MATEUS MUSSI BRUGNOLLI

**Predictive Adaptive Cruise Control in an embedded
environment**

São Paulo
2018

MATEUS MUSSI BRUGNOLLI

**Predictive Adaptive Cruise Control in an embedded
environment**

Thesis submitted to Escola Politécnica of
the Universidade de São Paulo for degree
of Master of Science.

São Paulo
2018

MATEUS MUSSI BRUGNOLLI

**Predictive Adaptive Cruise Control in an embedded
environment**

Thesis submitted to Escola Politécnica of
the Universidade de São Paulo for degree
of Master of Science.

Concentration area:

Systems Engineering

Advisor:

Bruno Augusto Angélico

Co-Advisor:

Armando Antônio Maria Laganá

São Paulo
2018

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

Brugnolli, Mateus Mussi

Predictive Adaptive Cruise Control in an embedded environment / M. M. Brugnolli -- versão corr. -- São Paulo, 2018.

80 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Telecomunicações e Controle.

1.VEÍCULOS AUTOMORES 2.DINAMÔMETROS 3.IDENTIFICAÇÃO DE SISTEMAS 4.RADAR 5.CONTROLE PREDITIVO I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Telecomunicações e Controle II.t.

ACKNOWLEDGMENTS

Thanks to my mother Maria Helly Mussi, my father Alceu Sandano Brugnolli and my sister Rebeca Mussi Brugnolli, for their support, patience and time.

Thanks to my advisor Prof. Dr. Bruno Augusto Angélico, for his support and friendship.

Thanks to Gabriel Pereira das Neves, Fábio Yukio Toriumi, Arthur Castello Branco de Oliveira, Fernando dos Santos Barbosa, Kennedy Xisto Barros Peixoto and other LCA members, for their friendship and assistance.

Thanks to teacher Armando Antônio Maria Laganá and his automotive group, for their cooperation and help with materials.

Thanks to Centro Paulo Souza, for the initial scholarship.

Thanks to Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), for the scholarship, Process 132237/2017-2.

Thanks to Escola Politécnica da Universidade de São Paulo, for the Master's program, courses and technical support.

“When do you think people die? When they are shot through the heart by the bullet of a pistol? No. When they are ravaged by an incurable disease? No. When they drink a soup made from a poisonous mushroom? No. It’s when they are forgotten.”

-- Eiichiro Oda

RESUMO

A inclusão de sistemas avançados para assistência de direção (ADAS) tem beneficiado o conforto e segurança através da aplicação de diversas teorias de controle. Um destes sistemas é o Sistema de Controle de Cruzeiro Adaptativo. Neste trabalho, é usado uma distribuição de duas malhas de controle para uma implementação embarcada em um carro de um Controle de Cruzeiro Adaptativo. O modelo do veículo foi estimado usando a teoria de identificação de sistemas. O controle da malha externa utiliza dados de um radar para calcular uma velocidade de cruzeiro apropriada, enquanto o controle da malha interna busca o acionamento do veículo para atingir a velocidade de cruzeiro com um desempenho desejado. Para a malha interna, é utilizado duas abordagens do controle preditivo baseado em modelo: um controle com horizonte de predição finito, e um controle com horizonte de predição infinito, conhecido como IHMPC. Ambos controladores foram embarcados em um microcontrolador capaz de comunicar diretamente com a unidade eletrônica do veículo. Este trabalho valida estes controladores através de simulações com sistemas variantes e experimentos práticos com o auxílio de um dinamômetro. Ambos controladores preditivos apresentaram desempenho satisfatório, fornecendo segurança para os passageiros.

Palavras-Chave – Sistemas ACC, Radar, Dinamômetro, Identificação de Sistemas, Controle preditivo baseado em modelo, IHMPC.

ABSTRACT

The development of Advanced Driving Assistance Systems (ADAS) produces comfort and safety through the application of several control theories. One of these systems is the Adaptive Cruise Control (ACC). In this work, a distribution of two control loops of such system is developed for an embedded application to a vehicle. The vehicle model was estimated using the system identification theory. An outer loop control manages the radar data to compute a suitable cruise speed, and an inner loop control aims for the vehicle to reach the cruise speed given a desired performance. For the inner loop, it is used two different approaches of model predictive control: a finite horizon prediction control, known as MPC, and an infinite horizon prediction control, known as IHMPC. Both controllers were embedded in a microcontroller able to communicate directly with the electronic unit of the vehicle. This work validates its controllers using simulations with varying systems and practical experiments with the aid of a dynamometer. Both predictive controllers had a satisfactory performance, providing safety to the passengers.

Keywords – ACC systems, Radar, Dynamometer, System Identification, Model Predictive Control, IHMPC.

LIST OF FIGURES

1	Representation of an ACC System	15
2	Photo of the Polo vehicle	19
3	Photo of the customized ECU	20
4	Diagram of the ECU	21
5	Photo of the dynamometer	22
6	Photo of the radar ARS300	23
7	Photo of the ACC Module	24
8	System step response.	26
9	Schematic of the tuning simulation.	44
10	First three simulations of MPC using the same models as target systems. .	45
11	More three simulations of MPC using the same models as target systems. .	45
12	Last three simulations of MPC using the same models as target systems. .	46
13	Objective functions of all nine simulations of MPC.	46
14	Schematic of the simulation with changing load models.	47
15	System response to three different MPC for a system with changing models.	47
16	Schematic of the simulation with changing speed models.	48
17	System response to three different MPC for a system acting as nonlinear. .	48
18	Schematic of the ACC Simulation 1.	49
19	Simulation 1 of ACC with MPC for a system with changing models.	49
20	Corresponding distances and control signal of Figure 19.	50
21	Schematic of the ACC Simulation 2.	50
22	Simulation 2 of ACC with MPC for a system with changing models.	51
23	Corresponding distances and control signal of Figure 22.	51

24	First three simulations of IHMPC using the same models as target systems.	52
25	More three simulations of IHMPC using the same models as target systems.	52
26	Last three simulations of IHMPC using the same models as target systems.	53
27	Objective functions of all nine simulations of IHMPC.	53
28	System response to three different IHMPC for a system with changing models.	54
29	System response to three different IHMPC for a system acting as a nonlinear.	54
30	Simulation 1 of ACC with IHMPC for a system with changing models. . .	55
31	Corresponding distances and control signal of Figure 30.	55
32	Simulation 2 of ACC with IHMPC for a system with changing models. . .	56
33	Corresponding distance to leading vehicle of Figure 32.	56
34	Schematic of the Experiment 1	57
35	Schematic of the Experiment 2	58
36	Speeds for the Experiment 1, using the MPC as inner controller	58
37	Distances and Control Signal for Experiment 1 with the MPC	59
38	Speeds for the Experiment 1, using the IHMPC as inner controller	59
39	Distances and Control Signal for Experiment 1 with the IHMPC	60
40	Speeds for the Experiment 2, using the MPC as inner controller	60
41	Distances and Control Signal for Experiment 2 with the MPC	61
42	Speeds for the Experiment 2, using the IHMPC as inner controller	61
43	Distances and Control Signal for Experiment 2 with the IHMPC	62
44	Normalized double exponential function.	77
45	Normalized double exponential function for three models.	78

LIST OF TABLES

1	CAN Messages for the ACC System.	21
2	PRBS data specification	28
3	Models designation	29
4	Coefficients and time delay of ARX models	29
5	FIT (%) for system validation, with several configurations of steps ahead .	30

LIST OF ABBREVIATIONS

ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
AIC	Akaike Information Criterion
ARX	AutoRegressive with exogenous input
CACC	Cooperative Adaptive Cruise Control
CC	Cruise Control
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CRC	Cyclic Redundancy Check
ECU	Electronic Control Unit
EPUSP	Escola Politécnica da Universidade de São Paulo
IEE	Instituto de Energia e Ambiente
IHMPC	Infinite Horizon Model Predictive Control
LCA	Laboratório de Controle Aplicado
MDL	Minimum Description Length
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
OBD	On-Board Diagnosis
OPOM	Output Prediction Oriented Model
PI	Proportional-Integral
PRBS	Pseudo-Random Binary Sequence
PWA	PieceWise-Affine
SPI	Serial Peripheral Interface
QP	Quadratic Programming
UFMG	Universidade Federal de Minas Gerais

LIST OF SYMBOLS

$A(q)$	Output Polynomial of the ARX
$B(q)$	Input Polynomial of the ARX
na	Order of the Output Polynomial
nb	Order of the Input Polynomial
nk	Time delay for the ARX model (number of sampling)
θ	Model time delay [s]
T_s	Sampling Time [s]
f_s	Sampling Frequency [Hz]
d_{ref}	Security distance [m]
ℓ	Length of the vehicle [m]
d_s	Minimum distance to leading vehicle [m]
T_h	Time-headway [s]
v	Vehicle speed [m/s]
v_{ref}	Reference speed [m/s]
v_l	Leading vehicle speed [m/s]
K_p	Outer loop proportional gain
d	Distance to leading vehicle m
A, B, C, D	Matrices of the space state representation
$u(k-1)$	Control signal applied at the instant $k-1$
$u(k)$	Control signal applied at the instant k
$\Delta u(k)$	Control variation at the instant k
J_k^{MPC}	Objective function of the MPC
m	Control horizon
p	Prediction horizon
Q	Output error weight matrix
R	Control input weight matrix

$\bar{y}(k)$	Output prediction vector
Δu_k	Control input sequence
Φ, Γ	Prediction matrices to the horizon p
y^{sp}	Output Set-point m/s
$\bar{y}^{sp}$	Output Set-point vector to the prediction horizon
$\bar{Q}$	Output error weight matrix expanded to horizon p
$\bar{R}$	Control input weight matrix expanded to horizon m
A_c, b_c	Control input constraints
$u_{\max}$	Maximum control input for the actuator
$u_{\min}$	Minimum control input for the actuator
$\Delta u_{\max}$	Maximum control input variation for the actuator
$J_k^{\text{IHMP}}C$	Objective function of the IHMPC
δ_y	Slack variables
S_y	Slack variables weight matrix
$\bar{A}, \bar{B}$	Prediction matrices to the horizon $m + \theta_{\max}$
$\bar{Q}_y$	Output error weight matrix expanded to horizon $m + \theta_{\max} + 1$
x^s	OPOM steady state
x^d	OPOM dynamic states
K_f	Kalman filter gain

CONTENTS

1	Introduction	15
1.1	Objectives	18
2	Background	19
2.1	Vehicle	19
2.2	Dynamometer	21
2.3	Radar	22
2.4	ACC Module	23
3	System Identification	25
3.1	PRBS specification	27
3.2	ARX Models	28
3.3	Models Validation	30
4	Control Design	31
4.1	ACC Controller	31
4.2	Conventional MPC	32
4.2.1	Conventional MPC Design	32
4.3	Infinite Horizon MPC	35
4.3.1	IHMPC Design	36
4.4	Kalman Filter	40
4.4.1	Kalman Filter for the MPC	40
4.4.2	Kalman Filter for the IHMPC	41
5	Simulation and Practical Results	43

5.1	Simulation Experiments	44
5.1.1	Conventional MPC	44
5.1.2	Infinite Horizon MPC	50
5.2	Practical Experiments	55
6	Conclusions and future works	63
	References	64
	Appendix A – Hildreth’s Quadratic Programming	67
	Appendix B – Creating a state space with delay	69
	Appendix C – Creating an OPOM with time delay	71
	Appendix D – Transforming the OPOM with real states	75
	Appendix E – Switching models logic	77

1 INTRODUCTION

The automotive research is an abundant area with many control problems to be solved. One of the most discussed themes is the development and improvement of Advanced Driver Assistance Systems (ADAS). These systems aid the driver to handle the vehicle safely and with less stress (DELPHI, 2016). Initially, Cruise Control (CC) systems were developed within the ADAS to control the longitudinal speed of the vehicle to track a desired speed (known as cruise speed).

Such CC systems were improved using vision technologies, for example long-range radars. This change started the development of a complex control system, known as Adaptive Cruise Control (ACC) System. In this new problem, the cruise speed is constantly being computed to maintain a safe distance to any leading vehicles, which is measured by a radar positioned in front of the vehicle. The ACC System must track the constantly changing cruise speed reference.

For example, in (SHAKOURI; CZECZOT; ORDYS, 2015) the control design uses a structure of two control loops, an inner and an outer, as can be seen in Figure 1. For inner loop control, there are two input signals: cruise speed and system (vehicle) speed. The inner loop outputs are the signals for the actuators of the system. There are three input signals for outer loop control: user defined speed, vehicle speed and all important radar data, for example distance from the closest vehicle and its relative speed.

Source: Author.

Several control theories are being applied in ACC systems. Some examples are the usage of the Fuzzy logic (BENALIE et al., 2009; PANANURAK; THANOK; PAR-

NICHKUN, 2009), control barriers (AMES; GRIZZLE; TABUADA, 2014), sliding mode (GANJI et al., 2014) and balanced-based adaptive control (SHAKOURI; CZECHOT; ORDYS, 2012). In the later paper, it is important to remark that adaptive control theory does not present a direct relationship with the ACC theory, which will be explained in Section 4.

There are some works that analyses the ACC problem differently. In (ZHOU; PENG, 2005), it considers multiple vehicles with ACC systems. In such a paper, the main objectives are to reduce the distance spring effects between the vehicles and to improve the traffic flow. This effect is an oscillating phenomenon due to the lack of communication between the vehicles when there is a traffic change.

In (MARKVOLLRATH; SCHLEICHER; GELAU, 2011), a study of traffic changes is addressed comparing scenarios without any cruise systems, with CC systems and with ACC systems. Another work with a different approach is (MASSERA TERRA; WOLF, 2017), that studies Cooperative Adaptive Cruise Control (CACC) systems, considering a mutual communication between the controlled and the leading vehicle.

A control theory that has been very popular in ACC researches is the Model Predictive Control (MPC). This technique is quite particular, since its origins came from industrial control processes. In 2003, Badgwell and Joe published an interesting survey about the MPC development and its use in several engineering areas (QIN; BADGWELL, 2003). The development of MPC is shown to be much attached to the needs of industries of refineries, petrochemicals, chemicals and more. In 1999, the Adersa company had about seven automotive applications using MPC, when ACC Systems were in their early stages of development.

The fundamental theory for the model predictive control is to use a model representation of a system to predict the system performance. The MPC aims to compute a control law that minimizes a cost function, up to few sampling steps ahead (MACIEJOWSKI, 2002). Interestingly, Camacho compares the action of driving with the theory of MPC (CAMACHO; ALBA, 2013). The driver observes the environment around the vehicle and mentally designs a sequence of control actions to drive safely. Likewise, the MPC theory uses the data of the system and its model to design a sequence of control actions.

Next, it is discussed some papers that used MPC, in different forms, in ACC systems. After doing the control law design, (NAUS et al., 2010) shows a method to explicitly indicate the control law. Such method, known as explicit MPC, fragments the controller in regions aiding the controller's programming. In (OLIVERI et al., 2011), it also computes

an explicit MPC, but it uses techniques called PieceWise-Affine (PWA) to describe the multiple regions of the control law.

One of the most impressive papers also used PWA techniques for explicit MPC (CORONA; SCHUTTER, 2008). This paper does a great work comparing all different techniques by checking their system control and programming performances. In (WEISSMANN; GÖRGES; LIN, 2018), the authors combine the MPC theory with dynamic programming to reduce the energy consumption.

Two more papers with particularities are interesting to highlight. The first one (LUO; CHEN; ZHANG, 2016) aims to optimize the number of shifts between accelerator and brake through the control loop. This specification helps to reduce the discomfort of passengers. The second paper (SUN et al., 2016) uses multiple vehicles model in order to describe the system and for designing the model predictive controller.

The most significant paper that helped this research was (SHAKOURI; CZECHOT; ORDYS, 2015), from the same authors that made a translational vehicle model in Matlab (SHAKOURI; CZECHOT; ORDYS, 2010). The authors used this model as a simulated system to compare three controllers: a proportional-integral (PI) controller with Gain Scheduling, a Balanced-based adaptive controller (SHAKOURI; CZECHOT; ORDYS, 2012) and a nonlinear model predictive controller (NMPC). This paper also discusses a switching logic between ACC and cruise control (CC) modes. For the NMPC, the control law analyzes the last control input in order to select a linear model of acceleration or braking and recomputes the MPC matrices. Because the model for control is constantly switching, the authors called it as a nonlinear model predictive control.

One last paper to be discussed did a standard cruise control (DIAS; PEREIRA; PALHARES, 2015), however it did gave some interesting ACC suggestions. The authors from Universidade Federal de Minas Gerais (UFMG) did a simplified phenomenological modeling in order to estimate the system order. Afterward, the authors applied the system identification theory to obtain some system model representations. For the cruise controller, the inverted models were used each sampling time and it was applied a PI controller to decrease the steady state error.

Given the abundant presence of MPC in many ACC Systems researches, this work aims to contribute with an adapted outer loop controller and two different approaches of the MPC theory for the inner loop. The cost function of the MPC is designed with a finite horizon of prediction (conventional MPC) and with an infinite horizon of prediction (known as Infinite Horizon MPC, or IHMPC) (KOTHARE; BALAKRISHNAN;

MORARI, 1996; MARTINS; ODLOAK, 2016; ODLOAK, 2004). Furthermore, these controllers are addressed for an embedded implementation with a vehicle. Ultimately, this work aims the validation of the controllers using a dynamometer.

Most of the papers consider a simulation validation. The main contributions of this thesis are: (i) a practical validation of ACC in a real system with a customized hardware; (ii) two embedded applications of MPC; (iii) an application of an infinite horizon MPC for ACC; (iv) a comparison of two MPC controllers in an ACC application.

This work is structured in chapters to distribute the developed activities and few appendices to aid the comprehension of additional procedures. Firstly, Chapter 2 describes the vehicle used for this work and its particularities. Next, a suitable representation of the vehicle is addressed in Chapter 3. In Chapter 4, the control design for the outer and the inner loop are developed. The Chapter 5 discusses several simulations results in addition with practical experiments using the designed controllers. Lastly, the conclusions of the work and some mentions to future works are featured in Chapter 6.

1.1 Objectives

The main objectives of this research are:

- obtain a suitable model representation of a vehicle;
- study and design controllers with MPC theory;
- study the differences of CC and ACC and their implementations;
- evaluate control loops using simulations;
- program embedded controllers;
- evaluate controllers in embedded applications with a vehicle.

2 BACKGROUND

The platform available for this research consists of the following components: a Volkswagen vehicle, Polo Sedan 2004 with spark-ignition engine 2.0 L, shown in Figure 2, controlled by an open-source electronic control unit (ECU); an inertial dynamometer from NAPRO company; a ARS300 long-range radar from Continental company, for future on-road applications; and a module for ACC management.

2.1 Vehicle

Figure 2: Photo of the Polo vehicle

Source: Author.

The open-source ECU, shown in Figure 3, has been developed by the Automotive Electronics Group from Escola Politécnica da Universidade de São Paulo (EPUSP) aiming the research of ADAS.

Figure 3: Photo of the customized ECU

Source: Author.

The ECU was developed using a decentralized architecture of three microcontrollers, shown in Figure 4. Each microcontroller has a distinct functionality, being for management, synchronism and communication. The communication is available via CAN network of the vehicle and via USB for supervisory software on a computer.

This ECU has been improved in the last 5 years displaying power and torque performance and driving characteristics equivalent to the vehicle factory ECU. To achieve such results, several works were performed: open loop torque control, stoichiometric control of air and fuel mixture, electronic throttle valve position control and engine idling speed control with phase lead compensation for the ignition.

Furthermore, many other features have been implemented: identification of the set gear; speed calculation; increase of start engine idling speed; Serial Peripheral Interface (SPI) communication security between the blocks of the ECU, using Cyclic Redundancy Check (CRC); a supervisory software for computers; multiple operation modes for the ECU (normal, economic and sports); starting engine control; communication with the dashboard of the vehicle; identification and correction of eventual faults; and On-Board Diagnosis (OBD) (PEREIRA, 2015, 2017).

The CAN Messages used for this work are presented in Table 1. These messages were customized in order to easily operate and read the most relevant variables for the ACC system.

Figure 4: Diagram of the ECU

Source: Author.

Table 1: CAN Messages for the ACC System.

CAN Message	ID (Hex)	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Input Signal	200	"E" Token	"C" Token	"U" Token	Input Accelerator Pedal (0-100%)				
Configuration	201	"E" Token	"C" Token	"U" Token	Input Pedal Mode (On or Off)	Set Operation Mode	Idling Speed Mode	-	-
Output Signal	590	Accelerator Pedal (0-100%) Factor 0.01		Measured Speed (Km/h) Factor 0.01		Gear Engaged	Read Operation Mode	-	Error Code

2.2 Dynamometer

All experiments were performed in Instituto de Energia e Ambiente (known as IEE) which has a dynamometer to assist automotive researches, as shown in Figure 5. It has capability of setting a torque load and can be used to simulate different roads, for example higher slope or different terrains.

This load torque can be selected from 0% to 100%, which represents the amount of torque that will oppose the rotational movement, in relation to the maximum possible torque of the dynamometer. Unfortunately, the mapping relation into torque units have high complexity because it depends of the weight applied to the dynamometer axes,

according to the NAPRO engineers. Therefore, the load torque will be considered as an external disturbance.

Dynamometer load was used for increasing the number of identification experiments and, consequently, the number of linear models acquired. There are some control techniques which benefit from having several models to increase the system control performance (MARTINS; ODLOAK, 2016).

Figure 5: Photo of the dynamometer

Source: Author.

2.3 Radar

The ARS300 radar is manufactured by A.D.C. GmbH, a subsidiary of the Continental Corporation (CONTINENTAL, 2017). This radar is a third generation model of long range radars catalog of Continental. It can be seen in Figure 6.

ARS300 is able to provide several messages for different applications, for example distance of multiple objects in adaptive cruise control systems, headway control, traffic light approximation recognition, classification of multiple objects and some other applications.

It has a real time scanning of 15 measurements per second and the distance range is up to 200 meters, which is suitable for high velocity cruise control. For interface, ARS300 has a CAN bus available since its main application is for the automotive area.

Although the Automotive Electronics Group have the ARS300 at its disposal, its functionality is not available. This radar has a customized configuration to work with Mercedes trucks and to operate the radar in sensing mode, a wake-up sequence of messages

Figure 6: Photo of the radar ARS300

Source: Author.

must be sent. Unfortunately, at this present the wake-up messages were not provided. Whence, the ARS300 usage will be implemented in a future work.

2.4 ACC Module

The ACC Module, shown in Figure 7, has been developed to permit a communication between a microcontroller LPC1768, the vehicle ECU and, in the future, with an ARS300 Radar. The LPC1768 (MBED, 2017) is 32-bit ARM Cortex M3 microcontroller, 96 MHz, with 512 kB of memory flash and 32 kB of RAM.

This microcontroller is compatible with the Mbed Compiler and it has two CAN Channels (MBED, 2013). These features will be essential because the CAN network frequencies of the vehicle (500 kbits per second) and the Radar (666 kbits per second) are different, therefore, the ACC module will act as an intermediary between the two CAN networks. With the assistance of Mbed libraries, the ACC algorithm is embedded in the LPC1768.

Other key components to ensure the communication works properly are the two CAN Transceivers MCP2561 (MICROCHIP, 2014) of Microchip, each for both CAN Channels, and the two terminating resistors of $120\ \Omega$. Some coupling capacitors were also added to the circuit in order to improve the quality of the power signals. Finally, there are two connectors, for the Vehicle CAN network and for the Radar network.

Figure 7: Photo of the ACC Module

Source: Author.

3 SYSTEM IDENTIFICATION

As said in Chapter 2, this work uses a vehicle provided by EPUSP as system to implement ACC techniques. The equations for modeling the longitudinal dynamics of a vehicle are ordinary to obtain (SHAKOURI; CZECHOT; ORDYS, 2010). However, the biggest issue for phenomenological modeling is to gather the system parameters, in particular, for the Polo vehicle.

Given this situation, an alternative method for obtaining the vehicle model is using system identification theory (LJUNG, 1999). This modeling method consists of several experiences and collecting the input and output data. Using these data, it is possible to achieve discrete transfer functions that have similar output as the vehicle system, when excited by the same input.

There are several discrete transfer functions that can be used as systems models, likewise characteristics of the input signal, such as its waveform frequency and intensity limits. The most appropriate transfer functions and input signals for system identification varies for every system since there are particularities, for example, its frequency bandwidth.

Although the vehicle has a customized ECU, it does not have an electronic brake system. Thus, this research discarded using the brake pedal as a control input. The control signal chosen for the vehicle is the accelerator pedal, which range varies from 0% (not pushed) to 100% (fully pushed). The vehicle output of an ACC system is the translational vehicle speed (assumed in meters per second). Both signals have specific CAN Messages, as shown in Table 1.

The main goal of this identification is to obtain a model representation of the system, with few parameters, but still a reliable model to the vehicle. The ARX structure was chosen due its simplicity, assuming that the signal-to-noise ratio is good and the disturbance is a white noise (LJUNG, 1999). Certainly, the noise variance is very low due to the robustness of the CAN network.

The selected input signal for the identification was Pseudo-random Binary Sequence,

known as PRBS, due its statistical characteristics similar to a white noise signal (LJUNG, 1999). This signal can be easily generated by using command *idinput* from System Identification Toolbox of Matlab. The signal parameters for generating a PRBS signal are number of samples, frequency bandwidth, and minimum and maximum signal limit.

The number of samples affects the amount of data for analysis, then it is not affected directly by the system. Usually the frequency bandwidth is determined considering the slowest time constant of the system. This time constant, named as τ , might be identified applying a step input and analyzing the system output.

For an experiment of 60 seconds, in the first 30 seconds a CAN message of 30% of accelerator pedal was sent to the ECU. In the last 30 seconds, this CAN message was changed to 15% of accelerator as input. The system step response is presented in Figure 8. The dynamometer torque load was set to 0% and the sampling time was $T_s = 0.5$ seconds ($f_s = 2$ Hz).

Source: Author.

The step response pointed out that the system has a time delay of one sampling time of $\theta = 0.5$ s and a time constant approximately $\tau = 5$ s. A sufficient PRBS frequency bandwidth can be calculated with $f_s/20$ Hz to avoid short bit times, which are the slowest interval that PRBS remains constant during all of its sequence (LJUNG, 1999). Such frequency bandwidth would produce a PRBS signal that remains constant up to $1/0.1$ seconds, or 10 seconds.

The next step is to determine PRBS limit values and that is quite a challenge. The system is nonlinear and for this research the system identification algorithm produces

linear models. Additionally, the vehicle speed response changes with each set gear. Given these circumstances, the following procedure was considered.

Firstly, all system identification and control were established with the vehicle set in the third gear. Experimentally, the vehicle reached speeds from 15 km/h to 120 km/h, under safe conditions of use. Thus, this proposal has a wide speed range and sufficient for ACC.

During the identification experiments, there are intervals that the PRBS maintains constant, either its maximum or minimum designed value. However, the front wheels of the vehicle are in constantly rolling over the dynamometer and, if the accelerator pedal is not enough, the motor engine stalls. Experimentally, this minimum value was identified as 10%. Moreover, if the accelerator pedal is overly pushed, the engine will exceed 6000 rotations per minute, which is detrimental for the engine. Again experimentally, this maximum value was identified as 35%.

Nevertheless, using a PRBS with its range from 10% to 35% would produce an inefficient linear model, because there are nonlinearities in the system that would strongly affect the identification. This research proposes to split accelerator range into intervals and each one would produce a linear model. Each model are identified with different input signals, from 10% to 20%, another signal from 20% to 30% and lastly a signal from 30% to 35%.

3.1 PRBS specification

The characteristics of the PRBS signals used in the system identification are presented in Table 2. The sampling time chosen was $T_s = 0.5$ seconds, or $f_s = 2$ Hz. Every experiment lasted 4 minutes, resulting in 480 samples. The first 3 minutes, or 360 samples, were used for system identification and the last minute of experiment, 120 samples, were used for system validation.

Three dynamometer loads were used: 0%, 10% and 15%. In order to identify the system with each load, thus reducing correlations within each experiment, each input range created 12 minutes of data, or 1440 samples. Within these 12 minutes, the first 4 minutes were used for load 0% (named as load A), the following 4 minutes were used for load 10% (named as load B) and the last 4 minutes for load 15% (named as load C).

Table 2: PRBS data specification

	Frequency bandwidth (Hz)	Minimum value	Maximum Value
Range 10-20	[0 0.1]	10%	20%
Range 20-30	[0 0.1]	20%	30%
Range 30-35	[0 0.1]	30%	35%

Source: Author.

3.2 ARX Models

An autoregressive with exogenous input (ARX) model with one input and one output has the following structure:

$$\begin{aligned}
 y(t) + a_1y(t-1) + \dots + a_nay(t-na) &= b_1u(t-nk) + \dots + b_nbu(t-nk-nb+1) + e(t) \\
 A(q)y(t) &= B(q)u(t) + e(t)
 \end{aligned}$$

Given an input u and an output y , an ARX model calculates all a_i and b_i coefficients in order to reduce the error $e(t)$. The system time delay is defined as nk and the orders of $A(q)$ and $B(q)$, respectively na and nb .

Given $T_s = 0.5$ seconds, for all ARX models the time delay order was defined as $nk = 1$, since the time delay for step response was $\theta = 0.5$ seconds. Using the nine experimental input and output data, the function *ident* swept each combination of na and nb from 1 to 10. This function offered several models that presented higher FIT indexes and those that minimized the Akaike Information Criterion (AIC) (AKAIKE, 1998) or Minimum Description Length (MDL) criterion (RISSANEN, 1978).

The model that minimizes the MDL criterion is the one that has the best compression of data. It means that this model has a high FIT index with the lowest number of system parameters. For presenting a simpler but still reliable model, the MDL criterion was decisive for choosing the orders of the ARX model.

Among all nine model sweeps, the orders $na = 2$ and $nb = 3$ had the best MDL criterion in most cases. The order nb needs to be higher than na because in Section 4, while creating an Output Prediction Oriented Model, it was necessary for the transfer function $B(q)/A(q)$ to be strictly proper.

Since all identification experiments were made with the vehicle in the third gear, at start it was necessary to manually press the accelerator pedal, avoiding the motor engine

to stall. After setting up a sufficient speed, the vehicle's ECU was ready to receive PRBS input messages. For this reason, each experiment had an initial condition not equal to zero and due to system nonlinearities, there might appear some phenomena affecting the system identification.

A possible solution is to pre-process all input and output data, in order to improve the identification of linear models. The pre-processing function used was *detrend* from Matlab, that removes linear trends of matrices. This function removes the mean value of input and output data, so essentially it takes out the initial condition effect.

From now on, the ARX models will be referenced as shown in Table 3. The number represents the PRBS range and the letter represents the dynamometer load used for identification.

Table 3: Models designation			
	Load A	Load B	Load C
PRBS Range 10-20	Model 1A	Model 1B	Model 1C
PRBS Range 20-30	Model 2A	Model 2B	Model 2C
PRBS Range 30-35	Model 3A	Model 3B	Model 3C

Source: Author.

In Table 4, it is shown all nine ARX models coefficients, changing the minimum and maximum input values and the dynamometer load. Every system has one sample delay ($nk = 1$) with $T_s = 0.5$ seconds.

Table 4: Coefficients and time delay of ARX models

	a_1	a_2	b_1	b_2	b_3	n_k
Model 1A	-1.31	0.40	1.78	3.87	-0.78	1
Model 1B	-0.98	0.15	5.60	1.94	-0.07	1
Model 1C	-1.20	0.36	2.78	3.03	-0.14	1
Model 2A	-1.42	0.46	4.70	1.75	-1.97	1
Model 2B	-1.30	0.36	6.23	0.84	-1.00	1
Model 2C	-1.33	0.40	4.98	2.53	-1.31	1
Model 3A	-1.52	0.56	5.06	-1.28	-0.14	1
Model 3B	-1.33	0.38	7.50	-0.66	-1.23	1
Model 3C	-1.27	0.33	7.58	-0.10	-1.15	1

Source: Author.

3.3 Models Validation

The experiment data selected for validation were used to compare ARX models response with the corresponding experimental vehicle speed. The FIT indexes (LJUNG, 1999) were calculated for 1, 5, 10, 50 and infinite (simulation) steps ahead and their results are presented in Table 5. Despite the evident decrease of the FIT index with higher prediction steps, these results are remarkably high given that in the worst situation, infinite steps ahead, the lowest index was 73.87%, which is generally a decent value.

Table 5: FIT (%) for system validation, with several configurations of steps ahead

	1	5	10	50	∞
Model 1A	95.56	86.49	83.03	84.56	82.28
Model 1B	96.14	89.73	87.95	91.72	87.37
Model 1C	94.01	84.47	83.29	83.88	83.27
Model 2A	96.15	87.80	83.35	85.87	79.62
Model 2B	96.60	90.12	86.83	84.86	84.15
Model 2C	94.40	81.55	75.41	77.51	74.09
Model 3A	95.10	84.95	78.45	76.48	73.87
Model 3B	94.90	88.97	85.83	84.39	83.93
Model 3C	92.54	82.83	77.28	84.97	74.47

Source: Author.

4 CONTROL DESIGN

In this research, one controller is designed for the outer loop and two controllers for the inner loop using the MPC theory. The outer loop controller must be able to change between Cruise Control (CC mode) and Adaptive Cruise Control (ACC mode). For CC mode, the user defined speed should be the cruise speed, while for ACC mode, a new cruise reference speed must be calculated in order to maintain a safe distance.

4.1 ACC Controller

The algorithm for the outer loop controller was adapted from (SHAKOURI; CZECHOT; ORDYS, 2015). Designating the vehicle speed as v , the user defined speed as v_{user} , the measured distance d with the closest vehicle, a security distance d_{ref} with the closest vehicle, a relative speed (v_r) between the closest vehicle and the system vehicle.

In real applications, the closest vehicle speed v_l is estimated by the progression of the distance d to the same vehicle. Some radar models, such as ARS 300, is able to receive the controlled vehicle speed as a message and provide an estimated relative speed v_r with the closest vehicle.

The controller for the outer loop must compute d_{ref} in order to maintain a safe distance to the leading vehicle. In (SHAKOURI; CZECHOT; ORDYS, 2015), the authors suggest computing d_{ref} as in Equation (4.1). The parameter ℓ is the length of the system vehicle, d_s it is an additional distance to avoid crashes and T_h is known as constant-time headway, which estimates a human driver reaction time. Usually this value varies between 0.8 and 2 seconds.

$$d_{ref} = \ell + d_s + T_h v \quad (4.1)$$

The same authors suggest computing a reference speed, named as v_{ref} , as shown in Equation (4.2) (SHAKOURI; CZECHOT; ORDYS, 2015). If the distance d for the closest

vehicle is the same as the security distance d_{ref} , the system is safe to travel with the same speed of the closest vehicle v_l , commonly described as leading vehicle. Otherwise, it is necessary to change the security speed v_{ref} using a proportional gain K_p .

$$v_{ref} = v_l - K_p(d_{ref} - d) \quad (4.2)$$

The adapted control law is shown in Algorithm 1. Firstly, the controller computes d_{ref} and v_{ref} . Next, it compares if the reference speed is greater than the user defined speed. If this statement is true, the cruise speed must be limited to v_{user} , causing the controller to enter in CC mode. Otherwise, the cruise speed will be adjusted to v_{ref} , in order to keep a safe distance from the leading vehicle (ACC mode).

Algorithm 1 Outer loop controller

- 1: $d_{ref} = \ell + d_s + T_h v$;
 - 2: $v_{ref} = v_l - K_p(d_{ref} - d)$;
 - 3: If $v_{ref} > v_{user}$, $v_{cruise} = v_{user}$ (CC Mode);
 - 4: Otherwise, $v_{cruise} = v_{ref}$ (ACC Mode).
-

4.2 Conventional MPC

For the Model Predictive Control to accomplish its functions, there must be a suitable model to work with. Recently in MPC researches, a state space model is used as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k) \end{aligned} \quad (4.3)$$

This model is in an incremental form, that has an input signal given by $\Delta u(k) = u(k) - u(k-1)$. The incremental form is able to reduce output offsets if the desired state is reachable.

Also, if the model has time delays, it is necessary to include them in the model. Thus, in Appendix B it is detailed every ARX model transformation to fulfil MPC model requirements.

4.2.1 Conventional MPC Design

The objective function for Conventional MPC, at any instant k , can be described as:

$$J_k^{\text{MPC}} = \sum_{j=1}^p (y(k+j|k) - y^{sp})^T Q (y(k+j|k) - y^{sp}) + \sum_{j=0}^{m-1} \Delta u(k+j|k)^T R \Delta u(k+j|k) \quad (4.4)$$

The element y^{sp} is the desired output value, m is the control horizon, p is the prediction horizon, Q and R are weight matrices related to output error and control input, respectively.

Foremost, the output prediction vector is assembled, considering that after the sample $k+m$ there are no more control inputs ($\Delta u(k+m|k) = \Delta u(k+m+1|k) = \dots = 0$). The Equation (4.5) shows the prediction to the horizon p , compressing all m control signals into Δu_k .

$$\bar{y}(k) = \Phi x(k) + \Gamma \Delta u_k, ; \Delta u_k = \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix} \quad (4.5)$$

Where:

$$\Phi = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^m \\ CA^{m+1} \\ \vdots \\ CA^p \end{bmatrix}; \Gamma = \begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ CA^{m-1}B & CA^{m-2}B & \dots & CB \\ CA^mB & CA^{m-1}B & \dots & CAB \\ \vdots & \vdots & \vdots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \dots & CA^{p-m}B \end{bmatrix}$$

Considering that the output set-point y^{sp} for any output prediction, then the set-point vector will be $\bar{y}^{sp} = \left[\underbrace{y^{sp^T} \dots y^{sp^T}}_p \right]^T$. In order to expand the summation in (4.4), it is also necessary to consider weight matrices Q and R in their respective horizons, obtaining $\bar{Q} = \text{diag} \left[\underbrace{Q \dots Q}_p \right]$ and $\bar{R} = \text{diag} \left[\underbrace{R \dots R}_m \right]$.

Expanding all elements of (4.4), the following expression is obtained:

$$J_k^{\text{MPC}} = (\Phi x(k) + \Gamma \Delta u_k - \bar{y}^{sp})^T \bar{Q} (\Phi x(k) + \Gamma \Delta u_k - \bar{y}^{sp}) + \Delta u_k^T \bar{R} \Delta u_k \quad (4.6)$$

The objective function J_k^{MPC} can still be reduced to a quadratic form:

$$J_k^{\text{MPC}} = \Delta u_k^T H \Delta u_k + 2c_f^T \Delta u_k + c \quad (4.7)$$

where

$$\begin{cases} H = \Gamma^T \bar{Q} \Gamma + \bar{R}; \\ c_f^T = (\Phi x(k) - \bar{y}^{sp})^T \bar{Q} \Gamma; \\ c = (\Phi x(k) - \bar{y}^{sp})^T \bar{Q} (\Phi x(k) - \bar{y}^{sp}). \end{cases}$$

The control law for conventional MPC will be the solution of the following Quadratic Programming (QP):

$$\begin{aligned} \min_{\Delta u_k} \quad & \Delta u_k^T H \Delta u_k + 2c_f^T \Delta u_k \\ \text{s.t.} \quad & \\ & -\Delta u_{\max} \leq \Delta u(k+j|k) \leq \Delta u_{\max}, j = 0, 1, \dots, m-1 \\ & u_{\min} \leq u(k+j|k) \leq u_{\max}, j = 0, 1, \dots, m-1 \end{aligned} \quad (4.8)$$

For MPC Quadratic Programming implementation, it is required to adjust all constraints for Hildreth Quadratic Programming (Appendix A) into constraints in the form $A_c \Delta u_k \leq b_c$. Using the identity $u(k) = \Delta u(k) + u(k-1)$, A_c and b_c will be:

$$A_c = \begin{bmatrix} I_{nu} & 0 & 0 & \dots & 0 \\ 0 & I_{nu} & 0 & \dots & 0 \\ 0 & 0 & I_{nu} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & I_{nu} \\ -I_{nu} & 0 & 0 & \dots & 0 \\ 0 & -I_{nu} & 0 & \dots & 0 \\ 0 & 0 & -I_{nu} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -I_{nu} \\ I_{nu} & 0 & 0 & \dots & 0 \\ I_{nu} & I_{nu} & 0 & \dots & 0 \\ I_{nu} & I_{nu} & I_{nu} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I_{nu} & I_{nu} & I_{nu} & \dots & I_{nu} \\ -I_{nu} & 0 & 0 & \dots & 0 \\ -I_{nu} & -I_{nu} & 0 & \dots & 0 \\ -I_{nu} & -I_{nu} & -I_{nu} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -I_{nu} & -I_{nu} & -I_{nu} & \dots & -I_{nu} \end{bmatrix}; b_c = \begin{bmatrix} \Delta u_{\max} \\ \Delta u_{\max} \\ \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \\ \Delta u_{\max} \\ \Delta u_{\max} \\ \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \\ u_{\max} - u(k-1) \\ u_{\max} - u(k-1) \\ u_{\max} - u(k-1) \\ \vdots \\ u_{\max} - u(k-1) \\ -u_{\min} + u(k-1) \\ -u_{\min} + u(k-1) \\ -u_{\min} + u(k-1) \\ \vdots \\ -u_{\min} + u(k-1) \end{bmatrix}.$$

Finally, the control law for implementing the MPC will be the solution of:

$$\begin{aligned} \min_{\Delta u_k} \Delta u_k^T H \Delta u_k + 2c_f^T \Delta u_k \\ s.t. \\ A_c \Delta u_k \leq b_c \end{aligned} \tag{4.9}$$

4.3 Infinite Horizon MPC

Similar to conventional MPC, it is recommended to use specific models for accomplishing IHMPC control functions. One model suggestion is Output Prediction Oriented Model (OPOM), which is detailed in Appendix C (MARTINS; ODLOAK, 2016).

This model is also in the incremental form. It has incorporated time delays and has some advantages when used as model for IHMPC.

However, the OPOM might have complex matrices, resulting in complex states. This occurrence is detrimental because some QP solvers do not operate with complex problems. In Appendix D it is shown a method to transform the OPOM to only have real states.

4.3.1 IHMPC Design

The objective function for IHMPC can be described as (ODLOAK, 2004):

$$J_k^{\text{IHMPC}} = \underbrace{\sum_{j=0}^{\infty} (y(k+j|k) - y^{sp} - \delta_y)^T Q (y(k+j|k) - y^{sp} - \delta_y)}_{J_1} + \sum_{j=0}^{m-1} \Delta u(k+j|k)^T R \Delta u(k+j|k) + \delta_y^T S_y \delta_y \quad (4.10)$$

For this control technique, it is essential to use slack variables δ_y for each output, because the control law will converge to an expression with equality constraints. Without slack variables, it is possible to have unfeasible solutions. The solution for IHMPC control law must calculate control input Δu and slack variable δ_y . With this control design, S_y is the weight matrix related to the slack variables.

At first, an expansion of the infinite summation of J_k^{IHMPC} is expressed:

$$J_1 = \underbrace{\sum_{j=0}^{m+\theta_{max}} (y(k+j|k) - y^{sp} - \delta_y)^T Q (y(k+j|k) - y^{sp} - \delta_y)}_{J_{1a}} + \underbrace{\sum_{j=m+\theta_{max}+1}^{\infty} (y(k+j|k) - y^{sp} - \delta_y)^T Q (y(k+j|k) - y^{sp} - \delta_y)}_{J_{1b}}$$

Similar to the conventional MPC, an output prediction vector is calculated with $m + \theta_{max}$ as the prediction horizon and all m control signals are compressed into Δu_k , as shown in Equation (4.11).

$$\bar{y}(k) = \bar{A}x(k) + \bar{B}\Delta u_k; \Delta u_k = \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix} \quad (4.11)$$

Where:

$$\overline{A} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^m \\ \vdots \\ CA^{m+\theta_{max}} \end{bmatrix}; \overline{B} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ CB & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ CA^{m-1}B & CA^{m-2}B & \dots & CB \\ \vdots & \vdots & \vdots & \vdots \\ CA^{m+\theta_{max}-1}B & CA^{m+\theta_{max}-2}B & \dots & CA^{\theta_{max}}B \end{bmatrix}$$

Consequently, creating an output set-point vector $\overline{y}^{sp} = \left[\underbrace{y^{sp^T} \dots y^{sp^T}}_{m+\theta_{max}+1} \right]^T$, a supporting vector $\overline{I}_{ny} = \left[\underbrace{I_{ny} \dots I_{ny}}_{m+\theta_{max}+1} \right]^T$ and adjusting the weighting matrix Q as $\overline{Q}_y = \text{diag} \left[\underbrace{Q \dots Q}_{m+\theta_{max}+1} \right]$, the first expansion element of J_1 can be calculated as:

$$J_{1a} = (\overline{A}x(k) + \overline{B}\Delta u_k - \overline{y}^{sp} - \overline{I}_{ny}\delta_y)^T \overline{Q}_y (\overline{A}x(k) + \overline{B}\Delta u_k - \overline{y}^{sp} - \overline{I}_{ny}\delta_y)$$

For the second element of J_k^{HMPC} , the output prediction of the OPOM for any time instant j after $m + \theta_{max}$ can be described as:

$$y(k + m + \theta_{max} + j|k) = x^s(k + m + \theta_{max}|k) + \Psi x^d(k + m + \theta_{max} + j|k) \quad (4.12)$$

The definition of the states x^s and x^d and the matrices Ψ and F are described in Appendices C and D. Replacing expansion (4.12) in J_{1b} :

$$J_{1b} = \sum_{j=1}^{\infty} (x^s(k + m + \theta_{max}|k) + \Psi x^d(k + m + \theta_{max} + j|k) - y^{sp} - \delta_y)^T Q_y (x^s(k + m + \theta_{max}|k) + \Psi x^d(k + m + \theta_{max} + j|k) - y^{sp} - \delta_y)$$

Being the matrix F stable, the condition for J_{1b} to be bounded is:

$$x^s(k + m + \theta_{max}|k) - y^{sp} - \delta_y = 0$$

If the above condition is satisfied, J_{1b} remains as follows:

$$\begin{aligned}
J_{1b} &= \sum_{j=1}^{\infty} (\Psi x^d(k+m+\theta_{max}+j|k))^T Q_y (\Psi x^d(k+m+\theta_{max}+j|k)) \\
&= \sum_{j=1}^{\infty} (\Psi F^j x^d(k+m+\theta_{max}|k))^T Q_y (\Psi F^j x^d(k+m+\theta_{max}|k)) \\
&= x^d(k+m+\theta_{max}|k)^T \underbrace{\left(\sum_{j=1}^{\infty} F^{jT} \Psi^T Q_y \Psi F^j \right)}_{\overline{Q_d}} x^d(k+m+\theta_{max}|k) \quad (4.13)
\end{aligned}$$

The matrix $\overline{Q_d}$ can be calculated as solution of the following discrete Lyapunov equation:

$$\overline{Q_d} = F^T \Psi^T Q_y \Psi F + F^T \overline{Q_d} F$$

Next, the predicted states $x^s(k+m+\theta_{max}|k)$ and $x^d(k+m+\theta_{max}|k)$ are obtained:

$$\begin{cases} x^s(k+m+\theta_{max}|k) = N_s A^{\theta_{max}} (A^m x(k) + W \Delta u_k) \\ x^d(k+m+\theta_{max}|k) = N_d A^{\theta_{max}} (A^m x(k) + W \Delta u_k) \end{cases}$$

where

$$\begin{cases} N_s = \begin{bmatrix} I_{ny} & 0_{ny \times nd} & 0_{ny \times \theta_{max}} \end{bmatrix} \\ N_d = \begin{bmatrix} 0_{nd \times ny} & I_{nd} & 0_{nd \times \theta_{max}} \end{bmatrix} \\ W = \begin{bmatrix} A^{m-1} B & A^{m-2} B & \dots & B \end{bmatrix} \end{cases}$$

The next operation consists of replacing the previous expressions in $J_k^{\text{IHMP}}C$:

$$\begin{aligned}
J_k^{\text{IHMP}}C &= (\overline{A}x(k) + \overline{B}\Delta u_k - \overline{y}^{sp} - \overline{I}_{ny}\delta_y)^T \overline{Q_y} (\overline{A}x(k) + \overline{B}\Delta u_k - \overline{y}^{sp} - \overline{I}_{ny}\delta_y) \\
&+ (N_d A^{\theta_{max}} (A^m x(k) + W \Delta u_k))^T \overline{Q_d} (N_d A^{\theta_{max}} (A^m x(k) + W \Delta u_k)) \\
&+ \Delta u_k^T \overline{R} \Delta u_k + \delta_y^T S_y \delta_y
\end{aligned}$$

As well as in MPC formulation, $J_k^{\text{IHMP}}C$ can be reduced into a quadratic form:

$$J_k^{\text{IHMP}}C = \begin{bmatrix} \Delta u_k & \delta_y \end{bmatrix} \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \delta_y \end{bmatrix} + 2 \begin{bmatrix} c_{f1} & c_{f2} \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \delta_y \end{bmatrix} + c$$

where

$$\left\{ \begin{array}{l} H_{11} = (\overline{B})^T \overline{Q}_y (\overline{B}) + (N_d A^{\theta_{max}} W)^T \overline{Q}_d (N_d A^{\theta_{max}} W) + \overline{R}; \\ H_{12} = -(\overline{B})^T \overline{Q}_y (\overline{I}_{ny}); \\ H_{21} = H_{12}^T; \\ H_{22} = (\overline{I}_{ny})^T \overline{Q}_y (\overline{I}_{ny}) + S_y; \\ c_{f1} = (\overline{A}x(k) - \overline{y}^{sp})^T \overline{Q}_y (\overline{B}) + (N_d A^{\theta_{max}+m} x(k))^T \overline{Q}_d (N_d A^{\theta_{max}} W); \\ c_{f2} = -(\overline{A}x(k) - \overline{y}^{sp})^T \overline{Q}_y (\overline{I}_{ny}); \\ c = (\overline{A}x(k) - \overline{y}^{sp})^T \overline{Q}_y (\overline{A}x(k) - \overline{y}^{sp}) + (N_d A^{\theta_{max}+m} x(k))^T \overline{Q}_d (N_d A^{\theta_{max}+m} x(k)). \end{array} \right.$$

The control law for IHMPC will be the solution of the following QP:

$$\begin{aligned} \min_{\Delta u_k, \delta_y} \quad & \begin{bmatrix} \Delta u_k & \delta_y \end{bmatrix} \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \delta_y \end{bmatrix} + 2 \begin{bmatrix} c_{f1} & c_{f2} \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \delta_y \end{bmatrix} \\ \text{s.t.} \quad & \\ & -\Delta u_{\max} \leq \Delta u(k+j|k) \leq \Delta u_{\max}, j = 0, 1, \dots, m-1 \\ & u_{\min} \leq u(k+j|k) \leq u_{\max}, j = 0, 1, \dots, m-1 \\ & N_s A^{\theta_{max}} (A^m x(k) + W \Delta u_k) - y^{sp} - \delta_y = 0 \end{aligned} \tag{4.14}$$

Similar to implementing conventional MPC, the constraints must be transformed into the form $A_c \begin{bmatrix} \Delta u_k & \delta_y \end{bmatrix}^T \leq B_c$, in order to use the Hildreth's Quadratic Programming (Appendix A). The control law for implementing IHMPC will be the solution of the following minimization problem and the matrices A_c and b_c for IHMPC are shown next.

$$\begin{aligned} \min_{\Delta u_k, \delta_y} \quad & \begin{bmatrix} \Delta u_k & \delta_y \end{bmatrix} \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \delta_y \end{bmatrix} + 2 \begin{bmatrix} c_{f1} & c_{f2} \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \delta_y \end{bmatrix} \\ \text{s.t.} \quad & \\ & A_c \begin{bmatrix} \Delta u_k \\ \delta_y \end{bmatrix} \leq b_c \end{aligned} \tag{4.15}$$

$$A_c = \begin{bmatrix} I_{nu} & 0 & 0 & \dots & 0 & 0 \\ 0 & I_{nu} & 0 & \dots & 0 & 0 \\ 0 & 0 & I_{nu} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & I_{nu} & 0 \\ -I_{nu} & 0 & 0 & \dots & 0 & 0 \\ 0 & -I_{nu} & 0 & \dots & 0 & 0 \\ 0 & 0 & -I_{nu} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -I_{nu} & 0 \\ I_{nu} & 0 & 0 & \dots & 0 & 0 \\ I_{nu} & I_{nu} & 0 & \dots & 0 & 0 \\ I_{nu} & I_{nu} & I_{nu} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ I_{nu} & I_{nu} & I_{nu} & \dots & I_{nu} & 0 \\ -I_{nu} & 0 & 0 & \dots & 0 & 0 \\ -I_{nu} & -I_{nu} & 0 & \dots & 0 & 0 \\ -I_{nu} & -I_{nu} & -I_{nu} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -I_{nu} & -I_{nu} & -I_{nu} & \dots & -I_{nu} & 0 \\ [& N_s A^{\theta_{max}} W &] & -1 \\ [& -N_s A^{\theta_{max}} W &] & 1 \end{bmatrix}; b_c = \begin{bmatrix} \Delta u_{\max} \\ \Delta u_{\max} \\ \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \\ \Delta u_{\max} \\ \Delta u_{\max} \\ \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \\ u_{\max} - u(k-1) \\ u_{\max} - u(k-1) \\ u_{\max} - u(k-1) \\ \vdots \\ u_{\max} - u(k-1) \\ -u_{\min} + u(k-1) \\ -u_{\min} + u(k-1) \\ -u_{\min} + u(k-1) \\ \vdots \\ -u_{\min} + u(k-1) \\ y_{sp} - N_s A^{\theta_{max}+m} x(k) \\ -y_{sp} + N_s A^{\theta_{max}+m} x(k) \end{bmatrix}.$$

4.4 Kalman Filter

Both in MPC and IHMPC, the estimated system states are required for control design. The states are used in computing c_f for MPC and in computing c_{f1} and c_{f2} for IHMPC. One of the most effective procedures to estimate states is using a Kalman filter (KALMAN, 1960).

4.4.1 Kalman Filter for the MPC

At each time sampling, the Algorithm 2 is executed in order to correctly estimate the system states for the model of the MPC.

Algorithm 2 Kalman Filter for the MPC

- 1: Estimate model output (Equation (4.16));
 - 2: Correct the states using the Kalman gain (Equation (4.17));
 - 3: Compute ACC cruise speed and MPC QP;
 - 4: Update the states using the model (Equation (4.18)).
-

The first step of algorithm is to simply compute the model output using the current states $\bar{x}(k)$ (Equation (4.16)). Note that at the start of the simulation or experiment, the states are equal to zero and the estimated output would be zero. This is due to the absence of sensor data at that moment.

$$y_c(k) = C_{\text{MPC}}\bar{x}(k) \quad (4.16)$$

Next, the current system output $y(k)$, obtained from the CAN Message that provides the vehicle speed, is used to correct the states values $x(k)$ (Equation (4.17)). The Kalman Filter K_f^{MPC} is calculated previously using the model for the MPC (KALMAN, 1960).

$$x(k) = \bar{x}(k) + K_f^{\text{MPC}}.(y(k) - y_c(k)) \quad (4.17)$$

After computing the cruise speed, the states $x(k)$ are used to compute c_f and the Hildreth's QP algorithm computes the next MPC control input. The next step is the update procedure with the new control input (Equation (4.18)), using the model for the MPC.

$$\bar{x}(k+1) = A_{\text{MPC}}x(k) + B_{\text{MPC}}\Delta u(k) \quad (4.18)$$

4.4.2 Kalman Filter for the IHMPC

At each time sampling, the Algorithm 3 is executed in order to correctly estimate the system states for the OPOM.

Algorithm 3 Kalman Filter for the IHMPC

- 1: Estimate model output (Equation (4.19));
 - 2: Correct the states using the Kalman gain (Equation (4.20));
 - 3: Compute ACC cruise speed and IHMPC QP;
 - 4: Update the states using the model (Equation (4.21)).
-

The first step of algorithm is to simply compute the model output using the current

states (Equation (4.19)).

$$y_c(k) = C_{\text{IHMPC}}\bar{x}(k) \quad (4.19)$$

Next, the current system output $y(k)$, obtained from the CAN Message that provides the vehicle speed, is used to correct the states values (Equation (4.20)). The Kalman Filter K_f^{IHMPC} is calculated previously using the OPOM (KALMAN, 1960), which is different to Kalman Filter for the MPC due to the distinct state space representation.

$$x(k) = \bar{x}(k) + K_f^{\text{IHMPC}}.(y(k) - y_c(k)) \quad (4.20)$$

After computing the cruise speed, the states $x(k)$ are inserted in order to compute c_{f1} and c_{f2} . Afterward, the Hildreth's QP algorithm computes the next IHMPC control input. The final step is the update procedure with the new control input (Equation (4.21)), using the OPOM.

$$\bar{x}(k+1) = A_{\text{IHMPC}}x(k) + B_{\text{IHMPC}}\Delta u(k) \quad (4.21)$$

5 SIMULATION AND PRACTICAL RESULTS

The results of this work are divided into simulation and practical sections for each designed controller. Foremost, the tuning process was executed using cruise control simulations (CC mode).

For enhancing tuned controllers analysis, a switching logic for multiple models was developed (Appendix E). This conception allowed further simulations of different driving situations, for example, terrain change and system speed response closer to the vehicle. Lastly, ACC mode simulations are presented using preset data to perform as a radar, using the controller of Section 4.1.

There are two main variables for the outer loop controller, d_{ref} and v_{ref} . The time-headway T_h is the time of action and reaction of a driver, which varies from 0.8 to 2 seconds, without any adverse conditions (SHAKOURI; CZECHOT; ORDYS, 2015). For the simulations and practical experiments, the slower situation was chosen. The security distance d_{ref} was computed with:

$$\ell = 4\text{ m}; d_s = 1\text{ m}; T_h = 2\text{ s}$$

In all experiments, the distance d with the leading vehicle was estimated integrating the relative speed v_r . In on-road applications, this distance should be given by a long-range radar. The tuning process of the K_p was empirically performed considering several experiments. High values of K_p increased the oscillations of the cruise speed v_{cruise} and lower values presented a slower control performance. The computing of v_{ref} (Equation (4.2)) was tuned with:

$$K_p = 0.1$$

The system actuator usually has three constraints: maximum value, minimum value and maximum slew rate. As discussed in Chapter 3, the maximum value of the accelerator

pedal is 1 and the minimum value is 0. Since there is no explicit limitations regarding accelerator slew rate, this constraint was chosen as a tuning parameter for a smoother controller response. The maximum slew rate chosen was $\Delta u_{max} = 0.1$, for positive and negative input variations.

5.1 Simulation Experiments

5.1.1 Conventional MPC

Firstly, the ARX models were transformed using the theory of Appendix B. The tuning parameters of the MPC are: prediction horizon p , control horizon m , output error weight matrix Q and control input weight matrix R . An increase in p designs a control that predicts more sampling times ahead, and an increase in m gives a smoother control actions. An increase in Q penalizes the output prediction error and R penalizes higher control input actions (MACIEJOWSKI, 2002).

The first MPC controller simulation was set in CC mode, with an output set-point $y^{sp} = 15$ m/s. The schematic for the tuning simulation is shown in Figure 9. For each ARX model, a simulation was conducted using the same model for the MPC computing and for the system. The chosen tuning parameters for all the MPC were:

$$p = 10; m = 5; Q = 5; R = 1000.$$

Figure 9: Schematic of the tuning simulation.

Source: Author.

During the simulation, in the instant $t = 25$ seconds the set-point was reduced by 20%. At $t = 40$ seconds, an input disturbance of $\Delta u = -0.15$ was applied. The simulation for Models 1A, 1B and 1C are presented in Figure 10.

Figure 10: First three simulations of MPC using the same models as target systems.

Source: Author.

The simulation for Models 2A, 2B and 2C are presented in Figure 11.

Figure 11: More three simulations of MPC using the same models as target systems.

Source: Author.

The simulation for Models 3A, 3B and 3C are presented in Figure 12. Note that each model has a distinct steady state gain and time constant τ .

Furthermore, for each previous simulations the objectives functions J_k were calculated and presented in Figure 13. All nine controllers reached a minimum value equal to zero,

Figure 12: Last three simulations of MPC using the same models as target systems.

Source: Author.

since each controller managed to track their set-points. Even with the presence of a disturbance at $t = 40$ seconds, the MPC was able to recover the set-point tracking.

Figure 13: Objective functions of all nine simulations of MPC.

Source: Author.

The first simulation with the developed switching models logic was carried out to represent a vehicle driving through a soft terrain into a harder terrain, for example with higher slope hill, snow or a dirt road. In each of the three simulations, it was desired to track the set-point even with the system model changing from a low load model (A), to a medium load model (B), and finally through a high load model (C). For example, the MPC with the model 2A had to track the set-point while the system changed between the models 2A, 2B and 2C. The schematic of this simulation is shown in Figure 14.

Figure 14: Schematic of the simulation with changing load models.

Source: Author.

The first simulations with changing models are presented in Figure 15. In $t = 25, 75$ seconds, the switching models logic is not as smooth as the rest of the simulation, causing a sudden system output change.

Figure 15: System response to three different MPC for a system with changing models.

Source: Author.

The second simulation with switching models logic was set to represent an increase in speed between the linear models, making it similar to a nonlinear system. Starting with a set-point of 10 m/s and slowly increasing up to 20 m/s, the controllers needed to follow the changing set-point. The system models change between 1,2 and 3, with the same load as the controller. For example, Model 1A simulation goes through 1A, 2A and 3A; Model 1B goes through 1B, 2B and 3B. The schematic of this simulation is shown in Figure 16 and their results are presented in Figure 17.

In both simulations with constantly changing models, the MPC, with just one model in the control design, was able to track the set-point accordingly.

Figure 16: Schematic of the simulation with changing speed models.

Source: Author.

Figure 17: System response to three different MPC for a system acting as nonlinear.

Source: Author.

In order to inspect the ACC controller operation cooperatively with MPC, two simulations were proposed. In Simulation 1, the controlled vehicle with ACC and MPC started with zero speed and another vehicle passes through with a speed of 12 m/s. The desired speed for the controlled vehicle is 15 m/s. The schematic of the Simulation 1 is shown in Figure 18.

In Figure 19, the MPC with Model 1A tries to follow the computed cruise speed. In $t = [20, 50, 70]$, the leading vehicle slowly accelerates by 10%. Additionally, the system model changes between models with higher loads, as the simulation of Figure 15.

In Figure 20, the distance d to the leading vehicle, the safety distance d_{ref} and the control signal of Simulation 1 are shown. During all the experiment, the ACC System managed to maintain a safe distance to the leading vehicle. After $t = 70$ seconds, the leading vehicle surpassed the user desired speed v_{user} . The outer loop controller did not track further the leading vehicle speed because the maximum value in CC Mode is v_{user} .

Figure 18: Schematic of the ACC Simulation 1.

Source: Author.

Figure 19: Simulation 1 of ACC with MPC for a system with changing models.

Source: Author.

For Simulation 2, the controlled vehicle with ACC and MPC also started with zero speed and another vehicle passes through with a speed of 12 m/s. The desired speed for the controlled vehicle is also 15 m/s. The schematic of the Simulation 2 is shown in Figure 21.

In Figure 22, the MPC with Model 1A tries to follow the computed cruise speed. In $t = [20, 50, 70]$, the leading vehicle slowly accelerates by 10%. Additionally, the system model changes between models representing higher speeds, as the simulation of Figure 17.

In Figure 23, the distance d to the leading vehicle, the safety distance d_{ref} and the control signal of Simulation 2 are shown. As in Simulation 1, the MPC managed to maintain a safe distance to the leading vehicle during all Simulation 2, even with the system changing models.

Figure 20: Corresponding distances and control signal of Figure 19.

Source: Author.

Figure 21: Schematic of the ACC Simulation 2.

Source: Author.

5.1.2 Infinite Horizon MPC

The ARX models were transformed using the theory of Appendixes C and D. Similarly to MPC, the maximum slew rate chosen was $\Delta u_{max} = 0.1$.

The tuning parameters of IHMPC are: control horizon m , output error weight matrix Q , control input weight matrix R and slack variables weight matrix S_y . An increase in m also gives a smoother control action as in MPC. An increase in Q penalizes the output prediction error and an increase in R penalizes higher control input actions. However, it is important that $S_y > Q$, otherwise it is possible for the control to find a minimum with slacks nonzero, which will allow set-point errors. The tuning parameters chosen for the IHMPC were:

Figure 22: Simulation 2 of ACC with MPC for a system with changing models.

Source: Author.

Figure 23: Corresponding distances and control signal of Figure 22.

Source: Author.

$$m = 10; Q = 0.1; R = 100; S_y = 1000.$$

The same simulations with the MPC were performed with the IHMPC. Firstly, the simulation with Models 1A, 1B and 1C are presented in Figure 24.

The simulation for Models 2A, 2B and 2C are presented in Figure 25.

The simulation for Models 3A, 3B and 3C are presented in Figure 26.

Furthermore, for each previous simulations the objective functions J_k were calculated

Figure 24: First three simulations of IHMPC using the same models as target systems.

Source: Author.

Figure 25: More three simulations of IHMPC using the same models as target systems.

Source: Author.

and presented in Figure 27. All nine controllers reached a minimum value equal to zero since each controller managed to track their set-points.

The simulations with the switching models logic with IHMPC were the same as with MPC. Firstly, a simulation of changing the terrain, in Figure 28.

Secondly, the controllers needed to follow a changing set-point. The system model changed between 1, 2 and 3, with the same load as the controller. For example, Model 1A simulation goes through 1A, 2A and 3A; Model 1B goes through 1B, 2B and 3B. These results are presented in Figure 29.

Figure 26: Last three simulations of IHMPC using the same models as target systems.

Source: Author.

Figure 27: Objective functions of all nine simulations of IHMPC.

Source: Author.

The Simulations 1 and 2 were remade with the outer loop controller and with the IHMPC. In Figure 30, the IHMPC with Model 1A tries to follow the computed cruise speed. The system model changes to models with higher loads during the experiment.

In Figure 31, the distance d to the leading vehicle, the safety distance d_{ref} and the control signal of Simulation 1 are shown. During all the experiment, the ACC System managed to maintain a safe distance to the leading vehicle.

In Figure 32, the IHMPC with Model 1A tries to follow a cruise speed computed by the ACC controller. The system model changes to models representing higher speeds

Figure 28: System response to three different IHMPC for a system with changing models.

Source: Author.

Figure 29: System response to three different IHMPC for a system acting as a nonlinear.

Source: Author.

during the simulation.

In Figure 33, the distance d to the leading vehicle, the safety distance d_{ref} and the control signal of Simulation 2 are shown.

Figure 30: Simulation 1 of ACC with IHMPC for a system with changing models.

Source: Author.

Figure 31: Corresponding distances and control signal of Figure 30.

Source: Author.

5.2 Practical Experiments

The controllers were embedded using MBED C++ compiler with few supporting libraries. The CAN Messages were easily defined using LPC1768 functions for CAN Networks. The ACC algorithm was programmed in such way that the user just needed to select the model for the control, select MPC or IHMPC as active, and its tuning parameters. With the software active, it automatically updates the dimensions and values of the matrices H , c_f , A_c and b_c regarding the selected control law.

The duration of the solution of each QP was also inspected. The highest interval

Figure 32: Simulation 2 of ACC with IHMPC for a system with changing models.

Source: Author.

Figure 33: Corresponding distance to leading vehicle of Figure 32.

Source: Author.

of the MPC's QP was 0.0439 seconds, which is more than 10 times quicker than the time sampling ($T_s = 0.5$ seconds). When the MPC arrives close to the cruise speed, the QP solution is found in less than 0.3 milliseconds. The IHMPC's QP is determined between 0.188 and 0.164 seconds. The IHMPC takes longer to solve the QP, since the equality constraint is stricter. For both solvers, the maximum loop iterations was set to 100 obtaining suboptimal control signals to avoid possible unfeasible solutions.

Two experiments were proposed to analyze the controllers performance in several environments, using the same tuning parameters of Subsections 5.1.1 and 5.1.2. The Experiment 1 had three main objectives: check the performance of the control system in

CC Mode, its dynamics while changing into ACC Mode and its sensitivity to disturbances. The Experiment 2 had the objective to check the performance of the ACC System with all possible scenarios regarding the leading vehicle. This experiment required a leading vehicle with lower, equal and higher speed in relation to the v_{user} .

For the following experiments, the Model 1A was selected to predict the system. Any of the nine models could be chosen for the prediction, although the performance would differ while using the same tuning parameters.

The procedure of Experiment 1 is shown in Figure 34. During the start of the Experiment 1, the user defined speed is 15 m/s and it is assumed that at 10 m of distance there is a leading vehicle at constant 15 m/s. At 30 s of the experiment, there is an Event A. This event is described as the leading vehicle slowly breaking until reaching the speed of 10 m/s. At 70 s, there is an Event B, in which the load torque of the dynamometer slowly increases from 0% to 15%, representing the inclusion of disturbances.

Figure 34: Schematic of the Experiment 1

Source: Author.

The procedure of Experiment 2 is shown in Figure 35. During the start of the Experiment 2, the user defined speed is 15 m/s and it is assumed that at 10 m of distance there is an leading vehicle at constant 10 m/s. At 40 s of the experiment, there is an Event A. This event is described as the leading vehicle slowly accelerating until reaching the speed of 15 m/s. At 80 s, there is an Event B, in which the leading vehicle slowly accelerates until reaching the speed of 20 m/s.

The practical results of the Experiment 1 using the MPC are shown in Figure 36 and Figure 37. The first figure has the relevant speeds of the experiment and the latter figure has the estimated distance to the leading vehicle, the calculated distance d_{ref} and the control signal (accelerator pedal). In less than 5 seconds, the vehicle speed reaches the cruise speed (set-point). The controller keeps on tracking the cruise speed, as it converges

Figure 35: Schematic of the Experiment 2

Source: Author.

into the v_{user} . The output presented a small amplitude of oscillation due to number of quantization bits of the control signal (only 8 bits).

After 30s, the Event A started. Correspondingly, the outer loop controller changed into ACC Mode and the MPC tracked the changing cruise speed. At 70s of experiment (Event B), the load torque of the dynamometer started to increase and the control signal changed in order to maintain the vehicle speed close to the cruise speed. Even with the disturbances, the vehicle speed has barely changed. Most importantly, the ACC system maintained a considerable distance to the leading vehicle.

Source: Author.

Next, the practical results of the Experiment 1 using the IHMPC are shown in Figure 38, the relevant speeds of the experiment, and Figure 39, the estimated distance d to the leading vehicle, d_{ref} and the control signal. The controller quickly accelerates, presenting

Source: Author.

a small overshoot, and later it tracks the cruise speed smoother than the MPC.

After the Event A, the outer loop controller changed into ACC mode and the IHMPC also tracked the changing cruise speed. After the Event B, the control signal was increased in order to maintain the vehicle speed close to the reference. The distance to the leading vehicle remained close to safe values during all the experiment.

Source: Author.

Regarding the Experiment 2, the practical results using the MPC are shown in Figure 40 and Figure 41. The MPC tracked the cruise speed during all the experiment.

Source: Author.

After Event B, the leading vehicle accelerates even more to 20 m/s, however, the cruise speed remained at 15 m/s, as well as the vehicle speed. This occurred for two reasons: the maximum speed for the outer loop controller is $v_{user} = 15$ m/s and the ACC System is not a vehicle following system, but a speed tracking system while maintaining a safe distance. Therefore, the controlled vehicle did not follow the leading vehicle after Event B, increasing the distance between the two vehicles.

Source: Author.

Using the IHMPC as inner controller, the practical results of the Experiment 2 are shown in Figure 42, the relevant speeds of the experiment, and Figure 43, the estimated

Figure 41: Distances and Control Signal for Experiment 2 with the MPC

Source: Author.

distance d to the leading vehicle, d_{ref} and the control signal. Likewise, the IHMPC managed to track the cruise speed.

Source: Author.

Source: Author.

6 CONCLUSIONS AND FUTURE WORKS

The ARX models granted satisfactory FIT indexes, even with prediction of infinity steps ahead. Therefore, this indicates that the models appear to be validated. With the aid of customized CAN messages, this work successfully designed embedded predictive controllers with direct communication to the ECU of the vehicle. Both MPC formulations were implemented as quadratic programming problems, even with distinct prediction horizons and different model representations.

The practical experiments 1 and 2 provided a wide analysis of the ACC system. The adapted algorithm of the outer loop controller between CC and ACC modes performed adequately. Both predictive controllers also managed to track the constantly changing cruise speed with a satisfactory performance. Most importantly, each controller tracked the cruise speed maintaining a safe distance to the leading vehicle, achieving the ACC system main objective. Even with the outer loop controller having only one tuning parameter, its performance was satisfactory.

For future works, the on-road application requires that the ARS 300 radar must be integrated with the ACC Module using the appropriate messages for wake-up. Furthermore, the ACC module allows further control theories to be implemented in the ACC System.

REFERENCES

- AKAIKE, H. *Selected Papers of Hirotugu Akaike*. New York, NY: Springer, 1998. 199-213 p. ISBN 978-1-4612-1694-0. Available from Internet: <https://doi.org/10.1007/978-1-4612-1694-0_15>.
- AMES, A. D.; GRIZZLE, J. W.; TABUADA, P. Control barrier function based quadratic programs with application to adaptive cruise control. In: *Conference...* [S.l.]: IEEE, 2014. p. 6271–6278. ISSN 0191-2216.
- BENALIE, N. et al. Improvement of adaptive cruise control system based on speed characteristics and time headway. In: *Conference...* [S.l.]: IEEE, 2009. p. 2403–2408. ISSN 2153-0858.
- CAMACHO, E.; ALBA, C. *Model Predictive Control*. Springer London, 2013. (Advanced Textbooks in Control and Signal Processing). ISBN 9780857293985. Available from Internet: <<https://books.google.com.br/books?id=tXZDAAAQBAJ>>.
- CONTINENTAL. *Main page of radar ARS 300*. 2017. <<https://www.continental-automotive.com/Landing-Pages/Industrial-Sensors/Products/ARS30X>>. Accessed: 2017-10-10.
- CORONA, D.; SCHUTTER, B. D. Adaptive cruise control for a smart car: A comparison benchmark for mpc-pwa control methods. *IEEE Transactions on Control Systems Technology*, v. 16, n. 2, p. 365–372, March 2008. ISSN 1063-6536.
- DELPHI. *Delphi Adaptive Cruise Control*. 2016. Accessed: 2016-10-25. Available from Internet: <<http://www.delphi.com/docs/default-source/old-delphi-files/1a5875e3-7d50-461b-ad1d-9464017913bf-pdf.pdf>>.
- DIAS, J. E. A.; PEREIRA, G. A. S.; PALHARES, R. M. Longitudinal model identification and velocity control of an autonomous car. *IEEE Transactions on Intelligent Transportation Systems*, v. 16, n. 2, p. 776–786, April 2015. ISSN 1524-9050.
- GANJI, B. et al. Adaptive cruise control of a hev using sliding mode control. *Expert Systems with Applications*, v. 41, n. 2, p. 607 – 615, 2014. ISSN 0957-4174. Available from Internet: <<http://www.sciencedirect.com/science/article/pii/S0957417413005861>>.
- HILDRETH, C. A quadratic programming procedure. *Naval Research Logistics Quarterly*, Wiley Subscription Services, Inc., A Wiley Company, v. 4, n. 1, p. 79–85, 1957. ISSN 1931-9193. Available from Internet: <<http://dx.doi.org/10.1002/nav.3800040113>>.
- KALMAN, R. E. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, v. 82, n. Series D, p. 35–45, 1960.

KOTHARE, M. V.; BALAKRISHNAN, V.; MORARI, M. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, v. 32, n. 10, p. 1361 – 1379, 1996. ISSN 0005-1098. Available from Internet: <<http://www.sciencedirect.com/science/article/pii/0005109896000635>>.

LJUNG, L. *System Identification (2nd Ed.): Theory for the User*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999. ISBN 0-13-656695-2.

LUO, L.; CHEN, J.; ZHANG, F. Integrated adaptive cruise control design considering the optimization of switching between throttle and brake. In: *Conference...* [S.l.]: IEEE, 2016. p. 1162–1167.

MACIEJOWSKI, J. *Predictive Control: With Constraints*. Prentice Hall, 2002. (Pearson Education). ISBN 9780201398236. Available from Internet: <https://books.google.com.br/books?id=HV__Y58c7KiwC>.

MARKVOLLRATH; SCHLEICHER, S.; GELAU, C. The influence of cruise control and adaptive cruise control on driving behaviour - a driving simulator study. *Accident Analysis & Prevention*, v. 43, n. 3, p. 1134 – 1139, May 2011. ISSN 0001-4575. Available from Internet: <<http://www.sciencedirect.com/science/article/pii/S0001457510004008>>.

MARTINS, M. A.; ODLOAK, D. A robustly stabilizing model predictive control strategy of stable and unstable processes. *Automatica*, v. 67, n. Supplement C, p. 132 – 143, 2016. ISSN 0005-1098. Available from Internet: <<http://www.sciencedirect.com/science/article/pii/S0005109816000479>>.

MASSERA F. C.; TERRA, M. H.; WOLF, D. F. Safe optimization of highway traffic with robust model predictive control-based cooperative adaptive cruise control. *IEEE Transactions on Intelligent Transportation Systems*, v. 18, n. 11, p. 3193–3203, Nov 2017. ISSN 1524-9050.

MBED. *CAN Handbook Mbed Page*. 2013. <<https://os.mbed.com/handbook/CAN>>. Accessed: 2017-05-08.

MBED. *LPC1768 Mbed Page*. 2017. <<https://os.mbed.com/platforms/mbed-LPC1768/>>. Accessed: 2017-05-08.

MICROCHIP. *MCP2561 Overview Page*. 2014. <<http://www.microchip.com/wwwproducts/en/MCP2561>>. Accessed: 2017-05-10.

NAUS, G. et al. Design and implementation of parameterized adaptive cruise control: An explicit model predictive control approach. *Control Engineering Practice*, v. 18, n. 8, p. 882 – 892, 2010. ISSN 0967-0661. Available from Internet: <<http://www.sciencedirect.com/science/article/pii/S0967066110000882>>.

ODLOAK, D. Extended robust model predictive control. *AIChE Journal*, American Institute of Chemical Engineers, v. 50, n. 8, p. 1824–1836, 2004. ISSN 1547-5905. Available from Internet: <<http://dx.doi.org/10.1002/aic.10175>>.

OLIVERI, A. et al. Low-complexity approximations of pwa functions: A case study on adaptive cruise control. In: *Conference...* [S.l.]: IEEE, 2011. p. 669–672.

- PANANURAK, W.; THANOK, S.; PARNICHKUN, M. Adaptive cruise control for an intelligent vehicle. In: *Conference...* [S.l.]: IEEE, 2009. p. 1794–1799.
- PEREIRA, B. C. F. *Evolução de uma unidade de gerenciamento eletrônico de um motor VW 2.0 L e desenvolvimento de controle de cruzeiro: Projeto Otto IV*. Dissertação (Master's Thesis) — Escola Politécnica da Universidade de São Paulo, 2017.
- PEREIRA, B. S. *Controle da mistura ar/combustível em um motor a combustão interna: sistema em malha fechada*. Dissertação (Master's Thesis) — Escola Politécnica da Universidade de São Paulo, 2015.
- QIN, S.; BADGWELL, T. A. A survey of industrial model predictive control technology. *Control Engineering Practice*, v. 11, n. 7, p. 733 – 764, 2003. ISSN 0967-0661. Available from Internet: <<http://www.sciencedirect.com/science/article/pii/S0967066102001867>>.
- RISSANEN, J. Modeling by shortest data description. *Automatica*, v. 14, n. 5, p. 465 – 471, 1978. ISSN 0005-1098. Available from Internet: <<http://www.sciencedirect.com/science/article/pii/0005109878900055>>.
- SHAKOURI, P.; CZECHOT, J.; ORDYS, A. Longitudinal vehicle dynamics using simulink/matlab. In: UKACC, 2010. *Conference...* [S.l.]: IET, 2010. p. 1–6.
- SHAKOURI, P.; CZECHOT, J.; ORDYS, A. Adaptive cruise control system using balance-based adaptive control technique. In: *Conference...* [S.l.]: IEEE, 2012. p. 510–515.
- SHAKOURI, P.; CZECHOT, J.; ORDYS, A. Simulation validation of three nonlinear model-based controllers in the adaptive cruise control system. *Journal of Intelligent and Robotic Systems*, v. 80, n. 2, p. 207–229, Nov 2015. ISSN 1573-0409. Available from Internet: <<https://doi.org/10.1007/s10846-014-0128-4>>.
- SUN, Y. et al. Multi-performance predictive cruise control of connected vehicles on urban roads. In: *Conference...* [S.l.]: IEEE, 2016. p. 9270–9275.
- WANG, L. *Model Predictive Control System Design and Implementation Using MATLAB*. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 1848823304, 9781848823303.
- WEISSMANN, A.; GÖRGES, D.; LIN, X. Energy-optimal adaptive cruise control combining model predictive control and dynamic programming. *Control Engineering Practice*, v. 72, p. 125 – 137, 2018. ISSN 0967-0661. Available from Internet: <<http://www.sciencedirect.com/science/article/pii/S0967066117302691>>.
- ZHOU, J.; PENG, H. Range policy of adaptive cruise control vehicles for improved flow stability and string stability. *IEEE Transactions on Intelligent Transportation Systems*, v. 6, n. 2, p. 229–237, June 2005. ISSN 1524-9050.

APPENDIX A – HILDRETH’S QUADRATIC PROGRAMMING

Foremost, a quadratic programming problem can be described as a minimization of J :

$$J = \frac{1}{2}x^T Hx + c_f^T x$$

restricted by

$$A_c x \leq b_c.$$

For the minimization of J with restrictions, this problem is reformulated with Lagrange multipliers λ (HILDRETH, 1957):

$$\min_x \frac{1}{2}x^T Hx + c_f^T x + \lambda^T (A_c x - b_c)$$

The first step for the minimization process is to take the first partial derivatives of x and λ and equate these terms to zero, obtaining:

$$\begin{cases} \frac{\partial J}{\partial x} = Hx + c_f + A_c^T \lambda = 0 \\ \frac{\partial J}{\partial \lambda} = A_c x - b_c = 0 \end{cases}$$

The solution for this problem can be easily calculated by:

$$\begin{cases} \lambda = -(A_c H^{-1} A_c^T)^{-1} (b_c + A_c H^{-1} c_f) \\ x = -H^{-1} (c_f + A_c^T \lambda) \end{cases}$$

The solution for x is quite interesting if the terms are expanded:

$$x = -H^{-1}(c_f + A_c^T \lambda) = H^{-1}c_f + H^{-1}A_c^T \lambda$$

The first element represents the global optimal solution if there is no active constraints, and the second element is to adjust the solution if there is active constraints with Lagrange multiplier λ . Consequently, the Hildreth's Algorithm initially tries to solve the problem without active constraints (WANG, 2009). If this solution attends the constraints, this will be the final solution. If the constraints are not satisfied, then the algorithm recalculates λ and corrects the solution interactively. For this reason, Hildreth's Quadratic Programming can be quite fast and suitable for implementation.

APPENDIX B – CREATING A STATE SPACE WITH DELAY

Given a matrix with discrete transfer functions from system identification, there are three steps required for obtaining a model suitable for conventional MPC. The first step is to compute a state space realization for $G(z)$, without considering time delay. For incorporating time delays, it is possible to create new states representing delayed inputs.

For example, considering a time delay of $\theta = 3$, a state space representation of a generic ARX Model can be described as:

$$\begin{cases} z(k+1) = Az(k) + Bu(k-3); \\ y(k) = Cx(k) + Du(k-3). \end{cases}$$

Using this system, the second step is to incorporate a θ time delay using command *absorbDelay* from Control System Toolbox of Matlab. For example, the resulting system, with nu inputs, would be as follows:

$$\begin{cases} \begin{bmatrix} z(k+1) \\ u(k-2) \\ u(k-1) \\ u(k) \end{bmatrix} = \begin{bmatrix} A & B & 0 & 0 \\ 0 & 0 & I_{nu} & 0 \\ 0 & 0 & 0 & I_{nu} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z(k) \\ u(k-3) \\ u(k-2) \\ u(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ I_{nu} \end{bmatrix} u(k); \\ y(k) = \begin{bmatrix} C & D & 0 & 0 \end{bmatrix} \begin{bmatrix} z(k) \\ u(k-3) \\ u(k-2) \\ u(k-1) \end{bmatrix}. \end{cases}$$

This new system will have θ more states than the previous state space system. Each new state represents past system inputs and for each sampling these states are shifted to the next time delayed state, for example: $x(k) = [z(k) \ u(k-3) \ u(k-2) \ u(k-1)]^T$.

The final required step is to modify the input u into an incremental input, using the

identity $u(k) = \Delta u(k) + u(k-1)$. Using the same previous example, the model in the incremental form will be described as follows:

$$\left\{ \begin{array}{l} \begin{bmatrix} z(k+1) \\ u(k-2) \\ u(k-1) \\ u(k) \end{bmatrix} = \underbrace{\begin{bmatrix} A & B & 0 & 0 \\ 0 & 0 & I_{nu} & 0 \\ 0 & 0 & 0 & I_{nu} \\ 0 & 0 & 0 & I_{nu} \end{bmatrix}}_{A_{\text{MPC}}} \begin{bmatrix} z(k) \\ u(k-3) \\ u(k-2) \\ u(k-1) \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ I_{nu} \end{bmatrix}}_{B_{\text{MPC}}} \Delta u(k); \\ y(k) = \underbrace{\begin{bmatrix} C & D & 0 & 0 \end{bmatrix}}_{C_{\text{MPC}}} \begin{bmatrix} z(k) \\ u(k-3) \\ u(k-2) \\ u(k-1) \end{bmatrix}. \end{array} \right.$$

This new $(A_{\text{MPC}}, B_{\text{MPC}}, C_{\text{MPC}})$ model is appropriate for the MPC since it has system time delays and it is in incremental input form.

APPENDIX C – CREATING AN OPOM WITH TIME DELAY

The method for creating Output Prediction Oriented Model is similar to (ODLOAK, 2004), but with subtle differences for including time delays in the model. Given any matrix of discrete transfer functions $G(z)$ with its elements:

$$G_{i,j}(z) = \frac{b_{i,j,1}z^{na-1} + b_{i,j,2}z^{na-2} + \dots + b_{i,j,nb}z^{na-nb}}{z^{na} + a_{i,j,1}z^{na-1} + a_{i,j,2}z^{na-2} + \dots + a_{i,j,na}}; i = 1, \dots, ny; j = 1, \dots, nu$$

It will be discussed later that it is recommended for all $G_{i,j}(z)$ being strictly proper transfer functions. Also, na and nb are the highest orders of denominator and numerator for all $G_{i,j}(z)$. In this OPOM formulation, it will be considered that there are no integrating poles and there are no repeating poles.

The discrete step response of $G_{i,j}$ can be written as:

$$S_{i,j}(k) = d_{i,j}^0 + \sum_{l=1}^{na} [d_{i,j,l}^d \cdot r_l^k]$$

The poles of $G_{i,j}$ are r_l and $d_{i,j}^0, d_{i,j,1}^d, d_{i,j,2}^d, \dots, d_{i,j,na}^d$ coefficients can be obtained by partial fractions expansion of the step response of $G_{i,j}$.

The OPOM is a not minimal realization that has the same properties of an analytical step response of $G(z)$. Each $G_{i,j}$ is affected by discrete step transfer function, $\frac{z}{z-1}$, and then the *residue* command from Matlab is used in order to collect poles and their respective residues. Because all $G_{i,j}$ are strictly proper transfer functions, there will be no direct terms in the partial fractions expansion.

Analyzing the residues obtained, the first residue is related to a pole located at 1. Since there is no integrating pole in the original system, this pole occurs because of the additional pole of step transfer function. This residue is determined as $d_{i,j}^0$ and its

function is system stationary gain. Each remaining residue corresponds to the coefficients $d_{i,j}^d$ regarding each system poles.

A state vector is defined with dimension nx related to the number of inputs, outputs and poles for the system. The dimension of x^s is the same as the dimension of output, since x^s represents the stationary gain to each output.

$$x = \begin{bmatrix} x^s \\ x^d \end{bmatrix}, ns = ny, nd = ny \times nu \times na$$

The first OPOM with θ time delay will be determined as follows:

$$\begin{bmatrix} x_{i,j}^s(k+1) \\ x_{i,j}^d(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & F_{i,j} \end{bmatrix} \begin{bmatrix} x_{i,j}^s(k) \\ x_{i,j}^d(k) \end{bmatrix} + \begin{bmatrix} B_{i,j}^s \\ B_{i,j}^d \end{bmatrix} \Delta u_j(k-\theta) \quad (C.1)$$

$$y_i(k) = \begin{bmatrix} 1 & \Psi_{i,j} \end{bmatrix} \begin{bmatrix} x_{i,j}^s(k) \\ x_{i,j}^d(k) \end{bmatrix} \quad (C.2)$$

Where

$$F_{i,j} = \text{diag}(r_{i,j,1} \dots r_{i,j,na}), \Psi_{i,j} = \begin{bmatrix} \underbrace{1 \dots 1}_{na} \end{bmatrix}$$

$$\begin{bmatrix} B_{i,j}^s \\ B_{i,j}^d \end{bmatrix} = \begin{bmatrix} D_{i,j}^0 \\ D_{i,j}^d F_{i,j} N \end{bmatrix}, D_{i,j}^0 = d_{i,j}^0, D_{i,j}^d = \text{diag}(d_{i,j,1}^d \dots d_{i,j,na}^d), N = \left. \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right\}_{na}$$

The system representation C.1 is not practical for control purposes since each $G_{i,j}$ has its own time delay and each control input have different instant of application. Therefore, it will be defined a new state vector (Equation (C.3)), where θ_{max} is the highest time delay among all inputs and outputs.

$$x(k) = \begin{bmatrix} x^s(k) \\ x^d(k) \\ \Delta u(k-1) \\ \Delta u(k-2) \\ \vdots \\ \Delta u(k-\theta_{max}) \end{bmatrix} \quad (C.3)$$

The new OPOM representation with time delay for all input and output signals is described in equations (C.4) and (C.5).

$$\begin{bmatrix} x^s(k+1) \\ x^d(k+1) \\ \Delta u(k) \\ \Delta u(k-1) \\ \vdots \\ \Delta u(k-\theta_{max}+1) \end{bmatrix} = A_{OPOM} \begin{bmatrix} x^s(k) \\ x^d(k) \\ \Delta u(k-1) \\ \Delta u(k-2) \\ \vdots \\ \Delta u(k-\theta_{max}) \end{bmatrix} + B_{OPOM} \Delta u(k) \quad (C.4)$$

$$y(k) = C_{OPOM} \begin{bmatrix} x^s(k) \\ x^d(k) \\ \Delta u(k-1) \\ \Delta u(k-2) \\ \vdots \\ \Delta u(k-\theta_{max}) \end{bmatrix} \quad (C.5)$$

Where

$$A_{OPOM} = \begin{bmatrix} I_{ny} & 0 & B_1^s & B_2^s & \dots & B_{\theta_{max}-1}^s & B_{\theta_{max}}^s \\ 0 & F & B_1^d & B_2^d & \dots & B_{\theta_{max}-1}^d & B_{\theta_{max}}^d \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & I_{nu} & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & I_{nu} & 0 \end{bmatrix}; B_{OPOM} = \begin{bmatrix} B_0^s \\ B_0^d \\ I_{nu} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$C_{OPOM} = \begin{bmatrix} I_{ny} & \Psi & 0 & 0 & \dots & 0 \end{bmatrix}$$

$$F = \text{diag}(r_{1,1,1} \dots r_{1,1,na} \dots r_{1,nu,1} \dots r_{1,nu,na} \dots r_{ny,nu,1} \dots r_{ny,nu,na}), \mathbb{C}^{nd \times nd}$$

$$\Psi = \begin{bmatrix} \Phi & & 0 \\ & \ddots & \\ 0 & & \Phi \end{bmatrix}, \Psi \in \mathbb{R}^{ny \times nd}, \Phi = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}, \Phi \in \mathbb{R}^{nu \times na}$$

The definition of every B^s and B^d is a bit more problematic. Matrices B_l^s , for $l = 0, \dots, \theta_{max}$ are calculated as follows:

- If $l \neq \theta_{i,j}$, there is no delay θ in element l for i, j , then $[B_l^s]_{i,j}=0$;
- Otherwise, $[B_l^s]_{i,j} = d_{i,j}^0$.

If there is no time delay, $\theta_{max} = 0$ and $l = 0$, B_l^d is calculated as $B_0^d = D^d F N$, where:

$$D^d = \text{diag}(d_{1,1,1}^d \dots d_{1,1,na}^d \dots d_{1,nu,1}^d \dots d_{1,nu,na}^d \dots d_{ny,nu,1}^d \dots d_{ny,nu,na}^d), \mathbb{C}^{nd \times nd}$$

$$N = \left\{ \begin{bmatrix} J \\ \vdots \\ J \end{bmatrix} \right\}_{ny}, N \in \mathbb{R}^{nd \times nu}, J = \begin{bmatrix} 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}, J \in \mathbb{R}^{nu \times na}$$

However if in $G_{i,j}$ there is time delay l , nonzero elements of matrix $D^d F N$ must be reallocated to their respective B_l^d in the same position they were in matrix $D^d F N$. After reallocating every element of $D^d F N$, all other elements of B_l^d are completed with zeros.

APPENDIX D – TRANSFORMING THE OPOM WITH REAL STATES

For example, an OPOM with $\theta_{max} = 1$ and with a pair of complex states x_1^d and x_2^d is represented as follows:

$$\begin{bmatrix} x^s(k+1) \\ x_1^d(k+1) \\ x_2^d(k+1) \\ \Delta u(k) \end{bmatrix} = \begin{bmatrix} I_{ny} & 0 & 0 & B^s \\ 0 & f+gi & 0 & B_1^d \\ 0 & 0 & f-gi & B_2^d \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^s(k) \\ x_1^d(k) \\ x_2^d(k) \\ \Delta u(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ I_{nu} \end{bmatrix} \Delta u(k)$$

$$y(k) = \begin{bmatrix} I_{ny} & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x^s(k) \\ x_1^d(k) \\ x_2^d(k) \\ \Delta u(k-1) \end{bmatrix}$$

The elements $x_1^d(k)$, $x_2^d(k)$, $f+gi$, $f-gi$, B_1^d and B_2^d are complex and $x^s(k)$, B^s , $\Delta u(k-1)$ and $y(k)$ are real. The pair of complex poles, $f+gi$ and $f-gi$, are already in discrete time and $B_1^d = B_2^{d*}$.

It can be shown that the free response of this OPOM is the same free response of the following pair (A,C):

$$A = \begin{bmatrix} I_{ny} & 0 & 0 & B^s \\ 0 & f & g & B_{real1}^d \\ 0 & -g & f & B_{real2}^d \\ 0 & 0 & 0 & 0 \end{bmatrix}, C = \begin{bmatrix} I_{ny} & 1 & 0 & 0 \end{bmatrix}$$

The next step is to determine B_{real1}^d and B_{real2}^d for maintaining input response. The

required matrix D_d in order to do that it is defined as follows:

$$D_d = \begin{bmatrix} \text{real}(B_1^d) - \text{imag}(B_1^d) & -\text{real}(B_1^d) - \text{imag}(B_1^d) \\ -\text{real}(B_1^d) - \text{imag}(B_1^d) & \text{real}(B_1^d) - \text{imag}(B_1^d) \end{bmatrix}$$

Finally, B_{real1}^d and B_{real2}^d are calculated as follows:

$$\begin{bmatrix} B_{real1}^d \\ B_{real2}^d \end{bmatrix} = D_d \begin{bmatrix} f & g \\ -g & f \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

APPENDIX E – SWITCHING MODELS LOGIC

For improving simulations analysis, it was desired to use more than one model for the system and smoothly changing between them. A candidate function for a smooth transition is the double exponential function, which was normalized for X and Y axes between 0 and 1, as seen in Figure 44.

Figure 44: Normalized double exponential function.

Source: Author.

Using this function, the X-axis represents a time factor and the Y-axis represents a weight factor. Considering a simulation with two models A and B, starting with model A and performing a smooth transition into model B. An assumption is made that one can use a system model acting as a linear combination of the two models responses.

At the start of the simulation, the time factor is 0 and it increases linearly until 1, when the simulation stops. Next, the normalized double exponential function is applied to model B and its complement of 1 is applied to model A. Therefore, at start the weigh

factor of A will be higher than B and it will decrease following the time factor until the end of simulation. When it is reached, the weight factor of B will equal to 1 and the weight factor of A will be 0.

In order to change between three models, it is possible to increase the time factor to 2 as follows in Figure 45. The first half is the same as the previous example. Next, just concatenate the functions with appropriate levels: Model A receives null importance, Model B receives high importance into low importance (complement of 1 of double exponential) and Model C receives low importance into high (double exponential).

Source: Author.

For this new simulation, the time factor should be 0 at the start and 2 at the ending. The system response will be a linear combination of the three models responses, each one multiplied by their weight factor corresponding to the current time factor.

This proposed switching models logic considers a linear combination of three models, considering a double exponential as weighting function. If the switching logic is not cautious, it can insert instability to the system. However, this method performed a linear combination with a smooth weight function, reducing the possibility of instability.