

We also include an integrator in the open-loop system for tracking purposes, that is, to eliminate steady-state error in the relative position. The controlled variable which was denoted \( v_{\text{ref}} \) in Figure 2.7 and Figure 2.8, is then included to the state vector. The LQ controller will compute the derivative of the speed reference to be given to the cruise control. We denote \( u \) the derivative of \( v_{\text{ref}} \), \( u \) is the input to the extended system, and we have

\[ v_{\text{ref}}[k+1] = v_{\text{ref}}[k] + T_s u[k] \]

where \( T_s \) represents the sampling time of the controller. The relative position from the tracked-vehicle to the ego-vehicle is denoted \( \epsilon_p \) and is given as

\[ \epsilon_p[i+1] = p_{\text{lead}}[i + 1] - p_{\text{ego}}[i + 1] = \epsilon_p[i] + T_s (v_{\text{lead}}[i] - v_{\text{ego}}[i]), \]

where \( p_{\text{lead}} \) and \( p_{\text{ego}} \) are the position of the tracked-vehicle and ego-vehicle, respectively and \( v_{\text{lead}} \) and \( v_{\text{ego}} \) their respective velocities.

Let \( \Phi, \Gamma \) be the state transition matrix and the input matrix of the extended system, respectively and \( G \) is the matrix representing the influence of the reference vector on the state dynamics. The extended dynamics of the ego-vehicle in tracking is then given as

\[ x[i + 1] = \Phi x[i] + \Gamma u[i] + G r[k], \]

(3.8)

\(^1\)http://en.wikipedia.org/wiki/Pontryagin’s_minimum_principle
where $x$ is the state vector of the extended system. The state and different matrices of the state space representation are given as

$$x = \begin{pmatrix} x_{ego} \\ v_{ref} \\ c_{p} \end{pmatrix}, \quad \Phi = \begin{pmatrix} \Phi_{ego} & \Gamma_{ego} & 0 \\ 0 & 1 & 0 \\ -C_{v} T_{s} & 0 & 1 \end{pmatrix}, \quad \Gamma = \begin{pmatrix} 0 \\ T_{s} \\ 0 \end{pmatrix}, \quad G = \begin{pmatrix} 0 \\ 0 \\ T_{s} C_{v}^{lead} \end{pmatrix},$$

where the output vectors $C_{v}, C_{a}, C_{v}^{lead}, C_{d}^{lead}$ and $C_{a}^{lead}$ are defined as

$$C_{v}^{lead} = (0 \ 1 \ 0), \quad C_{a}^{lead} = (1 \ 0 \ 0), \quad C_{d}^{lead} = (0 \ 0 \ 1), \quad C_{a} = (1 \ 0 \ 0), \quad C_{v} = (0 \ 1 \ 0),$$

which translates the relations

$$C_{a}^{lead} r = a_{lead}, \quad C_{v}^{lead} r = v_{lead}, \quad C_{d}^{lead} r = d_{lead}, \quad C_{a} x_{ego} = a_{ego}, \quad C_{v} x_{ego} = v_{ego}.$$

The tracking error vector is then written as

$$e[i] = C_{e} x[i] + M r[i],$$

which is expressed in correspondence with Equation 3.6:

$$C_{e} = \begin{pmatrix} -C_{a} & 0 & 0 \\ -C_{v} & 0 & 0 \\ -h C_{v} & 0 & 1 \end{pmatrix}, \quad M = \begin{pmatrix} C_{a}^{lead} \\ C_{v}^{lead} \\ -C_{d}^{lead} \end{pmatrix} \quad (3.9)$$

### 3.2.3 Solving the optimisation problem

The optimal control problem is

$$\min_{u[.]} \Psi(x[k + N]) + \sum_{i=k}^{k+N} f_{0}(k, x[.], u[.], r[.]),$$

s.t. $x[i + 1] = f(i, u[.], x[.], r[.]) \quad (3.10)$

where the notation $\chi[.]$ refers to all sample of the variable $\chi$ in the prediction horizon. Problem 3.10 is solved according to Pontryagin Minimum Principle (PMP) [18]. PMP states that if the optimisation problem 3.10 admits the solution $\{u_{i}^{*}\}_{i=k}^{k+N-1}$, and if we let $\{x_{i}^{*}\}_{i=k}^{k+N}$ be the corresponding state trajectory, there exists an adjoin variable, the Lagrange multiplier $\{\lambda_{i}^{*}\}_{i=k}^{k+N}$, such that,

$$\frac{\partial H}{\partial u}(i, x_{i}^{*}, u_{i}^{*}, r_{i}, \lambda_{i+1}) = 0, \quad i = k, \ldots, k + N - 1, \quad (3.11)$$

$$\lambda_{i} = \frac{\partial H}{\partial x}(i, x_{i}^{*}, u_{i}^{*}, r_{i}, \lambda_{i+1}), \quad i = k, \ldots, k + N - 1, \quad (3.12)$$

$$\lambda_{N} = \frac{\partial \Psi}{\partial x}(x^{*}[k + N]), \quad (3.13)$$

where $H$ is the Hamiltonian of the problem and it is defined as

$$H(i, x^{*}[i], u^{*}[i], r[i], \lambda[i + 1]) = f_{0}(i, x, u, r) + \lambda^{T} f(i, x, u, r).$$

The Hamiltonian here admits the expression

$$H(i, .) = \frac{1}{2} \{x_{i}^{T} C_{e}^{T} Q C_{e} x_{i} + 2 r_{i}^{T} M^{T} Q C_{e} x_{i} + r_{i}^{T} M^{T} Q M r_{i} + u_{i}^{T} R u_{i}\} + \lambda_{i+1}^{T} (\Phi x_{i} + \Gamma u_{i} + G r_{i}).$$
Equation 3.11 yields
\[ Ru_i^* + \Gamma^T \lambda_{i+1} = 0, \] (3.14)
and Equation 3.12 gives
\[ \lambda_i = C_e^T QC_e x_i + C_e^T Q M r_i + \Phi^T \lambda_{i+1}. \] (3.15)

Let \( \bar{Q} = C_e^T QC_e \), Equations 3.10, 3.14, 3.15 represent the Euler-Lagrange equations [18] of the problem, which can be summarized as
\[ \begin{pmatrix} \lambda_i \\ x_{i+1} \end{pmatrix} = \begin{pmatrix} \bar{Q} & 0 & \Phi^T \\ 0 & R & \Gamma \\ \Phi & \Gamma & 0 \end{pmatrix} \begin{pmatrix} x_i \\ u_i \\ \lambda_{i+1} \end{pmatrix} + \begin{pmatrix} C_e^T Q M \\ 0 \\ G \end{pmatrix} r_i. \] (3.16)

Let the Lagrange multiplier be decomposed as a term function of the state \( x_i \) and a component independent of \( x_i \), i.e.,
\[ \lambda_i = P_i x_i + g_i, \] (3.17)
where \( P_i \) represents a time varying matrix and \( g_i \) a scalar term independent of the state \( x_i \) at any time-sample \( i \).

Replacing \( \lambda_{i+1} \) and \( r_i \) following Equation 3.8. Via replacement of \( \lambda[i+1] \), Equation 3.16 becomes
\[ \begin{pmatrix} \lambda_i \\ 0 \end{pmatrix} = \begin{pmatrix} Z_{xx} & Z_{xu} \\ Z_{xu}^T & Z_{uu} \end{pmatrix} \begin{pmatrix} x_i \\ u_i \end{pmatrix} + \begin{pmatrix} \phi_{i+1} \\ \Gamma \end{pmatrix} g_i + \begin{pmatrix} Z_{xr} \\ Z_{ur} \end{pmatrix} r_i, \] (3.18)
with
\[ \begin{pmatrix} Z_{xx} & Z_{xu} \\ Z_{xu}^T & Z_{uu} \end{pmatrix} = \begin{pmatrix} \bar{Q} + \phi_{i+1} \phi_{i+1}^T \\ \phi_{i+1}^T \Gamma \end{pmatrix}, \] (3.19)
and
\[ \begin{pmatrix} Z_{xr} \\ Z_{ur} \end{pmatrix} = \begin{pmatrix} C_e^T Q M + \phi_{i+1}^T \Gamma P_{i+1} G \\ \Gamma^T P_{i+1} G \end{pmatrix}, \] (3.20)

Combining the two equations of Equation 3.18 to replace the term \( u[i] \) yields
\[ \lambda_i = (Z_{xx} - K Z_{xu}) x_i + (\phi_{i+1}^T - \Gamma K^T) g_i + (Z_{xr} - K Z_{ur}) r_i, \] (3.21)
where
\[ K = K[i] = Z_{xx} Z_{uu}^{-1}. \]

Identifying the decomposition of \( \lambda_i \) from Equation 3.21 to the one in Equation 3.17, and replacing the terms defined in Equation 3.19 we get to
\[ P_i = \bar{Q} + \phi_{i+1}^T \phi_{i+1} - \phi_{i+1}^T \Gamma (R + \Gamma^T P_{i+1} \Gamma)^{-1} \Gamma^T P_{i+1} G, \] (3.22)
and
\[ g_i = (\phi - \Gamma K^T) g_{i+1} + (Z_{xr} - K Z_{ur}) r_i. \] (3.23)

The Equation 3.22 can be recognised as the recursive Riccati difference equation.

\[^2Z_{xx}, Z_{xu}, Z_{uu}, Z_{xr} \text{ and } Z_{ur} \text{ are time dependent, which is not included in the already heavy notation.}\]
**Infinite time horizon problem**  In the case of an infinite time-horizon, the focus is put on the steady state of these equations, $P_i$ is now independent of the time sample $i$ and is the solution to the discrete time algebraic equation

$$P = \bar{Q} + \Phi^T P \Phi - \Phi^T P \Gamma (R + \Gamma^T P \Gamma)^{-1} \Gamma^T P G.$$  \hspace{1cm} (3.24)

Consequently, at steady state, the vectorial term $g$ from Equation 3.17 becomes a constant vector expressed as a function of the reference vector $r$, that is,

$$g(r) = (I - \Phi^T + K \Gamma^T))^{-1}(Z_{xr} - KZ_{ur}) r.$$  \hspace{1cm} (3.25)

Besides, Equation 3.17 leads to

$$\lambda_{k+1} = P (\Phi x[k] + \Gamma + Gr[k]) + \Gamma,$$

which, along with Equation 3.14 once $u[k]$ is isolated, yields

$$u[k] = -(R + \Gamma^T P \Gamma)^{-1}(\Gamma^T P \Phi x[k] + \Gamma^T (PG + \Gamma)r[k]).$$

The control is then derived as

$$u[k] = K_x x[k] + K_r r[k],$$

where

$$K_x = -(R + \Gamma^T P \Gamma)^{-1}\Gamma^T P \Phi, \hspace{1cm} K_r = -(R + \Gamma^T P \Gamma)^{-1}\Gamma^T (PG + \Gamma).$$

**Finite time horizon problem**  In the case of a finite time horizon $N$, the solution to the Riccati equation has to be computed recursively from the boundary condition formulated in Equation 3.13. The boundary condition translates here to

$$P[N] = S_f, \hspace{1cm} g[N] = 0.$$  

Using the recursion of Equation 3.22, the matrix $P[i]$ corresponding to the value of $P$ at a given instant $i$ in the prediction horizon is computed for all time instant in the prediction horizon until the current instant $k$ is reached. We proceed exactly the same way with regards to the term $g$ obtained with the recurrence from Equation 3.23. This time around, an assumption must be done on the values taken by the reference signal. We will consider the reference kept constant over the prediction horizon to match what is done in the case of infinite time horizon. The optimal control is then given as

$$u[k] = -R^{-1}\Gamma^T \lambda_{k+1},$$

with

$$\lambda_{k+1} = P_{k+1}(\Phi x[k] + \Gamma u[k] + Gr[k]) + g[k + 1],$$

i.e.

$$u[k] = -(R + \Gamma^T P_{k+1} \Gamma)^{-1}(\Gamma^T P_{k+1}(\Phi x[k] + Gr[k]) + g[k + 1]).$$

In the Figures 3.1 and 3.2, the structure of the two different sorts of controllers are depicted. The main difference resides in the computation of the solution of the Riccati equation and the feedback gains which is done
beforehand in the infinite time-horizon case and on-line in the finite time-horizon case, this is symbolised in the pictures by the black color for on-line computations and red for off-line computation.

In infinite time horizon, the dynamics of the reference vector’s components are neglected and the reference vector is assumed constant. The opposite would seem physically more accurate but considering that the optimisation is computed over an infinite time horizon, if at time $k$, the acceleration of the vehicle is measured as positive, considering that the velocity would evolve as an integral of the acceleration would lead to considering an actually short acceleration as lasting for several seconds. However, in case of finite-time horizon LQ, the assumption of constant acceleration will not have the same repercussion as we limit the scope of the integration in terms of duration. This will be elaborated in the case of the Speed Convergence Platooning Strategy.
3.2.4 PWA approach

LQ control only consider the control of linear plants. MPC can, to some extent, be seen as a generalisation of LQ which accepts different sorts of cost functions and more complex system dynamics. One of the system dynamics usable in MPC corresponds to the class of piece-wise affine systems. In MPC, we envisage the use of complex solvers based on computational geometry. This, allows to take into consideration the switch between different sub-dynamics of the plant inside the prediction and optimisation computations. This is not conceivable in the case of LQ control. However, if multiple LQ controller are computed based on different plant models and if we associate every single of these plant models to a given set-point of the vehicle speed, switching between the different controllers as the speed evolves might help capture a better approximation of the model behaviour.

However, it has to be understood that switching between different controllers as described above may not be harmless when considering string-stability of the platoon and would require a closer study of the system’s behaviour when switching between two sub-models.

3.3 Model Predictive Control

3.3.1 Introduction

In MPC much like in LQ control, the control problem is formulated as an optimisation problem (minimisation of an objective function) constrained by the dynamic of the system. However, MPC adds to the optimisation problem constraints on the states and input.

A general formulation is given as: determine for the current time instant $k$, the control solution $u[k]$ to

$$
\min_{u[k]} \sum_{i=k}^{i=k+N} j(x[i], u[i]) \quad \text{s.t.} \begin{cases} 
0 \leq g(x[i], u[i]) & \forall i \in [k, k + N], \\
x[i + 1] = F(x[i], u[i]) & \forall i \in [k, k + N],
\end{cases}
$$

(3.26)

where $N$ is the prediction horizon length of the problem, positive integer, $x$ and $u$ are the state and input of the system. Multiple MPC problems can be formulated from Equation 3.26. The function $F$, describing the system’s dynamics can be linear, piece-wise affine or even non-linear. The objective function itself, here denoted $j$, can be chosen linear, quadratic or even non-linear. The same applies to the constraints represented by $g$, which describe admissible value sets for the state and input. The way the solution is computed depends on the choice of $F$, $j$ and $g$. The classical formulation of MPC, limiting to some extent the complexity of the problem, concerns a quadratic cost and linear dynamics and constraints. The general formulation of Equation 3.26 then translates to
\[
\begin{align*}
\min_{u[k]} & \quad x[k+N]^T S_f x[k+N] + \sum_{i=k}^{k+N-1} (x[i]^T Q x[i] + u[i]^T R u[i]) \\
s.t. & \left\{ \begin{array}{l}
   x[i+1] = \Phi x[i] + \Gamma u[i], \quad \forall i \in [k, k+N-1], \\
   0 \leq K \begin{bmatrix}
   x[k+1] \\
   \vdots \\
   x[k+N]
\end{bmatrix} + L \begin{bmatrix}
   u[k] \\
   \vdots \\
   u[k+N-1]
\end{bmatrix} + g,
\end{array} \right.
\end{align*}
\] (3.27)

where \( K \), \( L \) and \( g \) are matrices of appropriate sizes. They define any number of linear constraints concerning all states and inputs in the prediction horizon. The so-called constraints, restrict the input and state values into polytopes. In our case, the constraints will only be formulated on the output of the system (maximum and minimum speeds for example).

The LQ problems presented in Paragraph 3.2.1 and Paragraph 3.2.3 accept a static explicit solution when formulated with infinite time-horizon. In the case of MPC, the constraints change the nature of the optimisation and modify the application of PMP which do not allow the same explicit solution. In MPC, an optimisation problem is solved for every sample. In the case of a quadratic cost and a linear model and constraints, a quadratic-programming solver is used. Methods exist allowing determination of an explicit control rule. The explicit control is obtained after computing the solution to the quadratic program for every possible state of the plant. Explicit MPC is characterised by a very heavy computation to be done offline. Solving the quadratic program introduced in Equation 3.27 for all possible states calls for the use of multi-parametric quadratic-programming (MP-QP).

### 3.3.2 Optimisation problem formulation

**On-line formulation** A quadratic programming problem is formulated as

\[
\begin{align*}
\min_{u} & \quad \frac{1}{2} u^T H u + c^T u, \\
\text{s.t.} & \quad E u \leq d,
\end{align*}
\] (3.28)

where \( u \) is a vector of unknowns, parameters of the optimization. To formulate the MPC problem as a quadratic program, we note that for any integer \( l \in \{k, k+N\} \), the state of the plant can be written as

\[
x[k+l] = \Phi^l x[k] + \sum_{i=1}^{l-1} \Phi^{i-1} \Gamma u[i + k - 1].
\] (3.29)

The Problem 3.27 can be reshaped to fit the form of a classic quadratic program as in Problem 3.28. To do so, we define the following matrices

\[
\hat{Q} = \begin{pmatrix}
   Q & 0 & \cdots & 0 & 0 \\
   0 & Q & \cdots & \vdots & \vdots \\
   \vdots & \vdots & \ddots & \vdots & \vdots \\
   0 & \cdots & 0 & Q & 0 \\
   0 & \cdots & 0 & \cdots & S_f
\end{pmatrix}, \quad \hat{R} = \begin{pmatrix}
   R & 0 & \cdots & 0 \\
   0 & R & \cdots & \vdots \\
   \vdots & \vdots & \ddots & \vdots \\
   0 & \cdots & 0 & R
\end{pmatrix},
\]
\[ x = \begin{pmatrix} x[k] \\ \vdots \\ x[k+N] \end{pmatrix}, \quad u = \begin{pmatrix} u[k] \\ \vdots \\ u[k+N-1] \end{pmatrix}, \]

\[ \hat{A} = \begin{pmatrix} I \\ \Phi \\ \vdots \\ \Phi^N \end{pmatrix}, \quad \hat{B} = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ \Gamma & 0 & \cdots & 0 \\ \Phi \Gamma & \Gamma & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \Phi^{N-1} \Gamma & \Phi^{N-2} \Gamma & \cdots & \Gamma \end{pmatrix}, \]

\[ H = \hat{B}^T \hat{Q} \hat{B} + \hat{R} \quad F = \hat{A}^T \hat{Q} \hat{B}. \]

Thus, we can write Equation 3.29 as

\[ x = \hat{A} x[k] + \hat{B} u, \]

the cost function from Equation 3.27 can then be written as

\[ x[k+N]^T S_f x[k+N] + \sum_{i=k}^{k+N-1} (x[i]^T Q x[i] + u[i]^T R u[i]) = x^T \hat{Q} x + u^T \hat{R} u. \]

We can now identify the formulation of the MPC problem in Equation 3.27 as the standard formulation of a quadratic program in Equation 3.28 with

\[ H = \hat{B}^T \hat{Q} \hat{B} + \hat{R}, \quad c = 2x[k]^T F; \]

\[ E = K \hat{B} + L, \quad d = K \hat{A} x[k] + g. \]

A solver such as quadprog in Matlab can then be used to solve the optimisation problem at each time-instant.

**Explicit formulation** The computational complexity of MPC comes from the necessity to solve an optimisation problem at each iteration. The computation time of MPC depends on the size and condition number of the Hessian matrix \( H \). At every single sampling time \( k \), the problem to be solved depends on the value of the current state \( x[k] \). To make the process of computing quicker, using MP-QP (mentioned in 3.3.1), the solution of every quadratic program for every single possible value of \( x[k] \) is computed. The high complexity of the real-time computation in on-line MPC, are traded against an extremely high off-line complexity in the case of explicit MPC. An algorithm for MP-QP resolution is described in [14]. The aim of the algorithm is to precompute all optimal controls beforehand and build a partition of the state-space. Each partition is a convex subspace over which an affine state feedback is available to solve the quadratic program. Once the partition and feedbacks are computed, the remaining task to be performed on-line consists of finding the affine feedback corresponding to the current state \( x[k] \) of the plant. The piece-wise affine feedback is then given as

\[ u^*(x) = K_i x + \Gamma_i \text{ for } x \in P_i, \]
where $P_i$ is the i-th polytope of the parameter space, and $K_i$ and $I_i$ are parameters of the affine state feedback over this polytope. The methods involved in MP-QP rely much on computational geometry. The first step of the algorithm, as described in [14], consists of solving a linear programming problem to locate a feasible state to start from in the parameter space. A quadratic program is then solved using an active-set method and the active set computed in this problem is evaluated to generate critical regions of the parameter space. These critical regions are then used to determine the polytopes over which MPC is an affine feedback. The reader interested in MP-QP will find exhaustive descriptions in [14], [13] and [15]. The methods used to determine the state feedback corresponding to a given state $x[k]$ also make use of computational geometry. Luckily toolboxes were created to handle MP-QP and the computational geometry mentioned here. The use of one of these toolboxes is presented in Chapter 4.

Figure 3.3: Principle description of On-line MPC. The ego-vehicle state $x_{ego}$ and the tracked-vehicle state $x_{leader}$ are merged along with the desired inter-vehicular distance $d_{ref}$ into an extended state vector $\hat{X}$. The controller computes $v_{ref}$, speed reference, to be provided to the cruise-control.

3.3.3 Tracking problem with linear plant

As stated in the previous section, MPC based on a linear model and a quadratic objective function share the same global structure as LQ. The formalisation of the tracking applied to the platooning problem studied here then resembles the structure exposed in Section 3.2.2. However, contrary to the LQ case, the toolbox used in this thesis did not allow any other formulation than the canonical one given in Equation 3.27. The reference vector and the state vector used in Section 3.2 have to be merged into the state vector of an extended system. In LQ we used the reference as an external signal influencing the control, we need to incorporate that part into the state vector to solve it with the tool given to us. The results regarding optimality should be the same, the same objective function and as a consequence, the same cost criterion will be used in the exact same way.

Plant dynamics We still consider a reference-vehicle to be tracked by our ego-vehicle. The reference-vehicle sends us a reference vector that we
Figure 3.4: Principle description of explicit MPC. The ego-vehicle state $x_{ego}$ and the tracked-vehicle state $x_{leader}$ are merged along with the desired inter-vehicular distance $d_{ref}$ into an extended state vector $\hat{X}$. The controller computes $v_{ref}$ speed reference to be provided to the cruise-control.

consider to be of good quality (filtered and ready to use). The reference vector is unchanged from the LQ formulation and remains

$$
\mathbf{r} = \begin{pmatrix} a_{lead} \\ v_{lead} \\ d_{lead} \end{pmatrix}.
$$

The error vector $\mathbf{e}$ that we want to minimize also remains unchanged, that is,

$$
\mathbf{e} = \begin{pmatrix} \Delta_a \\ \Delta_v \\ \Delta_p \end{pmatrix} \quad \text{with} \quad \begin{cases} 
\Delta_a = a_{lead} - a_{ego} \\
\Delta_v = v_{lead} - v_{ego} \\
\Delta_p = \epsilon_p - d_{lead} - h v_{ego}
\end{cases},
$$

and the cost function is written

$$
J(x(\cdot), u(\cdot), r(\cdot)) = \frac{1}{2} \sum_{i=0}^{N} e[i]^T Q e[i] + u[i] R u[i].
$$

The differences between LQ and MPC appear with the state vector changing from

$$
\mathbf{x}_{LQ} = \begin{pmatrix} x_{ego} \\ v_{ref} \\ \epsilon_p \end{pmatrix},
$$

\footnote{as a reminder the $v_{ref}$ in the state vector is the speed reference given to the cruise control, it is the input to our system which is added to the state vector to include an integrator to the closed-loop, the input $u$ generated by our controller corresponds to the derivative of the speed reference given to the cruise control. Besides, $x_{ego}$ represent the two-dimensional state vector of the model built based on identification}
In MPC, the dynamic of the reference-vehicle is included in the dynamic of the model. Thus, the matrices given in Equation 3.8 (p.32) will be changed to include the reference vector. Yet, the input given to the leading-vehicle at the current instant cannot be evaluated. The assumption that the leading-vehicle will hold the same acceleration can for example be done. We would then expect the leading-vehicle’s speed to be given as the integral of this constant acceleration. This seemed like a good idea, but two problems resulted from it. Apart from starting or stopping situations, the phases where a vehicle accelerates or decelerates are of short length. But if the prediction horizon is long\(^4\), if the leading-vehicle is having an acceleration different from zero, we will, with the assumption of constant acceleration over the prediction interval, possibly falsely create infeasibility problems in the prediction. We can for example imagine the case of a vehicle accelerating towards the speed limitation as represented in Figure 3.5. In LQ, the reference vector is chosen as constant. Speed and acceleration are kept constant in the prediction horizon. The same is used in the following. An alternative is proposed with the so-called speed-convergence strategy in Section 3.5.

To take into account the reference vector in the state vector of the MPC formulation, the matrices $G$ and and $M$ presented in Section 3.2.2 must now be merged into the state space description as requested by the toolbox and \(^4\)the prediction is meant to encompass the rise time of the system for the optimization to be consistent

$$x_{MPC} = \begin{pmatrix} x_{ego} \\ v_{\text{ref}} \\ \epsilon_p \\ r \end{pmatrix}. $$
presented in Section 3.3.2, that is,

\[ x[k + 1] = \hat{A} x[k] + \hat{B} u[k]. \quad (3.32) \]

\[
\begin{pmatrix}
  x_{ego} \\
  v_{ref} \\
  \epsilon_p \\
  r
\end{pmatrix}
\]

\[
\hat{A} = \begin{pmatrix}
  \Phi_{ego} & \Gamma_{ego} & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  -C_s T_s & 0 & 1 & T_s C_{lead} \\
  0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
\hat{B} = \begin{pmatrix}
  0 \\
  T_s \\
  0 \\
  0
\end{pmatrix}
\]

The output matrix, 

\[ e[k] = C_e x[k], \]

is

\[
C_e = \begin{pmatrix}
  -C_a & 0 & 0 & C_{lead} \\
  -C_v & 0 & 0 & C_{lead} \\
  -h C_v & 0 & 1 & -C_{dlead}
\end{pmatrix}
\]

**Cost function**  The objective function given in Equation 3.27 is characterized by three different matrices. Two penalise the state and the last one penalises the input. An effective tracking minimizes the absolute value of every component in the error vector \( e \). An objective function compatible with this formulation is given as

\[
J(u, x) = (C_e x[N])^T Q_e (C_e x[N]) + \sum_{i=k}^{i=k+N-1} (C_e x[i])^T Q_e (C_e x[i]) + u[i]^T R u[i],
\]

where,

\[
Q_e = \begin{pmatrix}
  Q_{\Delta_a} & 0 & 0 \\
  0 & Q_{\Delta_v} & 0 \\
  0 & 0 & Q_{\Delta_p}
\end{pmatrix}
\]

The weights in the basic MPC formulation then are written

\[ Q = S_f = C_e^T Q_e C_e. \]

The matrix \( R \) remains a constant to be chosen along with the weights on the diagonal of \( Q_e \), during the tuning phase.

**Constraints**  The constraints required by the GCDC protocol concern, as stated in Chapter 1, the minimum and maximum values of the acceleration and speed but also, and more importantly, the minimum value of the inter-vehicular distance which has to be higher than a certain safety distance at all time. Constraint handling is the only reason justifying the use of MPC over LQ. This does not mean that LQ control or a simpler controller will not allow us to respect the constraints from the GCDC protocol, but rather that the handling will be external to the optimisation\(^5\). MPC we include these constraints in the controller so that they are taken into account at all instant and in a way that the control will remain optimal. The parameters to be constrained can be expressed as linear combinations of the state and input of our system’s model which gives the matrices \( K \), \( L \) and \( g \) introduced in

\(^5\)plain saturation of the speed reference, emergency braking if the distance becomes too small
Equation 3.27 (p.38) in a simple form. In our case, the acceleration, velocity and position are directly expressed by a linear combination of the states\(^6\)

\[
a_{\text{ego}}[i] = \hat{C}_{a}^{\text{ego}} x[i], \quad v_{\text{ego}}[i] = \hat{C}_{v}^{\text{ego}} x[i], \quad \epsilon_{\text{ego}}[i] = \hat{C}_{\epsilon}^{\text{ego}} x[i].
\]

The constraints related to these parameters have to be formulated for every single sample in the prediction horizon, that is for \(i\) from \(k\) to \(k+N\). Then, the \(L\) matrix in Equation 3.27 is empty and the \(K\) and \(g\) matrices become

\[
K = \begin{pmatrix}
\hat{C}_{a}^{\text{ego}} & 0 & \cdots & 0 \\
0 & \hat{C}_{a}^{\text{ego}} & \cdots & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & \hat{C}_{a}^{\text{ego}} \\
0 & \cdots & 0 & -\hat{C}_{a}^{\text{ego}} \\
0 & \cdots & 0 & 0 \\
\hat{C}_{v}^{\text{ego}} & 0 & \cdots & \vdots \\
0 & \hat{C}_{v}^{\text{ego}} & \cdots & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & \hat{C}_{v}^{\text{ego}} \\
0 & \cdots & 0 & -\hat{C}_{v}^{\text{ego}} \\
0 & \cdots & 0 & 0 \\
\hat{C}_{\Delta p}^{\text{ego}} & 0 & \cdots & \vdots \\
0 & \hat{C}_{\Delta p}^{\text{ego}} & \cdots & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & \hat{C}_{\Delta p}^{\text{ego}} \\
0 & \cdots & 0 & -\hat{C}_{\Delta p}^{\text{ego}} \\
0 & \cdots & 0 & 0
\end{pmatrix}, \quad g = \begin{pmatrix}
-a_{\text{min}} \\
-\vdots \\
-a_{\text{min}} \\
-a_{\text{max}} \\
-\vdots \\
-a_{\text{max}} \\
v_{\text{min}} \\
-\vdots \\
v_{\text{min}} \\
v_{\text{max}} \\
-\vdots \\
v_{\text{max}} \\
\Delta p_{\text{min}} \\
-\Delta p_{\text{min}} \\
-\Delta p_{\text{min}} \\
\Delta p_{\text{min}} \\
-\Delta p_{\text{min}} \\
-\Delta p_{\text{min}}
\end{pmatrix}.
\]

\[\text{(3.35)}\]

3.4 Proportional-Integral Controller

This work was initially supposed to result in the implementation of an MPC controller for the CoAct demonstration. Unfortunately, only a simple proportional controller could be implemented and tuned. The controller was developed during the GCDC 2011 and adapted to the hardware updated this year.

**Presentation**  This controller was designed and tuned by Henrik Petterson from Scania. Keeping the definition of \(\Delta p\) introduced in Equation 3.6 and used for the tracking formulation in the case of LQ and MPC, the speed reference given by the controller to the cruise control, is

\[
v_{\text{ref}} = v_{\text{lead}} + K_{H} \Delta p.
\]

We need to keep in mind that an interface controller as introduced in Section 2.3.3, is still present in this system and provides an acceleration reference to the breaking system if the acceleration is intended to be lower

\[\text{6Previously we noted } \hat{C}_{a}^{\text{ego}} \text{ the vector outputting the acceleration from the state vector of the ego-vehicle, } \hat{C}_{\epsilon}^{\text{ego}} \text{ correspond to the equivalent padded with zeros to output the same variable from the extended system's state vector.}\]
than the threshold $a_{\text{switch}}$. The Proportional-Integral (PI) controller is not supposed to exhibit an optimal behaviour with regards to a given objective function nor does it include constraint handling. Yet experiments proved satisfactory after using the value $K_H = 0.1$ for the gain on the inter-vehicular distance $\Delta_p$. The controller feeds back speed of the leading vehicle and the integral of the speed error between the ego and leading vehicles.

![PI regulated system](image)

Figure 3.6: PI regulated system. The gain $K_H$ quantifies the amount of control produced by a discrepancy in relative distance from the ego-vehicle to the tracked-vehicle. The tracked-vehicle and ego-vehicle velocities are noted $v_{\text{leader}}$ and $v_{\text{ego}}$ respectively. The speed reference given to the cruise-control is noted $v_{\text{ref}}$. The ego-vehicle is here characterized by two poles $p_1$ and $p_2$ obtained through system identification.

**Study of the controller** In Chapter 2, the ego-vehicle was considered as being sufficiently well characterized by two poles $p_1$ and $p_2$ and a unit gain, which gives, in the frequency domain with the Laplace variable $s$, the relation

$$V_{\text{ego}}(s) = \frac{p_1 p_2}{(s-p_1)(s-p_2)} V_{\text{ref}}(s).$$

If considering that the distance in between the two vehicles should be regulated to 0, the PI control is expressed

$$V_{\text{ref}}(s) = V_{\text{leader}}(s) - \frac{1}{s} (V_{\text{leader}}(s) - V_{\text{ego}}(s)).$$

The transfer function from tracked-vehicle’s speed to ego-vehicle’s speed is then

$$\frac{V_{\text{ego}}(s)}{V_{\text{leader}}(s)} = \frac{(s + K)p_1 p_2}{s(s-p_1)(s-p_2) + p_1 p_2 K},$$

and consequently, the transfer function from tracked-vehicle’s speed to relative-position is

$$\frac{\Delta_p(s)}{V_{\text{leader}}(s)} = \frac{1}{s} \frac{V_{\text{leader}}(s) - V_{\text{ego}}(s)}{V_{\text{leader}}(s)} = \frac{s^2 - s(p_1 + p_2)}{s^3 - (p_1 + p_2)p_2 s^2 + p_1 p_2 s + kp_1 p_2}.$$  

The problem of tuning such a controller can be considered as a pole placement problem with only the gain $K$ as degree of freedom. In a neighbourhood of 0 the transfer function is similar to the transfer function:

$$F(s) = -\frac{s(p_1 + p_2)}{k p_1 p_2}.$$  

The final value theorem applied to this transfer function indicates that the inter-vehicular distance is regulated to zero without steady state error if the leader displays a constant speed. In case of a ramp in the leader’s velocity
(acceleration phases), the distance will be regulated with a steady state error. This characterisation of the controller is not sufficient to guarantee collision avoidance or good platooning performances. The simple modelling of the truck allows us to show that what is expected from the controller will be obtained at steady state. A study of the transient state would be necessary to conclude on the dynamical properties of the system. A two-pole representation of the truck is in that sense limited. We decide to describe the efficiency of this controller with an experimental approach, as exposed in Chapter 5.

3.5 Platooning strategies

In the above sections, several different kinds of controller were presented. However, the platooning formulation was only considered as a two vehicles problem. This problem is the most fundamental of the platooning concept. We assume that if a vehicle is able to follow a single vehicle with decent performances and with a safe behaviour, displaying the same kind of behaviour in a platoon of many vehicles should be feasible. Tracking a single vehicle in or out of a platoon is the same operation. In the context of the GCDC, every member of the platoon makes its speed, acceleration and position available to the whole platoon. The problem that we want to tackle in the present section is to evaluate if taking into account the information coming from the vehicles of the platoon situated ahead of us could bring an improvement in the platooning behaviour.

3.5.1 Average platoon speed

The reference speed given to the cruise controller was, in the case of the PI controller from Section 3.4 (p. 44), given as the speed of the leading vehicle altered by a function of the distance in between the two vehicles. In the case where more than two vehicles are considered in the platoon, the initial term is changed from the speed of the vehicle ahead of us, to a weighted average of the speeds of all vehicles ahead of the ego-vehicle, that is,

\[ v_{av} = \sum \alpha_i v_{vehicle_i}. \]

Such a behaviour should help reducing the ”inertia” of the platoon. To illustrate this concept of inertia, we can take the case of start-up, where all vehicles depart from stand-still. Without platooning strategy, the second vehicle in the platoon follows the platoon leader’s behaviour with a delay. The third vehicle follows the second vehicle with a delay. The information takes time to reach the last vehicles of the platoon. When the last vehicle starts moving, the platoon leader is already far away. If now, the speed reference of a vehicle back in the platoon starts increasing as soon as the platoon leader starts gaining speed, the delay is potentially reduced. The situation is similar, when the platoon leader starts decelerating. The benefits brought by such a method is greatly conditioned by the behaviour of other vehicles in the platoon. If no other vehicle implement the strategy, only a limited benefit will be obtained. If all vehicle of the platoon use such a strategy, we can expect the platoon’s behaviour to be greatly improved.
The different weights set on the other vehicles speeds, represent additional tuning parameters to be considered along with the parameters introduced with every single controller. In order to apply the strategy to the different controllers presented in this report, the description of the system dynamics must be slightly altered. The reference vector of the LQ formalisation must include the speeds of every single vehicle in the platoon, similarly in the case of the MPC controller, which contains its reference vector as a part of the state vector of the dynamic model, the state vector should be augmented. The error vector $e$ defined in Equation 3.31 (p.41) is redefined to include

$$\Delta v = \sum \alpha_i v_{\text{vehicle }i} - v_{\text{ego}}.$$ 

Yet the average-speed strategy do not provide the same benefits in the case of LQ control. The infinite time-horizon LQ controller is characterised by the two feedback gains $K_x$ and $K_r$ in which the contribution of every single state or reference signal can be analysed easily by looking at the corresponding component of the adequate gain. The components relative to the speeds of the vehicles added by the platooning strategy (the vehicles which are not the tracked-vehicle) is always equal to 0. This can be analysed as follows. The inter-vehicular distance $\epsilon_p$ is the only distance\(^7\) regulated by our system as we want it to converge to a predefined value when $\Delta_p$ is regulated to zero. This convergence assumes convergence of the ego-vehicle’s speed to the tracked-vehicle speed. Even if a weight is put on the speed differences with regards to the platoon leader for example, infinite time-horizon will not allow to have the speed of the ego-vehicle different than the speed of the tracked-vehicle. Having a speed set in between the tracked-vehicle’s and the platoon leader’s would induce a divergence of the state $\epsilon_p$. If the weight on the $\Delta_p$ is set to zero, the speed at steady-state becomes the weighted average of the speeds considered as references. The tracking of the relative positions is then lost. This platooning strategy will then only be appropriate in the case of the PI controller.

3.5.2 Most restrictive control strategy

The second platooning strategy investigated was developed by the winning team of the GCDC 2011 [19]. In this strategy, all vehicles in the platoon are considered one after the other as being part of a two-vehicle platoon. Only in this case, the most restrictive control is taken into account. The method is computationally intense in the sense that the control algorithm runs as many times as the number of vehicles ahead of the ego-vehicle in the platoon. For a simple explicit controller, this may not be a problem but is not be reasonable in the case of on-line MPC.

The advantages of this platooning strategy are different than for the method described in the previous paragraph. The control is always taken as the most conservative one. In the case of acceleration phases, the vehicles further ahead in the platoon have speeds higher than the ego-vehicle’s. During an acceleration phase, the control based on the tracked-vehicle is thus the most restrictive one. The most restrictive control strategy does not

---

\(^7\)the inter-vehicular distances relative to the other vehicles could be included, but conflicting goals would emerge which would imply uncontrollability
change the control generated by the two-vehicle platoon control scheme in acceleration phases but is effective in deceleration phases; in this sense, team AnnieWays’s strategy can be considered as safety oriented. It computes its control based on the vehicle displaying the biggest deceleration or smallest velocity.

A problem concerning this method needs to be mentioned. When computing the optimal control in a two-vehicle platoon, a reference distance (which as stated in the introduction is desired to be kept above and close to a value \( d = d_0 + hv_{ego} \)\(^8\)) is used. When the vehicle considered as platoon leader of the two-vehicle platoon is the closest one to the ego-vehicle, this distance can be directly controlled. However, if in the perspective of the platooning strategy we are studying here, the vehicle considered as leader is situated \( n \) vehicles further away in the platoon, the distance between this leader and the ego-vehicle is not bound to the ideal value of \((n + 1)d\). The first reason for this is that the desired distance cannot be held at all time and many perturbations have to be accounted for and will accumulate the more the two vehicles are spread out in the platoon. The second reason comes from the diversity of controllers present in the platoon. Not all vehicles are supposed to regulate the same distance to the vehicle directly ahead of them.

The GCDC specifies a lower bound for the distance between two vehicles and in the performance assessment, it is favourable to keep the distance as small as possible at all time. Nevertheless, to avoid having the inter-vehicular distance reach too low values, it was common practise at the previous GCDC to augment the desired distance with a safety parameter. This safety parameter is not supposed to be known. The actual inter-vehicular distance to the vehicles far away in the platoon will most likely be bigger than the expected distance \( d \). In this context, the platooning strategy will end-up regulating its speed on the sole vehicle right in front of the ego-vehicle most of the time. To avoid this problem, we decided that in the case of vehicles far away in the platoon, the current distance is always taken as the target distance. As a consequence, the most conservative control policy is applied and it considers the distance error to the tracked-vehicle only if it is right ahead of us. Otherwise, the control is computed the same way but assuming that the distance is perfect. The controller will try to keep the current distance. The platooning policy exposed in the previous paragraph is not concerned by the same problem. In this former method, the vehicles of the platoon are hierarchised, only the distance to the closest vehicle is taken into account, the other vehicles being considered only in terms of speed and, potentially, acceleration.

3.5.3 Speed convergence strategy

In the Section 3.2.3 we show that in the case of finite time-horizon LQ, the control is computed at each time sample after solving two recursions. The value of the reference vector at every instant of the prediction horizon is used in the recursions. It was noted before that the reference could be taken constant over the whole prediction horizon as it was done in the case

\(^8\)In the discussions concerning the inter-vehicular distances, we will consider here the vehicle length to be 0, in the actual system, vehicle length was broadcast by every vehicle and taken into account in the computations.
of infinite time-horizon LQ and MPC control. A better idea would reside in finding a better prediction of the values for the reference variables. A first solution exposed earlier was to consider the acceleration of the leader constant and make its velocity evolve accordingly. Given the fact that the acceleration is often noisy and subject to oscillations, this assumption of constant acceleration over the prediction may not be sensible. However, the speeds of other members of the platoon can provide us with insight on the future speed of the tracked-vehicle. If the ego-vehicle is situated at the fourth position in the platoon, the speeds of the platoon leader and second vehicle in the platoon can be used to guess the future values of the third vehicle’s speed. We decide to assume that at the end of the prediction horizon, the speeds of the tracked-vehicle will have converged to the platoon leader’s. This speed convergence strategy will be used in the simulations involving the finite time-horizon LQ controller.

3.5.4 Considerations on overtaking manoeuvres

When a vehicle changes position in the platoon, a set of manoeuvres has to be executed. When opening a gap to let another vehicle come into the platoon, or in the opposite, when closing gaps, \(d_{\text{ref}}\) included in the reference vector is modified. Such an abrupt change in the distance to track provokes abrupt braking. The values of the \(d_{\text{ref}}\) to be regulated by the controller are generated by the logic unit of the system. The logic coordinates the different operations of the manoeuvres. A low-pass filter should be used on the signal generated by the logic to make the steps in reference distance smoother. When the ego-vehicle is taking over or is being overtaken by another vehicle, the controller has to regulate its speed and position on a subset of the vehicles of the platoon. In normal situations, the logic transmits to the controller a platoon vector integrating all the vehicles ahead in the platoon. During an overtaking, the vehicle taking over should be removed from the array of vehicles to be considered by the controller. This aspect of control was integrated to the logic part of the system[6].

3.5.5 Platoon filtering

The different vehicles of the platoon display a bit of an oscillatory behaviour when running in real experiments. No platoon filter was included in the system. The decision was made to include low-pass filters in the controller in order to attenuate the disturbances. By setting a low value of the low-pass filter’s cut-off frequency, we also realized that the low-pass filtering could be beneficial for the platooning application. By only considering the average trends of vehicle far away from us in the platoon, we can expect to have a smooth application of the platooning strategies introduced in this chapter.
Chapter 4

Implementation

In this chapter, we discuss the constraints imposed by the hardware architecture used in the context of the experimentations led by team Scoop. This will hence be an opportunity to comment the interface between the controller and the vehicle and some precautions to take in the implementation phase. MPC proved to be computationally unstable, we hence describe the problems encountered in the implementation of this kind of controller and justify the reason why MPC was not used in simulations.

4.1 Hardware and software resources

The implementation of the controller and other components of the system was done in the xPC Target real-time software environment from Mathworks. The details of the implementation and the complete structure of the system are discussed in [4]. The environment consists of a real-time kernel for standard x86 computers and a set of libraries allowing automatic compilation of Matlab/Simulink to C-code in the real-time target. This solution exists as a prototyping platform. Simulink offers a simulation environment allowing tests of the technical solution in a closed and simple framework. xPC Target exists as a transition solution between software testing inside Simulink and actual design of a specific hardware solution. It allows the use of the Simulink test code inside a safe and flexible architecture. As mentioned earlier, the computer running the software is a standard x86 computer. This computer is much less powerful than the full-fledged desktop computers of today but still of a more powerful nature than the specific embedded computers used in standard Scania trucks.

The real-time framework set some restrictions on the implemented software and does not allow the use of any ready-made Matlab software. This is a problem when concerning the solvers to be used, for example, in the case of on-line MPC for which only the quadprog solver, part of any standard Matlab distribution, could be used.

The computation of the LQ controllers tested in the following chapter was performed using the dare function in Matlab when the prediction-horizon is infinite. The recursive algebraic Riccati equation is solved online in the case of finite time horizon. This function provides solution to the discrete algebraic Riccati equation for the set of matrices given in Section 3.2.3 (p. 33). Implementation of the LQ controller then only consists of simple
algebraic operations easily written in Matlab code. The same also applies to Henrik’s controller as presented in Section 3.4 (p. 44).

The case of MPC control implementation is more complicated. The \texttt{quadprog} software has to be run at every time instant with the same arguments but updated at each iteration with the current value of the state $x[k]$. The use of \texttt{quadprog} follows exactly what was presented in Section 3.3.2 (p. 38). In the case of explicit MPC, the main part of the computations has to be done off-line and consists of the multi-parametric quadratic programming solving. Two toolboxes can be used for that purpose [26] [27]. The first one, called Hybrid Toolbox was developed by Alberto Bemporad, and is meant to be used as an extension of the MPC toolbox in Matlab. In the beginning of this work, we decided not to rely on the MPC toolbox from Matlab which contains components not usable in the framework offered by xPC-Target. We instead opted for the use of the Multi-Parametric Toolbox ($\texttt{MPT}$\textsuperscript{1}) which allows more flexible representations of dynamical systems and compatibility with the hardware architecture. The \texttt{MPT} toolbox requests the description of the control problem thanks to two objects of the type struct. These two objects, describe altogether the whole MPC problem description formulated in Paragraph 3.3.3 (p. 40). These two objects are used by the \texttt{MPT} toolbox in order to formulate and solve the MP-QP problem as formulated in Paragraph 3.3.2. Once this problem is solved, an object of the type controller is created by the toolbox. This object contains the partition of the parameter state and the list of all piecewise affine state feedback to be applied to the system on every subspace of the partition. This object is not directly usable in the environment of the xPC-Target. Luckily, the \texttt{MPT} toolbox contains a certain number of tools allowing conversion of this controller object to a search tree, more universally used in computer science and well appropriate to describe the structure of the parameter space. The Table 4.1 gathers information about the implementation of the controllers. The data type of the objects to be stored and the computation to be done varies in function of the controller.

When beginning this thesis work, model predictive control was known to be computationally intensive. High hopes were put on the explicit version of such a controller to reduce the on-line complexity of the control evaluation. Explicit MPC moves all the complexity of the control off-line. Instead of solving particular optimization problems at every instant of the control, explicit MPC tries to compute beforehand all possible controls. Even if the parameter space to be searched can be limited, the off-line computation complexity is in the case of systems of dimension bigger than 3, tremendous. A computation of an explicit controller was considered to have failed after 14 hours of computation without result. Such a length was common with the formulation of the tracking problem presented in Chapter 3 on controllers.

### 4.2 Interface system / controller

As described in Figure 1.1 (p.11), the controller receives its inputs from two different sources. The data from the ego-vehicle itself is measured, filtered\footnote{http://control.ee.ethz.ch/~mpt/}
<table>
<thead>
<tr>
<th>Controller</th>
<th>Off-line computation</th>
<th>On-line computation</th>
<th>Storage needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>nil</td>
<td>matrix multiplication</td>
<td>scalar gain $K_H$</td>
</tr>
<tr>
<td>$\infty$ time-horizon LQ</td>
<td>Riccati equation resolution, using matlab functions</td>
<td>matrix multiplications</td>
<td>two vectorial gains $K_x$ and $K_r$</td>
</tr>
<tr>
<td>finite time-horizon LQ</td>
<td>nil</td>
<td>matrix recurrences, algebraic expressions</td>
<td>set of matrices describing cost function, system dynamics to form the matrix recurrences</td>
</tr>
<tr>
<td>online MPC</td>
<td>nil</td>
<td>optimisation problem using quadprog</td>
<td>Set of matrices describing cost function, system dynamics and constraints to be given as arguments to quadprog</td>
</tr>
<tr>
<td>explicit MPC</td>
<td>Computation of the piece-wise affine control law using MPT toolbox</td>
<td>Extraction of the current control law depending on current state</td>
<td>Search tree</td>
</tr>
</tbody>
</table>

Table 4.1: Implementation specificities of different controllers.

and directly routed to the controller. The filter used is supposed to improve the quality of the data but also performs simple sensor fusion, combining the information coming from the GPS and the vehicle sensors. The GPS is indeed a computer of its own and derivates not only longitude, latitude and altitude of the vehicle but also its heading speed and acceleration. The data coming from the different vehicles in the platoon is not, for now, filtered by any kind of filter. This means that we rely entirely on the quality of the data received through wireless transmission. In some cases (packet losses and delays for example) this assumption is too rough and a platoon estimator should be implemented in later iterations of the project. The data transferred from the wireless unit to the controller has to be conditioned to be usable by the controller. The lane position of every vehicle is not included in the wireless communication and has to be interpolated from GPS measurement. This interpolation is not trivial for example in curves. Besides, overtaking vehicles may or may not be part of the platoon and should not always be considered for control. This corresponds to the sorting function mentioned Figure 1.1 (p.11). The second pre-treatment concerned speed and acceleration of the vehicles. We decided to apply a low pass filter over every single vehicle’s speed and acceleration allowing to get rid of potential perturbation that were actually measured even in the case of the ego-vehicle.

The input to the controller block contains formally two elements. A vector containing the measured state of the ego-vehicles and an array characterizing the whole platoon. The array contains the vehicles ordered from the closest to the furthest to the ego-vehicle and contains, the relative distance to the vehicles, the expected distance requested in between the two vehicles, their speeds and accelerations.
4.3 Safety measures

The problem of stability of the platoon formation was mentioned Equation 1.1 (p. 11) in the introduction. In [2], the study of this criterion is performed for a linear representation of the vehicles involved in the platoon and with the use of identical PID controllers for every vehicle in the platoon. The conclusion made is that in such a context, arranging the trucks in decreasing order of weights would systematically provide the platoon with string stability. In the context of the GCDC and CoAct, this condition is not necessarily respected. Besides, as formulated in introduction, to guarantee satisfaction of the frequency domain criterion specified by the GCDC is difficult practically. The vehicles considered are non-linear and the different vehicles’ dynamics and controllers in the platoon are heterogeneous and unknown. We do not try to analyse this criterion ourselves. String stability is only assessed through experimentations.

String stability needs to be considered with regards to tuning. The main concern regarding safety is the ability every vehicle in the platoon has to keep a security distance to the closest vehicle ahead of him. In [1] it was determined that a distance bigger \( d > 2 \) meters between two vehicles would ensure collision avoidance in the case of identical vehicle dynamics and a total delay between sensing and braking of 500 ms. The scenario studied concerned two vehicles going from a velocity of 25 m/s to stand still with a maximum retardation. The situation represents a worst case scenario but in the GCDC, smaller personal vehicle of completely different dynamics are enlisted along with trucks. The acceleration profile smaller vehicles can display may be more abrupt than in the case of trucks. The distance between two vehicles is in the case of our controller is regulated to a value of always strictly bigger than 15 meters. It was decided to implement an emergency brake to the controller which is triggered by a value of the inter-distance smaller than 15 meters.

4.4 Model Predictive Control implementation

4.4.1 Introduction of slack variables

Our system is modelled in a simple way. MPC generates its control based on prediction of the state several steps ahead in time. The difference in between actual system and model will generate problem when considering constraint handling. The actual system might fall into a forbidden state even if this was not predicted base on our simpler model. Non-respect of a constraint generates infeasibility for the quadratic program. Increasing the prediction horizon should help mitigating feasibility problem. However, this comes at great cost as the Hessian matrix \( H \) of the quadratic program grows bigger with the prediction horizon. Slack variables are used to make the constraints on the states soft. The constraints on the states are formulated

\[
0 \leq Kx + g.
\]

The value of the slack variable is chosen by the controller. The slack variables are then introduced as inputs to the system which never directly influence any of the state but are taken into account in the constraints. If one slack
A variable is introduced to soften one particular kind of constraint⁴, the input matrix of the system introduced in Equation 3.33 (p. 43) is then augmented with zeros,

\[
\Gamma = \begin{pmatrix}
0 & 0 \\
T_s & 0 \\
0 & 0 \\
0 & 0
\end{pmatrix}.
\]

The constraint softening is then obtained by injecting the slack variable in the desired constraint. For instance if we consider the acceleration, we get

\[
a_{\text{min}} < \hat{C}_{\text{ego}}^x x + u_{\text{slack}}^{a_{\text{min}}},
\]

(4.1)

Maximum and minimum value constraints may be imposed on the slack variables to prevent the constraints from being too harshly violated but this restrict the ability of the slack variable to reduce infeasibility. The number of constraints added is equal to \(2N\) times the number of slack variables introduced⁵. The addition of constraints on the slack variables extends the \(K\) matrix defined in Equation 3.35 (p. 44) with zeros as the state is not involved with these constraints, the \(g\) matrix is extended with the maximum and minimum admissible values for the slack variables in the same fashion as done with other constraints. The \(L\) matrix in Equation 3.27 (p. 38) now includes non-zero elements for every constraint concerning a slack variable. In the following equation we show how to change the matrices \(K, L\) and \(g\) to add slack variables on the lower bound on the acceleration,

\[
K = \begin{pmatrix} K_{\text{old}} \\
0 & \ldots & 0 \\
\vdots & \ddots & \vdots \\
0 & \ldots & 0
\end{pmatrix}, \quad g = \begin{pmatrix} g_{\text{old}} \\
-u_{\text{slackmin}} \\
\vdots \\
-u_{\text{slackmin}}
\end{pmatrix}, \quad L = \begin{pmatrix} 0 & 1 \\
\vdots \\
1 \\
0 \\
\vdots \\
0
\end{pmatrix},
\]

(4.2)

where we renamed the former \(K\) and \(g\) matrix with the script \(\text{old}\) and concatenated them inside the new matrices. The last element from the MPC formulation to be changed to include the slack variables is the input weight \(R\) which should not be scalar anymore but a square matrix of dimension 1 plus the number of slack variables introduced. \(R\) is chosen diagonal as we only want to penalise the magnitude of the slack variable. The weights addressed to every slack variable should be chosen carefully to limit the violations of the constraints. If the weights are too low, the slack is used even when not necessary. We want to use it only to prevent infeasibility problems. To put a very high weight on the slack variables limits violations while preventing infeasibility problems when external factors lead the system to a constraint violating state (if a perturbation leads the speed higher than the speed limit

---

²for us this would be all constraints over the lower (respectively higher) bound of the speed (respectively acceleration or inter-vehicular distance)

³lower and higher bound for every slack variable at every single sample in the prediction horizon
for example). Every slack variable added imply a lot more complexity to the optimization problem to be solved at every step. The controller then needs to compute a vectorial control which makes the computation time for the quadratic program skyrocket.

4.4.2 Computational considerations

The Model Predictive Control is known as being computationally demanding. In former work carried out at Scania [3] the computational load was considered too great for any concrete implementation. However, big hopes were based on the perspective of efficiency presented by explicit MPC. This method transfers the computational load off-line where the computational time is virtually free. Based on the conclusion of the MPC implementation done in [3] our goal became to analyse the performance gains obtained by reducing as much as possible the complexity of the model used.

Even with such simple models, multiple instability problems are experienced when computing controllers based on a plant including integrators. We will introduce in this Section an elementary example showing the sort of computational instability encountered.

Let us consider the system characterised by the state space representation

$$
x[k + 1] = Ax[k] + Bu[k] \\
y[k] = Cx[k],
$$

(4.3)

with

$$
A = \exp \left( \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} T_s \right), \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.
$$

(4.4)

The state vector $x$ can be seen as representing two decoupled systems, one driven by the input and characterised by a stable discrete pole $e^{-T_s}$. The two remaining states evolve independently from the first one. They are characterized by two poles on the unit circle. The global system is driven using MPC based on the quadratic cost determined by the weights,

$$
Q = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad R = 0.1.
$$

(4.5)

The system is initialised with the state

$$
x[0] = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.
$$

The states $x_2$ and $x_3$ display oscillations of magnitude 1 and frequency 1Hz and given the state weight used, the MPC control will do its best to have the state $x_1$ as close as possible to $x_3$. To these specifications, we can add the constraints

$$
x_1 \leq 0.8 \\
-0.09 \leq u \leq 0.09.
$$

(4.6)
With a sampling time of 0.1s and a prediction horizon of 30 samples, the Multi-Parametric Toolbox (MPT), Matlab tool for multi-parametric optimization which provides a framework to specify online and explicit MPC [26], computes 10s of simulation in 6.41s on a fast desktop computer. The result of the simulation are shown in Figure 4.1 where the expected tracking of the third state by the first one is observed. This elementary problem is meant to represent to some extent the tracking formulation done in Section 3.3.3. The state vector is decomposed in two blocks. The first one represents the ego-vehicle state, the second one represents the reference as coming from the tracked-vehicle. The analogy to the tracking problem gives us the state $x_1$ as the ego-vehicle’s velocity, $x_2$ and $x_3$ as the velocity and acceleration of the tracked-vehicle.

Figure 4.1: MPC simulation for a simple model, a 3-states plant analogous to the tracking problem, with constraints on state and input. The input without constraint was obtained in a different simulation with the same model and without any constraint applied to the system. It is included for information. $x_1$ represents the speed of the ego-vehicle, $x_3$ is the speed of the tracked-vehicle. A constraint is formulated on the maximum speed of the ego-vehicle and on the maximum and minimum value of the input.

To match more closely the form of the tracking as described in the description of MPC, we increase the problem formulation to contain the distance between the two vehicles and modify the cost function and state weight
\( Q \) to penalise the distance. The system’s state is extended with the state \( x_4 \). The state represents the inter-vehicular distance. The new state space representation of the system and the new cost function are then characterised by the matrices

\[
\hat{A} = e^t \begin{bmatrix}
-1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & -1 & 0 & 0 \\
-1 & 1 & 0 & 0
\end{bmatrix} T_s, \quad \hat{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},
\]

(4.7)

\[
\hat{Q} = \begin{bmatrix}
1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}.
\]

(4.8)

The on-line MPC is computed using MPT with a sampling time of 0.1s and a prediction horizon of 30 samples, like in the case of the example given in Figure 4.1. The simulation crashes after 4 seconds of computation when the state \( x_1 \) gets closer to the value of its constraint. When the prediction horizon is decreased to 15 samples, the simulation runs for 11 seconds but eventually crashes with the same infeasibility error. A prediction horizon of 10 samples however allows the system to compute the 20 seconds of the simulation without any crash as pictured in Figure 4.2. The figure shows an appropriate constraint handling. The control is expected to be periodical, and no infeasibility problem should appear. The infeasibility error message given by MPT then comes completely unexpected. Infeasibility can come up for a number of reasons. Model error induces differences in between the behaviour of the actual plant and the prediction done for the optimisation.
The control that would keep the linear model inside the bounds of the constraints, may not reach the same objective with the actual plant. In the context of an actual implementation, after a constraint violation, the MPC cannot be used anymore. A backup controller needs to be used in order to bring the system’s state in the polytope defined by the constraints. The solution provided by slack variables described earlier cannot be considered here as the computational complexity would further increase. Infeasibility can also be provoked by a bad configuration of the constraints. In the case of vehicle platooning, an inherently infeasible constraint could emerge from a simultaneous constraint on the minimum acceleration of the vehicle and minimum relative distance to the tracked-vehicle. In the example discussed in this paragraph, the second and third states are not controllable through the input $u$. The two states represent references, if a constraint was to be applied to it, the controller will be in no circumstances able to cope with it. Such problems need to be avoided through careful study of the constraints applied to the system. In some cases, infeasibility can be avoided via increase of the prediction horizon. The bigger the prediction horizon, the greater is the number of control move dedicated to respecting the constraints. The problem encountered here is different than the two mentioned here above, a decrease in the prediction horizon resulted in an improvement of the situation with regards to infeasibility. The simulation was realised without model mismatch. The same model was used for building the controller and for the simulations. The problem encountered here has numerical origins and possibly relates to the computational complexity of the problem. The exact same problem solved with CVX, a package for specifying and solving convex programs in Matlab [22], leads to a different result. The simulation continues without interruption or any problem for the values of the prediction horizon’s length of 15 or 30 samples. The solvers in CVX are different than the Matlab function quadprog used by MPT. The numerical stability is increased by the use of CVX based controllers but the computational efficiency is decreased. The tracking problem introduced in the current paragraph, was simulated with help of CVX and is pictured in Figure 4.3. It’s computation already too difficult to be implemented in real time as 20 seconds of simulation were computed in more than 40 seconds on a fast desktop computer.

Different quadratic programming solvers are available on the market. The company IBM offers an academic license allowing the use of the CPLEX [23] optimisation suite including bindings to different programming languages and a range of solvers which contains among others, a quadratic programming solver. This solver proved way more efficient and numerically stable than the ones used in CVX and the standard solver offered by Matlab. The numerical problems encountered on-line emerged from the combined effect of increasing the size of the state vector of the problem and the presence of critically stable poles in the system. The critically-stable poles come from the integrator included for tracking purposes and the relative distance taken as integral of the relative speed.

MPT does more than offering a framework to design, in a user-friendly way, an online-MPC. It also allows us to deal with the problem of multi-parametric programming and, as a consequence, explicit-MPC. In the first of the two systems introduced in this paragraph to illustrate simple track-
Figure 4.3: CVX test, 4-states plant with constraints on state and input for \( N = 30 \). The distance from the ego-vehicle to the tracked-vehicle is represented by \( x_4 \). \( x_1 \) represents the speed of the ego-vehicle, \( x_3 \) is the speed of the tracked-vehicle. A constraint is formulated on the maximum speed of the ego-vehicle and on the maximum and minimum value of the input.

After more than 20 hours of computation the computation was assumed failed.
tion needs to be stripped down to its simplest expression. This results, for example in the omission of the integrator at the input of the plant, and of the acceleration reference whose impact on the control proved to be mathematically negligible in the example of LQ control. Besides, the reference distance $d_{ref}$ included as a reference to the state of the extended system, is a constant whose value is decided off-line. In order to reduce the length of the state vector, $d_{ref}$ has to be directly subtracted from the measurement of the physical distance in between the vehicles, as measured through the GPS embedded in the vehicles, in which case we can only ask the system to drive the position error $\Delta p - hv_{ego}$ to zero. The state vector is then reduced to a four-dimensional vector, that is

$$X = \begin{bmatrix} v_{ego} \\ a_{ego} \\ \epsilon_p \\ v_{leader} \end{bmatrix}$$

The rise time in speed for the Scania truck used in experimentation was roughly 5 to 16 seconds. Using the rule of thumb which requires ten samples to be taken in a rise time, we set a value of 0.5s for the sampling time.

Concerning infeasibility relative to modelling errors, the complexity of the MPC control scheme required us to model the system as simply as possible. Ironically, the main motivation for the study of MPC control for the platooning application is the addition of constraints to the LQ control scheme, to take into account the specifications formulated by the GCDC authority and avoid collisions. With such a simple model, coping with constraints on the acceleration is a real challenge. The second order model we identified cannot possibly capture the trucks behaviour closely enough to reliably describe the rather high frequency content of the acceleration. The advantages provided by MPC then become marginal. MPC can only be used if the system is as simple as possible but constraints cannot be handled if the system representation is too simple. Besides, the computational instability problem subsist. As a consequence, the results shown in Chapter 5 do not include MPC.

4.4.3 PWA and non-linear plants

As mentioned in Chapter 2 concerning modelling, piece wise affine and non-linear plants can be used in the framework offered by model predictive control. The formulation of the optimisation problem is then, from a formal point of view, nearly identical to the one given in Equation 3.27. The system dynamics is however changed and the solver used to solve the optimisation problem is adapted. A PWA plant implies the use of mixed-integer parametric-programming [15] which extends the definition of the cost and the constraints of quadratic programming with a vector of integers. The MPT toolbox used in this thesis for experimentations on explicit MPC comes with such a solver. The toolbox also allows a simple way to describe PWA plants and to design an adapted MPC controller either on-line or explicit.

5 the feedback gain on the ego-vehicle’s acceleration with any value of the acceleration weight $Q_a$ close to 0 after solving of the Riccati equation
The vector of integers allow in this sort of plant, the introduction of the discrete states from the PWA formulation in the optimisation problem. Mixed-integer optimisation increases the computational complexity of the problem. In the case of non-linear MPC, solvers aimed at finding a constrained minimum to any scalar function of several variables need to be used. However, in case of non-convex functions, no guarantee on resolvability can be given. Considering the problems encountered with a classic on-line implementation of MPC, these two solutions will not be further investigated. They both involve an increase in the control complexity.
Chapter 5

Analyses

In this chapter, the different controllers studied in this report are applied to the platooning problem in the simulation environment provided by PreScan. A platoon of four vehicles is studied. Every vehicle in the platoon is equipped with the same controller within one iteration of the test. We measure accelerations, velocities, relative positions and controller inputs for every vehicle. The relative position is taken as the position in the fourth vehicle’s referential. The position of the fourth vehicle will then always be 0. On the position profile plot, we include the position reference relative to every vehicle. These position reference are computed as being $p_{vehicle_i} + d_{ref}$. The $d_{ref}$ was here chosen constant equal to 27 meters. On one graph we can visualise the relative positions and position errors for every vehicle. Two scenarios are included for every simulation. In a first phase (time = 0), the vehicles start at stand still and reach a steady state (phase end at time = 80s), in the second phase, the first vehicle displays a strong retardation (at time = 80s) for a brief instant and reaches a steady state (phase ends at time = 160s).

A list of the controllers used in these simulation is given in Table 5.1 and Table 5.2. The figures of this chapter will gather only a subset of whole data measured during the simulation. In the next paragraphs, different controller solution will be compared based on this controllers. When comparing two controller, we will mention the parameters modified, the parameters unmentioned are kept equal. Their values are listed in the two tables.

Table 5.1: List of PI controllers used in simulation. The $\alpha$ parameters refer to the weights of the average speed platooning strategy.

<table>
<thead>
<tr>
<th>Name</th>
<th>$K_H$</th>
<th>Second vehicle</th>
<th>Third vehicle</th>
<th>Fourth vehicle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\alpha_{leader}$</td>
<td>$\alpha_{leader}$</td>
<td>$\alpha_{vehicle_2}$</td>
</tr>
<tr>
<td>PI 1</td>
<td>0.1</td>
<td>1</td>
<td>0.3</td>
<td>0.7</td>
</tr>
<tr>
<td>PI 2</td>
<td>0.1</td>
<td>1</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>PI 3</td>
<td>0.1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>PI 4</td>
<td>0.3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>PI 5</td>
<td>0.05</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>PI 6</td>
<td>0.3</td>
<td>1</td>
<td>0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>
Table 5.2: List of LQ controllers used in simulation.

<table>
<thead>
<tr>
<th>Name</th>
<th>$N$</th>
<th>$Q_a$</th>
<th>$Q_v$</th>
<th>$Q_p$</th>
<th>$R$</th>
<th>model</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LQ 1</td>
<td>$\infty$</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>500</td>
<td>sys56</td>
<td>No</td>
</tr>
<tr>
<td>LQ 2</td>
<td>$\infty$</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>500</td>
<td>sys56</td>
<td>No</td>
</tr>
<tr>
<td>LQ 3</td>
<td>$\infty$</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>500</td>
<td>sys1011</td>
<td>No</td>
</tr>
<tr>
<td>LQ 4</td>
<td>$\infty$</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>500</td>
<td>sys1920</td>
<td>No</td>
</tr>
<tr>
<td>LQ 5</td>
<td>$\infty$</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>500</td>
<td>sys56</td>
<td>No</td>
</tr>
<tr>
<td>LQ 6</td>
<td>$\infty$</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>75</td>
<td>sys56</td>
<td>No</td>
</tr>
<tr>
<td>LQ 7</td>
<td>$\infty$</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>25</td>
<td>sys56</td>
<td>No</td>
</tr>
<tr>
<td>LQ 8</td>
<td>90</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>500</td>
<td>sys56</td>
<td>No</td>
</tr>
<tr>
<td>LQ 9</td>
<td>150</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>500</td>
<td>sys56</td>
<td>Convergence</td>
</tr>
<tr>
<td>LQ 10</td>
<td>$\infty$</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>75</td>
<td>sys56</td>
<td>Restrictive</td>
</tr>
<tr>
<td>LQ 11</td>
<td>150</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>500</td>
<td>sys56</td>
<td>No</td>
</tr>
<tr>
<td>LQ 12</td>
<td>$\infty$</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>75</td>
<td>sys56</td>
<td>Restrictive</td>
</tr>
<tr>
<td>LQ 13</td>
<td>$\infty$</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>75</td>
<td>sys56</td>
<td>No</td>
</tr>
</tbody>
</table>

5.1 Controllers’ Tuning

The three different controllers presented in this thesis are all characterized by a certain number of parameters. Some of these parameters are closely related to the implementation and computational performances of the system. This is the case of the prediction horizon in MPC and finite time-horizon LQ control.

**Prediction horizon** This parameter is constrained by the hardware implementation. This is an important factor in the case of MPC control which is, even in its most simple formulation prone to computational instability related to algorithm complexity. The Hessians size increases proportionally to the time-horizon. The finite time-horizon LQ controller doesn’t show the same kind of unstable behaviour, but the computation time of its control for a given sample $k$ does depend on the size of the prediction horizon. A rule of thumb requires the prediction horizon for a receding horizon control scheme to be of size comparable to the rise time of the system. The rise time in speed was measured of the order of magnitude $6$ to $15$ seconds. Taking a prediction horizon $N = 150$ for a sampling time $T_s = 0.1$ gave a computation time of less than $0.1 s$ per real-time second on the desktop computer used. The finite time-horizon LQ remains computationally efficient even considering the reduced performance of the embedded computer used in the actual implementation. To compare, the time requested for computing an explicit infinite time-horizon LQ control or to perform the computation of the simple PI controller was slightly inferior to $0.01 s$. The finite time-horizon LQ scheme computes recursively the control which makes the procedure at least one order of magnitude bigger. However the computation time increases dramatically in the case of MPC control. To respect the condition given on the prediction horizon $15 \leq N * T_s$ and limit the computation time, we decide to take the biggest value allowed for sampling times as computed earlier in this report using a thumb’s rule $T_s = 0.6s$. This would require a prediction-horizon $N = 25$ to cover the same $15s$ of time-horizon. An
interesting fact in the case of MPC is that the computation of the solution to the quadratic programming problem formulated at a given time, depends on the value of the current state. The optimisation is more quickly solved when the state is situated far away from any constraint. This increase is presented here in the case of the simpler problem formulation as introduced in Section 4.4.2. We summarize in Table 5.3, the detail concerning the computation time of every method given in terms of controller time to the real time elapsed ratio as measured on a standard desktop computer.

Table 5.3: Computation time for different controller choices.

<table>
<thead>
<tr>
<th>Controller</th>
<th>sampling time</th>
<th>prediction horizon N</th>
<th>active constraints</th>
<th>comput-time ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>infinite horizon LQ</td>
<td>0.1</td>
<td>NA</td>
<td>NA</td>
<td>2%</td>
</tr>
<tr>
<td>finite horizon LQ</td>
<td>0.1</td>
<td>150</td>
<td>NA</td>
<td>20%</td>
</tr>
<tr>
<td>PI</td>
<td>0.1</td>
<td>NA</td>
<td>NA</td>
<td>2%</td>
</tr>
<tr>
<td>MPC</td>
<td>0.6</td>
<td>25</td>
<td>none</td>
<td>56%</td>
</tr>
<tr>
<td>MPC</td>
<td>0.6</td>
<td>25</td>
<td>input maximum</td>
<td>129%</td>
</tr>
<tr>
<td>MPC</td>
<td>0.6</td>
<td>25</td>
<td>relative distance maximum</td>
<td>203%</td>
</tr>
<tr>
<td>MPC</td>
<td>0.6</td>
<td>10</td>
<td>none</td>
<td>45%</td>
</tr>
<tr>
<td>MPC</td>
<td>0.6</td>
<td>10</td>
<td>input minimum</td>
<td>93%</td>
</tr>
<tr>
<td>MPC</td>
<td>0.6</td>
<td>10</td>
<td>relative distance minimum</td>
<td>155%</td>
</tr>
</tbody>
</table>

The Simulink model used to compute these data contains the controller and linear representation of the plant identical to the one the controller used was based on. The different states of the plant were directly measured. In the case of MPC, the same simulation was launched one time for every case with or without constraint and for the two values chosen for $N$. At the first experiment no constraint was applied to the system and the simulation of real-time length $T_{real}$ was computed in $T_{noconst1}$. The computational ratio was given in the case without constraint as

$$r_{without} = \frac{T_{real1}}{T_{noconst1}}.$$ 

In a second simulation, the constraint was added on the input and gave a computation time of $T_{total}$ for the same real-time period $T_{real}$. The time the system spent with its input saturated at its constrained value was measured $T_{active}$ which gives a time spent away for the constraint $T_{total} - T_{active}$. If we call $r_{with}$ the efficiency of the computation when a constraint is active, and if we suppose that the computation efficiency is always of $r_{without}$ if no constraint is active, $r_{with}$ if we have an active constraint, we have the

---

1The measurements were realised through command-line simulation in Simulink, using the Matlab function clock before and after simulation and dividing the real elapsed time by the simulation length. The model used included in all cases was the same linear system for which the control was computed.
relation,
\[
\frac{T_{\text{active}}}{r_{\text{with}}} + \frac{T_{\text{total}} - T_{\text{active}}}{r_{\text{without}}} = \frac{T_{\text{real}}}{T_{\text{total}}}.
\]

The same method was applied for the case where the constraint was applied on the inter-vehicular distance.

The computation time decreases indeed with the prediction horizon, but even for small values of the prediction horizon, the computational efficiency remains low. The biggest load comes from the constraint on relative distance. The processor occupation will always be high with this formulation of the MPC controller. Better solvers than CVX, the package used here for specifying and solving convex programs [22], like CPLEX could be used with better performance.

**Dynamical tuning** The remaining parameters defining the different controllers have an important effect on the system’s performances. In the PI controller, only I parameter, used to feedback the relative-position to the tracked-vehicle was left for tuning. In the case of an LQ or MPC controller, the tuning parameters are represented by the set the different weights constituting the matrix weight \(Q\) applied to the vector of output; and the scalar weight \(R\) on the system’s input. The linear model used to represent the vehicle’s dynamics can also be considered as a tuning parameter. In the Chapter 2, three different models obtained around three different speed set-points were determined. The controller displays different behaviours depending on the model chosen to be used in the optimisation problem. The tuning problem is seen as a trade-off between speed and stability. The focus must be put on the platoon behaviour and the ability the system has to display good regulation of the inter-vehicular distances. All of the controller were tuned by trial and errors. When increasing the \(K_H\) parameter from the PI controller, the speed characterising the convergence to a steady state in the tracking increases. With this rapidity, the stability deteriorates, oscillations appear in the behaviour of the different vehicles in the platoon. The effect in bigger the further the vehicle is situated in the platoon, these behaviours are displayed in Figure 5.1. With the quicker controller exhibiting overshoot, the distance to the following vehicles can be led to problematic values, in the case of a deceleration, for example in figure 5.1 where a collision is obtained between vehicle 2 and vehicle 3. A compromise was reached with the value \(K_H = 0.1\).

Tuning becomes more complicated in the case of optimal controllers as the parameters are more numerous. The speed-reference given to the cruise-control of the vehicle is part of the state vector and is weighted in the \(Q\) matrix, the input weight \(R\) applies to increments of this speed reference. The braking system used through the interface controller introduced earlier induce an important model error for negative speed reference increments. When the weight \(R\) is close to one, quick variation of the speed reference tend to be observed which induce heavy use of the braking system. The braking system having a rather abrupt behaviour, the truck’s behaviour is highly affected and displays oscillations. The weigh on the speed reference itself is chosen equal to 0. It is not required to penalize the magnitude taken by this parameter as in steady-state we don’t expect it to be zero, but equal to the speed of the leading vehicle.
The speed profile of the vehicles becomes more and more oscillatory as we get closer to the tail of the platoon. Without platooning strategy, these oscillations are caused by a string effect. The information propagates from one vehicle to the other with a delay which needs to be compensated for. In the Figure 5.2, appears that for high values of the control increment weight $R$ these oscillations are more prominent. This statement applied to both the phases of acceleration and deceleration. The Figure 5.3 shows that in the case of control increment weight $R = 500$, the control increases with a lower rate than for a weight $R = 25$, the controller takes more time to react to the acceleration taken by the tracked-vehicle. This increases the delay between consecutive vehicle action. To compensate for this delay, the control is characterised by a bigger overshoot which explains the oscillatory behaviour.

The model used is characterised by two poles that we identified based on step responses of the system. The original thought was that the step response was measured for a given set-point of the ego-vehicle velocity and that several of these models could be used in a PWA approach. The poles of the model can however be adjusted like any other tuning parameter inherent to the controller. Among the three models identified in the Section 2.3 sys56 and sys1920 were characterized by quicker dynamics than sys1011, Figure 5.4 show that disparity in rise time implies disparity on the velocity profiles of the controllers. The controlled system based on sys1011 is quicker and more damped than the two other systems in the acceleration scenario but becomes more oscillatory in the deceleration phase in particular for the vehicle 4 at the tail of the platoon. The controllers based on sys56 and sys1920 show really close velocity profiles. The poles of the system model should be tuned to achieve stability and correct rapidity. The modelling used in this work is extremely simple. The model should only give some insight into the behaviour of the truck.

5.2 Platooning schemes

The scenarios used in these experimentations can be considered in a way as unfavourable to the platoon stability. The platoon leader follows here an abrupt control signal. It reaches the constant speed of $15m/s$ with the greatest acceleration possible and brakes abruptly after a while. This corresponds to a situation where a driver tries reach the speed limitations as soon as the corresponding traffic sign comes within sight. The vehicles following the platoon leader display more oscillations in their speed profile when no platooning strategy is used.

**PI controller** In the case of the PI controller, we stated that the platooning strategy consisting of considering an average weight of the vehicle ahead of the current vehicle would provide better performances both in case of accelerations and decelerations. We tested several combination of weights for a PI controller set with a $K_H$ parameter fixed at 0.1. The weights are given in the Table 5.4.

The controller labelled PI 3 is a PI controller without platooning strategy. PI 2 and PI 3 use the same strategy but with different sets of weight.
Table 5.4: Different choice of weights for PI controller platooning strategy.

<table>
<thead>
<tr>
<th>Controller</th>
<th>Third vehicle</th>
<th>Fourth vehicle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_{\text{leader}}$</td>
<td>$\alpha_{\text{vehicle}_2}$</td>
</tr>
<tr>
<td>PI 1</td>
<td>0.3</td>
<td>0.7</td>
</tr>
<tr>
<td>PI 2</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>PI 3</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

In the case of PI 1 the main contribution to the reference speed comes from the closest vehicle when it comes from the platoon leader in the second case. The results of the simulation relative to the speed is given Figure 5.5. The platooning strategy does bring an improvement to the speed regulation by decreasing the overshoot in speed and by thus reducing the settling time of the system. The impact is bigger as we observe a vehicle far away from the platoon leader. We limited here the platoon size to four vehicle but the effect of the platoon strategy are already noticeable in the case of the fourth vehicle. The weights introduced onto the velocity of the different vehicles in the platoon represent new tuning parameters to be taken into account. To guarantee a good tracking of the relative distance to the vehicle directly ahead of the ego-vehicle, we should keep the main part of the control effort as coming from the so-called vehicle. However, this is when the head of the platoon give the main contribution to the speed reference (LQ2) that the speed displays the smallest overshoot.

The effect on the relative position is not as noticeable. In the Figures 5.6 and 5.7, we display the positions of the different vehicles in the referential of vehicle 4 and the respective position reference to be tracked by every vehicle. We can see that the distance between two successive vehicles remain quite critical few seconds after the braking has begun. The effect of the platooning is here difficult to evaluate as a global trend cannot be observed regarding the different vehicles.
Figure 5.1: $K_H$ parameter impact on velocity (PI1, PI4 and PI5).
Figure 5.2: Velocity profiles for tuning of the input increment weight R of the LQ controller, (LQ5, LQ6, LQ7).
Figure 5.3: Control input comparison for different input increment weights, (LQ5, LQ6, LQ7).
Figure 5.4: System model comparison, for a given LQ controller (LQ2, LQ3, LQ4).
Figure 5.5: Velocity profiles for the three different set of weights in the platoon average speed strategy (PI1, PI2, PI3).
Figure 5.6: Platoon vehicles’ positions in vehicle 4’s referential, deceleration scenario (PI2: with average sped strategy).
Figure 5.7: Platoon vehicles' positions in vehicle 4’s referential, deceleration scenario (PI3: no average speed strategy).
**LQ controllers** Two versions of the LQ control were studied in this work. The difference between the two of them being the length of the prediction horizon which was infinite in one case, finite in the other. For a prediction horizon sufficiently big such as 150 samples for a sampling time of 0.1s, no difference between infinite horizon and finite horizon was noticeable. For a prediction horizon of 90 samples, the difference become more important. In the acceleration phase pictured in Figure 5.8, the difference are anecdotal but as pictured in Figure 5.8, oscillations that already existed in the infinite time-horizon case are now amplified right after braking of the leader vehicle. This could be compensated via tuning but is solved with an appropriate time horizon.

![Vehicle 2's speed depending on the prediction horizon used](image1)

![Vehicle 3's speed depending on the prediction horizon used](image2)

![Vehicle 4's speed depending on the prediction horizon used](image3)

Figure 5.8: Influence of the prediction horizon on speed profiles (LQ8,LQ11).
The position profile remains similar even with the effect of the limited time horizon. The error in position for the fourth vehicle becomes however important 30 seconds after braking, the oscillations don’t allow the vehicle 4 to get back to position after the braking phase which led it to increase the distance to the third vehicle. This behaviour will be amplified as vehicles are added to the platoon. The jump in position noticed at second 93 came from GPS jam introduced by the PreScan environment.

The two platooning strategies planned to be implemented concerning the LQ controllers were the Team AnnieWay’s strategy, or most restrictive control, and a strategy only implementable in the case of finite time-horizon formulations which consisted of assuming, in the prediction horizon, that the speed of the closest vehicle in the platoon would converge to the leader’s speed at a constant rate.

The case of the Team AnnieWay’s strategy was tested in the case of
both the finite and infinite time-horizon LQ and the same results were reported. As expected, the strategy of most restrictive control doesn’t change much the phases of acceleration where all vehicles try to catch up with the vehicles ahead of them, in an acceleration phase, the most restrictive control is almost always the control computed on the vehicle precisely ahead of the ego-vehicle. Different scenarios, such as the deceleration phase studied in the current examples however make use of the platoon strategy exposed here. In Figure 5.10, Figure 5.11 and Figure 5.12 are displayed results from the simulation of the controllers LQ9 and LQ10, designed identically if not considering the platooning strategy used. The Figures show that in the case of most restrictive control, the speeds of the third and fourth vehicle remain closer to the leader’s speed while decelerating. Taking the most restrictive control thus allows a less brutal braking but which is effected slightly earlier than in the case without strategy. The second vehicle doesn’t take any benefit from the strategy as the control always remain on the leader vehicle in his case. The position graph, in the referential of the fourth vehicle show that the position of every vehicle remain in both cases relatively close to the references. However, the platooning strategy allows the relative positions to remain globally constant when oscillations are bigger in the case without strategy. Without platooning strategy, the minimum size of the platoon in that phase is of 60 metres when the most restrictive control strategy doesn’t allow the minimum size under 68 metres.

It has to be noticed here, that the position profiles of the platoon is in the case of the LQ controllers presented here are characterised by much less oscillations than the ones displayed in the cases of all PI controllers presented previously an depicted in Figure 5.6 and Figure 5.7. This means that the distance to the fourth vehicle remain more stable. The positions are also much closer to their respective references. This comes from the fact that in the case of the LQ controllers, many parameters are available to tune the performance of the controlled system, when only one parameter is available in the case of the PI controllers. In the PI tuning, the tuning was ended when a good trade-off between damping and speed of the response in velocity was reached. In Figure 5.7, the total platoon length goes down to a value of roughly 25 meters some 20 seconds after inception of the braking phase. In the case of the LQ controller as studied in Figure 5.12, the total length of the platoon doesn’t get smaller than 60 meters in the case without platooning strategy.

The second platooning strategy described in this work concerned the finite time-horizon LQ controller, the data shown in the Figure 5.13 and Figure 5.14 was obtained using the same simulation process with four vehicles, using the controllers LQ5 and LQ9 meant to be identical apart form the strategy used. This strategy that we decided to call by the name of speed convergence strategy was devised as a way to adapt the average speed strategy used in the case of PI controllers, to the case of LQ controllers. A theoretical problems was encountered and prevented direct application of the so-called strategy. The efficiency this strategy depends greatly on the prediction horizon used for the finite time-horizon \( N \) used. In the case of LQ controllers, the reference vehicle is always the one directly ahead of the ego-vehicle. We assume in the speed convergence strategy that the reference vehicle’s speed will precisely be equal to the platoon leader’s speed at the
Figure 5.10: Inputs comparison with and without most restrictive control strategy (LQ9, LQ10).

end of the prediction horizon and this with a constant acceleration. If the prediction horizon is too big, the assumption done on speed will not be accurate enough as the platoon leader would probably have changed its speed before then. In these conditions, a significant positive effect on the platoon behaviour cannot be expected. This remark also applies in the case where the prediction horizon is too small in which case the acceleration needed for the reference vehicle to reach the platoon leader’s speed might be excessively big. To keep the prediction horizon close to the vehicle speed rise time might however remain efficient. As shown in the figures named here above, the result of this platooning strategy is a better damped speed profile in both the acceleration and deceleration phase in the case of vehicles far away in the platoon. The effect of the strategy on the speed of the vehicles situated immediately after the platoon leader should however be considered as negative. Oscillations quickly damped in the second vehicle’s speed profile at braking were indeed amplified. The distance between vehicle is greatly im-
vehicle 2’s speeds depending on the regulator used

vehicle 3’s speed depending on the regulator used

vehicle 4’s speed depending on the regulator used

Figure 5.11: Speeds comparison with and without most restrictive control strategy (LQ9, LQ10).

proved as shown in Figure 5.14. The vehicles remain closer to their reference position, and similarly to the case of the most restrictive control strategy, the platoon size remains superior to 78 meters at all time when it reached a value of 60 meters without platooning strategy. The advantage of this strategy is the positive effect observed in the acceleration phase which could not be observed in the case of the most restrictive control strategy. The platoon size pikes up to the value of 140 meters with the most restrictive control strategy when it’s contained under 120 meters with the platooning strategy introduced with the finite time-horizon. As a reminder, the size of the four-vehicle platoon should remain as close as possible to 81 meters at all time with a regular repartition in between all vehicles.

To summarize, the results of simulations exposed here show that the different platooning strategies proposed indeed increase the stability of the platoon. Oscillations in the velocities of vehicles far away in the platoon are decreased. This effect will be even more noticeable in longer platoons.
Figure 5.12: Position profiles comparison with and without most restrictive control strategy (LQ9, LQ10).

When no platooning strategy is used, during the acceleration phase, the second vehicle of the platoon tries to catch up with the first vehicle and displays an overshoot in velocity. The third vehicle will catch up with the second vehicle displaying an overshoot compared to the vehicle 2. Without any platooning strategy, the vehicles close to the tail of the platoon will display big overshoots in speed. This behaviour make oscillations appear in the speed and positions profile. The objective of the platooning strategies is to smooth out the control generated and dissipate these oscillations. In the context of the speed convergence platooning strategy, the ego-vehicle assumes that the tracked-vehicle’s speed will converge to the platoon leader speed before the end of the prediction horizon. This convergence corresponds to adding a very simple dynamics modelling relative to the reference vector. In a standard formulation of the LQ problem, the reference is considered as constant in the prediction horizon. This strategy is made possible
thanks to the structure of the finite time-horizon problem where the control is computed recursively based on the values taken by the reference vector at every sample of the prediction horizon. In the next paragraph we compare the behaviour of the three different classes of controllers and corresponding platooning strategy which proved to behave the best. The values of the different parameters characterising these three controllers are given in Table 5.5 and Table 5.6. The controller chosen as best member of the class of the PI controllers studied in this thesis is characterized by a parameter $K_H = 0.3$. Earlier we described that for the value of this parameter, the behaviour of the platoon regarding velocities was too conciliatory. Yet as shown in Figure reffig:LQstrat3, Figure reffig:LQstrat4, for values of the $K_H$ coefficient lower than 0.3, the tracking of the positions references becomes poorer. The tuning of the LQ controllers is facilitated by the multiplicity of the parameters to modify. In the PI controller, one lever only balances the whole system.

With the speed convergence strategy, the advantage over the most restrictive control which also smooth out the velocity profiles by looking at the speed of the vehicles ahead, is that the effect of this strategy is present both in the acceleration and deceleration phases. This appears in the Figure 5.15 where the controller LQ9 using the speed convergence strategy display a much smoother control than the LQ controller using most restrictive control and than the controller PI6, which is the PI controller used in this work which displayed the best behaviour of the class of PI controllers. However, when observing the velocity profiles in deceleration phase, as depicted in Figure 5.16, the most restrictive control manage to less abrupt. The acceleration of the different vehicles has a smaller magnitude in the case of most restrictive control. The consequence of this is a settling time increased which advantages to some extent the finite time horizon controller with speed convergence strategy. The advantage of speed convergence strategy over most restrictive control however appear in the comparison of the position profiles of the platoon’s vehicles as depicted in Figure 5.17. In the most restrictive control strategy, the tracked-vehicle does not remain the vehicle just ahead of the ego-vehicle. The controller switch on vehicles further ahead in the platoon when their speed becomes small. While switching, the relative position to the vehicle just ahead is disregarded and a control too restrictive is applied which lead the inter-vehicular distance to increase. The PI controller used in this comparison behaves comparably to the most restrictive controller for what concerns the relative positions in the platoon if we consider that in the deceleration phase, the minimum length of the platoon reaches 60 meters for these two controllers when the speed convergence strategy manages to keep the minimum distance above 75 metres. The same comment applies to the acceleration scenario where the speed convergence methods contains the platoon size inferior to 110 meters when this size reaches 150 meters in the case of the two other controllers as observed in Figure 5.18.

If new experimentations were allowed with the Scania vehicle, the finite time-horizon LQ control with platooning strategy would have to be implemented. The tuning phase will however give different results as the actual truck dynamics differs from this of the PreScan truck used in this study.
### Table 5.5: Controllers LQ9 and LQ10 parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>$N$</th>
<th>$Q_a$</th>
<th>$Q_v$</th>
<th>$Q_p$</th>
<th>$R$</th>
<th>model</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LQ 12</td>
<td>150</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>500</td>
<td>sys56</td>
<td>Convergence</td>
</tr>
<tr>
<td>LQ 13</td>
<td>$\infty$</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>75</td>
<td>sys56</td>
<td>Restrictive</td>
</tr>
</tbody>
</table>

### Table 5.6: Controller PI6 parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>$K_H$</th>
<th>Third vehicle</th>
<th>Fourth vehicle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_{leader}$</td>
<td>$\alpha_{vehicle_2}$</td>
<td>$\alpha_{leader}$</td>
</tr>
<tr>
<td>PI 6</td>
<td>0.3</td>
<td>0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

### 5.3 Experimental conclusion

In this chapter, controller tuning was realized based on trial and error approaches. The aim of this study was to show the differences in the behaviour of two particular controllers: the PI controller implemented in 2011 and the LQ controller, belonging to the class of optimal controllers. The conclusion taken from this study are that based on simulation, the use of an LQ controller will improve the performances of the platoon. This result was expected in the sense the PI controller is taken as a rudimentary controller and compared to the LQ controller which allows a more thorough way to adapt to the system through the modification of many parameters. The most restrictive control strategy was not used in the comparisons with the PI controller, we considered that the fact that the most restrictive control strategy is only useful in deceleration phases was limiting enough not to include more entities in a list already long of controllers.

The platooning strategies proved to be really efficient. This can be observed for example with the total length of the platoon which is always smaller than the nominal value in a deceleration phase and bigger in case of accelerations. The differences being way smaller when a platooning strategy is applied. The key is to keep the speeds of the different vehicles of the platoon as homogeneous as possible at all time. The platoon leader is the vehicle of the platoon displaying speed changing. Its task is to adapt its velocity to the speed signs encountered on the way and to avoid collision with the potential other vehicles on the lane. Using the platoon leader’s speed as a reference is thus beneficial to the platoon. The speed convergence strategy proved to be particularly efficient in the case of the LQ controller. It is to be noted, though, that by placing a big importance on the leader’s velocity reduces the considerations attributed to the other vehicles in the platoon, which in case of failure for one of the vehicles which is not the leader, could be detrimental from a safety point of view.

None of the controllers tested here can be certified safe. In the context of the CoAct, emergency measures were to be taken by a driver, permanently monitoring the distances to the the closest vehicle. The tests realized here were thought of with the same spirit.
Figure 5.13: Speed comparison with and without speed convergence strategy (LQ8, LQ9).
Figure 5.14: Position profiles comparison with and without speed convergence strategy (LQ8, LQ9).
Figure 5.15: Comparison of controllers with platooning strategies, velocities of the different vehicles in function of the controller used, acceleration scenario (PI6, LQ9, LQ10).
Figure 5.16: Comparison of controllers with platooning strategies, velocities of the different vehicles in function of the controller used, deceleration scenario (PI6, LQ9, LQ10).
Figure 5.17: Comparison of controllers with platooning strategies, positions of the different vehicles in function of the controller used, deceleration scenario (PI6, LQ9 and, LQ10).
Figure 5.18: Comparison of controllers with platooning strategies, positions of the different vehicles in function of the controller used, acceleration scenario (P16, LQ9 and LQ10).
Chapter 6

Conclusion

The master thesis work which lead to this report mainly focused on the implementation of an MPC controller for the platooning application exposed here. The implementation was unfortunately not successful and the different controllers presented and compared are LQ and PI controllers. Several methods and different solvers and toolboxes were tested for MPC.

MPC is known to be very computationally intensive, this tried to evaluate the possibilities of improvement brought by two options. The first one came from the idea that a plant simple enough and a tracking formulation as simple as possible could bring the computation time of the MPC control in an order of magnitude appropriate to the embedded computer used in the CoAct demonstration. This idea was proved wrong in the sense that even this most simple formulation of the MPC control could lead to oversized computation time and even infeasibility problems on neighbourhoods of constraints. These infeasibility problem are related to numerical issues intern to the solvers. A possible way to adapt the work presented here and make the formulation introduced run on the embedded computer used in the demonstrations, would be to work on the adaptation of complex and sophisticated solvers such as The IBM CPLEX solver which proved to be much more reliable than quadprog and CVX used in this work.

The second option which was explored concerned the case of explicit MPC. This form of MPC control was not thoroughly described in this report since no concrete successful implementation was obtained. The technical aspects of this method are complex, and we would refer to [14] and [15] for thorough descriptions. Given the problems encountered in the on-line formulations, this result can be expected in the sense the explicit computation corresponds to an off-line computation of all possible QP program given the values of the system’s state at all instant. The CPLEX solver was not successful in computing the explicit control rule for numerical reasons close to the ones encountered in online MPC with simpler solvers.

MPC controller is often compared to LQ fitted with constraint handling. This fact is true when the formulation of the MPC problem follows a quadratic cost function and a linear plant. While taking something other than a linear plant should not improve the computational aspects of MPC, taking a linear cost function decreases the complexity of the system and should be explored.

Beyond MPC, this thesis led to the concrete implementation of a PI controller which gave coherent performances when compared with the other
contestant of the CoAct demonstration. A different class of controllers and several platooning strategies were presented. These were not tested but represent a starting point for future experimentations. The PreScan simulation environment proved to be useful and helped simulating with models sufficiently close to reality to provide insights on the way the controller would work on the real system. This simulation environment provides a simple and user-friendly way to test implementations of different controllers and helps grasping the main stakes encountered in their tuning and adaptation. PreScan should be used as a way to proof-test a solution before trying to face the actual hardware.

The scope of this thesis was reduced to the use of overly simple models. This simplification was motivated by the study of MPC which requires models as simple as possible. However LQ control, evaluated in the Chapter 5 does not present the same computational problems but could have benefited from more complex system representations. Yet, the use of simple plants allows a simple solution to some implementation problems related to observation of the system’s state during the demonstration. The simulation environment used to test LQ and PI controllers do not require the controller’s computation to be done real time. This could allow to test MPC and evaluate the improvement brought by constraint handling in the context of vehicle platooning. The reason why MPC was not tested is that none of the implementation tested was protected from numerical infeasibility problems. Understanding the precise cause of the infeasibility in the QP solvers used in MPC control should be a priority in the future attempt of MPC implementation concerning vehicle platooning.
Bibliography


[5] Magnus Almroth *Simulation, validation and design of cooperative driving systems using PreScan*, Master’s Degree Project, 2013


[27] http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/, Hybrid System Toolbox

94