UNIVERSITÀ DEGLI STUDI DI CAGLIARI

Facoltà di Ingegneria

Dipartimento di Ingegneria Elettrica ed Elettronica

Delft University of Technology

Delft Center for Systems and Control

DESIGN OF AN ADAPTIVE CRUISE CONTROLLER VIA MPC/PWA-BASED METHODS

Relatori/Advisors:
Prof. Alessandro Giua,
Prof. Bart De Schutter,
Dr. Daniele Corona

Tesi di laurea specialistica
Master's Thesis di:
Nicola Manca

Corso di Laurea in Ingegneria Elettronica
Anno Accademico 2005/2006

A TUDelft che ha reso possibile
la realizzazione di questo lavoro,
ai miei supervisors e a chi ha
sempre creduto in me.

Abstract

An adaptive cruise controller (ACC) is a modern device of the automotive industry that aims to control the engine actuators so that the position and the speed of a leading vehicle are tracked. This project focuses on the comparison of two design methods for the ACC. In particular, a PI-based (proportional integral action) as it is currently used in the automotive industry, and an existing MPC-based (model predictive control) approach, developed for nonlinear and piecewise affine (PWA) systems, are compared. The former is a standard PI where the coefficients are optimally tuned off-line using a set of pre-designed curves. The second approach leads to a norm-1 mixed integer optimization problem that is solved on-line via a branch cutting strategy that prunes the tree of switching possibilities, reducing the number of linear programs. The added value of the MPC method is to allow the implementation of constraints that model physical specifications, safety/comfort issues, and mechanical stress of the vehicle. One of the most challenging aspect of this study is that the gear-shift and some of the engine/friction nonlinearities are included in the model of the system.

Sommario

Un adaptive cruise controller (ACC) è un dispositivo moderno dell'industria automotiva che si occupa di controllare gli attuatori del motore per poter inseguire la velocità e la posizione del veicolo leader. Questo tesi di laurea mette a confronto due metodi per l' ACC. In particolare vengono confrontati, un metodo basato sul PI (azione proporzionale integrale) come viene usato nell'industria automotiva e un metodo basato sull'MPC (model predictive control) sviluppato per sistemi non lineari e piecewise affine (PWA). Il primo è un PI standard nel quale i coefficienti vengono ottimizzati off-line usando un set di curve già note. Il secondo ci conduce ad un problema di ottimizzazione con norma 1 mixed integer che viene risolto on-line utilizzando una strategia che pota l'albero costruito su tutti i possibili eventi, riducendo il numero di problemi lineari. Il valore aggiunto dell'MPC riguarda la possibilità di inserire limitazioni sulle specifiche del veicolo, sicurezza, comfort e stress meccanico sugli organi del motore. Uno degli aspetti più importanti di questo lavoro riguarda l'inserimento di elementi non lineari come il cambio e l'attrito.

Contents

Chapter 1

Introduction

1.1 Adaptive cruise control

In the field of vehicle control, conventional cruise control systems have been available on the market for many years. During the last years, modern cars include more and more electronic systems. These systems are often governed by a computer or a network of computers programmed with powerful software. One of those new services is Adaptive Cruise Control (ACC), which extends the conventional cruise control system to include automated car following when the preceding car is driving at a different speed than the desired set-speed.

The conventional cruise control system (sometimes known as speed control or Autocruise) is a system to automatically control the speed of an automobile. The driver sets the speed and the system will take over the throttle of the car to maintain the same speed. Historically the conventional speed control with a centrifugal[1] governor was used in automobiles as early as the 1910s, notably by Peerless [2]. Peerless advertised that their system would

[1]A centrifugal governor is a specific type of governor that controls the speed of an engine by regulating the amount of fuel admitted, so as to maintain a near constant speed whatever the load or fuel supply conditions. It uses the principle of proportional control.

[2]Peerless was a United States automobile produced by the Peerless Motor Company of Cleveland, Ohio.

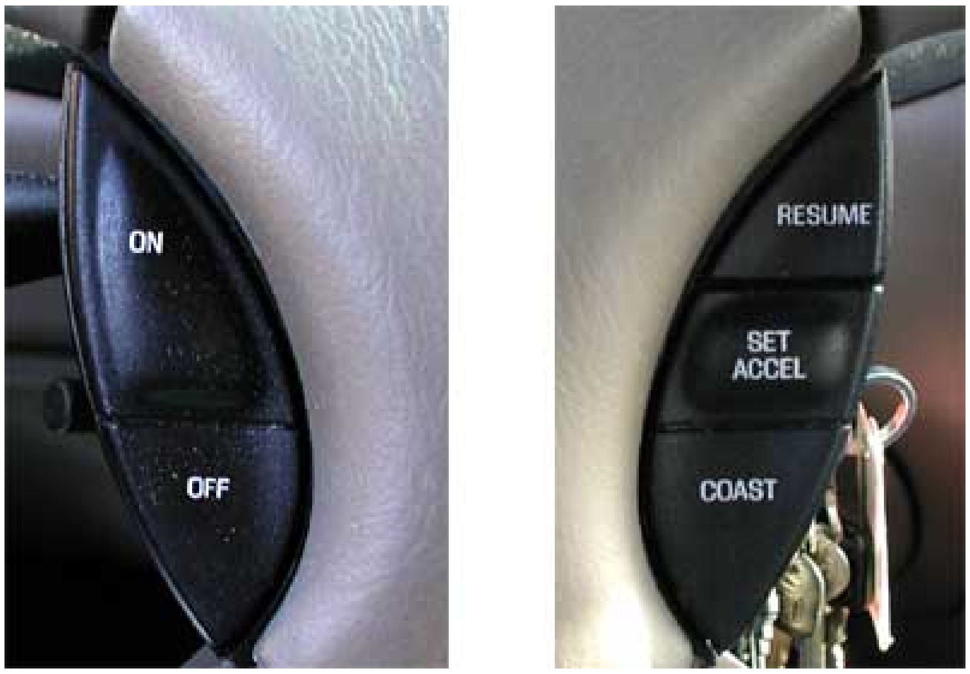


Figure 1.1: *Buttons of a classic cruise control.*

“maintain speed whether up hill or down”. The technology was invented by James Watt and Matthew Boulton in 1788 for use in locomotives. It uses centrifugal force to adjust throttle position as the speed of the engine changes with different loads (e.g. when going up a hill). Modern cruise control (also known as a speedostat) was invented in 1945 by the blind inventor and mechanical engineer Ralph Teetor. His idea was born out of the frustration of riding in a car driven by his lawyer, who kept speeding up and slowing down as he talked. The first car with Teetor’s system was the Chrysler Corporation Imperial in 1958. This system calculated ground speed based on driveshaft rotations and used a solenoid to vary throttle position as needed.

In modern designs, the cruise control may or may not need to be turned on before use, in some designs it is always “on” but not always enabled, others have a separate “on/off” switch, while still others just have an “on” switch that must be pressed after the vehicle has been started. Most designs have buttons for “set”, “resume”, “accelerate”, and “coast” functions. Some also have a “cancel” button. Alternatively, tapping the brake on most cars equipped with cruise control will disable it. The system is operated with controls easily within the driver’s reach, usually with two or more buttons on the steering wheel spokes or on the edge of the hub like those on Honda vehicles, on the turn signal stalk like in some General Motors vehicles or on a dedicated stalk like those found in Toyota and Mercedes-Benz vehicles. Early designs used a dial to set speed choice.

The driver must bring the car up to speed manually and use a button to set the cruise control to the current speed (an example image is shown in Figure 1.1 and in Figure 1.2). The cruise control takes its speed signal from a rotating driveshaft, speedometer cable, speed sensor (found on the wheels) or from the engine revolutions per minute (RPM). Most systems do not allow the use of the cruise control below a certain speed (normally 55 km/h) to discourage use in city driving. The car will maintain that speed by pulling the throttle cable with a solenoid or a vacuum driven servomechanism. On the latest vehicles fitted with electronic throttle control, cruise control can be easily integrated into the vehicle’s engine management system. Most systems can be turned off both explicitly and automatically, when the driver hits the brake or clutch. Cruise control often includes a memory feature to resume the set speed after braking and a coast feature to reset the speed lower without braking. When the cruise control is in effect, the throttle can still be used to accelerate the car, but once it is released the car will then slow down until it reaches the previously set speed.

The cruise control is independent of the environment, e.g. other vehicles on the road. When the vehicle ahead is traveling slower than the desired speed, the driver must at some point intervene with the brake pedal to avoid a collision. Alternatively, he must overtake the vehicle. The ACC concept extends the conventional cruise control system to include car following, see also [45], [16], [32]. In the scenario above, the ACC would have automatically lowered the speed of the car to match the speed of the vehicle ahead and to maintain an appropriate distance. If the preceding vehicle would have later on increased its speed, the ACC system would have automatically increased the speed (thereby following the car), unless it becomes greater than the desired speed set by the driver. Devices of ACC are currently being introduced by several car manufacturers in their latest car models. These systems consist of a sensor, mounted in the front of the car, that measures the preceding vehicle's velocity and distance. The sensor could either be of optical or radar type, but the radar sensor is often preferred since it is much less influenced by the weather conditions than the optical sensor. The sensor information is transmitted to an ACC controller (a computer) that controls the engine and brake systems. The first generation of ACC will only allow gentle acceleration and deceleration. A major reason for this is that the driver should never be surprised by the actions of the ACC system and the driver should always be able to intervene if the system does not comply with the driver's intentions. These systems will only work in highway traffic, where the required speed changes are moderate. The second generation of ACC will allow a greater acceleration and deceleration, which is necessary in suburban areas where the speed and distance is relatively low, but where the relative speed may be rather high. It is important to remember that the ACC is only a service to help the driver, not a replacement of the driver. The driver is still in charge of the car at any moment, regardless whether the ACC system is active or not [45].

A difficulty that arises in suburban areas is that it might be hard to find a model of the car dynamics that is good enough. Since the controller must be able to accelerate and decelerate quite hard, it must be carefully designed so it does not behave in an inconvenient manner which causes discomfort to the driver. There must not be any big overshoots and also if the acceleration is hard, it must be smooth. Hence, the design of the controller requires a good model of the car dynamics. But even if a good model is known, it still might be hard to design a suitable controller.

The advantages to use intelligent adaptation are manifold. As first it offers a comfortable optional to built in the cars and it gives an advantages

Figure 1.2: *An example of a cruise control on a Jeep Steering Wheel.*

for the driver. It is also well known that excessive vehicle speed on roadways is a major cause of traffic accidents, injuries, deaths. Additionally bad use of the vehicles often results in high fuel consumption and excessive pollutant emissions(e.g., carbon monoxide (CO), hydrocarbons (HC), and oxides of nitrogen (NOx)). This has been documented in many accident reports and safety studies [26], [49].

1.2 Model predictive control

Model predictive control (MPC) (also referred to as receding horizon control) is a control strategy that offers attractive solutions for the regulation of constrained linear or nonlinear systems and, more recently, also for the regulation of hybrid systems. Within a relatively short time, MPC has reached a certain maturity due to the continuously increasing interest shown for this distinctive part of control theory. Since 1970 various techniques have been developed for the design of model based control systems for robust multivariable control of industrial unit processes (see [6, 14, 15, 19–22, 46, 47, 53, 56],).

Predictive control was pioneered simultaneously by Richalet *et al.* [53] and Cutler and Ramaker [15]. MPC technology has evolved from a basic multivariable process control technology to a technology that enables operation of processes within well defined operating constraints.

One of the reasons for the fruitful achievements of MPC algorithms consists in the intuitive way of addressing the control problem. In comparison with conventional control, which often uses a pre-computed state or output feedback control law, predictive control uses a discrete-time model of the system to obtain an estimate (prediction) of its future behavior. At each discrete-time instant k, the measured variables and the process model (linear, nonlinear or hybrid) are used to (predict) calculate the future behavior of the controlled plant over a specified time horizon, which is usually called the prediction horizon and is denoted by N_p as depicted in Figure 1.3. This is achieved by considering a future control scenario as the input sequence applied to the process model, which must be calculated such that certain desired constraints and objectives are fulfilled. To do that, a performance oriented cost function is minimized subject to constraints, yielding an optimal sequence of controls over a specified time horizon, which is usually called control horizon and is denoted by N_c (see Figure 1.3). The feedback control law is then obtained in a receding horizon manner by applying to the system only the first element of the computed sequence of optimal controls, and repeating the whole procedure at the next discrete-time step.

MPC is built around the following key principles:

- The explicit use of a process model for calculating predictions of the future plant behavior;
- The optimization of an objective function subject to constraints, which yields an optimal sequence of controls;
- The receding horizon strategy, according to which only the first element of the optimal sequence of controls is applied on-line.

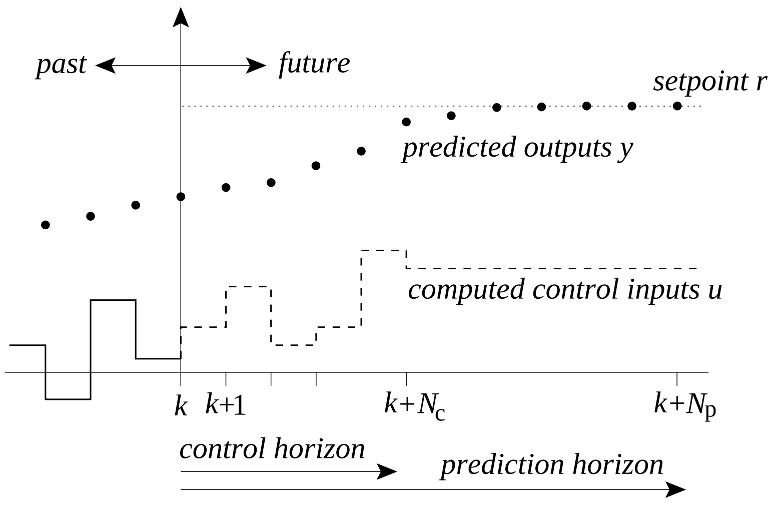
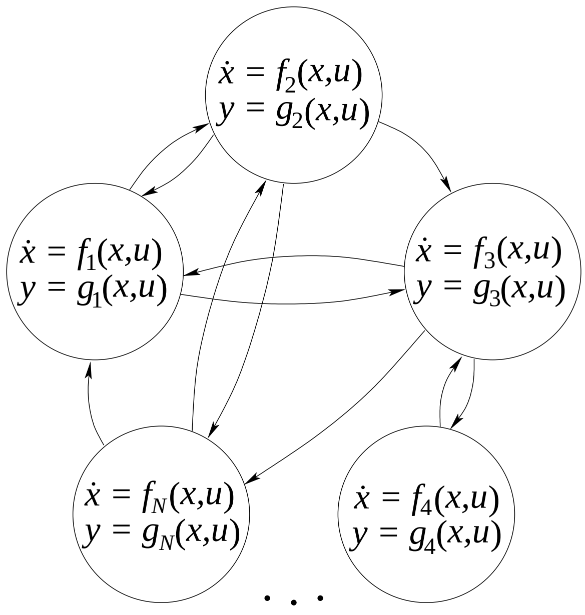


Figure 1.3: *A graphical illustration of Model Predictive Control [54].*

1.2.1 Stability

We distinguish two different cases, namely the unconstrained and constrained case. A predictive controller which is stable for the unconstrained case is not automatically stable for the constrained case. When inequality constraints are absent, the problem of stability can be easily obtained, being the controller a linear and time-invariant system (LTI). If we introduce an end-point constraints the closed-loop system becomes asymptotically stable. Since the end-point constraint is an equality constraint, it still holds that the resulting controller is linear and time-invariant. To guarantee stability we can also choose an infinite control orizon ($N_\mathrm{p} = \infty$) in the unconstrained case.

As reported in [59], the issue of stability becomes even more complicated in the case where constraints have to be satisfied. When there are only constraints on the control signal, stability can be ensured if the process itself is stable [58, 66]. In the general case with constraints on input, output and states, the main problem is feasibility. The existence of a stabilizing control law is not at all trivial. The two main results on stability in constrained predictive control are the papers by Rawlings and Muske [52] and Kothare *et al.* [38]. For more information about stability and robustness see also [40, 59].



$\dot{x} = f_2(x,u)$ $y = g_2(x,u)$
$\dot{x} = f_1(x,u)$ $y = g_1(x,u)$
$\dot{x} = f_3(x,u)$ $y = g_3(x,u)$
$\dot{x} = f_N(x,u)$ $y = g_N(x,u)$
$\dot{x} = f_4(x,u)$ $y = g_4(x,u)$
. . .

Figure 1.4: *Schematic representation of a hybrid system [54].*

1.3 Hybrid systems

Hybrid means, in general, heterogeneous in nature or composition and the term hybrid systems means systems with behavior defined by entities or processes of distinct characteristics. Hybrid[3] systems arise, for instance, from the combination of an analog continuous-time process and a digital time asynchronous controller. Hybrid dynamical systems generate variables or signals, that are mixed signals consisting of combinations of continuous or discrete value or time signals, and through them interaction with other systems and the environment occurs. More specifically, some of these signals take values from a continuous set (the set of real numbers) and others take values from a discrete, typically finite set (the set of symbols). Furthermore, these continuous or discrete-valued signals depend on independent variables such as time, which may also be continuous or discrete [1]. In general we could say that a hybrid system can be in one of several modes whereby in each mode

[3]Term used in this context for the first time by Witsenhausen in 1966, for describing the combination of continuous dynamical and discrete event systems [51].

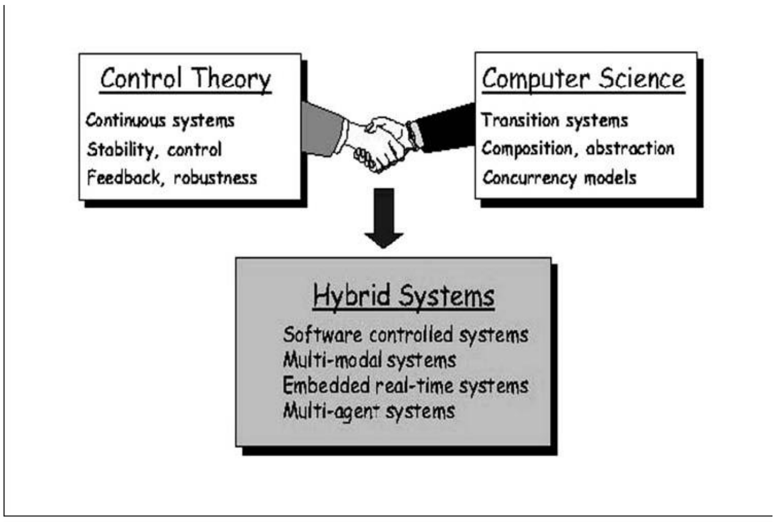


Figure 1.5: *The control community shakes hands with the computer science community to better understand the role and the benefits of hybrid systems [40].*

the behavior of the system can be described by a system of difference or differential equations, and that the system switches from one mode to another due to the occurrence of an event (see Figure 1.4) [54]. The investigation of hybrid systems is creating a new and fascinating discipline bridging control engineering, mathematics and computer science (see Figure 1.5). There has been significant research activity in the area of hybrid systems in the past decade involving researchers from several areas. It is not difficult to come up with interesting applications in the mechanical area: control of robotic manipulators driving nails or breaking objects [8], reduction of rattling in gear boxes of cars, drilling machines [50], simulation of crash-tests, regulating landing maneuvers of aircraft, design of juggling robots [9], and so on. Examples are not only found in the mechanical domain. Nowadays switches like thyristors and diodes are used in electrical networks for a great variety of applications in both power engineering and signal processing.

1.3.1 Hybrid systems classes

As described in [40] there are several approaches to model hybrid systems. Hybrid automata [7] can model a large class of hybrid systems as they consider a discrete event system where the continuous dynamics in each discrete state are modeled by an arbitrary differential (difference) equation. Such models are used in [7] to formulate a general stability analysis and controller synthesis framework for hybrid systems. Results for modeling and stability analysis of hybrid systems have also been presented in the more recent works [44], where dynamical properties of hybrid automata are investigated, and [29], where a new formalism based on hybrid time domains is defined for hybrid systems and it is employed to derive results on stability. These results are very important because they provide a unified view on solution concepts and stability theory for general hybrid systems (see also [61] for a comprehensive overview). However, a general model of hybrid systems, although it can capture a lot of situations (high modeling power), usually leads to a high level of complexity with respect to analysis and controller design techniques. Indeed, for a particular instance, a high analytical and computational complexity originates from the fact that the structure of particular classes of hybrid systems is not exploited. Therefore, in each choice of modeling formalism there is always the trade-off between the modeling power and the complexity of the analysis. That is why the research in hybrid systems also focuses on particular subclasses that have a simpler representation and more structure, but still include a wide range of industrially relevant processes. The following classes of hybrid systems are part of this category:

- Piecewise affine (PWA) systems [25, 36, 57];
- Switched systems [43];
- Mixed logical dynamical (MLD) systems [2, 4];
- Linear complementarity (LC) systems [33, 60];
- Discrete event systems extended to include time driven dynamics [10];

- Max-min-plus-scaling (MMPS) systems [55].

In particular, PWA systems have become popular due to their accessible mathematical description on one hand, and their ability to model a broad class of (hybrid) systems on the other hand: in [34] it has been proven that PWA systems are equivalent under certain mild assumptions with other relevant classes of hybrid systems, such as MLD systems, LC systems and MMPS systems. Also, it is well known that PWA systems can approximate nonlinear systems arbitrarily well [57] and they can arise from the interconnection of linear systems and automata [24]. The modeling power of PWA systems has already been shown in several applications, such as switched power converters [18], optimal control of DC-DC converters and direct torque control of three-phase induction motors [27], applications to automotive systems [3, 5, 62], and systems biology [23, 48], to mention just a few.

1.3.2 An example

A familiar simple example of a practical hybrid control system is the heating and cooling system of a typical home. The furnace and air conditioner, along with the heat flow characteristics of the home, form a continuous-time system, which is to be controlled. The thermostat is a simple asynchronous discrete-event system, which basically handles the symbols [`too hot`, `too cold`, `normal`]. The temperature of the room is translated into these representations in the thermostat and the thermostat's response is translated back to electrical currents, which control the furnace, air conditioner, blower, etc. There are several reasons for using hybrid models to represent dynamic behavior of interest in addition to the ones already mentioned. Reducing complexity was and still is an important reason for dealing with hybrid systems. This is accomplished in hybrid systems by incorporating models of dynamic processes at different levels of abstraction; in fact in this example the thermostat in the above example sees a very simple, but adequate for the task in hand, model of the complex heat flow dynamics [1].

1.4 Organization and contribution of this thesis

The thesis moves from the general concepts of hybrid systems to derive a physical model of a car (in particular a SMART) describing the gearshift and the friction as hybrid system. The hybrid model derived is then used as benchmark for two adaptive cruise controllers. The first is modeled using the model predictive control (MPC) and the second using a proportional and integral action (PI). Both controller are simulated, commented and compared. Finally it is given a comment on the comparisons and some further investigations. More specifically:

- The second chapter gives a mathematical description of the SMART and introduces the constraints on it. This chapter uses hybrid system theory to give the physical model of the car approximating the gearshift and the friction. At the end of the chapter we give an approximate PWA model, which is used to make the predictions. Moreover we provide a description of the ACC design goals, problematic and constraints.
- In Chapter 3, we present two methods used to build an adaptive cruise controller on the SMART. We first describe the model predictive control and the algorithm used to implement it. The algorithm [42] is based on branch and bound strategy which reduces the number of analyzed possible paths through a tree of possibilities. As second instance we describe the PI-based controller: how it is structured, how many parameters are needed and how we design the parameters.
- In Chapter 4, we enumerate five references scenarios used for the simulations:
    1. A constant references.
    2. A time-varying references.
    3. A constant references in the fifth/sixth switching gear value.

4. A sinusoidal varying velocity.
5. An emergency stop.

We expose the results for each scenarios using the two methods comparing the on-line number of variables used, the efficiency and other important results. Finally we comment and evaluate the results pointing out advantages & disadvantage for each scenario.

- In the last chapter, we give some final comments and we describe some of the open research topics. Firstly, we suggest to improve the accuracy of the model better describing the physical system. As second instance, we propose an idea to develop a more efficient MPC controller, improving the branch and bound algorithm. Finally, it is also necessary to implement this control strategy on a real car.

Chapter 2

Model description

2.1 Problem formulation

In this section we describe the model for the experimental set-up that will be used as a benchmark. The aim of an ACC is to ensure a minimal separation between the vehicles and a speed adaptation. In a basic ACC application two cars are driving one after the other (see Figure 2.1). In general platoons of cars can also be considered, see for instance [28], in a multi-agent framework, but here we restricted to the basic experimental condition of only two vehicles, allowing better insight into the physics of the global system with a reduced number of variables. We assume that the front vehicle communicates its speed and position to the rear vehicle, which has to track them as good as possible. So, for the control design purpose, only the dynamics of the rear vehicle can be considered.

An accurate model of the system considers the air drag proportional to the square of the speed and a constant *road-tire* static friction, proportional to the weight of the vehicle. The differential equation describing the dynamics of the rear vehicle is [12] :

$$m\ddot{s}(t) + (c\dot{s}^2 + \mu mg)\mathrm{sgn}(\dot{s}(t)) = b(j, \dot{s})u(t), \tag{2.1}$$

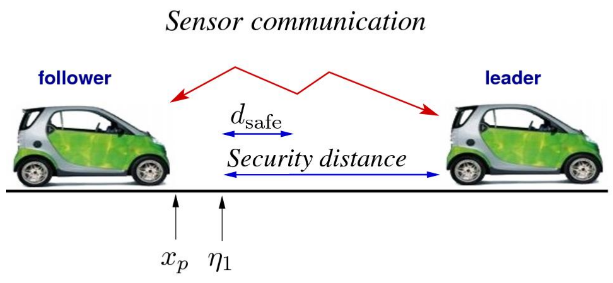


Figure 2.1: ACC set-up.

Parameter	Description	Numerical value	Unit
m	Mass of vehicle	800	kg
R	Average wheel radius	0.28	m
c	Viscous coefficient	0.5	kg/m
μ	Coulomb friction coefficient (dry asphalt)	0.01	—
g	Gravity acceleration	9.8	m/s^2
$w_{\min}$	Minimum engine rotational speed	105	rad/s
$w_{\max}$	Maximum engine rotational speed	630	rad/s
$p(j)$	Transmission rate	—	—
$b(j, \dot{s})$	Traction force	—	N
u	Throttle control action	—	—
T_e	Engine torque	—	Nm

Table 2.1: *Definitions and values of the entries of equation* (3.19).

where $s(t)$ is the position at time t, $\dot{s}(t)$ is the velocity , $\ddot{s}(t)$ is the acceleration, $b(j, \dot{s})u(t)$ is the traction force, proportional to the normalized throttle/brake position $u(t)$, considered as an input. The symbols m, c, μ and g, with their numerical values are listed in Table 2.1. The value of the function $\operatorname{sgn}(\dot{s}(t))$ is equal to 1, 0 or -1 when its argument is positive, null or negative respectively. Note that the traction force depends on the current gear $j = \{1, \ldots, 6\}$ and on the ground speed $\dot{s}(t)$. Numerical values for the SMART actually measured for a real vehicle are listed in Table 2.1. Additionally we provide the function $b(j, \dot{s})$ in Figure 2.3, obtained from the transmission ratio of the engine torque curve [64] in the engine rotational velocity $w \in [w_{\min}, w_{\max}]$, see Table 2.1. More specifically we have

$$b(j, \dot{s}) = \frac{T_{\mathrm{e}}(w)p(j)}{R}, \tag{2.2}$$

$$\dot{s} = \frac{wR}{p(j)}, \tag{2.3}$$

where $T_{\mathrm{e}}(w)$ is the engine torque, R is the average radius of the wheels, $p(j)$ represents the gear ratios. Here we have omitted the dependency on time t of s, w and j. The values of $p(j)$, including also the efficiency of the transmission from engine to wheel, are listed in Table 2.2. Since the maximal engine torque ($T_{\mathrm{e,max}} = 80\,Nm$) may be considered constant [64] in the range $w \in [200, 480]$, we also give the values of $b(j)$ in the last column of Table 2.2.

Braking will be simulated by applying a negative throttle. Due to friction behavior in motion inversion [30], model (3.19) is valid as long as the ground speed $\dot{s}$ is different from zero, hence we impose $\dot{s}$ to be above a nonzero minimum velocity as shown in Table 2.4.

2.2 Physical constraints

The physical construction of the engine components, result in defining constraints on the behavior of the system, that limit the maximum and the

Gear j	Transmission rate $p(j)$	Traction force $b(j)$ (N)
I	14.203	4057
II	10.310	2945
III	7.407	2116
IV	5.625	1607
V	4.083	1166
VI	2.933	838

Table 2.2: *Values of the transmission rate and of the maximum traction force transmitted on the wheels in a* SMART.

minimum velocity. More precisely, for $t \geq 0$,

$$v_{\min} \leq \dot{s}(t) \leq v_{\max} \tag{2.4}$$

with $v_{\min}$ and $v_{\max}$ values listed on Table 2.4.

Additionally are considering a vehicle with 6 gears, hence, for all $t \geq 0$ the constraint,

$$j(t) \in [1, 2, 3, 4, 5, 6] \tag{2.5}$$

must be specified.

Another important constraint regarding the gearshift is about the maximum and the minimum switching velocity for each gears. In Figure 2.4 using asterisks are reported the minimum velocity allowed for each of them, using stars the maximum. Between the two symbols is included the allowed interval for each gears, for example the velocity interval for the first gear is $[3.94m/s, 9.46m/s]$. The other exact value are reported in Table 2.6. We can formalize the constraint saying:

$$v_L(j) \leq \dot{s}(t) \leq v_H(j), \tag{2.6}$$

where $v_L(j), v_H(j)$ are the maximum and minimum velocity mentioned above.

Note that these constraints can not be violated because they represent the mechanical limitations of the vehicle.

Parameter	Description	Numerical Value	Unit
a_{dec}	Comfort deceleration	2	m/s^2
a_{acc}	Comfort acceleration	2.5	m/s^2
$u_{\max}$	Maximum throttle brake	1	-

Table 2.3: *Value of the physical constraints regarding comfort, economy and safety*

Parameter	Description	Numerical Value	Unit
$v_{\min}$	Minimum velocity	2	m/s
$v_{\max}$	Maximum velocity	40	m/s

Table 2.4: *Value of the physical constraints.*

We are interesting in model comfort, economy, safety and fuel consumption. Hence It is necessary to formulate some constraints.

The first fixes an upper and a lower limit for acceleration:

$$a_{\text{dec}} \leqslant \dot{v}(t) \leqslant a_{\text{acc}}, \tag{2.7}$$

with a_{dec} and a_{acc} in Table 2.3. A gradual increment and decrement of the velocity give to the driver and to the passengers a good sensation of comfort.

The second is on the control input :

$$|u(t)| \leqslant u_{\max}, \tag{2.8}$$

where the normalized value of $u(t)$ represents a throttle action if it is positive and a breaking action if negative. This constraint reduces the fuel consumption and increase the safety on board.

Finally the third forbids jumps of gears, that usually provoke non optimal fuel consumption in up-shifting and mechanical stress in down-shifting:

$$|j(t + dt) - j(t)| \leq 1, \tag{2.9}$$

where dt is a finite small time-interval.

2.3 State space representation and approximations

Let now we define

$$x \triangleq [s, \dot{s}]^{\mathrm{T}} = [x_p, x_v]^{\mathrm{T}}, \tag{2.10}$$

the state variable; with

$$\dot{x}_p = \dot{s}(t) = x_v = v,$$

the velocity, and

$$\dot{x}_v = \ddot{s}(t) = a,$$

the acceleration.

Consequently, we are able to represent the system as:

$$\begin{bmatrix} \dot{x}_p \\ \dot{x}_v \end{bmatrix} = f(x,u) = \begin{bmatrix} x_v \\ -\frac{c}{m}x_v^2 - \mu g \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{b(j,x_v)}{m} \end{bmatrix} u(t). \tag{2.11}$$

Note that this model is nonlinear because of the quadratic term of the friction and of the traction force, and hybrid, because of the dependencies of b from the gear value.

Model (2.11) contains both state variables. To reduce the complexity of the simulation model we have chosen to consider only the contribution of the velocity. This led to reduced the amount of time needed to write the matlab code for the MPC controllers. Anyway to avoid any collisions it is possible to insert a constraints which does not allow the follower velocity to pass the leader velocity, but for our test this is to much restrictive and it will not be implemented. We intend to use the following model as simulation tool, for the tested controllers (the MPC and the PI):

$$\dot{x}_v = f(x,u) = -\frac{c}{m}x_v^2 - \mu g + \frac{b(j,x_v)}{m} \cdot u(t). \tag{2.12}$$

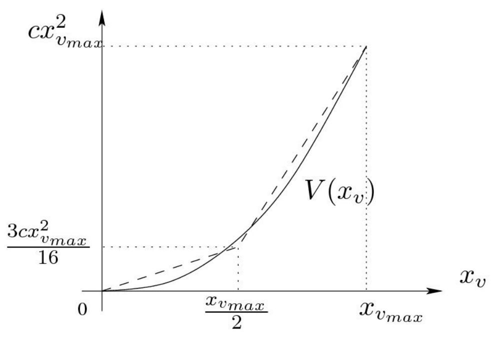


Figure 2.2: Nonlinear friction (solid) and PWA approximation (dashed).

constant	value
c_1	7.5
c_2	32.5
f_1	0
f_2	-500

Table 2.5: *Friction PWA approximation values.*

2.3.1 Friction nonlinearity approximation

The quadratic term of the friction introduces a nonlinearity, that can be tackled using the PWA approximation in Figure 2.2:

$$\frac{c}{m}x_v^2 \approx \begin{cases} -c_1 x_v + f_1 & x_v < \frac{v_{\max}}{2} \\ -c_2 x_v + f_2 & x_v \geq \frac{v_{\max}}{2} \end{cases}$$

with c_1, c_2, f_1, f_2 defined as in Table 2.5. Thus the model becomes:

$$f(x,u) \approx \begin{cases} -c_1 x_v + f_1 - \mu g + \frac{b(j,x_v)}{m}u(t) & x_v < \frac{v_{\max}}{2} \\ \\ -c_2 x_v + f_2 - \mu g + \frac{b(j,x_v)}{m}u(t) & x_v \geq \frac{v_{\max}}{2} \end{cases} \tag{2.13}$$

The first mode $i = 1$ is active when $v(t) < \frac{v_{\max}}{2}$ instead the second one $i = 2$ is active when $v(t) \geq \frac{v_{\max}}{2}$.

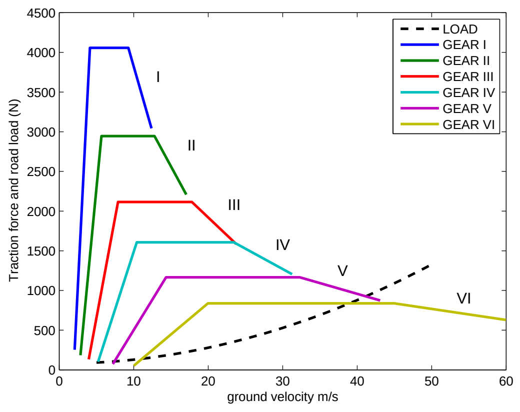


Figure 2.3: *Traction force transmitted to the wheel at maximum throttle input for different gears.*

2.3.2 Gearshift modeling

The admissible range of velocities for the given gear is explained in equation (2.6) and in Figure 2.4. Note that the switching condition is not uniquely defined, thus different gears are admitted for a specific value of the speed. For our interest we need a strategy to approximate the inequality (2.6) to avoid overlap and to assign for each velocity value a different gear. We choose to use an approximation that aims to emphasize the gear usage on the higher speed in order to meet efficiency of the engine defined as follow:

$$v_0 + v_1 j \leq x_v \leq v_0 + v_1(j+1), \tag{2.14}$$

which preserves linearity and a one-to-one relation between velocity and current gear. Within this condition the approximation depends only on the choice of the two values v_0 and v_1. Depicted as squares in Figure 2.4 there are the lower limit used for each gears and as rhombus the maximum values. In our model $v_0 = -7$, $v1 = 7$.

Consequently considering the b_j values from Table 2.2 we can write:

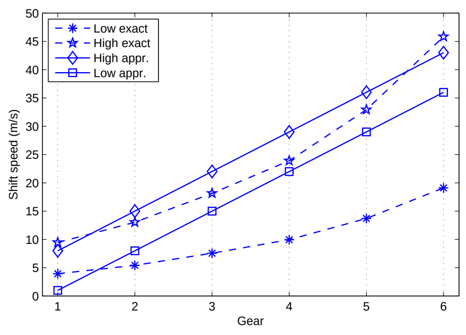


Figure 2.4: *Approximation of switching velocities for different gears.*

Gear	Min. Velocity (m/s)	Max. Velocity (m/s)
I	3.94	9.46
II	5.43	13.04
III	7.56	18.15
IV	9.96	23.90
V	13.70	32.93
VI	19.10	45.84

Table 2.6: *Ground velocity switching condition per gear shift position.*

$$f(x,u) \approx \begin{cases} -c_1 x_v + f_1 - \mu g + \frac{b_1}{m} u(t) & v_0 + v_1 \leq x_v \leq v_0 + 2v_1 \\ -c_1 x_v + f_1 - \mu g + \frac{b_2}{m} u(t) & v_0 + 2v_1 \leq x_v \leq v_0 + 3v_1 \\ -c_1 x_v + f_1 - \mu g + \frac{b_3}{m} u(t) & v_0 + 3v_1 \leq x_v \leq v_0 + 4v_1 \\ -c_2 x_v + f_2 - \mu g + \frac{b_4}{m} u(t) & v_0 + 4v_1 \leq x_v \leq v_0 + 5v_1 \\ -c_2 x_v + f_2 - \mu g + \frac{b_5}{m} u(t) & v_0 + 5v_1 \leq x_v \leq v_0 + 6v_1 \\ -c_2 x_v + f_2 - \mu g + \frac{b_6}{m} u(t) & v_0 + 6v_1 \leq x_v \leq v_0 + 7v_1, \end{cases} \tag{2.15}$$

this models the PWA approximation of the simulation model in equation (2.11).

2.4 The discrete time system

In the previous section we have derived the PWA model of the system :

$$\dot{x}_2 = c_i \cdot x_v + \frac{b_j}{m} \cdot u(t) + f_i - \mu g, \tag{2.16}$$

the discrete-time representation of (2.16) is

$$x_v(k+1) = A_i x_v(k) + B_j u(k) + F_i, \tag{2.17}$$

with $i \in [1,2]$; $j \in [1,2,3]$ when $i = 1$ and $j \in [4,5,6]$ when $i = 2$. This is shown also in the equation (2.15) where we have chosen the upper bound velocity of the third modes from the gearshift approximation equal to the upper bound velocity of the first mode of the friction approximation. For this reason we have six modes in total.

2.4.1 Numerical values

The numerical values of the matrices in equation (2.17) are:

$$
\begin{aligned}
A_1 &= 0.9907\\
A_2 &= 0.9602\\
B_1 &= 5.0476\\
B_2 &= 3.6640\\
B_3 &= 2.6326\\
B_4 &= 1.9685\\
B_5 &= 1.4283\\
B_6 &= 1.0265\\
F_1 &= -0.0975\\
F_2 &= 0.5164,
\end{aligned}
\tag{2.18}
$$

we have used sampling time T = 1s with *zero order hold* method.

2.5 Summary

In this chapter it was introduced and explained the model used for the control design. First it was given a physical model of the vehicle (SMART) with its constraints, then, due to the nonlinearity of friction and the integer variables of the gearshift, it was approximated as a PWA system. The discrete time model used in this thesis was also given.

Chapter 3

Implementation approaches

3.1 Hybrid MPC

3.1.1 Cost function minimization

The hybrid control signal $u(k), j(k)$ is designed by solving a constrained finite time optimal control problem, in an MPC receding horizon fashion. In this framework the prediction or acquisition of N_p samples ahead of the front vehicle trajectory is used to compute the optimal control law $u(k), j(k)$. The MPC approach is largely used to design the control action of constrained systems and in particular PWA systems (see [4,17,37,41] to cite a few). The control action is obtained by solving

$$\min_{\tilde{u}(N_\mathrm{p}),\tilde{\jmath}(N_\mathrm{p})} J(\theta(k), \tilde{u}(N_\mathrm{p}), \tilde{\jmath}(N_\mathrm{p})) \triangleq \\ \sum_{i=1}^{N_\mathrm{p}} ||Q_x\varepsilon(k+i)||_1 + ||Q_{\Delta u}\Delta u(k+i-1)||_1 + ||Q_{\Delta j}\Delta j(k+i-1)||_1, \tag{3.1}$$

subject to the particular prediction model that will be described in the sequel and the constraints deriving from physical and design specifications as described in Section 2.2. Note that we are interested to minimize the number of gear switchings Δj, the variation of the control input Δu and the error with a given reference trajectory communicated by the leading vehi-

$$\begin{tikzcd}
g_1 \arrow[loop above] \arrow[r, bend left] & g_2 \arrow[loop above] \arrow[l, bend left] \arrow[r, bend left] & g_3 \arrow[loop above] \arrow[l, bend left] \arrow[r, bend left] & g_4 \arrow[loop above] \arrow[l, bend left] \arrow[r, bend left] & g_5 \arrow[loop above] \arrow[l, bend left] \arrow[r, bend left] & g_6 \arrow[loop above] \arrow[l, bend left]
\end{tikzcd}$$

Figure 3.1: *Gearshift rules represented by an automaton.*

cle. Here $\varepsilon(k) \triangleq v(k) - \eta_v(k)$ is the tracking error (where $\eta_v(k)$ is the speed of the leader vehicle), $\tilde{u}(N_\mathrm{p}, k) \triangleq [u(k), \ldots, u(k + N_\mathrm{p} - 1)]^\mathrm{T}$ the sequence of control inputs, Q_x, $Q_{\Delta u}$ and $Q_{\Delta j}$ are weight matrices of appropriate dimension, $\theta(k)$ is a set of parameters containing the initial conditions and the N_p prediction of the reference trajectory. In this application we have $\theta(k) \triangleq [v(k), u(k-1), j(k-1), \eta_v(k+1), \ldots, \eta_v(k+N_\mathrm{p})]^\mathrm{T}$.

Additionally an appropriately tuned shorter control horizon $N_\mathrm{c} < N_\mathrm{p}$ may also be studied, i.e., $\Delta u(k+i) = 0$, $i = N_\mathrm{c}, \ldots, N_\mathrm{p} - 1$. This choice has the general advantage of reducing the number of variables and of providing a smoother solution. Nevertheless here we only consider $N_\mathrm{p} = N_\mathrm{c}$. The choice of the 1-norm in (3.1) offers a valid trade-off between the complexity of the optimization problem and the quality of the solution.

3.1.2 System constraints

The minimization of this cost function is subject to the constraints described in Section 2.2, which are formulated with a set of inequalities to model comfort, economy, safety and other limitations.

We can summarize the expressions of the constraints set in discrete as:

$$\begin{gathered}
v_\mathrm{min} \leq v(k) \leq v_\mathrm{max} \\
a_\mathrm{dec} T \leqslant v(k+1) - v(k) \leqslant a_\mathrm{acc} T \\
-u_\mathrm{max} \leq u(k) \leq u_\mathrm{max} \\
1 \leq j(k) \leq 6 \\
-1 \leq j(k+1) - j(k) \leq 1,
\end{gathered} \tag{3.2}$$

where k is the discrete time and T is the sampling time.

3.1.3 Linear problems enumeration

The simplest way to find the searched optimum consists in enumerating all the possible sequences of modes over the prediction horizon, solve all the LP subproblems with the corresponding sequences, compare them, and choose the optimum. For a given sequence of modes, two situations may occur: either there is no solution satisfying the constraints (infeasibility) or the LP subproblem is feasible. Let $I_{N_\mathrm{p}} = (i(0) \;\; i(1) \;\; \cdots \;\; i(N_\mathrm{p}-1))^T \in \mathbb{I}^{N_\mathrm{p}}$ be a sequence of modes on the horizon N_p, for each feasible sequence I_{N_p} an optimal continuous control sequence $\tilde{u}^*(N_\mathrm{p})$ and a corresponding cost $J^*_{N_\mathrm{p}}(\theta(k), I_{N_\mathrm{p}})$ can be obtained. The searched optimum is then given by the sequence of modes which minimizes (3.1):

$$J^*_{N_\mathrm{p}}(\theta(k)) = \min_{\tilde{\jmath}(N_\mathrm{p}),\tilde{u}(N_\mathrm{p},k)} (J_{N_\mathrm{p}}(\theta(k), \tilde{u}(N_\mathrm{p}), \tilde{\jmath}(N_\mathrm{p}))) \tag{3.3}$$

This method, called exhaustive enumeration, becomes untractable when the number of modes and/or the length of the prediction horizon increases because it leads to solve a number of linear subproblems that grows exponentially with N_p and with the numbers of the modes. In fact the problem is exponential [42]. For a given depth, the width of the tree is fixed by the number of possible modes for the system. Each leaf is LP subproblem and one of them is the searched optimum. The sequences of modes construction must follow the automaton reported on Figure 3.1 because it represents the physical constraints as reported in (2.4). The total number of LP subproblems is upper bounded by $3^{N_\mathrm{p}-1}$ because if the current discrete state at time k correspond to the second, the third, the fourth or the fifth, the possible discrete state at time $k+1$ takes values from a set of three elements. The $3^{N_\mathrm{p}-1}$ expression arises considering that in this worst case we have 3 LP for the first discrete depth of the tree ($P = 1$), $3 \cdot 3$ LP for $P = 2$ (because we have three new LP for each precedent nodes), and so on until $P = N_\mathrm{p}$. Consistently the total number it is lower bounded by $2^{N_\mathrm{p}-1}$ because if the current discrete state at time k corresponds to the first or the sixth gear the possible discrete state at time $k+1$ takes values from a set of two elements;

Current state δ	lower gear	same gear	higher gear
g_1	–	g_1	g_2
g_2	g_1	g_2	g_3
g_3	g_2	g_3	g_4
g_4	g_3	g_4	g_5
g_5	g_4	g_5	g_6
g_6	g_5	g_6	–

Table 3.1: *Transition Table for the automaton in Figure 3.1.*

first and last line in Table 3.1. This concept is more clear looking at the automaton in Figure 3.1 and the transition Table 3.1.

Automatically a tree of possibilities arises and its size depends on the horizon length. Each final leaf represents an LP problem with the relative gear-path and where each intermediate node is a partial path having its corresponding cost function; an example of this tree is shown in Figure 3.2 where we are considering the first gear as initial discrete mode.

3.1.4 PWA/MPC branch cutting method

Exhaustive enumeration consists of completely covering the tree of possibilities. In order to reduce the computational effort the exhaustive search can be pruned. The method was proposed by S. Leirens [42] and it is based on *branch and bound*(B&B) [39]. B&B is a general algorithmic method for finding optimal solutions of various optimization problems, especially in discrete and combinatorial optimization. It is basically an enumeration approach in a fashion that prunes the nonpromising search space. The general idea may be described in terms of finding the minimal or maximal value of a function $f(x)$ over a set of admissible values of the argument x called feasible region. Both f and x may be of arbitrary nature. A B&B procedure requires two tools.

The first one is a smart way of covering the feasible region by several

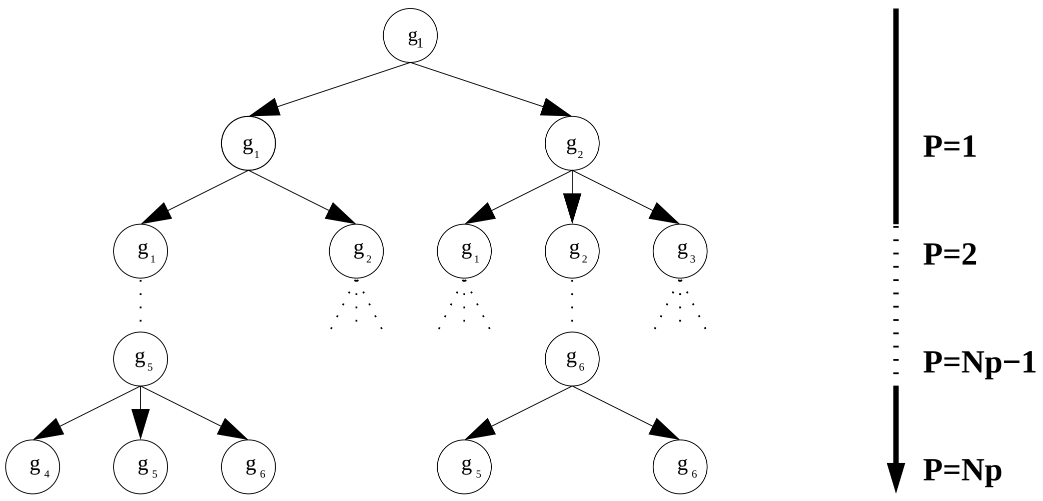


Figure 3.2: *Tree of possibilities for the event driven evolution following the graph in Figure 3.1.*

smaller feasible subregions (ideally, splitting into subregions). This is called branching, since the procedure may be repeated recursively to each of the subregions and all produced subregions naturally form a tree structure, called search tree or B&B tree or something similar. Its nodes are the constructed subregions.

Another tool is bounding, which is a fast way of finding upper and lower bounds for the optimal solution within a feasible subregion.

For a given problem space an efficient partition will divide the solution space into a small set containing high value (or low cost) solutions to be examined more closely and a larger set of their opposites (those to be ignored). While desirable, efficient divisions are often difficult to achieve in practice and so the creation of an effective algorithm is highly dependent upon the nature of the problem to be solved and the skill of the analyst creating the algorithm.

The core of the approach is a simple observation that (for a minimization task) if the lower bound for a subregion A from the search tree is greater than the upper bound for any other (previously examined) subregion B, then A may be safely discarded from the search. This step is called pruning. It is usually implemented by maintaining a global variable m that records the minimum upper bound seen among all subregions examined so far; any node whose lower bound is greater than m can be discarded.

It may happen that the upper bound for a node matches its lower bound; that value is then the minimum of the function within the corresponding subregion. Sometimes there is a direct way of finding such a minimum. In both these cases it is said that the node is solved. Note that this node may still be pruned as the algorithm progresses.

Ideally the procedure stops when all nodes of the search tree are either pruned or solved. At that point, all non-pruned subregions will have their upper and lower bounds equal to the global minimum of the function. In practice the procedure is often terminated after a given time; at that point, the minimum lower bound and the maximum upper bound, among all non-pruned sections, define a range of values that contains the global minimum. Alternatively, within an overriding time constraint, the algorithm may be terminated when some error criterion such as (max-min)/(min + max) falls below a specified value.

The efficiency of the method depends critically on the effectiveness of the branching and bounding algorithms used; bad choices could lead to repeated branching, without any pruning, until the sub-regions become very small. In that case the method would be reduced to an exhaustive enumeration of the domain, which is often impractically large. There is no universal bounding algorithm that works for all problems, and there is little hope that one will ever be found; therefore the general paradigm needs to be implemented separately for each application, with branching and bounding algorithms that are specially designed for it.

In our case the optimization problem associated with the predictive control has a particular structure: the cost is additive with positive terms. The key idea of the suggested partial enumeration algorithm consists in finding in an efficient way a a sub-optimum partial cost in order to prune the tree *i.e.*, cut branches that cannot lead to the optimum.

For a partial horizon $P(1 \leq P \leq N_{\mathrm{p}})$, *i.e.*, at a depth P in the tree, a partial cost is defined as follows:

$$\begin{aligned} J_P(\theta(k), \tilde{u}(N_{\mathrm{p}}, k), \tilde{\jmath}(N_{\mathrm{p}})) = \textstyle\sum_{i=1}^{P} ||Q_x \varepsilon(k+i)||_1 + ||Q_{\Delta u} \Delta u(k+i-1)||_1 + \\ ||Q_{\Delta j} \Delta j(k+i-1)||_1. \end{aligned} \tag{3.4}$$

The partial horizon P is a pointer, a cursor which moves itself from the first useful node of the tree ($P = 1$) until the end of the tree ($P = N_{\mathrm{p}}$), as shown in Figure 3.2. The top of the tree has a P cursor equal to zero because it does not represent a partial horizon. In fact a model predictive control with a horizon equal to zero does not exist.

The proposed approach is a recursive algorithm which is composed of a descent strategy to explore the tree of possibilities and a criterion of branch cutting. If we suppose to have the tree described in Section 3.1.3, we can summarize the method in two big steps.

Algorithm 1: Find the sub-optimum

INPUT: the tree of possibilities, N_{p} (the prediction horizon).
OUTPUT: J^*_{sub} (the sub-optimum cost).

Define $\tilde{\jmath}_P = [j_1, ..., j_P]$ as valid vector sequence for the automaton in Figure 3.1. We also define $\tilde{J}_{list}$ as a vector to store a list of needed partial cost function.

1. The pointer P is initialized with $P = 1$. The vector $\tilde{\jmath}^*_{P-1}$is initialized with the discrete label of the node at the top of the tree, $\tilde{\jmath}^*_{P-1} = j(k)$.

2. We define $\tilde{\jmath}_P = [\tilde{\jmath}^*_{P-1},\ succ(j_{P-1})]$ every possible path with fixed $\tilde{\jmath}^*_{P-1}$ and "floating[1]" last element tree

3. Minimize (3.4) as follows:

$$J^*_P(\theta(k)) = \min_{\tilde{\jmath}_P}(\min_{\tilde{u}_P} J_P(\theta(k), \tilde{u}_P, \tilde{\jmath}_P)), \tag{3.5}$$

[1]Floating in the sense that it is determined by the automaton in Figure 3.1.

exhaustively over every possible successors of (j_{p-1}). We add all these results to the vector $\tilde{J}_{list}$ because we will need them in step 2.

4. We define $\tilde{\jmath}_P^*$ the sub-optimum path vector as:

$$\tilde{\jmath}_{P-1}^*, \arg(\min_{\tilde{\jmath}_P}(\min_{\tilde{u}_P} J_P(\theta(k), \tilde{u}_P, \tilde{\jmath}_P))), \tag{3.6}$$

5. If $P = N_\mathrm{p}$ go to *step f*, else go back to *step b* and increment $P = P + 1$.

6. The vector $\tilde{\jmath}_P^*$ is completed. It became $\tilde{\jmath}_{N_\mathrm{p}}^*$, it is the path which reach the sub-optimum leaf. The last $J_{P_l}^*$ stored is the ***sub-optimum cost***, we call it J_{sub}^*.

■

An example is depicted in Figure 3.3 where we indicate the sub-optimum path in red bolded, we indicate the final sub-optimum leaf with a bolded circle. On the right side of each node is reported the partial cost function value calculated using (3.4).

- At the level $P = 1$ we have $\tilde{\jmath}_{P-1}^* = \{g_1\}$. The choice is here between a cost of 2 or a cost of 3. Since the lower is 2 (g_1) we update $\tilde{\jmath}_P^* = \{g_1, g_1\}$.
- At the level $P = 2$ the lower is 3 (g_1 again). Consequently $\tilde{\jmath}_P^* = \{g_1, g_1, g_1\}$.
- At this level we cut a part of the tree to give a full explanation. The missing part is depicted with a series of dots.
- At the level $P = N_\mathrm{p}$ we must chose between 9 (g_4), 8 (g_5) and 7 (g_6). It is intuitive than the final $\tilde{\jmath}_P^* = \tilde{\jmath}_{N_\mathrm{p}}^* = \{g_1, g_1, g_1, ..., g_5, g_6\}$ and $J_{sub}^* = 7$.

For a distract lector it seems now that the algorithm could end. But at level $P = 1$ the $\tilde{\jmath}_P^* = \{g_1, g_2\}$ has a partial cost function ($J = 3$) lower than the $J_{sub}^* = 7$ found in 6. It means that it is possible to find a lower sub-optimum exploring that part of the tree. This is explained in the next algorithm.

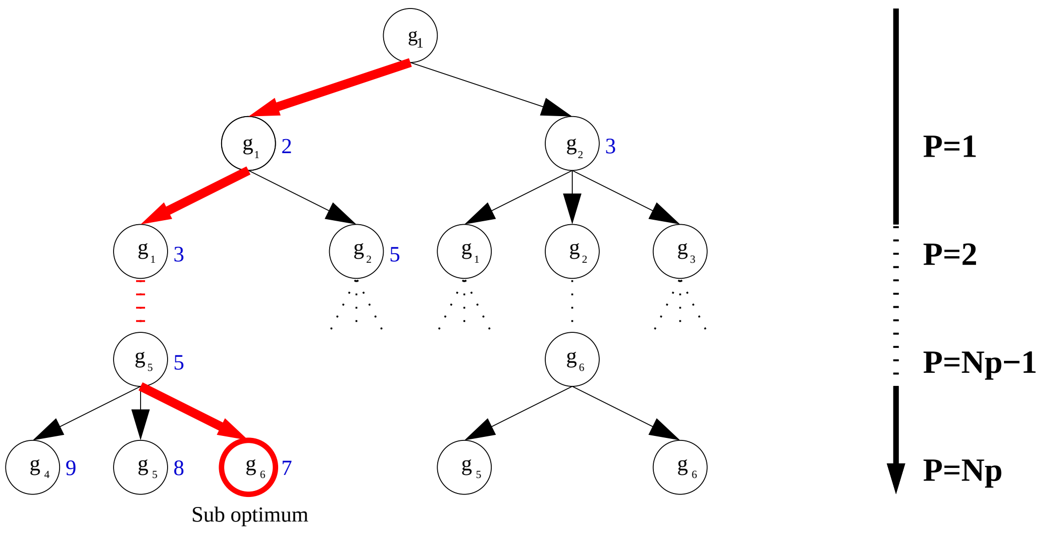


Figure 3.3: *First step of the B&B method.*

Algorithm 2: Find the optimum

INPUT: the tree of possibilities, N_p (the prediction horizon), J^*_{sub} (the sub-optimum cost).
OUTPUT: J^*_{opt} (the sub-optimum cost), $\tilde{\jmath}^*_{N_\mathrm{p}}$ (the optimum path vector).

Define q as a cursor which moves through the vector $\tilde{J}_{list}$. Each $\tilde{J}_{list}(q)$ value is a J^*_P partial cost mentioned in step 3 of algorithm 1. At every $\tilde{J}_{list}(q)$ vector corresponds a properly $\tilde{\jmath}_P(q)$ which is the partial sub-optimum path.

1. Initialize $q = 1$.

2. If $J^*_{sub} > \tilde{J}_{list}(q)$ we start to explore the part of the tree identified by the $\tilde{\jmath}_P(q)$ path.

   - If $P \neq N_\mathrm{p}$, minimize (3.4) as follows:

$$J^*_P = \min_{\tilde{\jmath}^*_P(q)}(\min_{\tilde{u}_P} J_P(\theta(k), \tilde{u}_P, \tilde{\jmath}^*_P(q))), \tag{3.7}$$

     exhaustively over every possible successors of $\tilde{\jmath}_P(q)$. We compare the obtained values with J^*_{sub}. For each $J^*_P \geq J^*_{sub}$ values or with infeasible minimization we cut that branch identified by $\tilde{\jmath}^*_P(q)$ because we do not need to explore it. We add each $J^*_P < J^*_{sub}$ values into the $\tilde{J}_{list}$ vector.

- If $P = N_\mathrm{p}$, we update the sub-optimum value $J^*_{sub} = \tilde{J}_{list}(q)$ and its relative $\tilde{\jmath}^*_{N_\mathrm{p}}$ path it is the ***new sub-optimum final path***.

3. If $J^*_{sub} \leq \tilde{J}_{list}(q)$ or $J^*_{sub} = \tilde{J}_{list}(q)$ and $P \neq N_\mathrm{p}$ or the minimization (3.7) is infeasible, then we cut that branch identified by $\tilde{\jmath}_P(q)$
4. If $J^*_{sub} = J^*_P$ and $P = N_\mathrm{p}$we use the ordering criterion in Section 3.1.5.
5. $q = q + 1$
6. If $q \leq length(\tilde{J}_{list})$ go back to step b.
7. The last J^*_{sub} stored is the ***optimum cost*** J^*_{opt}. Consequently its relative $\tilde{\jmath}^*_{N_\mathrm{p}}$ path vector is our ***optimum path***.

■

This second algorithm can be explained with the aim of the Figure 3.4 where we indicate the final sub-optimum leaf with a bolded red circle, the optimum path in green bolded and the optimum leaf with a bolded green circle. On the right side of each node is reported the partial cost function value calculated using (3.4). At this point $\tilde{J}_{list} = 3$ relative to $\tilde{\jmath}_P = [g_1, g_2]$ and $J^*_{sub} = 7$.

- $q = 1$
- $J^*_{sub} > \tilde{J}_{list}(1)$, we start to explore $[\tilde{\jmath}_P, g_1]$, $[\tilde{\jmath}_P, g_2]$, $[\tilde{\jmath}_P, g_3]$.
- The first has $J^*_P > J^*_{sub}$: we cut that branch with a cross.
  The second has $J^*_P < J^*_{sub}$and $P \neq N_\mathrm{p}$: $\tilde{J}_{list} = [\tilde{J}_{list}, J^*_P]$.
  The third has $J^*_P > J^*_{sub}$: we cut that branch with a cross.
- $q = 2$
- $length(\tilde{J}_{list}) = q$

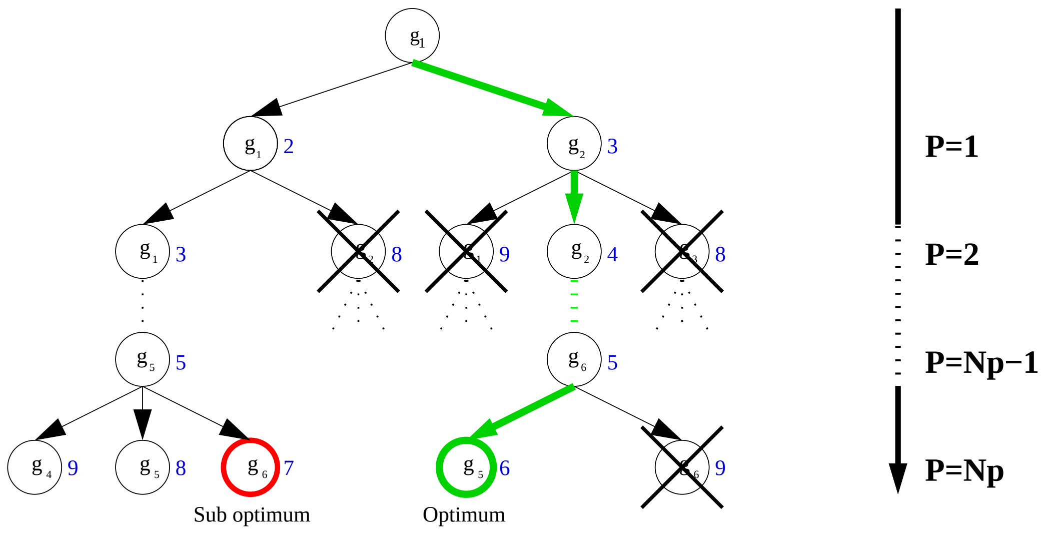


Figure 3.4: *Second step of the B&B method.*

- At this level we cut a part of the tree to give a full explanation. The missing part is depicted with a series of dots.
- $J^*_{sub} > \tilde{J}_{list}(N_p - 1)$, we start to explore $[\tilde{\jmath}_P, g_5]$, $[\tilde{\jmath}_P, g_6]$.
- The first has $J^*_P < J^*_{sub}$ and $P = N_p$: we update the sub-optimum value $J^*_{sub} = \tilde{J}_{list}(2)$ and its relative $\tilde{\jmath}^*_{N_p}$ path it is the ***new sub-optimum path***.
  The second has $J^*_P > J^*_{sub}$: we cut that branch with a cross.
- $q = 3$
- $length(\tilde{J}_{list}) < q$
- the last stored $J^*_{sub} = 6$ is the optimum cost J^*_{opt} and its relative $\tilde{\jmath}^*_{N_p} = \{g_1, g_2, g_2, ..., g_6, g_5\}$ path vector is our ***optimum path***.

3.1.5 Ordering criteria

In the particular case where the cost functions at two nodes of equal deepness (P) have the same value we must formulate a criterion to choose between two of them. Before giving the ordering criterion, let us call J_1 and J_2, with $J_1 = J_2$, the costs corresponding respectively to the node N_1 and the node N_2. We will denoted also the corresponding paths as $\tilde{\jmath}_1$ (reft., node N_1) and $\tilde{\jmath}_2$ (reft., node N_2). The ordering criterions is based on three rules and is given by the following algorithm.

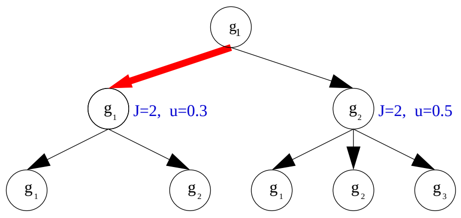
Algorithm 3: Ordering criteria

1. Compare u_1, u_2 where $u_1 = argmin(J_1)$, $u_2 = argmin(J_2)$.
   - If $u_1 > u_2$, we choose N_2.
   - If $u_1 < u_2$, we choose N_1.
   - If $u_1 = u_2$, go to the next step.
2. Compare $\tilde{\jmath}_{1_P}$ and $\tilde{\jmath}_{2_P}$ with $\tilde{\jmath}_{P-1}$.
   - If $\tilde{\jmath}_{1_P} = \tilde{\jmath}_{P-1}$ and $\tilde{\jmath}_{2_P} \neq \tilde{\jmath}_{P-1}$, we choose N_1.
   - If $\tilde{\jmath}_{2_P} = \tilde{\jmath}_{P-1}$ and $\tilde{\jmath}_{1_P} \neq \tilde{\jmath}_{P-1}$, we choose N_2.
   - If $\tilde{\jmath}_{1_P} = \tilde{\jmath}_{2_P} = \tilde{\jmath}(P-1)$, go to the next step.
3. Compare $\tilde{\jmath}_{1_P}$ with $\tilde{\jmath}_{2_P}$
   - If $\tilde{\jmath}_{1_P} > \tilde{\jmath}_{2_P}$, we choose N_2.
   - If $\tilde{\jmath}_{1_P} < \tilde{\jmath}_{2_P}$, we choose N_1.

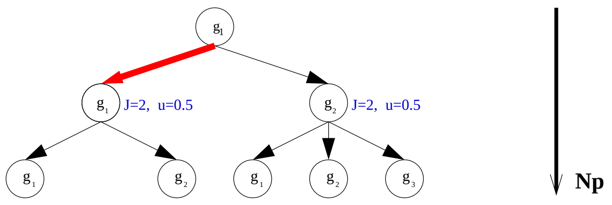
■

The rules come from physical considerations. They are enumerated in order of importance. The first prefers a smaller control. The second prefers as next gear the same of the last one reducing the switching number of the gears. The third chooses the lower gear to improve the security on board.

For example in Figure 3.5(a) the same cost function values ($J = 2$) for the two possible nodes is found. Hence we look at the control values to choose the best path. If it happens that also the control values are the same, we



(a) *Choose on the lower control*



(b) *Choose reducing the gears switching.*

Figure 3.5: *An example of the ordering criterion.*

choose the best path comparing the last gear as showed in Figure 3.5(b), where it is preferred the $1-1$ instead than the $1-2$ path.

3.1.6 Implementation example

Consider the following example with an horizon $N_{\mathrm{p}} = 3$ and choose as initial discrete state the first gear we need the enumerate all the state equation using (2.17) until reached the horizon:

$$\begin{cases} x_v(k+1) = A_i x_v(k) + B_j u(k) + F_i \\ x_v(k+2) = A_i x_v(k+1) + B_j u(k+1) + F_i \\ x_v(k+3) = A_i x_v(k+2) + B_j u(k+2) + F_i \end{cases} \tag{3.8}$$

Substituting $x_v(k+1)$ in $x_v(k+2)$ and then $x_v(k+2)$ in $x_v(k+3)$:

$$\begin{cases} x_v(k+1) = A_i x_v(k) + B_j u(k) + F_i \\ x_v(k+2) = A_i(A_i x_v(k) + B_j u(k) + F_i) + B_j u(k+1) + F_i \\ x_v(k+3) = A_i(A_i(A_i x_v(k) + B_j u(k) + F_i) + B_j u(k+1) \\ +F_i) + B_j u(k+2) + F_i \end{cases} \tag{3.9}$$

We do not know what will be the gear at $k+1$ and $k+2$ so we need to enumerate all the possibilities following the automaton switching rule on Figure 3.1. With $N_{\mathrm{p}} = 3$ the possible paths are:

$g_1 - g_1 - g_1$

$g_1 - g_1 - g_2$

$g_1 - g_2 - g_1$

$g_1 - g_2 - g_2$

$g_1 - g_2 - g_3$

If we use these possible paths to write all the systems remembering (2.15), we have the five following evolutions:

$$1)\begin{cases} x_v(k+1) = A_1 x_v(k) + B_1 u(k) + F_1 \\ x_v(k+2) = A_1(A_1 x_v(k) + B_1 u(k) + F_1) + B_1 u(k+1) + F_1 \\ x_v(k+3) = A_1(A_1(A_1 x_v(k) + B_1 u(k) + F_1) + B_1 u(k+1) + \\ F_1) + B_1 u(k+2) + F_1 \end{cases}$$

$$2)\begin{cases} x_v(k+1) = A_1 x_v(k) + B_1 u(k) + F_1 \\ x_v(k+2) = A_1(A1 x_v(k) + B_1 u(k) + F_1) + B_1 u(k+1) + F_1 \\ x_v(k+3) = A_2(A_1(A_1 x_v(k) + B_1 u(k) + F_1) + B_1 u(k+1) + \\ F_1) + B_2 u(k+2) + F_2 \end{cases}$$

$$3)\begin{cases} x_v(k+1) = A_1 x_v(k) + B_1 u(k) + F_1 \\ x_v(k+2) = A_2(A_1 x_v(k) + B_1 u(k) + F_1) + B_2 u(k+1) + F_2 \\ x_v(k+3) = A_1(A_2(A_1 x_v(k) + B_1 u(k) + F_1) + B_2 u(k+1) + \\ F_2) + B_1 u(k+2) + F_1 \end{cases}$$

$$4)\begin{cases} x_v(k+1) = A_1 x_v(k) + B_1 u(k) + F_1 \\ x_v(k+2) = A_2(A_1 x_v(k) + B_1 u(k) + F_1) + B_2 u(k+1) + F_2 \\ x_v(k+3) = A_2(A_2(A_1 x_v(k) + B_1 u(k) + F_1) + B_2 u(k+1) + \\ F_2) + B_2 u(k+2) + F_2 \end{cases}$$

$$5)\begin{cases} x_v(k+1) = A_1 x_v(k) + B_1 u(k) + F_1 \\ x_v(k+2) = A_2(A_1 x_v(k) + B_1 u(k) + F_1) + B_2 u(k+1) + F_2 \\ x_v(k+3) = A_3(A_2(A_1 x_v(k) + B_1 u(k) + F_1) + B_2 u(k+1) + \\ F_2) + B_3 u(k+2) + F_3 \end{cases} \tag{3.10}$$

In general we have:

$$\begin{cases} x_v(k+1) = f_1(x_v(k), \tilde{u}_{N_\mathrm{p}}, \tilde{\jmath}_{N_\mathrm{p}}) \\ x_v(k+1) = f_2(x_v(k), \tilde{u}_{N_\mathrm{p}}, \tilde{\jmath}_{N_\mathrm{p}}) \\ x_v(k+1) = f_3(x_v(k), \tilde{u}_{N_\mathrm{p}}, \tilde{\jmath}_{N_\mathrm{p}}) \end{cases} \tag{3.11}$$

where f_1, f_2, f_3 are linear expression of $\tilde{u}_{N_\mathrm{p}}$. Substituting in (2.12) we obtain:

$$||\min F(x_v(k), \tilde{u}_{N_\mathrm{p}})|| \tag{3.12}$$

subject to the set of constraints described in section 2.2. If we choose this specific sequence $\tilde{\jmath}_{N_\mathrm{p}} = [g_1, g_2, g_3]$ the above equation is a LP.

3.2 Proportional integral action

This a classic proportional-integral-derivative (PID) controller with the absence of the derivative actions. In general the controller takes a measured value from a process or other apparatus and compares it with a reference setpoint value. The difference (or "error" signal) is then used to adjust some input to the process in order to bring the process' measured value to its desired setpoint. Unlike simpler controllers, the PID can adjust process outputs based on the history and rate of change of the error signal, which gives more accurate and stable control. The "error" below is found by subtracting the measured quantity from the setpoint. "PI" is named after its two correcting terms, whose sum constitutes the output of the PI controller.

1. **Proportional**: To handle the immediate error, the error is multiplied by a constant P (for "proportional").
2. **Integral**: To learn from the past, the error is integrated and multiplied by a constant I (for "integral").

3.2.1 General description

As reported in (2.10) we define

$$x = [x_p, x_v]^\mathrm{T}, \tag{3.13}$$

as state vector where x_p is the position and x_v is the velocity of the follower vehicle. We recall, as described in section 3.1.1, the reference trajectory η, defined as follow

$$\eta = [\eta_p, \eta_v]^\mathrm{T}, \tag{3.14}$$

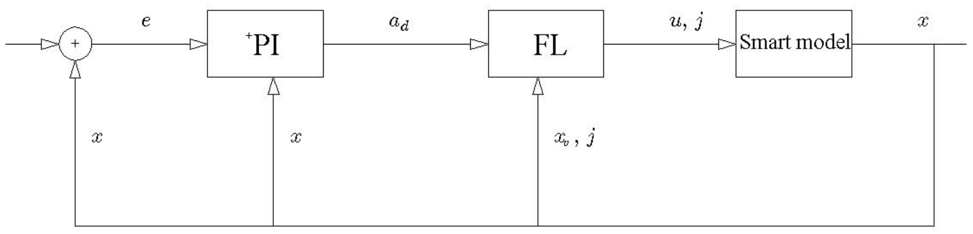


Figure 3.6: *Block diagram of the PI control system.*

where respectively η_p and η_v are the position and the velocity of the leader vehicle. Referring to the Figure 3.6, the summatory computes the velocity error

$$e_v = \eta_v - x_v, \tag{3.15}$$

and the position error

$$e_p = \eta_p - x_p. \tag{3.16}$$

The PI block uses this error to calculate the desired acceleration a_d using:

$$a_d = K_{dv} \cdot e_v + (K_{dx1} \cdot K_{dx2}) \cdot e_p. \tag{3.17}$$

In the above equation compare three parameters, K_{dv}, K_{dx1}, K_{dx2}. In the next section 3.2.2 it is explained how we choose and tune them. The desired acceleration is sent to the block FL, to calculate the control input as:

$$u \cdot b(j, \dot{s}) = m \cdot a_d + c\dot{s}^2 + \mu mg \tag{3.18}$$

It is fed into the Smart model block with the correspondent gear which integrates (3.19)

$$m\ddot{s}(t) + (c\dot{s}^2 + \mu mg)\text{sgn}(\dot{s}(t)) = b(j, \dot{s})u(t), \tag{3.19}$$

using the `Matlab ode45` subroutine. The output is the current state (3.13) with the current gear looped in feedback to the previous blocks.

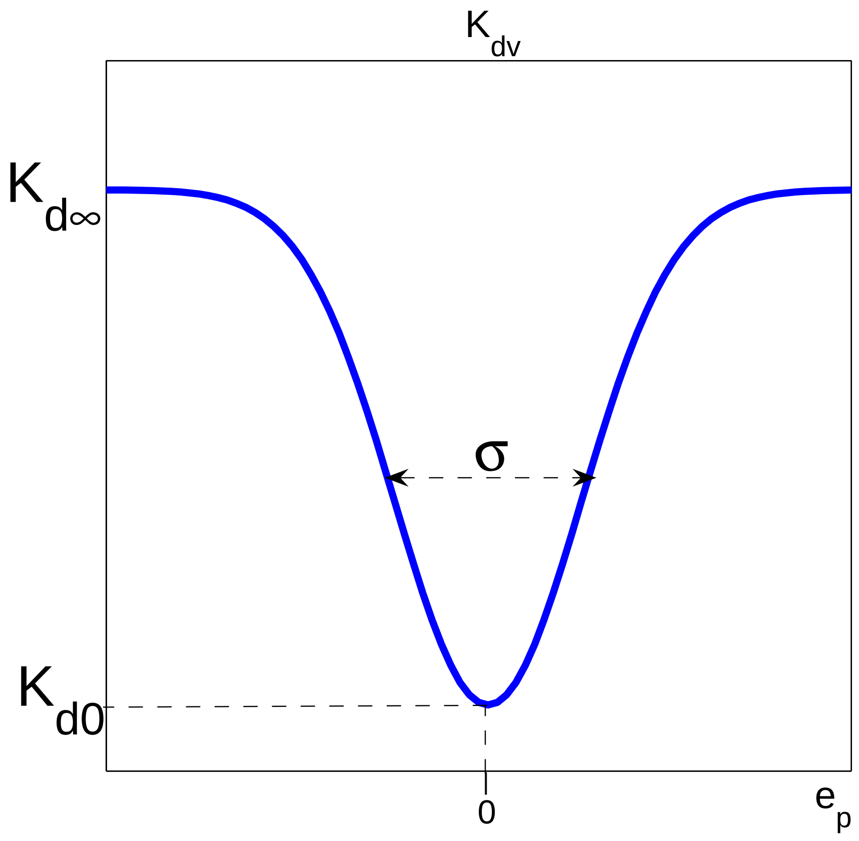


Figure 3.7: *Function k_{dv} versus its variables. The $\sigma, k_{d0}, k_{d\infty}$ are to be optimally tuned.*

3.2.2 Optimization of parameters

Better performance of the basic PI can be obtained by adapting the constants with the state/error value. This is done [65] by introducing some correction functions which depend on 3 parameters.

$$
\begin{aligned}
K_{dx1} &= K_{dx1_\infty} + (K_{dx1_0} - K_{dx1_\infty}) \cdot e^{-\sigma_{dx1} \cdot x_v^2}, \\
K_{dx2} &= K_{dx2_\infty} + (K_{dx2_0} - K_{dx2_\infty}) \cdot e^{-\sigma_{dx2} \cdot e_p^2}, \\
K_{dv} &= K_{dv_\infty} + (K_{dv_0} - K_{dv_\infty}) \cdot e^{-\sigma_{dv} \cdot e_p^2}.
\end{aligned}
\tag{3.20}
$$

As an example K_{dv} is depicted in Figure 3.7.
The nine parameters to be tuned are :

parameter	value
K_{dv_∞}	0.71
K_{dv_0}	0.26
σ_{dv}	0.007
K_{dx1_∞}	1.1
K_{dx1_0}	0.6
σ_{dx1}	0.3
K_{dx2_∞}	0.14
K_{dx2_0}	0.18
σ_{dx2}	0.026

Table 3.2: *Tuned set of parameters.*

1. K_{dv_∞}
2. K_{dv_0}
3. σ_{dv}
4. K_{dx1_∞}
5. K_{dx1_0}
6. σ_{dx1}
7. K_{dx2_∞}
8. K_{dx2_0}
9. σ_{dx2}

They are chosen optimizing the closed loop response. The parameters are obtained by solving

parameter	value
K_p	0.91
K_I	0.55

Table 3.3: *Ziegler-Nichols parameters.*

$$\min_{(K_{dx1},K_{dx2},K_{dv})} J(K_{dx1}, K_{dx2}, K_{dv}) \triangleq \sum_{i=1}^{isim} ||Q_x \varepsilon(k+i)||_1 + ||Q_{\Delta u} \Delta u(k+i-1)||_1 + ||Q_{\Delta j} \Delta j(k+i-1)||_1, \tag{3.21}$$

where $isim$ is the length of the simulation and the other terms are the same of the equation (3.1) in the chapter 3.1.1. The values of the weight matrix are in table 4.1. As reference trajectory η_{ref} for the optimization we chose the nominal scenario with constant velocity, see Section 4.2.

As initial values for these parameters we had a tuned set listed in Table 3.2. This configuration was obtained using a different experimental model and for this reason we are not allowed to reuse them for our model (3.19), for which they may not be optimal. For this reason we needed an offline part to compute a new optimal set of parameters. We set a search range tuning the PI coefficients via Ziegler-Nichols(ZN) method

$$a_d = K_{p_{ZN}} \cdot e_v + K_{I_{ZN}} \cdot e_x. \tag{3.22}$$

The Ziegler-Nichols values are reported in Table 3.2.2. Exploring variations around the ZN solution may result in better performance. To this purpose we fix a range on the 9 parameters.

The range is established using a mixed criterion, using ZN values and tuned set values.
More precisely

- For K_{dv_∞} and K_{dv_0}, we centered the interval on $K_{p_{ZN}}$. The lower bound (LB) is $K_p - 20\%$ and the upper bound (UB) is $K_p + 20\%$

upper bound	value	lower bound	value
$K_{dv_\infty}M$	1.365	$K_{dv_\infty}m$	0.455
$K_{dv_0}M$	1.365	$K_{dv_0}m$	0.455
$\sigma_{dv}M$	1000	$\sigma_{dv}m$	0
$K_{dx1_\infty}M$	0.37	$K_{dx1_\infty}m$	-0.22
$K_{dx1_0}M$	0.46	$K_{dx1_0}m$	-0.27
$\sigma_{dx1}M$	1000	$\sigma_{dx1}m$	0
$K_{dx2_\infty}M$	2.2	$K_{dx2_\infty}m$	-1.27
$K_{dx2_0}M$	1.8	$K_{dx2_0}m$	-1.03
$\sigma_{dx2}M$	1000	$\sigma_{dx2}m$	0

Table 3.4: *Upper and lower bound limits to search the optimal values for the parameters.*

- σ_{dv}, σ_{dx1} and σ_{dx2} represent the width of the bell shaped curve in Figure 3.7. They can variate between 0 and 1000.
- Upper bound and lower bound for K_{dx1_∞}, K_{dx1_0}, K_{dx2_∞} and K_{dx2_0} are derived from the following system

$$\begin{cases} K_{dx1_\infty}(UB) \cdot K_{dx2_\infty}(UB) = K_{I_{ZN}} + 50\% \\ K_{dx1_0}(UB) \cdot K_{dx2_0}(UB) = K_{I_{ZN}} + 50\% \\ K_{dx1_\infty}(LB) \cdot K_{dx2_\infty}(LB) = K_{I_{ZN}} - 50\% \\ K_{dx1_0}(LB) \cdot K_{dx2_0}(LB) = K_{I_{ZN}} - 50\% \\ K_{dx2_\infty}(UB) = K_{\infty_{set}} \cdot K_{dx1_\infty}(UB) \\ K_{dx2_0}(UB) = K_{0_{set}} \cdot K_{dx1_0}(UB) \\ K_{dx2_\infty}(LB) = K_{\infty_{set}} \cdot K_{dx1_\infty}(LB) \\ K_{dx2_0}(LB) = K_{0_{set}} \cdot K_{dx1_0}(LB), \end{cases} \tag{3.23}$$

where $K_{\infty_{set}}$ and $K_{0_{set}}$ are respectively 5.88 and 3.84. They are derived comparing the relative bell shaped set tuned curve. In Table 3.4 we report the intervals for all nine parameters.

We have done the search using two methods, each method has the first step different but the second in common:

1. (a) **Monte Carlo algorithm:** We ran 5000 simulations with random values choosing from the intervals in Table 3.4. Hence we choose the best set.

   (b) **Random line search:** The search fixes a random direction long which we minimize, this minimum fixes a new starting point to determine again a random direction until 5000 evaluations are made. More precisely the problem is

   $$\min_{\theta} f(\theta)$$

   $$s.t$$

   $$\theta_m \leq \theta \leq \theta_M$$

   where $f(\theta)$ is the cost of evolution over the nominal scenario, θ_m and θ_M are respectively the lower and the upper bound of $\theta \in \mathbb{R}^9$. We can summarize the method as follow

   i. Set θ_0.

   ii. Generate $\overline{\delta} \in \mathbb{R}^9$ `random` uniform distribution in $[-1, 1]$.

   iii. Define $\theta^* = \theta_0 + \alpha^* \overline{\delta}$
   with

   $$\alpha^* = argmin[f(\theta)]$$

   $$s.t$$

   $$\theta = \theta_0 + \alpha\overline{\delta}.$$

   This minimization is not exact: we sample the α range with a finite number of point and we choose as α^* the one that gives the lowest cost.

   iv. $\theta_0 = \theta^*$

   v. Go to step 1(b)ii if the total numbers of function evaluations has not reached a maximum value.

2. The sets found in the previous step is used as initial point for the `Matlab` function `fmincon` which attempts to find a constrained minimum of a scalar function of several variables starting from an initial estimate.

The output of `fmincon` is finally our optimal parameters set to use in the PI equation (3.17).

3.3 Summary

The goal of this chapter is to explain the methods used to implement an adaptive cruise controller applied to the SMART model described in the second chapter. It described two possible solutions:

1. A MPC/PWA-based method
2. An adaptive PI controller

The first minimized an appropriate cost function based on a prediction horizon to calculate ONLINE the optimal control. It was designed by solving a constrained finite time optimal control problem, in an MPC receding horizon fashion.

The second used a built-in equation with nine parameters to design the hybrid control signal. It had a preliminary OFFLINE part too, because we optimized the nine parameters adapting them to the simulation SMART physical model and the nominal scenario.

Chapter 4

Simulation

The two methods were implemented in `Matlab 7`, `Windows XP Version 2002 Service Pack 2 OS` on an `Intel Pentium 4, 2.99 GHz` processor with `0.99 GB` of RAM . The *MPC* problems is solved[1] with the programming subtourine `linprog` built in `Matlab 7` . The optimal choice of coefficients for method *PI* is obtained via the nonlinear programming subroutine `fmincon` built in `Matlab 7`. The integration of equation (3.19) is done with the `Matlab ode45` subroutine with `relative` and `absolute tolerances` 1.0×10^{-8}, `max step size` 1.0×10^{-3}, with the assumption that the input value of u, j remains unchanged during the sampling time interval[2]. The length of the whole simulation period is 52 s. The experiments, carried out in computer simulation, allowed us to establish the comparison issues among the different methods described previously. Additionally they exhibit a positive and encouraging motivation to perform a field trials. It should however be remarked that, for a possible embedded solution in a real SMART, several technical issues should be regarded, like the sensor system, the resources of the on-board electronics, the real-life disturbances and the actuators delays. The cost of the device

[1]More information about the used subroutines are given in Appendix A.

[2]In an MPC-based scenario the previously computed control signal is fed into the plant until the computation of the next value. In this preliminary study, mainly characterized by computer simulation, we neglect this aspect, that becomes relevant in a real-time application.

Description	Numerical value
State weight matrix Q_x	1
Input weight matrix $Q_{\Delta u}$	0.1
Shift weight matrix $Q_{\Delta j}$	0.5
Prediction horizon N_p	2
Control horizon N_c	2
Sampling time T	$1\,s$
Simulation time	$52\,s$
Throttle initial position	0
Initial gear	1
State initial condition	$5\ m/s$
Leading vehicle speed	$5\ m/s$

Table 4.1: *General implementation data and initial conditions for the simulation.*

is also a relevant discrimination parameter. Note that modern technology (differential GPS, laser sensors and extended Kalman filters [31]) provides fast and highly accurate measurements, up to $0.1m/s$ error in velocity. The general data common to all experiments are summarized in Table 4.1. Initial conditions are given and we assume that the vehicle has a low initial speed and it is in first gear [13].

4.1 Numerical data

As general simulation system we use the block scheme depicted in Figure 4.1. The output of the controller block are the control $u(k)$ and the gear $j(k)$. The controller measures its current state, receives the reference state, and *predicts*[3] the reference in the next N_p future samples. On the basis of previous gear and control input information it evaluates the optimal decision strategy.

[3]If the leading vehicle is human driven, it is not significant to predict the reference in a long term future. Hence we limited to $N_\text{p} = 2$ [13]. If automatically driven vehicles [35], then longer N_p may be taken.

parameters	PI1 values	PI2 values
$K_{dv_\infty}M$	1.0087	1.0087
$K_{dv_0}M$	0.9841	0.7333
$\sigma_{dv}M$	0.3986	0.2233
$K_{dx1_\infty}M$	0.0040	-0.0039
$K_{dx1_0}M$	-0.2700	0.1271
$\sigma_{dx1}M$	0.1912	0.1404
$K_{dx2_\infty}M$	-0.0001	1.0120
$K_{dx2_0}M$	1.7061	0.4977
$\sigma_{dx2}M$	0.3723	0.1542

Table 4.2: *PI1 & PI2 tuned parameters used for the simulation.*

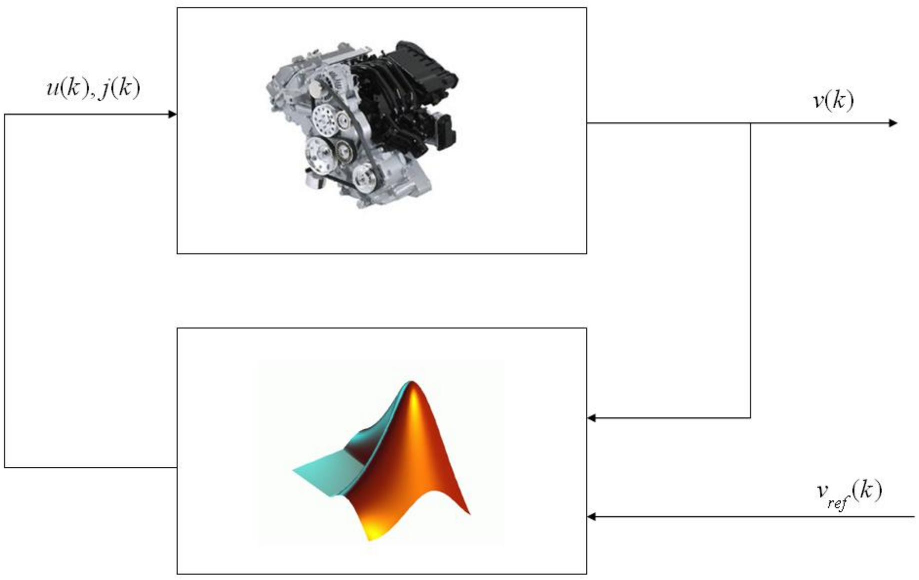


Figure 4.1: *Block scheme representation of the simulation system.*

In the on-line method this is done by solving an optimal control problem In the off-line methods by consulting a pre-scheduled table with the tuned parameters. The update of the state system is implemented using the non-linear “real” model (3.19) described in Section 2.1 as represented in Figure 4.1 with an engine picture.

We perform a comparison of 3 methods. The first is **MPC** as described in Section 3.1 with the initial condition reported in Table 4.1. The other two are 2 variants of PI. As explained in Section 3.2 **PI1** is tuned using Monte Carlo algorithm and **PI2** using random line search algorithm, both of the PI tuned parameters used in the simulation are reported in Table 4.2.

The 3 methods are simulated in five different scenarios with five different leader velocity.

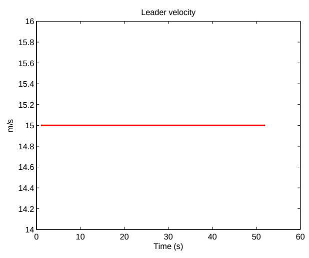
- In two scenarios the leading vehicle is moving with constant speed, in Figure 4.2(a) with $15m/s$ ($54Km/h$) and in Figure 4.2(c) with a constant speed of $35m/s$ ($126Km/h$). The vehicles reaches the desired constant velocity in both cases. The highest velocity is relevant to test the performance of the engine at high speed and of the gearbox actuators in an intermediate switching velocity.

- We chose two cases with varying velocity (Figure 4.2(b),Figure 4.2(d)) to simulate irregular movement of the leading vehicle, in particular in Figure 4.2(d) we will use also a constant horizon to check the behavior of the controller.

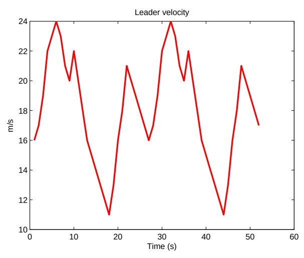
- Finally we test the model in an emergency breaking manoeuvre as depicted in Figure 4.2(e).

Most of the results are depicted for the entire simulation time, in particular for every methods implementation we consider five different plots for each scenarios.
These plots show:

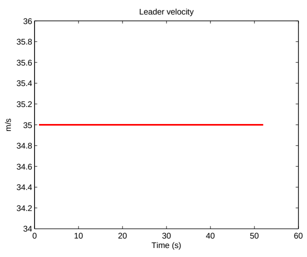
Control action: How the control action varies during the simulation.



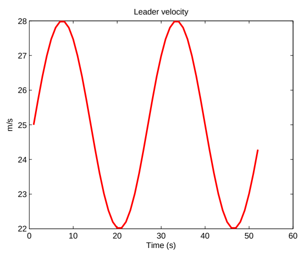
(a)



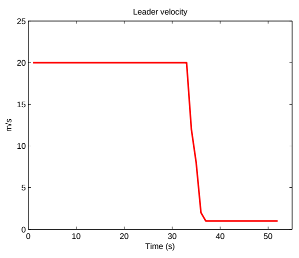
(b)



(c)



(d)



(e)

Figure 4.2: *The considered five possible scenarios for the reference velocity of the leading vehicle.*

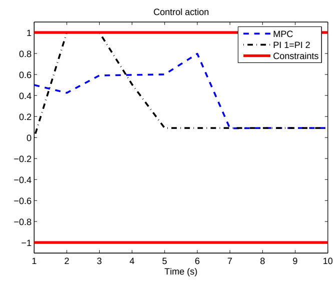
Control action variation: Related to energy saving and environmental issue.

Velocity curves: We plot the reference in red and the follower velocity in blue.

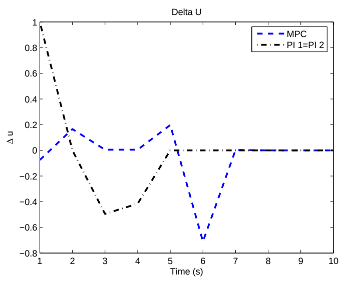
Acceleration: Related to comfort issues.

Gearshifts: It shows how many different gears are needed to reach and to track the reference.

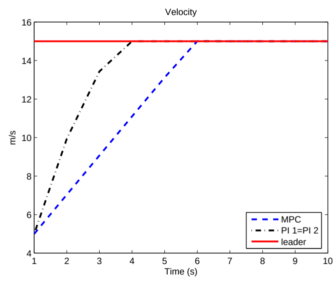
4.2 First scenario



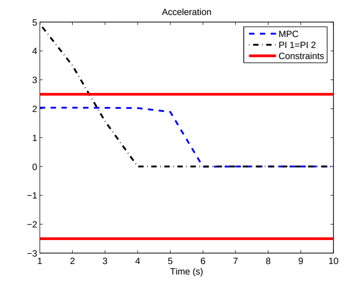
(a)



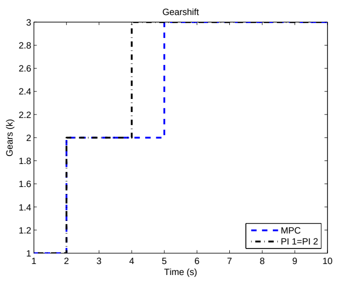
(b)



(c)



(d)



(e)

Figure 4.3: *Plots for the first scenario.*

In the first scenario the leading vehicle is moving with a constant velocity of $15m/s$ depicted using the red color in Figure 4.3(c). Obviously each method tries to reach the reference velocity as clear in all figures. PI 1 method and PI 2 method do not exhibit remarkable difference. They are practically identical for this scenario.

- Figure 4.3(a) and 4.3(b) plot, respectively, the control and its deriva-

Method	MPC	PI 1	PI 2
Computational features			
On-line time (s)	0.2272	5.8808e-004	6.1826e-004
Off-line time (s)	*0.2810*	5.8615e+004	5.5986e+004
Programming features			
Program class	LP	LP	LP
On-line method	Y	Y	Y
# variables	8	9	9
Solution features			
Cost of evolution	30.8148	7.8206	7.8206
Max acc. (m/s^2)	*2.0344*	4.9367	4.9367
Max dec. (m/s^2)	0.0077	0	0
Max $\Delta u(k)$	0.7979	1	1
Min $\Delta u(k)$	0.0874	0	0
Vel. overshoot (m/s)	0.0076	-5.5914e-010	3.0033e-008
Transient at 5% (s)	4	4	6
# gear-switches	3	3	3

Table 4.3: *Comparison issues for the* 3 *methods with the scenario in Figure 4.2(a) with* $N_\mathrm{p} = 2$.

tive. The PI methods apply initially a strong control action while MPC is more comfortable and it applies a control below the 0.8 value. Once reached the target velocity the PIs decrease the value of control action at 5s and the MPC at 7s, then they remain constant until the end of simulation.

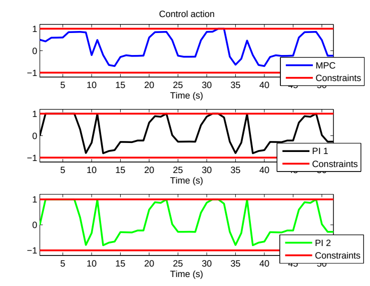
- Velocity and acceleration are showed in Figure 4.3(c) and 4.3(d). The PIs can not maintain the acceleration within the 2.5 comfort value, as depicted in the Figure 4.3(d). From the begining of the simulation until the third second the PI acceleration violates the constraint. It continues to decrease until the fourth second and from here it remains constant.

Instead the MPC curve shows a compliance to the acceleration upper bound constraint, and it respects the comfort standard. Starting from $5s$ it slowly reduces the acceleration in sight of the leader. Finally the MPC acceleration remains constant starting form $6s$.

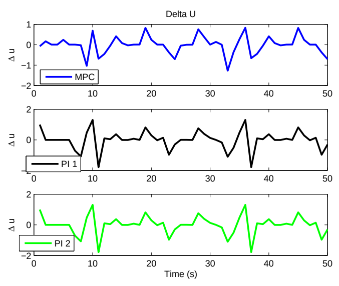
- In Figure 4.3(e) is depicted the gearshift. The final gear is the third and the MPC stays one second more using the second gear, because its action is more gradual.

In Table 4.3 are reported and compared relevant data for the simulation process. The on-line time values for the MPC case is significantly longer than the PIs because the quantity of used memory and of used variable requires a larger hardware usage. MPC solves on-line a significant number of LP (linear problem) instead PIs computes a series of arithmetic simple operations. The cost of the evolution is lower in the PIs because they do not embed specific constrains on the acceleration. This feature permits to reduce the error faster. Both PIs lose on the comfort, in fact the max acceleration value is much bigger than the MPC. With this scenario the number of gears are the same for all methods.

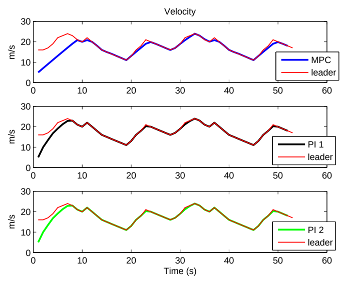
4.3 Second scenario



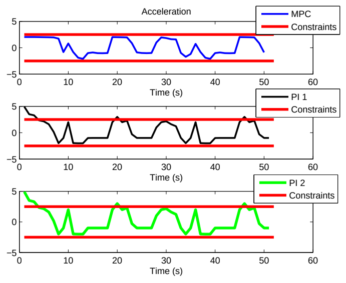
(a)



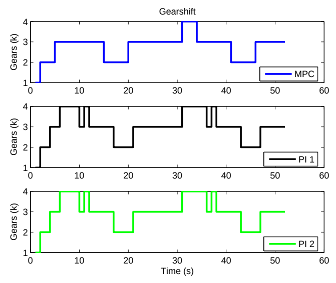
(b)



(c)



(d)



(e)

Figure 4.4: *Plots for the second scenario.*

In the second scenario the leading vehicle is moving with a variable speed. This scenario simulates a traffic situation velocity between $10 m/s$ and $25 m/s$ as depicted in Figure 4.4(c). Obviously each method tries to reach the reference, as clear in Figure 4.4(c). PI 1 method and PI 2 method have some minor differences which are explained below.

- In Figure 4.4(a) the MPC method shows a smoother feature ruled by

Method	MPC	PI 1	PI 2
Computational features			
On-line time (s)	0.1856	5.7059e-004	5.8933e-004
Off-line time (s)	*0.9220*	5.8615e+004	5.5986e+004
Programming features			
Program class	LP	LP	LP
On-line method	Y	Y	Y
# variables	8	9	9
Solution features			
Cost of evolution	144.3111	113.3920	111.3622
Max acc. (m/s^2)	2.0344	4.9367	4.9367
Max dec. (m/s^2)	2.1083	2.3887	2.6073
Max $\Delta u(k)$	0.8314	1.1456	1.2625
Min $\Delta u(k)$	1.2720	-2	-2
Transient at 5% (s)	–	–	–
# gear-switches	8	14	14

Table 4.4: *Comparison issues for the* 3 *methods with the scenario in Figure 4.2(b) with* $N_p = 2$.

the cost function minimization (3.1) as also shown of its derivative in Figure 4.4(b). There are also some differences between the two PI methods, in particular at the $27s$ of the simulation time probably because for the random relationship of the parameters.

- The velocity plot in Figure 4.4(c) shows that the MPC response, in blue, is lightly less efficient in tracking the reference. There are some difference between the two PI methods like in the control plotting at time $27s$ where the first PI gives smoother control behavior. For the first PI, we can also observe that the acceleration trend is rather similar to the MPC one.

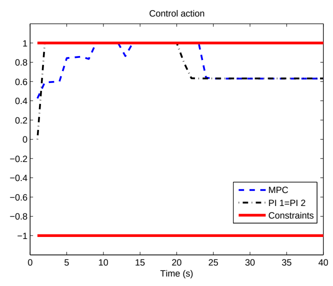
- In Figure 4.4(e) is reported the evolution of the gearshift. We immediately notice that the number of gear switches for the PI methods is higher than MPC method. We are interested in reducing the number of the gear switches in order to reduce the fuel consumption, the usury and the stress of the engine. It is evident in this scenario the big difference of switches, MPC switches 8 times and PIs switches 14 times, as reported in Table 4.4. From this point of view the MPC performs better than the PI methods.

The difference between the costs of the evolution in the MPC and in the PIs is here smaller than in the previous scenario. In fact the optimization of the PIs parameters is obtained with the nominal scenario in Figure 4.2(a), as mentioned in Section 3.2.2.

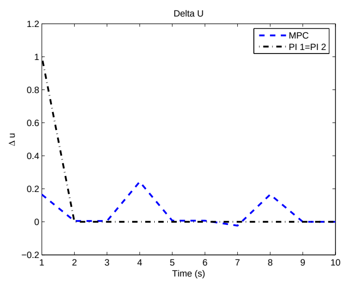
4.4 Third scenario

This scenario is similar to the first scenario (see Figure 4.2(a) and Figure 4.2(e)). We choose to use this scenario as benchmark because it tests the transient engine performance at high speed. Additionally, the velocity of $35m/s$ is exactly the switching velocity between the fifth and the sixth gear. In the PWA framework it is relevant to test the behavior of the different methods when it matters to regulate a hybrid system along the switching manifold [11].

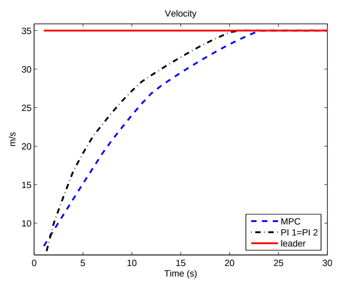
- In Figure 4.6(a) and 4.6(b) are plotted respectively the control and its derivative. The PI methods apply first a stronger control action instead MPC is less aggressive. Both methods saturate the maximum control value due to the high speed of the leader. Once reached the reference velocity, the PI 2 decreases its control input to 0.8, at $25s$. The PI 1 and the MPC at $27s$ reach the control stationary value of 0.6, then they remain constant until the end of simulation. The Figure 4.6(b) shows that the PI methods have a more aggressive control action especially before the time of $25s$. This aspects implies a bigger consume of fuel.



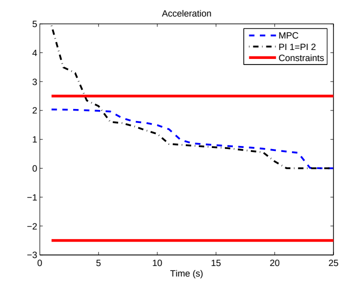
(a)



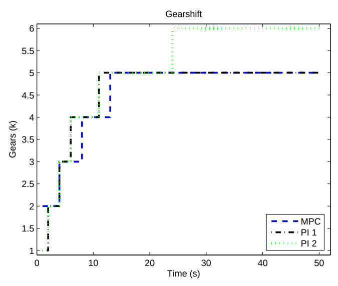
(b)



(c)



(d)



(e)

Figure 4.5: *Plots for the third scenario.*

- Velocity and acceleration are showed in Figures 4.6(c) and 4.6(d). The absence of a constraints in the PI method in the velocity Figure leads to a more aggressive behavior of the PIs. In fact, for the entire transient time, the velocity of the PIs is higher than the MPC one. Without any constraints the PIs acceleration can not stay under the 2.5 value comfort constraints. Before the simulation time of 4s the acceleration obtained using the PIS violates the constraints value losing on comfort and performance. The PIs acceleration remains constant starting form

$21s$. Instead the MPC curve shows a compliance to the acceleration upper bound constraint respecting the comfort standard. The MPC acceleration remains zero starting form $25s$.

- In Figure 4.6(e) is depicted the gearshift. The number of gear switches are similar for the three methods. The main difference regards the PI 2 which chooses as final gear the sixth. Probably because the PI 2 controller pass of a small value the leader velocity of $35m/s$.

4.5 Fourth scenario

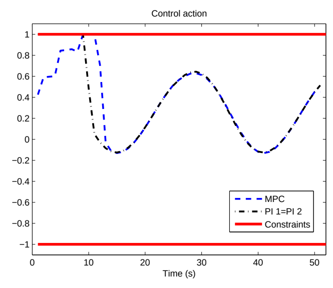
A varying velocity of the leader (see Figure 4.2(d)) here is proposed in order to test the main difference of the controllers. The irregular movements of the leader simulate a mixed traffic situation where the vehicles flow does not allow the drivers to maintain a constant velocity. Note that this realistic situation does not allow to use a traditional cruise controller.

- In Figure 4.6(a) and 4.6(b) are depicted respectively the control and its variation. The PI methods apply first a stronger control action instead MPC is less aggressive, like it has been seen in the previous testing scenarios. The control variation shows that the MPC method, in this case, is less regular mainly during the transient.

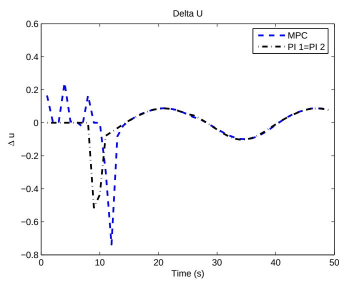
- Velocity and acceleration are showed in Figure 4.6(c) and 4.6(d).

  As explained in the previous scenarios, the absence of a constraint in the PI methods in the velocity Figure 4.3(c) lead to a more aggressive behavior. In fact, for the entire transient time , the velocity of the PIs is higher than the MPC one.

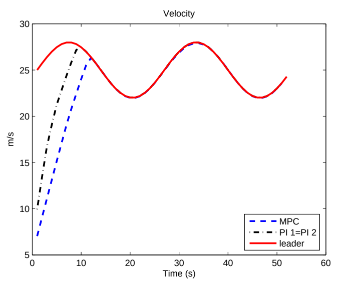
  Without any constraints the acceleration resulting from the PIs controllers can not stay within the $2.5m/s^2$ upperbound comfort requirements. Before the simulation time of $3s$ the PIs acceleration violate the



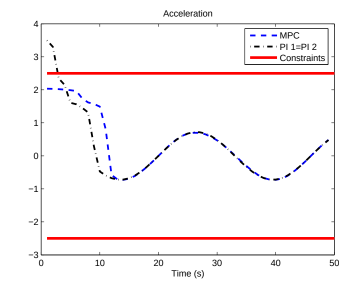
(a)



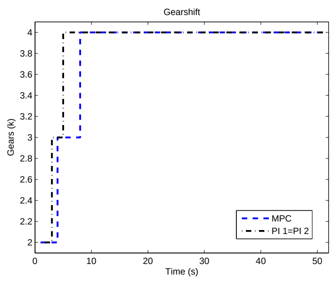
(b)



(c)



(d)



(e)

Figure 4.6: *Plots for the fourth scenario.*

constraints value losing on comfort. The MPC curve shows a compliance to the acceleration upper bound constraint respecting the comfort standard.

- In Figure 4.6(e) is depicted the gearshift. The number of gear switches are the same for the three methods.

In Table 4.5 are reported the most relevant value for a comparison. The online time shows a faster behavior of the PIs which means a faster reactions

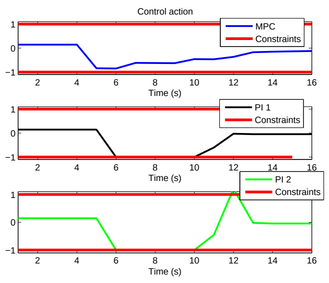
Method	MPC	PI 1	PI 2
Computational features			
On-line time (s)	0.1921	5.6762e-004	5.8341e-004
Off-line time (s)	*0.9220*	5.8615e+004	5.5986e+004
Programming features			
Program class	LP	LP	LP
On-line method	Y	Y	Y
# variables	8	9	9
Solution features			
Cost of evolution	155	105.91	105.91
Max acc. (m/s^2)	2.0344	3.4969	3.4969
Max dec. (m/s^2)	0.7287	0.7183	0.7183
Max $\Delta u(k)$	1	1	1
Min $\Delta u(k)$	0.1315	0.5142	0.5142
Transient at 5% (s)	10	10	12
# gear-switches	2	2	2

Table 4.5: *Comparison issues for the* 3 *methods with the scenario in Figure 4.2(d) with* $N_{\rm p} = 2$.

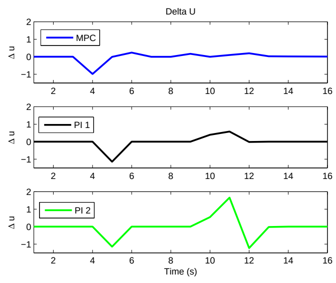
time. The cost of evolution is lower in the PIs cases because of absence of acceleration constraints which allow the PIs to use a bigger acceleration and so a smoother control action. In fact the maximum PIs acceleration resulting from value, listed in Table 4.5, are twice as much as the MPC ones.

4.6 Fifth scenario

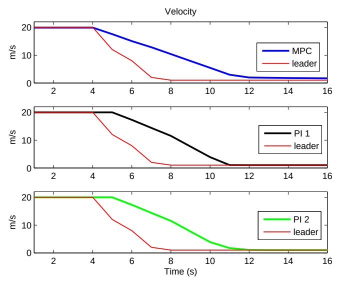
This scenario aims to simulate an emergency situation, where the leading vehicle brakes abruptly, see Figure 4.2. The maximum value of the leading vehicle deceleration is $8m/s^2$. This is a very challenging situation and it is difficult that the controllers follow faithfully the leader velocity.



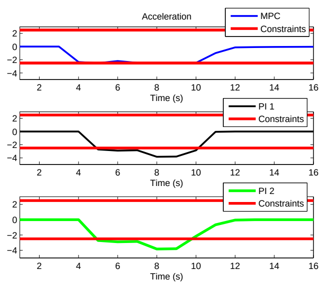
(a)



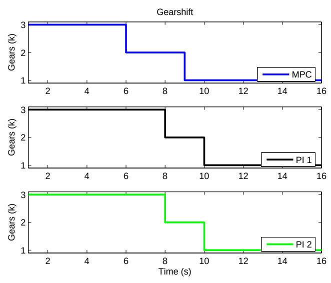
(b)



(c)



(d)



(e)

Figure 4.7: *Plots for the fifth scenario.*

- In Figure 4.6(a) and 4.6(b) are depicted respectively the control and its derivative. The MPC method is the first to notice first the deceleration of the leading vehicle. In fact it reacts at $4s$, one second before than PIs.

  The PI methods apply a stronger control action instead MPC is more comfortable. The PI 2 shows an excessive positive control at $12s$ in both figure. PI 1 and MPC are more efficient.

Method	MPC	PI 1	PI 2
Computational features			
On-line time (s)	0.1669	5.6006e-004	5.9498e-004
Off-line time (s)	*0.9220*	5.8615e+004	5.5986e+004
Programming features			
Program class	LP	LP	LP
On-line method	Y	Y	Y
# variables	8	9	9
Solution features			
Cost of evolution	122.34	88.073	89.045
Max acc. (m/s^2)	2.0344	4.9367	4.9367
Max dec. (m/s^2)	2.5130	3.8375	3.7456
Max $\Delta u(k)$	0.2425	1	1.6577
Min $\Delta u(k)$	0.9854	1.1450	1.2273
Transient at 5% (s)	–	–	–
# gear-switches	2	2	2

Table 4.6: *Comparison issues for the* 3 *methods with the scenario in Figure 4.2(e) with* $N_{\mathrm{p}} = 2$.

- Like explained in the all previous scenarios, the absence of a constraints in the PI methods in the velocity Figure leads to a more aggressive behavior of the PIs. In fact here we see a stronger deceleration of the PIs. Without any constraints the acceleration resulting from the PIs controllers can not stay within the $2.5m/s^2$ lowerbound comfort requirements. The MPC curve shows a compliance to the acceleration lower bound constraint respecting the comfort standard.

- In the last figure is depicted the gearshift evolution.For this scenario no particular remarks are necessary.

4.7 Constant prediction

A particular attention, here, is given to the MPC prediction horizon. In fact for every scenarios we assumed that the prediction of the reference is exactly known. This can be explained considering, for example, a series of vehicles in line which communicate their movements intentions to each others. Here we consider the case where there are not any communication between the vehicles and predictions are considered constant with value of the current state of the reference.

Using an horizon of $N_\mathrm{p} = 2$ it is not possible to remark any visible difference. Then we compare them with an horizon of $N_\mathrm{p} = 5$. The simulation plots are in Figure 4.8.

As expected the constant prediction horizon simulation gives a slower response to the future velocity of the leader. In fact using the exact curve as prediction horizon the method predicts what will happen, so the breaking system can react even before the leading vehicle reducing its velocity as can be find in Figure 4.8(a). Such anticipation is reflect also in the acceleration (Figure 4.8(c)) and in the gearshift (Figure 4.8(e)), which are anticipated at a distance of $1s$. If we consider longer horizon this trend will be amplified.

Until now it seems than we have only advantages to use a bigger window horizon prediction, but the disadvantages are the online-time. In fact the bigger is the prediction horizon the higher are the time needed to compute the optimal controller. In fact, as explained in section 3.1.3, the number of LP sub problems grows exponentially with N_p. This leads to much longer computational time. As an example a list of computational time as a function of N_p is given in Table 4.7.

4.8 Final remarks

In general the PIs approaches show a smaller cost function J compared to the MPC controller. This is explained, in the most cases, by the possibility to add the needed constraints in the MPC method. In fact in all scenarios

N_{p}	online-time s
3	0.5240
4	0.6961
5	0.9533
6	1.3957
7	1.4574
8	3.9320
9	7.6429

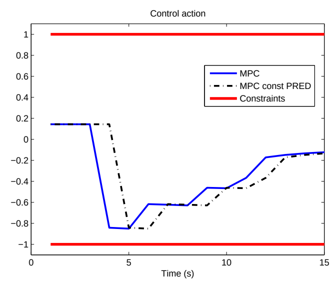
Table 4.7: *Comparisons of the fifth scenario online-time enlarging the prediction horizon* N_{p}.

the PIs violate the $2.5m/s$ imposed in the MPC; the error term in (3.1) is the most important responsible of this difference because the MPC needs more time to reduces the error due the acceleration constraints. We expect that if the PIs would have a way to add this constraint[4], the cost function became more similar.

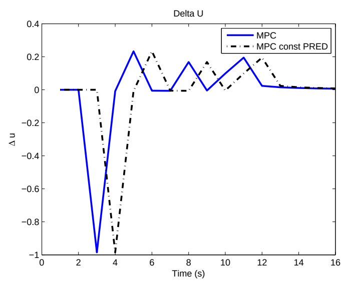
Generally the MPC needs a time response more bigger than the PIs. We must remark that the differences between the two methods is because the MPC solves mixed integer minimization on-line problem and the PIs compute only some arithmetic operations. In fact due to the nature of the problem the MPC requires an online computational time higher than that of the PIs. We use an algorithm described in Section 3.1.4 to drastically reduce them. Despite that, it is still not competitive with PI. However it is capable to handle constraints and provide overall more comfortable drive solutions.

The number of gear switches, in almost every simulation scenario, are the same for the MPC and the PI except for the variable scenario 4.2(b). Potentially the PI is instable in the gear switching because it is not capable to manage integers variable and it has the parameters tuned on the nominal scenario in Figure 4.2(a). This aspect implies that the switched gears can not

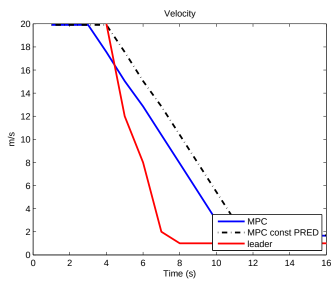
[4]Note that a further development of this study may consider to cascade the PI block with a saturation on the acceleration.



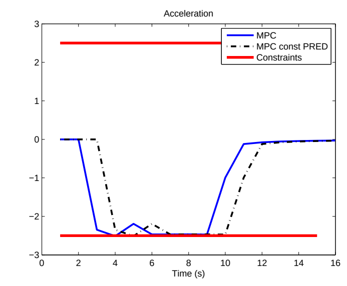
(a)



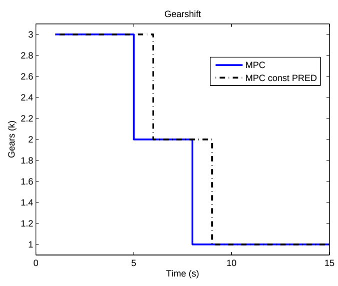
(b)



(c)



(d)



(e)

Figure 4.8: *MPC constant prediction horizon $N_\mathrm{p} = 5$ and MPC $N_\mathrm{p} = 5$ in the fifth scenario.*

be optimal in a different scenario.

An advantage of the MPC is the possibility to embed a set of constraints in the control problem. We also must remark the possibility to tune the weight matrixes Q , see Table 4.1, allowing to give more importance to the reduction of gear switching, or the reduction to the control change, or to the tracking error. The main advantage of the MPC is the smoother response obtained by the control predictive action. In fact the MPC uses an information about the future. This aspects implies than the control action can predict an abrupt manoeuvre of the reference.

With the aid of a Matlab tool [63] we calculated the used memory by the two methods. The tool gives as results the total memory allocated by Matlab. We find the result subtracting from the total allocated memory the amount needed by the software to run ($8.454MB$). In particular the amount needed by MPC is $3.776MB$ ($12.230MB - 8.454MB$) instead the amount needed by PI is $1.444MB(9,898MB - 8.454MB)$, the results are reported in Table 4.8. MPC needs more memory because of the `Matlab` function `linprog`.

MPC	PI
3.7776 MB	1.444 MB

Table 4.8: *Used ram memory by the MPC and the PIs simulations.*

4.9 Summary

In this chapter we described the results of the simulation carried out with the PI and the MPC approaches using five different scenarios depicted in Figure 4.2.
These scenarios included:

- Two varying velocities.
- Two constant velocities.
- One emergency manoeuvre, specifically an abrupt stop.

For each scenario we drew the plots of velocity and its derivative, the control action and its derivative and the switches during time evolution. Furthermore we had also collected, in comparison tables, other data of interest that cannot be conveyed with the mentioned plots.

At the end of the chapter we gave a final assessment between the two methods highlighting advantages and disadvantages.

Chapter 5

Conclusion

In this thesis we have presented a benchmark that serves as test bed to compare an ACC via a MPC/PWA method with a version of the ACC used in industry (based on PI methods) approach. The overall goal of the ACC is to track a given velocity reference. The significant challenge consists in the fact that a rather real simulation model of the system, including engine nonlinearities, hydraulic friction and tire versus ground static friction, was used. Additionally we have modeled the gear shift of the vehicle, using real field data. Constraints related to road safety, comfort, energy saving and mechanical stress are also considered. The main part of the work was devoted to adapt the method proposed by S. Leirens [42] to our physical model (3.19) as better explained in Appendix A. This method tackles the exponential number of LP deriving from exhaustive enumeration pruning the tree of derived possibilities via a Branch & Bound algorithm. The goal of this adaptation was to reduce drastically the number of linear problem in order to give a faster response to the MPC controller. We are successful in adapt this methods and the performance are near the PIs ones. But, as it can be seen in the simulations, carried out a slower time response from the MPC. It is slower but it has a smoother response with better comfort an it allows to insert the needed constraints in the optimization process. It is less aggressive than the PI and it has the possibilities to predict the events. This can not be done with the PIs controllers which is more difficult to adapt on a particular situation because it is tuned on the nominal scenario. Additionally to adapt the initial parameters shown in Table 3.2 of PI we must built an off-line part of the code which requested a lot of time to run.

The flexibility of the MPC method gives to it an added value which is missing in the PIs. Instead the PIs offer a more aggressive control.

As conclusion we can summarize that the PI has in general an efficient behavior but with a lot of limits derived from its strict structure. It offers a good response but it has not the possibility to include constraints. MPC shows a interesting behavior and generally we can say that it is efficient but it still need some adaption and some more test to tune correctly the weights and the parameters.

5.1 Further investigations

At least four open areas on which we can focus for a forthcoming research.

5.1.1 Physical model improvement

As reported in Chapter 2 the brake system is considered as a negative throttle so the braking force is computed using the traction force with a negative sign. But in a real car model the braking force is more intense than the traction force. This aspects implies that to use a model closer to a real vehicle it is necessary to know an exact modeling of it for the braking system of the Smart.

The modeling of the gearshift condition should be refined. Specifically, it should include not only a velocity criterion but also criteria based on acceleration, external condition or other various aspects.

5.1.2 MPC/PWA algorithm improvement and analysis

The method used here, proposed by S. Leirens [42], can be enhanced. In fact, as described in Section A and in Section 3.1.4, the second block of the algorithm explores again the sub-optimal potential nodes stored in a dynamical list. The access at this list and its sorting can be modified in order to reduce the total number of LP problems. In fact we implemented a simple FIFO access but could be interesting to formulate a list of access and sort criteria.

It can be very interesting to investigate how the method proposed by S. Leirens gains towards a brute-force enumeration of all the possible tree structure, for this specific application. In particular it would be interesting to quantify the efficiency of this algorithm of the B&B family. Which advantages, in terms of performance and complexity, could this method give?

5.1.3 PI method improvement

One of the most disadvantage of the PI is the absence of constraints. As it is shown in the simulation chapter one of the more important missing constraints is on the acceleration. This aspect implies a uncomfortable behavior as described in the previous chapter. It might be interesting, as future implementation, to force the PI in some manner to follow an acceleration constraint.

5.1.4 Implementation investigation

Every simulation run using `Matlab` but the ACC must be tested in a real car and in a real situation to achieve the most important results. The second step of this thesis should be a real implementation. In fact we do not know how can be performed a real implementation in a car in terms of hardware and mechanical parts. It can be also interesting to see if the needed hardware it is to much excessive.

For the sake of simplicity we did not consider the position from the state space. Including the position is important to prevent collisions. From a computational point of view the complexity would not increase because the position is built-in in the physical model.

Appendix A

Code description: relevant subroutines

In this appendix we give some technical explanation about the code used to adapt the MPC/PWA based method proposed by S. Leirens [42] to the physical model (3.19) in Chapter 2.1.

The function used to compute the minimization over the prediction horizon is called
`expl_found`, below is reported the input and the output:

```
[Uott,GEARott,Jott]= expl_found(Np,GEARk,GEARprec,vk,UPREC,xrefK);
```

The outputs are respectively the optimal control, the optimal gear and the optimal MPC cost function. The inputs respectively are the length of the horizon, the current gear, the previous gear, the current velocity, the previous control and the reference velocity.

This function is divided into two block:

1 Find the sub-optimum. As depicted in Figure 3.3 in Chapter 3, in the first block we find the sub-optimum comparing at each P depth the possible nodes (two or three). This is implemented using some `if` inside the `while` loop which flows the entire tree. Below is reported the `if` part where the choice is between three possible nodes comparing the MPC cost function value (3.21) in Section 3.1.1.

```
if(JJ(3)<JJ(1))
 if(JJ(3)<JJ(2))
   GEARk=GEARGEAR(3);memoria(2,q)=JJ(3);memoria(9,q)=uu(3);
                     memoria(3,q)=GEARGEAR(1);memoria(4,q)=
                     GEARGEAR(2);
                     memoria(5,q)=JJ(1);memoria(6,q)=JJ(2);
                     memoria(7,q)=uu(1);memoria(8,q)=uu(2);
 elseif(JJ(3)==JJ(2))
  if(uu(3)>uu(2))
   GEARk=GEARGEAR(2);memoria(2,q)=JJ(2);memoria(9,q)=uu(2);
                     memoria(3,q)=GEARGEAR(1);memoria(4,q)=
                     GEARGEAR(3);
                     memoria(5,q)=JJ(1);memoria(6,q)=JJ(3);
                     memoria(7,q)=uu(1);memoria(8,q)=uu(3);

  else
   GEARk=GEARGEAR(3);memoria(2,q)=JJ(3);memoria(9,q)=uu(3);
                     memoria(3,q)=GEARGEAR(1);memoria(4,q)=
                     GEARGEAR(2);
                     memoria(5,q)=JJ(1);memoria(6,q)=JJ(2);
                     memoria(7,q)=uu(1);memoria(8,q)=uu(2);
          end
    end
end
```

The matrix `memoria` stores the information about every explored node. At the end of the first block we have a `Jsuboptimum` and a `Usuboptimum` value.

2 Find the optimum. In the second block we compare the obtained sub-optimum with each stored values in matrix `memoria`, simply by flowing it. If the `J` value in matrix is smaller than the current `Jsuboptimum` and if it corresponds to a node at the end of the tree we simple update the sub-optimum values with the new ones:

```
 if(J1<Jsuboptimum)
           Jsuboptimum=J1; Usuboptimum=u1;GEARsubott=percorsoN(2);
        end
        if(J2<Jsuboptimum)
           Jsuboptimum=J2; Usuboptimum=u2;GEARsubott=percorsoN(2);
        end
        if(J3<Jsuboptimum)
           Jsuboptimum=J3; Usuboptimum=u3;GEARsubott=percorsoN(2);
        end
```

If the `J` value in matrix is smaller than the actual `Jsuboptimum` and if it corresponds to a node in the middle of the tree we explore the new part of the tree starting from that node,

```
if(J1<Jsuboptimum)
 memoria=[candidati;[percorsoN 1 zeros(1,(Np-1)-
 length(percorsoN))]];
end if(J2<Jsuboptimum)
 memoria=[candidati;[percorsoN 2 zeros(1,(Np-1)-
 length(percorsoN))]];
end if(J3<Jsuboptimum)
 memoria=[candidati;[percorsoN 3 zeros(1,(Np-1)-
 length(percorsoN))]];
end
```

If the relative node has a lower cost function (`J1, J2, J3`) we update the `memoria` matrix with that new path. We continue to flow the matrix `memoria` until is finished. At this point the last `Jsuboptimum` value stored became the optimum.

Bibliography

[1] P.J. Antsaklis and X.D. Koutsoukos. *Hybrid Systems Control*, volume 7. Encyclopedia of Physical Science and Technology, Third Edition, Academic Press, 2005.

[2] A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Trans. Automatic Contr.*, 49(5):832–838, May 2004.

[3] A. Bemporad, P. Borodani, and Mannelli. Hybrid control of an automotive robotized gearbox for reduction of consumptions and emissions. *In: Hybrid Systems: Computation and Control. Vol. 2623 of Lecture Notes in Computer Science*, pages 81–96, 2003.

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

[5] F. Borrelli. *Constrained Optimal Control of Linear & Hybrid Systems*, volume 290. Springer Verlag, 2003.

[6] S.P. Boyd and C.H. Barratt. *Linear Controller Design, Limits of performance.* Prentice Hall, Information and System Sciences Series, Englewood Cliffs, New Jersey, 1991.

[7] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, 1998.

[8] B. Brogliato, S. I. Niculescu, and P. Orthant. On the control of finitedimensional mechanical systems with unilateral constraints. *IEEE Transactions on Automatic Control*, pages 200–215, 1997.

[9] B. Brogliato and A. Z. Rio. On the control of complementary-slackness mechanical juggling systems. *IEEE Transactions on Automatic Control*, pages 235–246, 2000.

[10] C.G.Cassandras, D.L. Pepyne, and Y. Wardi. Optimal control of a class of hybrid systems. *IEEE Transactions on Automatic Control*, 46(3):398–415, 2001.

[11] D. Corona, I. Necoara, B. De Schutter, and T.J.J. van den Boom. Robust hybrid MPC applied to the design of an adaptive cruise controller for a road vehicle. In *CDC06*, pages 1721–1726, San Diego, USA, December 2006.

[12] D. Corona and B. De Schutter. Adaptive cruise controller for a smart: comparisson benchmark for mpc-pwa control methods. In *ADHS'06 second Conf. on Analysis and Design of Hybrid Systems*, volume 1, pages 1–6, june 2006.

[13] D. Corona and B. De Schutter. Adaptive cruise controller for a SMART: A comparison benchmark for MPC-PWA control methods. Technical Report 07-005, Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands, February 2007.

[14] C.R. Cutler. Dynamic matrix control, an optimal multivariable control algorithm with constraints. *Dissertation Abstracts Int. Part B: Science and Engineering*, 44(8):228, June 1983.

[15] C.R. Cutler and B.L. Ramaker. Dynamic matrix control - a computer control algorithm. In *Proc. Joint American Control Conf.*, San Francisco, CA, USA, 1980.

[16] J. Sjöberg D. Axehill. “Adaptive Cruise Control for Heavy Vehicles”. Master’s thesis, Linköping University, Institutionen för systemteknik, Linköping, Sweden, 2003.

[17] B. De Schutter and T.J.J. Van den Boom. MPC for continuous piecewise-affine systems. *Systems & Contr. Letters*, 52(3–4):179–192, July 2004.

[18] D.M.W.Leenaerts. Further extensions to chuas explicit piecewise linear function descriptions. *Int. Journal of Circuit Theory and Applications*, 24:621–633, 1996.

[19] J.C. Doyle. Analysis of feedback systems with structured uncertainties. In *IEE Proc.*, volume 129-D, pages 242–250, September 1982.

[20] J.C. Doyle. Lecture notes in advances in multivariable control. In *ONR/Honeywell Workshop on Advances in Multivariable Control, Tech. Rep.*, Honeywell, Minneapolis, MN, 1984.

[21] J.C. Doyle, B.A. Francis, and A.R. Tannenbaum. *Feedback control systems.* MacMillan Publishing Company, New York, USA, 1992.

[22] J.C. Doyle, A. Packard, and K. Zhou. Review on lfts, lmis and μ. In *Proc. of the 30th Conf. on Decision and Control*, pages 1227–1232, Brighton, UK.

[23] S. Drulhe, G. Ferrari-Trecate, H. de Jong, and A. Viari. Reconstruction of switching thresholds in piecewise-affine models of genetic regulatory networks. *In: Hybrid Systems: Computation and Control. Vol. 3927 of Lecture Notes in Computer Science*, pages 184–199, 2006.

[24] E.D.Sontag. Sontag, e. d., 1996. interconnected automata and linear systems: A theoretical framework in discrete-time. In *Hybrid Systems III: Verification and Control*, pages 1049–1056, Springer, New York, 1996.

[25] G. Ferrari-Trecate, F. A. Cuzzola, D. Mignone, and M. Morari. Analysis of discrete-time piecewise affine and hybrid systems. *Automatica*, 38(12):2139–2146, 2002.

[26] National Center for Statistics and Analysis. Traffic safety facts 2004: Speeding , [online document], http: http://www-nrd.nhtsa.dot.gov/pdf/nrd-30/ncsa/tsf2004/809915.pdf. 2005.

[27] T. Geyer, G. Papafotiou, and M. Morari. Model predictive control in power electronics: A hybrid systems approach. In *44-th IEEE Conf. on Decision and Control*, pages 5606–5611, Seville, Spain, 2005.

[28] D.N. Godbole and J. Lygeros. Longitudinal control of the lead car of a platoon. *IEEE Trans. Vehicular Technology*, 43(4):1125–1135, November 1994.

[29] R. Goebel and A. R. Teel. Solutions to hybrid inclusions via set and graphical convergence with stability theory applications. *Automatica*, 42(4):573–587, 2006.

[30] F. Gustafsson. Slip–based tire–road friction estimation. *Automatica*, 33(6):1087–1099, June 1997.

[31] R. Hallouzi, V. Verdult, H. Hellendoorn, and J. Ploeg. Experimental evaluation of a co-operative driving set-up based on inter-vehicle communication. In *Proc. IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, July 2004.

[32] P. R. Haney and M. J. Richardson. Adaptive cruise control, system optimisation and development for motor vehicles. *Journal of Navigation*, 53(1):42–47, January 2000.

[33] W. P. M. H. Heemels, J. M. Schumacher, and S. Weiland. Linear complementarity systems. *SIAM journal on applied mathematics*, 60(4):234–1269, 2000.

[34] W.P.M.H. Heemels, B. De Schutter, and A. Bemporad. On the equivalence classes of hybrid dynamical models. In *Proc. 40th IEEE Conf. on Dec. and Contr.*, pages 364–369, Orlando, USA, December 2001.

[35] P.A. Ioannou and C.C. Chien. Autonomous intelligent cruise control. *IEEE Trans. Vehicular Technology*, 42(4):657–672, November 1993.

[36] M. Johansson and A. Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid systems. In *IEEE Transactions on Automatic Control*, volume 43, pages 555–559, 1998.

[37] E.C. Kerrigan and D.Q. Mayne. Optimal control of constrained, piecewise affine systems with bounded disturbances. In *Proc. 41th IEEE Conf. on Dec. and Contr.*, pages 1552–1557, Las Vegas, USA, December 2002.

[38] M.V. Kothare, V. Balakrishnan, and M. Morari. Robust contrained predictive control using linear matrix inequalities. *Automatica*.

[39] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems econometrica. *Econometrica*, 28(3):497–520, July 1960.

[40] M. Lazar. *Model Predictive Control of Hybrid Systems: Stability and Robustness*. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2006.

[41] M. Lazar, W.P.M.H., S. Heemels, S. Weiland, A. Bemporad, and O. Pastravanu. Infinity norms as Lyapunov functions for model predictive control of constrained PWA systems. In *LNCS: Hybrid Systems: Computation and Control*, number 3414, pages 417–432, Zürich, Switzerland, 2005. Springer Verlag.

[42] S. Leirens, J. Buisson, P. Bastard, and J. L. Coullon. An efficient algorithm for solving model predictive control of switched affine systems.

ADHS'06 second Conf. on Analysis and Design of Hybrid Systems, 1(1–4):1–6, June 2006.

[43] D. Liberzon. *Switching in systems and control.* Birkhauser, Boston, 2003.

[44] J. Lygeros, K. H. Johansson, S. N. Simic, J. Zhang, and S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48:2–17, 2003.

[45] E. Hesslow M. Persson, F. Botling and R. Johansson. Stop and go controller for adaptive cruise control. In *Proc. IEEE Int. Conf. on Contr. application and IEEE Int. symposium on computer aided control system design*, pages 1692–1697, Kohala Coast-Island of Hawai, USA, August 1999.

[46] J.M. Maciejowski. *Multivariable Feedback Control Design.* Addison-Wesley Publishers, Wokingham, UK, 1989.

[47] M. Morari and E. Zafiriou. *Robust Process Control.* Prentice Hall, Englewood Cliffs, New Jersey, USA, 1989.

[48] M.W.J.M Musters and N.A.W. van Riel. Analysis of the transformating growth factor-beta1 pathway and extracellular matrix formation as a hybrid system. *In: 26th Conf. of the IEEE Engineering in Medicine and Biology Society*, pages 2901–2904, 2004.

[49] M. Barth O. Servin, K. Boriboonsomsin. An energy and emissions impact evaluation of intelligent speed adaptation. In *Proc. of the IEEE ITSC 2006 2006 IEEE Intelligent Transportation Systems Conf.*, Toronto, Canada, September 2006.

[50] F. Pfeiffer and C. Glocker. Multibody dynamics with unilateral contacts wiley. In *ONR/Honeywell Workshop on Advances in Multivariable Control*, Chichester, 1996.

[51] A. Yu. Pogromsky, M. Jirstrand, and P. Spangeus. On stability and passivity of a class of hybrid systems. In *Proc. of the 37-th IEEE Conf. on Decision and Control'98*, pages 3705–3710, Tampa (USA), 1998.

[52] J.B. Rawlings and K.R. Muske. The stability of constrained receding horizon control. *IEEE AC*.

[53] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, pages 413–428, 1978.

[54] B. De Schutter and M. Heemels. *Modeling and Control of Hybrid Systems.* Lecture notes for the course, TUDelft, 2005.

[55] B.De Schutter and T.J.J. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, 2001.

[56] A.R.M. Soeterboek. *Predictive control - A Unified Approach.* Prentice Hall, Englewood Cliffs, New Jersey, USA, 1992.

[57] E. D. Sontag. Nonlinear regulation: the piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–357, 1981.

[58] E.D. Sontag. An algebraic approach to bounded controllability of linear systems. *Int. J. Contr.*, pages 181–188, 1984.

[59] Ton J.J. van den Boom and Ton C.P.M. Backx. *Model Predictive Control.* Lecture notes for the course, TUDelft, 2005.

[60] A. J. van der Schaft and J. M. Schumacher. An introduction to hybrid dynamical systems. *IEEE Transactions on Automatic Control*, 43(3):483–490, 1998.

[61] A. J. van der Schaft and J. M. Schumacher. *An introduction to hybrid dynamical systems. Vol. 251 of Lecture Notes in Control and Information Sciences.* Springer, 2000.

[62] M. Vasak, M. Baotic, M. Morari, I. Petrovic, and N. Peric. Constrained optimal control of an electronic throttle. *Int. Journal of Control*, 79(5):465–478, 2006.

[63] Mathworks Website. `http://www.mathworks.com/matlabcentral/`.

[64] Smart Website. `http://www.smart-training-online.com/`.

[65] D. Yanakiev and I. Kanellakopoulos. Nonlinear spacing policies for automated heavy-duty vehicles. In *IEEE Transactions on vehicular technology*, volume 47, November 1998.

[66] A. Zheng, V. Balakrishnan, and M. Morari. Constrained stabilization of discrete-time-systems. *Int. Journal of Robust & Nonlinear Control*, 5(5):461–485, 1995.