CHALMERS

Motion Cueing in the Chalmers Driving Simulator: A Model Predictive Control Approach

Master of Science Thesis

BRUNO DANIEL CORREIA AUGUSTO
RUI JOSÉ LEAL LOUREIRO

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2009
Report No. EX057/2009

Abstract

Driving simulators are used for several different purposes. Ranging from entertainment, to research of automotive technologies, or even study of driver behaviors, a driving simulator is a very useful tool.

The goal of a driving simulator is defined as the reproduction of vehicle dynamics such as accelerations and rotations. This is an intricate task because the motion hardware displacements are subjected to limitations. Therefore, vehicle signals cannot be sent directly to the motion platform and a motion cueing strategy is required.

Motion cueing algorithms are used to translate the vehicle motion into valid simulator displacements. Standard motion cueing methods do not possess any kind of constraint awareness. This renders these methods somewhat conservative regarding the computation of motion commands.

This thesis work purposes a new motion cueing algorithm based on model predictive control (MPC) strategy. By applying MPC, workspace limitations are respected due to its capability of dealing with multivariable, constrained optimization problems.

We focused on the development and implementation of the new motion cueing strategy, on the Chalmers Driving Simulator. The human sensorial system was used as the internal model of the controller. The goal is now to reproduce the perceived signals in the vehicle on the motion simulator, rather than accelerations and rotations.

The developed controller was tested against the conventional motion cueing approaches installed in the Chalmers Driving Simulator. Tests showed that not only the perceived signals are better replicated by our controller, but the platform workspace is exploited in an optimal fashion.

Acknowledgments

First of all, we would like to thank our families for all the support in time of need as well as for providing us with the possibility to study abroad and expand our horizons. With your guidance, things would not have turned out as good as they did.

We would like to show our gratitude to all friends that followed us throughout the past six months. Thank you for all the good times and memories. Without them, these would have been, surely, hard times.

Thank you to Paolo Falcone for proving us with the opportunity to work on this wonderful project. Your help, support and constant disponibility guided us through this thesis work. Thank you for your encouragement that kept us going on until the end.

A special thanks for all the students and researchers that helped building and developing the Chalmers Vehicle Simulator. Without you we would not have been able reach where we stand.

We are very grateful for all the technical support provided by friends who did not have to. Specially, João Serra, for all the nights and days you did not give up on us despite all complications and, Sina Khosh Fetrat, for all the granted aid with a wide variety of topics. Also, we would like to show our appreciation to Josef Nilsson for all developed work in the CVS. With him, our task would surely be more complex.

Thank you to all persons who were not mentioned in the above acknowledgments, but were involved with our work, directly and indirectly.

Bruno: I would like to extend a very special thanks to both Erasmus coordinators, Carlos Cardeira and PANNE whose help was precious when dealing with all exchange issues. Also, i would like to show my gratitude towards Miguel Silva. Thank you very much for offering to be my coordinator in Portugal without hesitating.

Contents

List of Figures

List of Tables

List of Symbols

I	Identity matix	103
n_{in}	Number of inputs	104
$\lambda(t)$	Vector containing output predictions over H_p	104
ψ, θ	Matrices used to define the prediction model	104
ref	Vector containing references for the tracking variables	104
υ	Matrix used in the formulation of the optimization problem	104
p_c	Number of constrained inputs	104
p	Number of outputs	104
E	Free response of the system	105
$\otimes$	Kronecker product	105
U_t	Vector containing the input in the previous instant	105
κ	Matrix applied for computation of $U(t)$ from $\delta U(t)$	105

Subscripts

$()_S$	Simulator reference frame	14
$()_x$	x direction	16
$()_y$	y direction	16
$()_z$	z direction	16
$()_*$	Stands for rear or front	17
$()_A$	Vehicle reference frame	21
$()_D$	Simulator driver reference frame	21
$()_E$	Vehicle driver reference frame	21
$()_I$	Inertial reference frame	20
$()_\Lambda$	A general frame	24
$()_{scc}$	Semicicrcular canals sensation model	29
$()_{scc_x}$	Semicicrcular canals sensation model in the x direction	29
$()_{scc_y}$	Semicicrcular canals sensation model in the y direction	29
$()_{scc_z}$	Semicicrcular canals sensation model in the z direction	29
$()_{oto}$	Otolith model	31
$()_{oto_x}$	Otolith model for the x direction	32
$()_{oto_y}$	Otolith model for the y direction	32
$()_{oto_z}$	Otolith model for the z direction	32
$()_{vest}$	Vestibular model	34
$()_d$	Discretized model	41
$()_{pred}$	prediction model	101
$()_{aug}$	Augmented system	103
$()_{tr}$	Variables to be tracked	104
$()_c$	Variables to be hard constrained	104

Introduction

Driving simulators are often used for vehicle systems development, Human Machine Interface (HMI) studies and, as a tool for driving training in a safe controlled environment. They are virtual tools that give to the user the sensation of driving a real vehicle by associating generation of inertial cues, with visual and audio cues [25].

The information originating from the vehicle model that is simulated can not be directly applied to the platform in the form of motion commands. Since the motion platform has limited workspace, its boundaries would be quickly reached and the simulation of future motion cues would be prevented.

There are several well known conventional techniques applied for computation of motion cues named washout filters. The aim of these control methods is to replicate as well as possible the vehicle dynamics in the motion simulator while keeping the platform within its boundaries. This boundaries not only refer to physical constraints, as mentioned earlier, but also to input and input variation thresholds.

Several conventional motion cueing approaches are presented in [27] and [35]. Even though these techniques are well spread and present a good solution for the motion cueing problem, they have some disadvantages. In the formulation of the above methods there is no possibility of including constraint considerations. This obstacle forces the washout filters to be designed in a very conservative fashion limiting the platform workspace, hence the performance of the simulator.

This thesis work proposes the implementation of a Model Predictive Control (MPC) algorithm controlling the platform motion in a driving simulator. MPC is an effective technique for solving multivariable constrained optimal control problems. In MPC a model of the plant is used to predict the system behavior of a future finite time horizon. Based on these predictions a cost function is minimized over a sequence of future input commands. The first of such sequence is applied to the plant and, at the next time step, the optimization is repeated over a shifted horizon.

Finally, the plant model consists of the driver sensing system and the controller goal is to minimize the deviation between the specific forces and angular rates perceived in a vehicle and the ones sensed in a driving simulator. This is accomplished while fulfilling constraints on the platform workspace.

Chapter 1

Driving Simulators

Driving simulators are very useful research tool. Driving simulators provide a cheap and safe ways of testing new technologies to be implemented in vehicles. Besides research applications, driving simulators can also be used for entertainment purposes like integration on car games.

Driving simulators provide a realistic driving experience with the replication of several driving cues. These cues can be divided in three types: visual, hearing and inertial. Inertial cues are generated by a motion platform and even though, not all the driving simulators have the ability of generating these cues, without them, they are very incomplete [4]. Visual cues are generated by a graphical system based on information computed by a vehicle model. Basically, the latter consist of pictures of the road and surroundings to build the driving environment, figure 1.1. Finally, hearing cues focus on replicating the engine sounds based on current engine states, e.g., speed of the engine.

The generation of inertial cues is very important, since without them, the driver has no information about vehicle accelerations and rotations. Nevertheless, it is physically impossible to reproduce, flawlessly, those information in a driving simulator. Unlike a real vehicle, motion platforms have limited workspace that reduces the capabilities of reproducing rotations and accelerations. For the stated reason one can not simply extract signals from a vehicle model and apply them to the motion platform, therefore there is the need for a motion cueing algorithm.

Motion cueing algorithms produce the motion commands to be sent to the motion platform. The conventionally applied ones are called washout filters and can be divided in several categories. A succinct overview on this motion cueing technique and some of its variants is presented in chapter 2. Briefly, a washout filter translates the signals in a vehicle, like accelerations and rotations, to a set of displacement commands applied on the platform. This conversion is done in such a way that the platform will not hit is limitations.

In a driver simulator, the zone where the driver seats is called cockpit. This

Figure 1.1: Graphical environment

structure is mounted on top of the motion platform and its interior usually resembles with the one found in a vehicle. In the cockpit one can find all the vehicle controls like steering wheel and pedals.

Finally, one should address the vehicle model. As an essential component of a driving simulator, the vehicle model computes the accelerations and rotations generated in the body of the vehicle, given driver inputs such as steering, braking and gas pedal position. The signals extracted from this dynamical model are used by the motion cueing strategies, the sound cue system, and by the graphical environment.

This introduction served the purpose of familiarizing the reader with the different objectives and components of a typical driving simulator. To consolidate ideas, a general scheme of a driving simulator can be found in figure 1.2.

The next section will provide more details from the particular systems that form a driving simulator, and it is based on the Chalmers Driving Simulator, CVS.

1.1 Chalmers Driving Simulator

1.1.1 Vehicle model

The Chalmers vehicle simulator has installed uses a vehicle model developed by students. It is a 14 DOF model and it is responsible for the computation of all

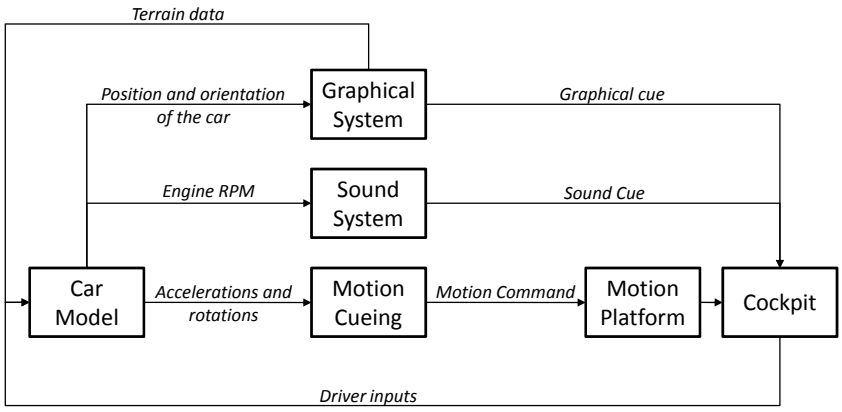

Figure 1.2: Scheme of a driving simulator

vehicle dynamics. All constants and relevant information to build this model were extracted from the Volvo X90 specifications.

The computed position and orientation of the vehicle are sent to the graphical system, which, in its turn returns all information concerning the type of road and path inclination. The vehicle model is also responsible for calculating the torque to be applied by the force feedback steering wheel [2]. The vehicle model is simulated in real time by Mathworks xPc target.

1.1.2 Cockpit

As introduced previously, the cockpit is where the driver is seated. It consists of a quarter of a Volvo car, and it is attached to the top of the motion platform. Figure 1.3 illustrates the CVS cockpit.

It has a fully functional dashboard that provides information such as vehicle velocity and RPM, as well as, all essential driving elements like steering wheel and pedals.

Figure 1.3: CVS cockpit

1.1.3 Motion platform

The CVS employs a six DOF Stewart motion platform. As presented in figure 1.4, the Stewart platform consists of two bases connected by six hydraulic actuators.

The six degrees of freedom allow the platform to perform a large set of complex and combined displacements, nevertheless, due to its size, the feasible amplitudes are very restricted. For information about the bounds and limits on displacements, refer to appendix B.

When controlling the Motion platform the user is given the possibility of choosing from two types of control inputs, either actuators displacement or degrees of freedom (Euler angles and positions of the simulator referential relatively to the inertial one). In the current work the latter were selected since the first hypothesis requires a physical model of the platform.

1.1.4 Graphical environment

The graphical environment receives information about the vehicle positions and orientations. These are used to place the vehicle in the virtual world. This tool is also responsible for generating information about the terrain that is used by the vehicle model. The generated graphical cues are projected into a screen attached to the cockpit, positioned in front of the driver (figure 1.1).

Figure 1.4: Motion Platform

1.1.5 Sound system

This application receives information about the engine states, RPM and gears. Then, a specific set of sound cues is computed given the current engine states. Two speakers located inside the dashboard play the latter.

1.1.6 Motion cueing strategies

All motion cueing algorithms are installed together with the vehicle dynamical model, inside the xPc target computer. There are eight different possibilities to choose from, including the one developed by this work. A more detailed discussion about this algorithms will be presented in the next chapter.

1.1.7 Sensorial system

This component is a part of the CVS owing to the motion algorithm developed in the current thesis. Since the goal is to work with sensed signals, the sensorial system will be included. Its function is to compute the sensed signals generated by accelerations and rotations applied to the driver. To the best of the authors knowledge, this system is not a common component of driving simulators.

The human being makes use of his senses to extract information about the

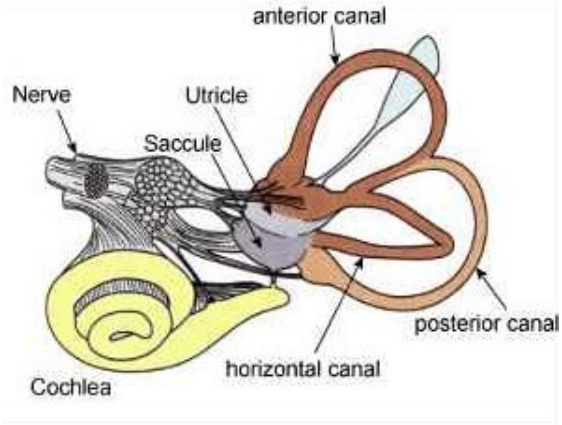

Figure 1.5: Vestibulum - Inner Ear [39]

environment that surrounds him. This allows the latter to place himself in space and interact with its surroundings. Bearing this in mind, it is possible to use sensorial information as measurement of the controller performance. If both, the sensations in a real vehicle and the sensations generated by the motion platform are available, an error between the two can be computed and the overall quality of the control approach measured. With the mentioned idea as our goal, a model of the human movement sensorial system is developed. It relates the accelerations and rotations applied to the driver with the one actually felt will be developed.

The sensory system is integrated in the human nervous system and its responsibility is to identify stimuli, either internal or external to our body. Mainly it can be divided into 5 sections: visual, vestibular, Proprioceptive, Kinesthesis and Tactile Sites, smell and hearing. This thesis will focus on the vestibular system. In the next section the latter is presented in more detail.

Vestibular system

The vestibular system main function is the control of the human sense of movement and balance. Present in every moment of one's life, this element of the sensorial system provides the human with information about accelerations and rotations applied to its body. Located in the vestibulum, inner ear figures 1.5 and 1.6, the vestibular system interacts with the human eye movements as well as, with the muscles that keep the latter upright [39] .It enables the human to maintain a posture, have notion of self and non-self motion, have spatial orientation, navigation, perform voluntary movement, oculomotor control and autonomic control [6].

The vestibular system can be sub-sectioned in two smaller systems:

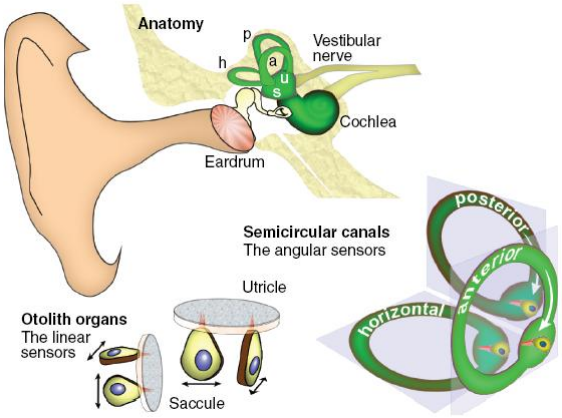

Figure 1.6: Vestibular System - Position relatively to the Ear [6]

Semicircular Channels The Semicircular channels are responsible for sensing rotations applied to the body and consist of three channels, horizontal, posterior and anterior whose interior is filled with a fluid called *endolymph*. The relative position of these channels, which are orthogonal to each other, enables sensing angular rotations about a normal to the plane in which each channel is included[23]. The three Semicircular channels have a bulbous cavity called *Ampulla* (Figure 1.7) where a gelatinous mass, named *Cupula*, (Figure 1.7) [35], supports hairs and sensory cells (Figure 1.7) [23].

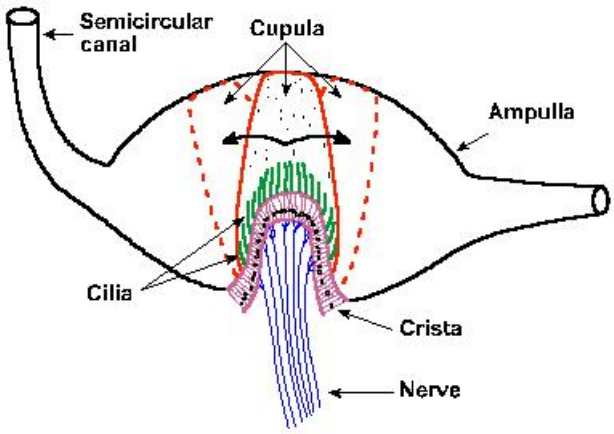

Figure 1.7: Inside the Semicircular Channel - Cupula [43]

The rotation of the head moves the *endolymph* inside the semicircular channels, causing a deflection of the cupula which excites the hairs and sensory cells. Depending on the excitation, informations regarding sensed rotations are sent to the brain.

Otoliths The otoliths detect linear motion. They have the responsibility of supplying our brain with information related to sensed specific forces by the human body. Specific force is a relation between translational acceleration and the gravity vector whose definition will be presented in the next chapter.

The human being is incapable of distinguishing translational acceleration from gravity force with the otoliths [35], [23]. Hence, another sensorial information is needed, e.g., rotational and/or visual-wise. This feature offers a precious advantage when developing motion cueing algorithms. In particular, low frequency components of the acceleration that would have required to extend the platform, can be reproduced by tilting the latter [27]. The tilting process is presented in sections 2.1 and 2.5.

There are two otolith organs in the human ear, the *Utricule* and the *Sacule* (Figure 1.5 and Figure 1.6). They sense horizontal and vertical motions respectively [35]. These two organs consist of an otolithic membrane (Figure 1.8) which is fixed on a sensory cell base, named *Macula*.

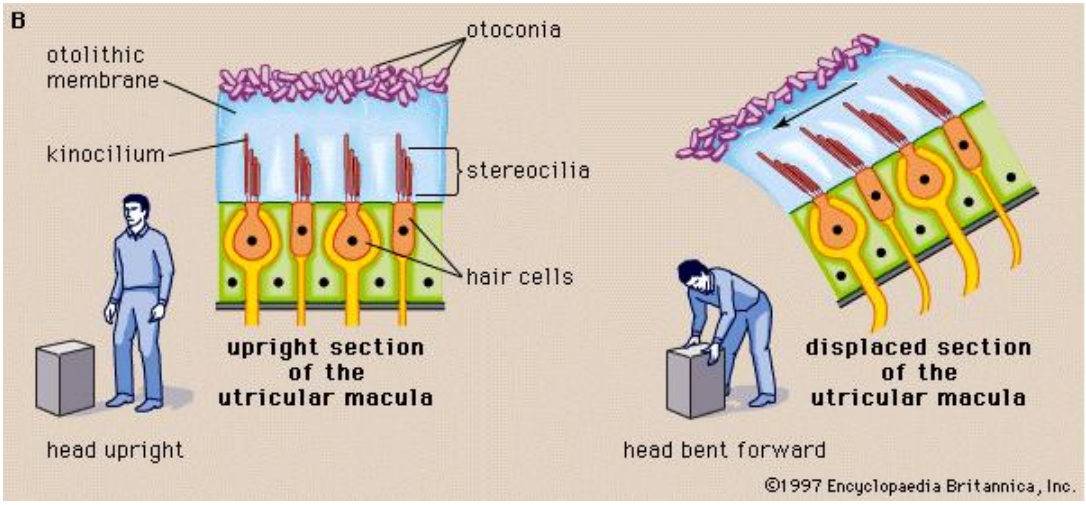

Figure 1.8: Otolithic Membrane [40]

The translational movements of the head cause deflection of the otolithic membrane. This excites the sensory cells, and sends informations about applied specific forces, to the brain.

In the next chapter a mathematical formulation of the sensorial system will be presented. It will allow the formulation of a dynamical model to be used by the proposed controller, MPC.

1.1.8 Communication

The CVS consists of several computers working together with the goal of, simulating a driving experience. The required tasks, like vehicle model simulation and motion control, graphical cues computation and sound cues creation are distributed between computers. Detailed information on this issue can be found in appendix B.

Chapter 2

Motion Cueing Algorithms

The previous chapter presented all the necessary components that form a driving simulator. The focus of this work is on motion cueing algorithms thus, from this point forward, this thesis will mostly contemplate information about the latter.

Importance of motion cueing algorithms arises from the existence of workspace and inputs limitations in the platform motion. As expected, it is not possible to transfer directly accelerations and rotations, generated by a vehicle, to the platform since the referred limits must not violated. A motion cueing algorithm plays the role of translating the movement of the vehicle to the platform actuators displacement, while satisfying all boundaries.

The conventional techniques applied in motion cueing problems are called washout filters. These have the disadvantage of not being aware of constrains, thus, most of the times, they are designed in a conservative fashion.

The Model predictive control, MPC, ability of dealing with large multivariable constrained control problems, presents this technique as an interesting approach for motion cueing algorithms. Due to the fact that this approach is constantly aware of all constraints, it is expected that the available platform displacements will be exploited in an optimal way. Implementation of this control method in the CVS is the ultimate goal of this thesis, and its development will be discussed in the current chapter.

2.1 Conventional Motion Cueing Strategies

The washout filter can be, in a very general way, split in three parts: rotational, translational and coordination [27].Figure 2.1 shows this layout.

- **Translational Channel** This part is responsible for simulating the vehicle translational movement. Its input is usually the translational acceleration of the vehicle.

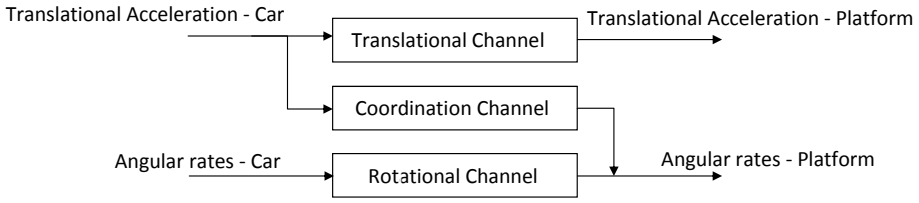

Figure 2.1: Washout filter scheme

- **Rotational Channel** This Channel role is to simulate the vehicle rotations. Its input is usually the angular rate of the vehicle, however angular displacements, [27], or accelerations can also be used.

- **Coordination Channel** The current channel uses a Tilt Coordination algorithm to simulate sustained accelerations that would eventually lead the platform beyond its physical limits. The output of this channel is summed to the one of the Rotational Channel thus establishing the final rotation cue.

Tilt Coordination establishes a relationship between the sustained accelerations, felt in the vehicle, with the Euler angles displacement of the platform. This algorithm enables the simulator to recreate the low frequency components of the accelerations. For example, tilting the platform about the Y-axis, a component of the gravity vector develops in the simulator X-axis (figure 2.2). From the driver point of view, it is the same as the vehicle acceleration, due to the inability of the Otoliths of distinguishing gravity from translational accelerations. This process, however, needs to be carried out with caution in order to avoid false cues. If the tilting rate is above the sensing threshold, the driver feels rotations that are not sensed in a real vehicle. Therefore, the output of this block should be limited in order to avoid the driver to fell platform rotations caused by tilting. In [35], these limits are presented and discussed. The used values are given in appendix B, together with all the constraints for the platform displacements.

The signals sent to each channel of the washout filter are subject to scaling and saturation. This is a nonlinear procedure whose goal is to change the input signals amplitude in an even way through out all its frequencies [36]. This process will provide an high gain to a small signal that would run the risk of not being felt after simulation, and will perform in an opposite fashion for higher signals that would try to extend the platform actuators beyond their limitations.

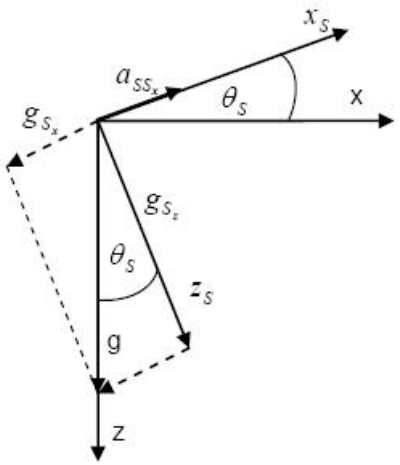

Figure 2.2: Simulated Specific force in the X-axis [27]

2.1.1 Types of Washout Filters

There exist several different types of washout filters:

The classical washout filter [27], [29], [28], [31], [4], [5], [34], [16], [44], is the most common approach and it makes use of high and low pass filters to deal with the problems of actuators over extension. One big disadvantage of Classical washout filter comes with the fact that its filters parameters are kept constant during all the simulation. This feature makes this method very inflexible owing to the fact that it demands a tuning process focused on the worst case situations [4], [28], [5], [16]. This leads to a poor usage of both workspace and platform displacements.

The adaptive washout filter [27], [36], [35], [29], [28], [4],[5], appeared to answer some shortages in the Classical format. In this approach the filters used in the classical washout filter are replaced by time varying gains thus, adapting its response to all situations. Moreover, the referred online adjusting of the parameters allows a better exploration of the platform potential.

The optimal washout filter [27], [36], [35], [29], [28], [4],[5], [33], [16], [31], takes a different approach to the motion cueing problem. The challenge is now to establish a transfer function that relates the accelerations and rotations of the vehicle with the ones that should be applied to the platform, such that the discrepancy between the felt signals in both is minimized. This technique requires the manipulation of several weights connected to states that often do not have a clear physical representation, making the tuning task very intricate [5]. Furthermore, the transfer function is only optimal for a set of different input scenarios

recreated by white noise. In other words, the final result is the one with best overall fit to several situations, which does not mean it will perform in the best possible way for all occasions.

The nonlinear washout filter [36],[35], is a combination of the features of both Optimal and Adaptive filters. It solves the optimal control problem online as a function of a gain much like the one used in the adaptive filter. This Nonlinear approach maximizes the use of motion hardware thanks to the generation of motion cues adapted to the current states of the platform.

From the above motion cueing methods, two of them can be found in the CVS. Those are the optimal and classical washout filters. The more recent versions of the latter were developed in thesis work, [27].

2.2 Reference Generation

Reference generation is an important step in any MPC problem. Knowing the future behavior of the controlled system allows the computation of optimal control actions. Bearing this in mind, it is easy to conclude that the quality of the generated reference is a factor that influences directly the overall performance of the control approach.

The general process for generating references is formulated in the next three steps:

a)- A batch of signals is extracted from the vehicle model whose dynamics are supposed to be duplicated in the simulator. These signals consist of the vehicle translational accelerations and velocities, Euler angular rates, steering ratio and longitudinal tire forces at the current time instant.

b)- Choosing one of three methods, a reference vector is created. It holds references for the Euler angular rates and translational accelerations. The size of this vector is determined by the formulation of the purposed controller. It will be discussed when this theme is introduced.

c)- The latter vector is filtered by the vestibular system to establish the final references in terms of tracking variables, felt rotations and specific forces presented in section 2.7.4.

There are three different methods of calculating the references in terms of accelerations and angular rates, second step.

The first method retrieves from the Euler angular rates and translational accelerations from the vehicle, step a). Then, this values are assumed constant through

the prediction horizon, creating the vector mentioned in step b). Following the procedure stated in step c), the desired references are created.

The second and third method use a simplified bicycle model presented in 2.2.3.

The second method extracts the necessary bicycle model states and inputs from the vehicle model, according to step a). The bicycle model is then simulated assuming constant inputs in the whole prediction horizon and the vector of references, step b), is created. The final predictions are generated following step c).

The third and last method uses the same information as method two in step a). The derivative of the bicycle model inputs is established, with a backward difference, and it is assumed constant for all the prediction horizon. The bicycle model is then ran, establishing the desired vector in step b). The references in terms of tracking variables are attained following the procedure presented by step c).

With the exception of the first method the remaining ones do not generate any kind of references, step b), for acceleration about the Z-axis, roll and pitch angular rates. This fact renders these references, somewhat, incomplete. Nevertheless, given that they are computed by a vehicle model, one might expect that the actual computed reference values, have a more realistic evolution along the prediction horizon

2.2.1 Vehicle Model

As stated in section 2.2, it is necessary to create a basic vehicle model, to predict the future behavior of the simulated vehicle. The simplified vehicle model studied and applied in our control application is extracted from [8]. This chapter will provide a short explanation of the model and how to implement it, for a more detailed explanation, the reader should consult [8].

Consider the figure 2.3:

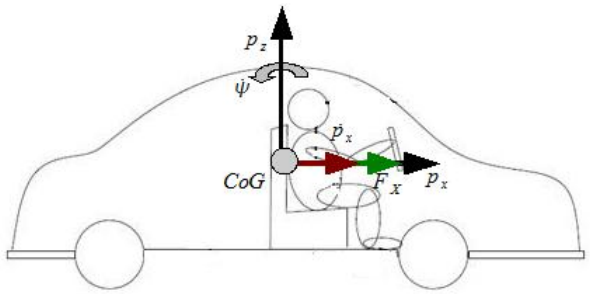

Figure 2.3: Forces acting on the center of gravity [27]

Note that the lateral axis p_y does not appear in the figure because its direction points to the reader.

By applying Newton's second law to the center of gravity (CoG) of the vehicle, the lateral, longitudinal and yaw motions are described by the following set of differential equations:

$$m\ddot{p}_y = F_y, \tag{2.1a}$$
$$m\ddot{p}_x = F_x, \tag{2.1b}$$
$$I_{car}\ddot{\psi} = M, \tag{2.1c}$$

where $\ddot{\psi}$ is the vehicle angular acceleration around the Z-axis (assumed positive counter clockwise as shown in figure 2.3), and M is the angular momentum around the Z-axis.

Furthermore, to model the longitudinal and lateral velocities with respect to the inertial fixed coordinates, X_m and Y_m, the following relationship is defined:

$$\dot{Y}_m = \dot{p}_x sin(\psi) + \dot{p}_y cos(\psi), \tag{2.2a}$$
$$\dot{X}_m = \dot{p}_x cos(\psi) - \dot{p}_y sin(\psi). \tag{2.2b}$$

2.2.2 Bicycle Model

In this thesis work we use a bicycle model to simulate the vehicle motion. This model considers longitudinal x, lateral y, and yaw ψ dynamics under the assumption of negligible lateral weight shift, roll and compliance steer. The bicycle model assumes perfect symmetry along the longitudinal axis of the vehicle. i.e., left and right sides are identical. This assumption allows to describe the front and rear dynamics of a vehicle in only one front and rear wheel. This originates the denomination of bicycle model.

The input of the bicycle model, 2.4, is the front steering wheel angle δ_f. Its states, 2.3, are the lateral and longitudinal velocities, $\dot{p}_y$ and $\dot{p}_x$, in the vehicle frame, yaw angle ψ , yaw angular rate $\dot{\psi}$ and the lateral and longitudinal positions with respect to the fixed inertial coordinates, Y_m and X_m, [8]. The bicycle model is presented in figure 2.2.3.

$$\dot{x} = \begin{bmatrix} \dot{p}_x & \dot{p}_y & \psi & \dot{\psi} & X_m & Y_m \end{bmatrix}, \tag{2.3}$$

$$u = \begin{bmatrix} \delta_f \end{bmatrix}. \tag{2.4}$$

Equation 2.1 defines the model dynamics. Moreover, the model 2.2 can be rewritten in the following form:

$$m\ddot{p}_y = -m\dot{x}\dot{\psi} + 2F_{y_f} + 2F_{y_r}, \tag{2.5a}$$
$$m\ddot{p}_x = m\dot{y}\dot{\psi} + 2F_{x_f} + 2F_{x_r}, \tag{2.5b}$$
$$I_{car}\ddot{\psi} = 2d_f F_{y_f} - 2d_r F_{y_r}, \tag{2.5c}$$

Let $* \in \{f, r\}$, denote the front and rear axles. F_{y*} and F_{x*} are defined as:

$$F_{y*} = f_y(F_{c*}, F_{l*}, \delta_*), \tag{2.6a}$$
$$F_{x*} = f_x(F_{c*}, F_{l*}, \delta_*), \tag{2.6b}$$

where,

$$F_{c*} = f_c(\alpha_*, s_*, \mu_*, F_{z_*}), \tag{2.7a}$$
$$F_{l*} = f_l(\alpha_*, s_*, \mu_*, F_{z_*}). \tag{2.7b}$$

F_{c*} and F_{l*} are the tires cornering and longitudinal forces, respectively. α_* is the tire slip angle and it represents the angle between the wheel velocity vector v_* and the direction of the wheel itself as in figure 2.2.3. The slip ratio s_*, is the relation between the theoretical and the actual longitudinal tire velocity. The theoretical velocity is calculated by the tire angular rotation rate. μ_* is the friction coefficient. F_{z_*} is the tire normal force.

The presented model defines a nonlinear relationship between the vehicle states. Since the goal is to compute predictions of a vehicle dynamics in real time a simpler model is needed. This motivates the next section, where a simplified version of the presented vehicle model is introduced.

2.2.3 Bicycle Model Based on Small Angles Approximation and Linear Tire Model

The nonlinear vehicle model can be simplified by assuming small angle approximation. In this model, the lateral and longitudinal tire forces are approximated by linear functions of the tire slip angle α_* and the slip ratio s_*, respectively .

The inputs of the simplified bicycle model 2.8 are the front steering wheel angle and tire slip ratios while its states are defined as in 2.3.

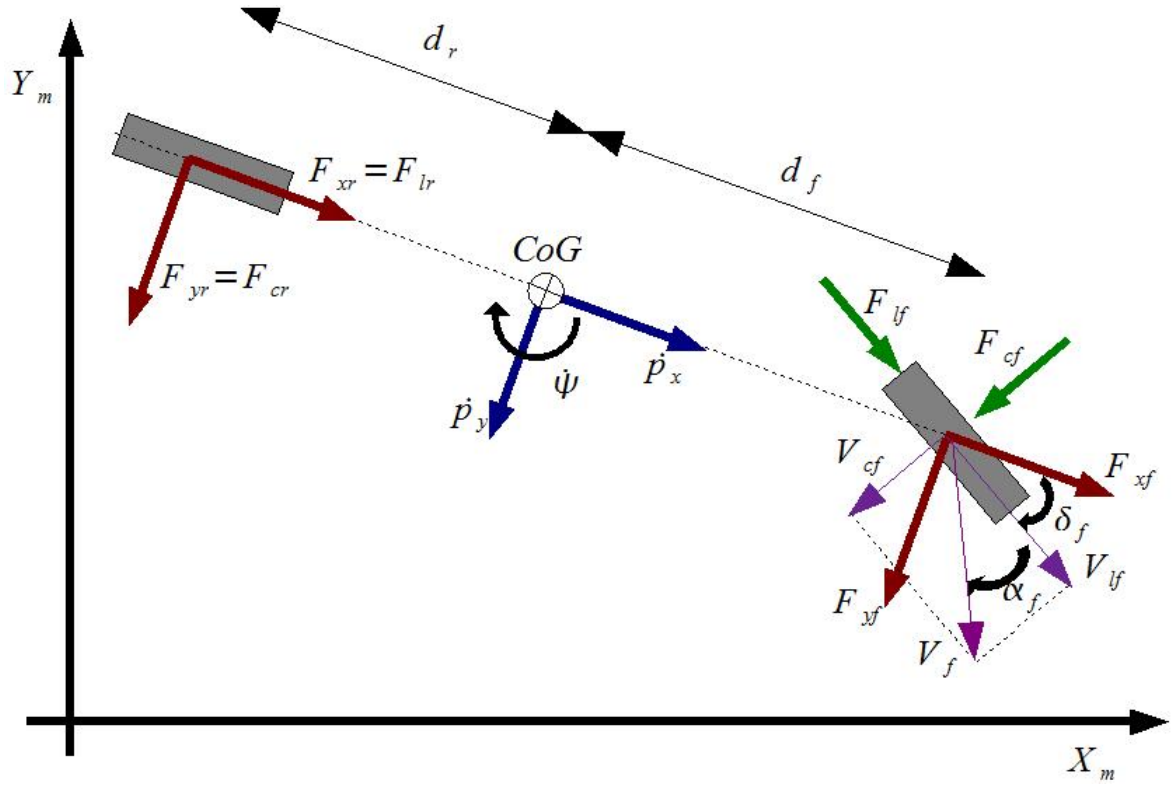

Figure 2.4: Linear Bicycle Model

$$u = \begin{bmatrix} \delta_f & s_f & s_r \end{bmatrix}. \tag{2.8}$$

For small values of the slip ratio (less than 10%) the longitudinal tire force F_{l*} is a linear function of the slip ratio s_* [8].

$$F_{l*} = C_{l*} s_*, \tag{2.9}$$

where, C_{l*} is the longitudinal stiffness coefficient.

In order to compute the lateral tire force F_{c*} as a linear function of the tire slip angle α_*, small values of the steering angle δ_f(less than 10 degrees). Thus the following linear equation can be presented [8]:

$$F_{cf} \cong C_{c,f}(\delta_f - \frac{\dot{p}_y + d_f\dot{\psi}}{\dot{p}_x}), \tag{2.10a}$$

$$F_{cr} \cong C_{c,r}(\frac{d_r\dot{\psi} - \dot{p}_y}{\dot{p}_x}). \tag{2.10b}$$

The stiffness coefficient C_{f*} is defined as the resistance of the tire to deformation by an applied force.

Furthermore, using the linearized longitudinal and lateral tire forces the following equations, that model the vehicle motion, are achieved:

$$m\ddot{y} = -m\dot{p}_x\dot{\psi} + 2[C_{c,f}(\delta_f - \frac{(\dot{p}_y + d_f\dot{\psi})}{\dot{x}} + C_{c,r}\frac{(d_r\dot{\psi} - \dot{p}_y)}{\dot{p}_x}], \tag{2.11a}$$

$$m\ddot{x} = m\dot{p}_y\dot{\psi} + 2[C_{l,f}s_f + C_{c,f}(\delta_f - \frac{(\dot{p}_y + d_f\dot{\psi})}{\dot{p}_x}\delta_f + C_{l,r}s_r], \tag{2.11b}$$

$$I\ddot{\psi} = 2[d_fC_{c,f}(\delta_f - \frac{(\dot{p}_y + d_f\dot{\psi})}{\dot{p}_x} - d_rC_{c,r}\frac{(d_r\dot{\psi} - \dot{p}_y)}{\dot{p}_x}]. \tag{2.11c}$$

By combining the equation 2.11 with equation 2.2 the motion of a vehicle subject to linear tire dynamics and small steering angle is completely defined.

In the current work there was no need for yaw, and positions about the inertial axis states except for model validation purposes.

2.3 Reference Frames

This section presents the reference frames used in the Chalmers Vehicles Simulator and shows how to translate variables between them. This is important in order to obtain the correct input to the vestibular system. Moreover, the relations between angular velocity and Euler angles, as well as inertial acceleration in rotating frames, are also introduced. The material presented in this chapter has been based on the works of Nikolce Murgovski [27] and Pieter van Balen [2].

To describe the development of motion cueing algorithms, five reference frames are introduced. Two of them are connected with a real vehicle two with the simulator and an inertial frame that relates to all. The implemented frames are right handed frames $X \times Y = Z$. Figure 2.3 shows the five reference frames.

2.3.1 Inertial Frame *I*

The inertial frame is fixed, its orientation and position do not change over time, and it is attached to the Earth. All the other frames in the simulator are defined in relation to the Inertial frame. The X_I-axis points forward, Y_I-axis points to the reader and Z_I-axis points downward. The origin of this frame is located under the simulator between the lower gimbals positions.

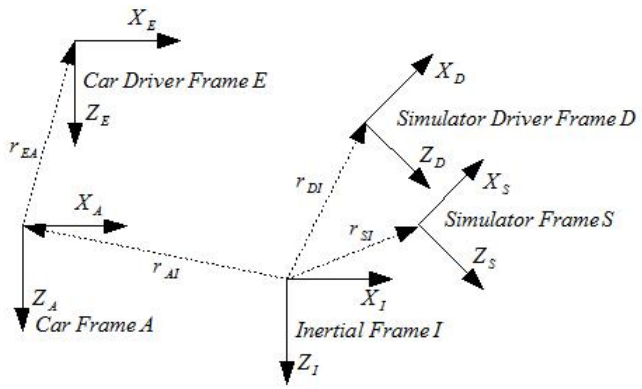

Figure 2.5: Overview of the reference frames [27]

2.3.2 Simulator Frame *S*

The simulator frame is connected to the Stewart platform. This frame follows the platform movements, thus it alters its position in relation to the inertial frame. The vector r_{SI} is defined in order to have information of the origin of the simulator frame in relation to the inertial one.

2.3.3 Driver Frame *D*

The driver frame has its origin attached to the center of the head of the driver in the simulator. The vector r_{SD} is defined in order to have information of the origin of the driver frame in relation to the simulator one. Note that this frame does not change in relation to the simulator frame because they are parallel, i.e. r_{DS} is constant. This frame is of special importance in this work since all felt signals generated by the vestibular system model are created regarding this specific location.

2.3.4 Vehicle Frame *A*

The vehicle frame is attached to the simulated vehicle. Its origin position is defined by introducing the vector r_{AI} defined as the position of the vehicle frame in relation to the inertial one. Its position and orientation varies over time with respect to the inertial frame.

2.3.5 Vehicle Driver Frame *E*

The vehicle driver frame has its origin attached to the head of the driver. Its origin is defined by introducing the vector r_{EA}. The latter is identified as the position of

the vehicle driver frame in relation to the vehicle one A. Furthermore, since it is parallel to the vehicle frame r_{EA} is constant.

2.3.6 Euler angles

The orientation of the frames is defined through their Euler angles. Frame S is taken as an example for defining orientation when the Euler angles are given. The Euler angles of S are:

$$\beta_s = \begin{bmatrix} \phi_s \\ \theta_s \\ \psi_s \end{bmatrix}. \tag{2.12}$$

To completely define frame S with respect to frame I, the orientation β_s and the position vector r_{SI} are necessary. It can be obtained through the following steps:

- Set the initial position of frame S equal to frame I,
- Rotate the frame S with angle ϕ_{SI}(roll) around the X_s-axis,
- Rotate the frame S with angle θ_{SI}(roll) around the Y_s-axis,
- Rotate the frame S with angle ψ_{SI}(roll) around the Z_s-axis,
- translate the frame S with r_{SI}.

We point out that the Euler angles are always defined with respect to the inertial frame. Then there is no need to use the index $_I$.

2.3.7 Translational acceleration

The translational acceleration vector consists of surge a_x, sway a_y and heave a_z accelerations, as in equation (2.13):

$$a = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}. \tag{2.13}$$

The acceleration in general frame Λ, can be found by multiplying the rotation matrix $L_{I\Lambda}$ by the acceleration in the frame I as in equation 2.14 .

$$a_{\Lambda I} = L_{I\Lambda} a_I. \tag{2.14}$$

By rotating the frames step by step, first around the X-axis with a roll angle, then around the Y-axis with a pitch angle, and finally about the Z-axis with a yaw angle, $L_{\Lambda I}$ can be found by multiplying the three rotations matrices as in equation 2.15. Figure 2.6 illustrates the rotation process described above.

$$L_{I\Lambda} = R_x \times R_y \times R_z. \tag{2.15}$$

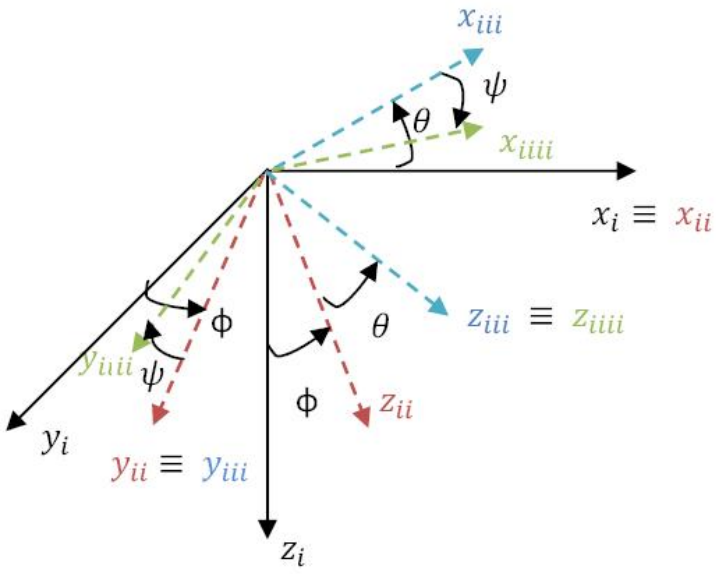

Figure 2.6: Rotation around X-axis(ϕ), Y-axis(θ) and Z-axis(ψ)

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\theta) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix}, \; R_y = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix}, \tag{2.16}$$

$$R_z = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.17}$$

With the three rotation matrices defined in 2.16 and 2.17, $L_{I\Lambda}$ can be easily calculated as:

$$L_{I\Lambda} = \begin{bmatrix} L_{11} & L_{12} & L_{13} \\ L_{21} & L_{22} & L_{23} \\ L_{31} & L_{32} & L_{33} \end{bmatrix}, \tag{2.18}$$

where

$$\begin{aligned}
L_{11} &= cos(\theta)cos(\psi), \\
L_{12} &= cos(\theta)sin(\psi), \\
L_{13} &= -sin(\theta), \\
L_{21} &= sin(\phi)sin(\theta)cos(\psi) - cos(\phi)sin(\psi), \\
L_{22} &= sin(\phi)sin(\theta)sin(\psi) + cos(\phi)cos(\psi), \\
L_{23} &= sin(\phi)cos(\theta), \\
L_{31} &= cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi), \\
L_{32} &= cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi)), \\
L_{33} &= cos(\phi)cos(\theta).
\end{aligned}$$

2.3.8 Frames rotation

The rotations can be expressed through their axes of rotation. The angular velocity vector consists of three components and its speed of rotation is defined by the magnitude of the vector ω in $[rad/s]$.

$$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \tag{2.19}$$

where p, q and r represent the components of ω about the x, y, and z respectively.

2.3.9 Relationship between angular velocity and Euler angular rates

The angular rate, ω, can be expressed by a relationship between Euler angular rates. To find the angular rate in a general Λ frame, given the Euler angular rates, equation 2.20 can be used.

$$\omega_\Lambda = R_\Lambda \dot{\beta}_\Lambda, \tag{2.20}$$

where the equation 2.20 is derived as follows,

$$\omega_\Lambda = \begin{bmatrix} \dot{\phi}_\Lambda \\ 0 \\ 0 \end{bmatrix} + R_x \begin{bmatrix} 0 \\ \dot{\theta}_\Lambda \\ 0 \end{bmatrix} + R_x R_y \begin{bmatrix} 0 \\ 0 \\ \dot{\phi}_\Lambda \end{bmatrix} = R_\Lambda \dot{\beta}_\Lambda, \tag{2.21}$$

and

$$R_\Lambda = \begin{bmatrix} 1 & 0 & -sin(\theta_\Lambda) \\ 0 & cos(\phi_\Lambda) & sin(\phi_\Lambda)cos(\theta_\Lambda) \\ 0 & -sin(\phi_\Lambda) & cos(\phi_\Lambda)cos(\theta_\Lambda) \end{bmatrix}. \tag{2.22}$$

The inverse operation, $\dot{\beta}_\Lambda = R_\Lambda^{-1} \times \omega_\Lambda$, can be performed by computing the inversion of R_Λ.

2.4 Vestibular System - Modelling

This section contains the development of the vestibular model. First the Semicircular channels model will be presented, secondly the otolith model is derived and finally the last two systems are aggregated and the complete vestibular system dynamical model is formulated.

2.4.1 Dynamic Modeling of the Semicircular Channels

Several number of models for the vestibular system are available in literature. In many existing models, the cupula deflection is modeled as a torsion pendulum. References as well as representations of the semicircular channels dynamical model can be found in [35]. In this thesis we used the modeling proposed in [31], sketched in figure 2.7, where ω and $\hat{\omega}$ are the rotation rate and felt rotation rate, respectively.

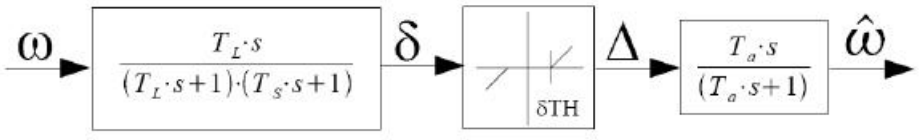

Figure 2.7: Dynamical Model of the Semicircular Channels [31]

The given representation is used for all the three dimensional rotations along the X, Y, Z axes. Each rotation component requires different model parameters

in the transfer functions as well as deadzone width. These values can be found in Table 2.1.

In figure 2.7, the first block refers to the cupula displacement model, i.e., the over damped torsional pendulum. The second block represents the human capability of sensing only accelerations higher than a certain threshold and the last block models the washout of human response to steady state rotational acceleration inputs [31].

Table 2.1: Coefficients of the Semicircular Channels Model [31]

	Roll(x)	Pitch(y)	Yaw(z)
$T_L\,[s]$	6.1	5.3	10.2
$T_s\,[s]$	0.1	0.1	0.1
$T_a\,[s]$	30	30	30
$\delta TH\,[deg/s]$	3.0	3.6	2.6

In order to establish the dynamical model the equivalent state space representation of the blocks, shown in 2.7, was developed. Let the block before the deadzone be *Block-A* and the block after *Block-B*.

Using the transfer functions presented in 2.7 the following state space models can be derived, for *Block-A* and *Block-B* respectively:

$$\begin{gathered} \dot{x} = Ax + B\omega, \\ \delta = Cx + D\omega, \\ A = \begin{bmatrix} -(T_l + T_s) & 1 \\ -1 & 0 \end{bmatrix}, B = \begin{bmatrix} T_l \\ 0 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \end{gathered} \tag{2.23}$$

and

$$\begin{gathered} \dot{x} = Ax + B\Delta, \\ \hat{\omega} = Cx + D\Delta, \\ A = \begin{bmatrix} -1 \end{bmatrix}, B = [-T_a], C = \begin{bmatrix} 1 \end{bmatrix}. \end{gathered} \tag{2.24}$$

The state feedthrough matrices are zero for both blocks.

By evaluating the output of *Block-A* in the deadzone with the characteristics specified in table 2.1, the system is completely defined and a nonlinear relationship between applied rotation rates and felt rotations is established.

2.4.2 Dynamic Modeling of the Otoliths

As for the semicircular channels, several number of models have also been proposed for the otoliths. Some of them evolved from an initial model built by measuring the subjective indication of direction [35]. This thesis uses the model in

[31], presented in figure 2.8, where f and $\hat{f}$ are the specific force, and felt specific

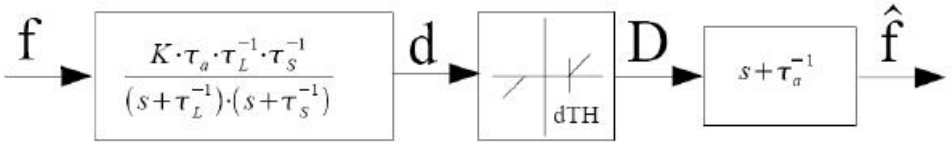

Figure 2.8: Otoliths Dynamic Model [31]

force, respectively. Let the input of the otoliths model be the specific force, and it is defined as $\vec{f} = \vec{a} - \vec{g}$..

The first block in figure 2.8 stands for the otolith actual mechanical behavior, the second is the mechanical threshold, while the last one represents a neural processing activity [31].

As for the semicircular channels case, the given transfer function represents felt specific forces for the Cartesian coordinates. Replacing the constants in the transfer function by the different sets of values, it is possible to build the three transfer functions, one for each axis X, Y, Z. The model parameters can be found in table 2.2.

Using the same procedure applied when defining the model of the semicircular channels, let the blocks before and after the deadzone ,in figure 2.8, be designated by *Block-A* and *Block-B*, respectively.

Using the canonical form of observability, the following state state model can be attained for *Block-A*:

$$\begin{aligned} \dot{x} &= Ax + Bf, \\ d &= Cx + Df, \end{aligned}$$
$$A = \begin{bmatrix} -(\tau_l^{-1} + \tau_s^{-1}) & 1 \\ -\tau_l^{-1}\tau_s^{-1} & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ k\tau_a\tau_l^{-1}\tau_s^{-1} \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{2.25}$$

Filtering d through the correspondent deadzone and then by *Block-B* leads to desired objective which is the establishment of the dynamical model for the otolith model. Nevertheless, the challenge defining this model comes with *Block-B*.

As it is possible to conclude, the transfer function of the *Block-B* is a non-causal system. This implies that the current output of the model has a dependence from the future ones. In order to deal with this problem, *Block-B* will be represented directly in its discretized form

$$\hat{f} = \frac{(D_m - D_{m_p})}{T_s} + \frac{D_m}{\tau_a}, \tag{2.26}$$

where D_{m_p} is the previous input of *Block-B*, D_m the current input and the input derivative is computed with a backward difference.

2.4.3 Dynamic Modeling of the Vestibular System

The model of the vestibular system is established by combining the two previously defined models. For constraint monitoring during the optimization problem, nine extra states are added to this model. These new states are the last nine values of the equation 2.65, presented in section 2.7.4.

The latter are added by integration of the translational accelerations as well as Euler angular rotations. In the discrete case they will be computed by backward difference.

Table 2.2: Coefficients of the Otoliths Model [31]

	a_x	a_y	a_z
$\tau_l\,[s]$	5.33	5.33	5.33
$\tau_s\,[s]$	0.66	0.66	0.66
$\tau_a\,[s]$	13.2	13.2	13.2
K	0.4	0.4	0.4
$dTH\,[m/s^2]$	0.17	0.17	0.28

2.5 Tilt Coordination

As pointed out previously, section 2.1, the sustained components of the acceleration tends to prolong the actuators, pushing the platform to its physical limitations. Tilt coordination is a technique used to simulate the specific forces by tilting the motion platform.

Considering the rotation matrices in equation 2.16, one can define the gravity vector represented in the simulator frame as:

$$g_{si} = R_x R_y g_i = \begin{bmatrix} -g\sin(\theta) \\ g\cos(\theta)\sin(\phi) \\ g\cos(\theta)\cos(\phi) \end{bmatrix}. \tag{2.27}$$

By the equation, $f_s = a_s - g_{si}$, the specific force is defined as:

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} a_x - g\sin(\theta) \\ a_y - g\cos(\theta)\sin(\phi) \\ a_z - g\cos(\theta)\cos(\phi) \end{bmatrix}. \tag{2.28}$$

Given equation 2.28 it is obvious to understand how the tilting of the platform contributes to simulate the specific force, because the components of the gravity about the different axes affect it.

In section 2.6, a small angles approximation will be taken in account. In that case, the previous equation changes to:

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} a_x + g\theta \\ a_y - g\phi \\ a_z - g \end{bmatrix}. \tag{2.29}$$

2.6 Vestibular System - Linearization

The plant introduced in the last section is nonlinear. This is a problem in terms of computational weight since it requires nonlinear optimization techniques to solve the MPC problem. To solve a simpler optimization methods, a linearization of the vestibular system is suggested, based on the work developed in [31].

2.6.1 Semicircular Channels - Linearization

The first step taken in order to build the proposed model is to disregard the dead-zone related to the threshold of felt rotations. The previous action will define the transfer function of the Semicircular Channels as:

$$TF_{scc} = \frac{T_l T_a s^2}{(T_a s+1)(T_l s+1)(T_l s+1)} = \frac{\hat{\omega}}{\omega}, \tag{2.30}$$

where the input and output were introduced previously in section 2.4.1.

Using the formulation found in [31], the previous transfer function can be rearranged in the following way:

$$TF_{scc} = \frac{\frac{1}{T_s}s^2}{s^3 + (\frac{1}{T_a} + \frac{1}{T_l} + \frac{1}{T_s})s^2 + (\frac{1}{T_l T_s} + \frac{1}{T_l T_a} + \frac{1}{T_s T_a})s + \frac{1}{T_l T_s T_a}}. \tag{2.31}$$

All the presented coefficients are function of the considered rotation about one of the three axes. This means that there will be one transfer function for each of the three rotations, and the MIMO system can be represented in the following way:

$$\begin{bmatrix} \hat{\omega} \end{bmatrix} = \begin{bmatrix} TF_{scc_x} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & TF_{scc_y} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & TF_{scc_z} \end{bmatrix} \begin{bmatrix} \omega \end{bmatrix}, \tag{2.32}$$

where the inputs and outputs can be represented by:

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \hat{\omega} = \begin{bmatrix} \hat{\omega}_x \\ \hat{\omega}_y \\ \hat{\omega}_z \end{bmatrix}. \tag{2.33}$$

The coefficients for the linear transfer functions of the Semicircular Channels can be found in table 2.1.

From the start of the current subsection until the present paragraph a linear model for the semicircular channels behavior has been presented, whose input and output are ω and $\hat{\omega}$, respectively. Nevertheless, since only the Euler angular rates, $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$, are available, it is important to establish a model that represents the relationship between the latter and felt rotations.

Taking the inverse of the relationship presented in equation 2.20 and calculating its linearization around $\beta = 0$ the following relationship is attained:

$$R_s^{-1} = \begin{bmatrix} 1 & 0 & \theta_s \\ 0 & 1 & -\phi_S \\ 0 & \phi_s & 1 \end{bmatrix}. \tag{2.34}$$

Assuming that the Euler angular displacement is small the presented matrix is reduced to an identity matrix which leads to the final result:

$$\dot{\beta} = \omega \tag{2.35}$$

This last result implies that the transfer function can now be presented in the following way:

$$\hat{\omega} = TF_{scc}\dot{\beta}. \tag{2.36}$$

Connecting the previously developed relationship with the one found in equation 2.31 a state space representation can be developed such as:

$$\begin{aligned} \dot{x} &= Ax + B\dot{\beta}, \\ \hat{\omega} &= Cx + D\dot{\beta}. \end{aligned} \tag{2.37}$$

The state space presented in 2.37 was developed recurring to the canonical form of the observer, resulting in the set of system matrices shown in equation 2.38.

$$A = \begin{bmatrix} -\frac{1}{T_a} - \frac{1}{T_l} - \frac{1}{T_s} & 1 & 0 \\ -\frac{1}{T_l T_s} - \frac{1}{T_l T_a} - \frac{1}{T_s T_a} & 0 & 1 \\ -\frac{1}{T_l T_s T_a} & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} \frac{1}{T_s} \\ 0 \\ 0 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}. \tag{2.38}$$

Given that the number of zeros is less than the number of poles there exists no feedthrough, which implies $D = 0$.

Like in the transfer function case, the presented state space is a general representation that fits each one of the three rotations, for the correspondent parameters (table 2.1). Finally, the complete Semicircular Channels model is defined in the following way:

$$\begin{aligned} \dot{x}_{scc} &= A_{scc}x_{scc} + B_{scc}\dot{\beta}, \\ \hat{\omega} &= C_{scc}x_{scc} + D_{scc}\dot{\beta}, \end{aligned} \tag{2.39}$$

where

$$A_{scc} = \begin{bmatrix} A_x & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & A_y & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & A_z \end{bmatrix}, B_{scc} = \begin{bmatrix} B_x & 0_{3\times1} & 0_{3\times1} \\ 0_{3\times1} & B_y & 0_{3\times1} \\ 0_{3\times1} & 0_{3\times1} & B_z \end{bmatrix}, \tag{2.40}$$

and

$$C_{scc} = \begin{bmatrix} C_x & 0_{1\times3} & 0_{1\times3} \\ 0_{1\times3} & C_y & 0_{1\times3} \\ 0_{1\times3} & 0_{1\times3} & C_z \end{bmatrix}, D_{scc} = \begin{bmatrix} 0 \end{bmatrix}, x_{scc} = \begin{bmatrix} x_{scc_x} \\ x_{scc_y} \\ x_{scc_y} \end{bmatrix}. \tag{2.41}$$

All the underscript symbols, x, y and z identify the state space matrices as part of a system related to rotations around that specific axis. For example, the matrices with underscript x belong to the state space model that associates the input $\dot{\phi}$ with the output $\hat{\omega}_x$.

2.6.2 Otoliths - Linearization

The development of the present model originates from the same principle as in the previous case, this means that the deadzone dynamics will be ignored and the system will only be regarded as the product of both transfer functions introduced in figure 2.8. The result of latter relationship is the following transfer function:

$$TF_{oto} = \frac{K(\tau_a s + 1)}{(\tau_l s + 1)(\tau_s s + 1)} = \frac{\hat{f}}{f}, \tag{2.42}$$

where the input and output were introduced previously in section 2.4.2.

The given transfer function can hold the following representation:

$$TF_{oto} = \frac{\frac{K\tau_a}{\tau_l\tau_s}s + \frac{K\tau_a}{\tau_l\tau_s}}{s^2 + \left(\frac{1}{\tau_l} + \frac{1}{\tau_s}\right)s + \frac{1}{\tau_l\tau_s}}. \tag{2.43}$$

As in the semicircular channels case, all the given coefficients are function of the specific force about a specific axis. This implies that the layout of the overall transfer function, representing the complete otolith model, is a MIMO system, 3×3, with the following format:

$$\begin{bmatrix} \hat{f} \end{bmatrix} = \begin{bmatrix} TF_{oto_x} & 0_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & TF_{oto_y} & 0_{2\times2} \\ 0_{2\times2} & 0_{2\times2} & TF_{oto_z} \end{bmatrix} \begin{bmatrix} f \end{bmatrix}. \tag{2.44}$$

where

$$f = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \hat{f} = \begin{bmatrix} \hat{f}_x \\ \hat{f}_y \\ \hat{f}_z \end{bmatrix}. \tag{2.45}$$

All the needed coefficients are stated in table 2.2.

The corresponding state space of the equation 2.43 can be formulated by:

$$\begin{aligned} \dot{x} &= Ax + Bf, \\ \hat{f} &= Cx + Df, \end{aligned} \tag{2.46}$$

where, through utilization of the canonical form of the observer, the state matrices are represented by:

$$A = \begin{bmatrix} -\frac{1}{\tau_l} - \frac{1}{\tau_s} & 1 \\ -\frac{1}{\tau_l \tau_s} & 0 \end{bmatrix}, B = \begin{bmatrix} \frac{K\tau_a}{\tau_l \tau_s} \\ \frac{K}{\tau_l \tau_s} \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{2.47}$$

Like in the previous model the state space feedthrough matrix, D, is zero.

The complete otolith model, attained by replacing all parameters for the different axes, can be presented by:

$$\begin{aligned} \dot{x}_{oto_f} &= A_{oto_f} x_{oto_f} + B_{oto_f} f, \\ \hat{f} &= C_{oto_f} x_{oto_f} + D_{oto_f} f, \end{aligned} \tag{2.48}$$

where the state space matrices have a similar format as the previously derived in equations 2.40 and 2.41, with correspondently sized zero matrices. The underscript f identifies the state space matrices as being defined for a system whose input consists of specific forces.

All the former work, in the present section, shows how the standard otolith model is built. Nevertheless, the resulting model can not be used in this project without an input transformation. Since specific forces are not available signals, a relation between the latter with translational accelerations and Euler angular rates needs to be developed.

Recollecting the tilt coordination technique, section 2.5, a relationship between the Euler angles and the components of the gravity vector about the three axes was developed. This means that f, in the relationship $f = a - g$, can be represented as a function of the Euler angles and accelerations (equation 2.28) and the following equation can be derived:

$$f = H\begin{bmatrix} \beta & a \end{bmatrix}^T, \tag{2.49}$$

and, considering the approximation applied to obtain equation 2.29 as well as disregarding its constant term in the component about the Z-axis,

$$H = \begin{bmatrix} 0 & g & 0 & 1 & 0 & 0 \\ -g & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.50}$$

If the input of the vestibular system consisted of Euler angles and accelerations, one could just use the product between B_{oto_f} and H to compute the new input matrix. However, that is not the case because the goal is to have Euler angular rates as input, as justified in section 2.7.4. This implies that the derived state space model needs to be modified to fit with the desired input:

$$u = \begin{bmatrix} \dot{\beta} & a \end{bmatrix}^T. \tag{2.51}$$

This goal is attained by adding three extra states, θ, ϕ, ψ, to the state space system. Recalling the input matrix,

$$B_{oto_f} = \begin{bmatrix} B_x & 0_{2\times1} & 0_{2\times1} \\ 0_{2\times1} & B_y & 0_{2\times1} \\ 0_{2\times1} & 0_{2\times1} & B_z \end{bmatrix}, \tag{2.52}$$

and, let $B_{oto_{\beta,a}} = B_{oto_f}H$, be the input matrix for the system using $[\beta \ a]$ as inputs. Assuming $B_{oto_{\beta,a}}$ can be partitioned in two, $B_{oto_{\beta,a}} = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$ then, $B_{oto_{\beta,a}}\begin{bmatrix} \beta & a \end{bmatrix}^T = B_1\beta + B_2a$. Bearing in mind this relationship, replacing f by u,equation 2.51 and dropping the underscript f in equation 2.48, the new state space matrices can be computed:

$$A_{oto} = \begin{bmatrix} A_x & 0_{2\times2} & 0_{2\times2} & B_{1(1:2,:)} \\ 0_{2\times2} & A_y & 0_{2\times2} & B_{1(3:4,:)} \\ 0_{2\times2} & 0_{2\times2} & A_z & B_{1(5:6,:)} \\ 0_{1\times2} & 0_{1\times2} & 0_{1\times2} & 0_{1\times3} \\ 0_{1\times2} & 0_{1\times2} & 0_{1\times2} & 0_{1\times3} \\ 0_{1\times2} & 0_{1\times2} & 0_{1\times2} & 0_{1\times3} \end{bmatrix}, B_{oto} = \begin{bmatrix} 0_{6\times3} & B_2 \\ I_{3\times3} & 0_{3\times3} \end{bmatrix}, \tag{2.53}$$

and

$$C_{oto} = \begin{bmatrix} C_x & 0_{1\times2} & 0_{1\times2} & 0 & 0 & 0 \\ 0_{1\times2} & C_y & 0_{1\times2} & 0 & 0 & 0 \\ 0_{1\times2} & 0_{1\times2} & C_z & 0 & 0 & 0 \end{bmatrix}, D_{oto} = \begin{bmatrix} 0 \end{bmatrix}, x_{oto} = \begin{bmatrix} x_{oto_x} \\ x_{oto_y} \\ x_{oto_z} \\ \phi \\ \theta \\ \psi \end{bmatrix}. \tag{2.54}$$

Given the approximation used to achieve the relationship between felt specific forces, accelerations and Euler angles, equation 2.50, the third output of the above model is not the felt specific force about the Z-axis.

In order to obtain the desired output, the constant term of the specific force vector about the Z-axis, disregarded previously, needs to be filtered by the vestibular system and added to the output correspondent output. This term represents the stationary gain of the Otolith transfer function for the felt force about the Z-axis. It can be regarded as an equilibrium term due to the constant presence of gravity.

2.6.3 Vestibular System - Linearization

Gathering all the information from the previous two subsections, it is possible to build the final dynamical model for the whole vestibular system. It consists of an aggregation of the two previous models plus the addition of integral states that will be needed in the MPC formulation.

The objective of the present section is to build the following model:

$$\begin{aligned} \dot{x}_{vest} &= A_{vest}x_{vest} + B_{vest}u, \\ Y_{vest} &= C_{vest}x_{vest} + D_{vest}u, \end{aligned} \tag{2.55}$$

where u is stated in 2.51. The state vector has the form

$$x_{vest} = \begin{bmatrix} x_{scc} \\ x_{oto} \\ p_x \\ v_x \\ p_y \\ v_y \\ p_z \\ v_z \end{bmatrix}, \tag{2.56}$$

and the state matrices, as well as the output vector, will be defined in the forthcoming steps.

In equation 2.56 there are six extra states besides the previously defined ones. The latter are integral states and will be used for constraint purposes in the MPC optimization problem.

To build the state matrix, the matrices developed in equations 2.40 and 2.53 need to be aggregated. Subsequently, extra elements must be added given the integral states presented in 2.56. Taking all the late steps into account, the state matrix is defined such as:

$$A_{vest} = \begin{bmatrix} A_{scc} & 0_{9\times9} & 0_{9\times6} \\ 0_{9\times9} & A_{oto} & 0_{9\times6} \\ 0_{6\times9} & 0_{6\times9} & A_e \end{bmatrix}, \tag{2.57}$$

where

$$A_e = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.58}$$

Like in the previous case, B_{vest} consists in the aggregation of the input matrices 2.40 and 2.53. Since there are six new states, a $[6\times6]$ block is added underneath the matrix formulated by the mentioned aggregation. This new block is defined such that the accelerations are equal to the state derivative of the velocity states. According to these instructions the system input matrix takes the following format:

$$B_{vest} = \begin{bmatrix} & B_{scc} & & & & \\ & B_{oto} & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.59}$$

Linking the output matrix in equation 2.41 with the one defined in 2.54, plus an identity matrix that defines the integral states as outputs, the final output matrix is formulated as:

$$C_{vest} = \begin{bmatrix} C_{scc} & 0_{3\times9} & 0_{3\times6} \\ 0_{3\times9} & C_{oto} & 0_{3\times6} \\ 0_{9\times9} & 0_{9\times6} & I_{9\times9} \end{bmatrix}. \tag{2.60}$$

The output matrix format determines the next set of outputs:

$$Y_{vest} = \begin{bmatrix} \hat{\omega}_x & \hat{\omega}_y & \hat{\omega}_z & \hat{f}_x & \hat{f}_y & \hat{f}_z & \phi & \theta \\ & & & \psi & p_x & v_x & p_y & v_y & p_z & v_z \end{bmatrix}^T. \tag{2.61}$$

2.7 MPC Formulation

The interest in the field of Model Predictive Control (MPC) started in the 1980s after the publication of dynamic matrix control (DMC) (Cutler & Ramaker, 1979, 1980), and since then, it has been extensively studied [26]. DMC had a tremendous impact in the oil industry and the initial research on MPC is defined by efforts to understand DMC. The latter has the ability to deal with constrained control problems, and it works with a step response or finite impulse model of the plant [26].

Its ability to deal with large multivariable constrained control problems, has made MPC quite popular in industry, over the past 30 years [42]. Firstly it was applied in Petrochemical industry [3], [21]. Currently with its development, MPC applications have been increasing in others sectors of the process industry such as, Automotive and Aerospace systems [30].

MPC is a control technique where the provided control actions are optimized according to a determined system model, plant, as well as a set of constraints on state evolution, input and input rates. The idea is to compute the control action by solving a finite horizon open-loop optimal control problem, at each sampling instant over a future fixed interval (Prediction Horizon) while always respecting the previously defined constraints. The current state of the system is used as initial state of the plant and the optimization process gives an optimal control sequence. The most common approach is to use the receding horizon technique, 2.7.2. wIt dictates that only the first value of the vector of optimized control actions is applied to the plant. This method will be introduced later on.

2.7.1 Advantages and Limitations of MPC

The main feature of MPC, that clearly distinguishes it from other control techniques, is its ability to handle input and output constraints (soft and/or hard), by direct incorporation into the optimization problem. This aspect of MPC is of great importance because most profitable operations are usually achieved when a process is running near its constraints [21]. It is a very versatile method which is able to control a several number of different processes, for instance, those with non-minimum phase, long time delay, or even open-loop unstable characteristics [3]. Furthermore, it also has advantages regarding the way in which the control

objective is defined. All control objectives are established in a single cost function where different parameters are weighted according to the user preferences for the resultant system behavior. MPC can also implement feedforward control in order to compensate the effects of disturbances [3].

On the other hand, MPC also has some disadvantages. Theoretical aspects associated with stability and performance properties of MPC have proven to be a problematic issue [37]. These properties are difficult to estimate and are currently an active area of research. Stability, for instance, is complicated to ascertain due to the introduction of constraints in the optimization process, which renders a non-linear closed-loop system even if the the plant dynamics are linear, [37]. Usually, to enforce stability, one can define a last state constraint that forces a selected group of states to achieve a certain value, [21].

A big disadvantage associated with MPC is the computational weigh associated with solving the optimization problem. Since at everytime instant an optimization problem needs to be solved, a lot of computational resources are needed, thus compromising real time applications.

MPC uses an internal model of the plant to be controlled, and this can be considered as an advantage and limitation simultaneously. It can be regarded as one advantage because the process model allows the controller to deal with a replica of the real process dynamics, leading to a much better set of control actions [25]. On the other hand, any available model, as good as it might be, is never so accurate as the real system behavior. This means that the predictions will differ from actual system values for the same set of conditions.

2.7.2 Receding Horizon

The receding horizon idea is related with the application of the control input, and is motivated by both the fact that the plant predictions are not so accurate, and by the presence of unmeasured disturbances that can lead the system states away from the ones predicted by the plant.

The main idea is that MPC only uses the first value of the control sequence

$$U(t) = [u(t|t), ..., u(t + H_p - 1|t)], \tag{2.62}$$

over the time horizon H_p, computed at each optimization. The rest of the control sequence is discarded hence, the name receding horizon. Figure 2.9 shows how the receding horizon idea would perform with $H_p = 6$. Each plot shows the minimizing control sequence, equation 2.62, that is computed at the sampling instant zero and one, respectively. The colored inputs are the ones that would be applied to the system and the remaining are discarded.

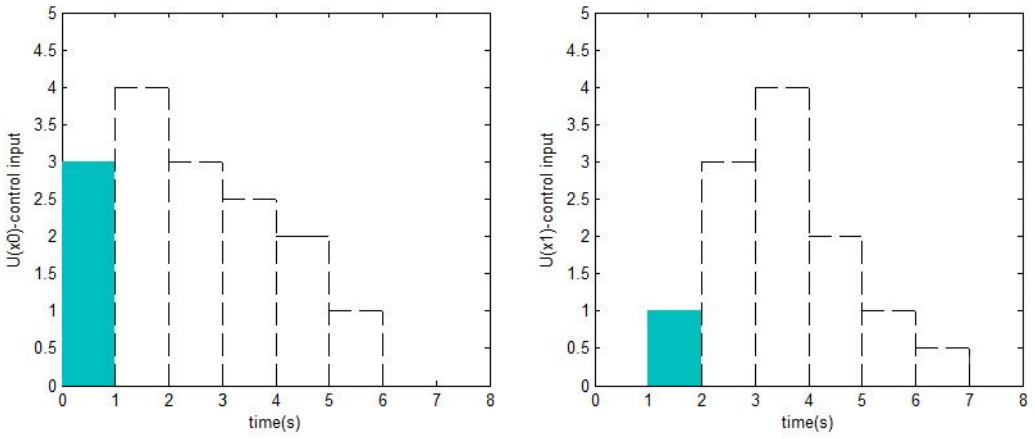

Figure 2.9: Example of receding horizon idea

2.7.3 MPC Problem

When using MPC, an optimization problem is solved online over a finite prediction horizon H_p. There are three different kinds of variables to be taken into consideration. These are known as the tracking variables, manipulated and constrained variables. The tracking are defined as the variables that need to follow a given reference, the manipulated are the plant inputs and the constrained are the variables limited with upper and/or lower bounds. It is important to notice that the tracking and the manipulated variables may also be constrained. In addition to the previously defined prediction horizon it is also common to establish two extra important parameters for the definition of the MPC problem.

- The control horizon H_u determines the number of time instants for which the input u is going to be optimized .For instants above and equal to H_u the control input is equal to the last computed one. By other words, there are no input variations after H_u time instants. The relationship between the control and prediction horizon is stated as $H_u \leq H_p$.

- The constraint horizon, H_c, establishes the number of sampling instants where the constraints will be enforced and its relation with the prediction horizon is as follows, $H_c \leq H_p$.

2.7.4 Working Variables

In order to understand the problem at hand there is a set of working variables that needs to be presented. This group includes three smaller sets that refer to Manip-

ulated variables, Constrained variables and Tracking variables. These smaller sets might have common variables as it is going to be shown.

Manipulated variables - MV This set contains all the variables used to control the plant. They can be stated as:

$$\left[\begin{array}{cccccccc} \dot{\phi}_r & \dot{\theta}_r & \dot{\psi} & a_x & a_y & a_z & \dot{\phi}_a & \dot{\theta}_a \end{array} \right]. \tag{2.63}$$

From left to right, the first two variables represent the roll and pitch angular rates used to simulate rotations in the motion platform referential. The third one represents the yaw angular rate. The following three variables are the translational accelerations of the motion platform referential. The last two variables are the roll and pitch angular rates, in the motion platform referential. They are used to simulate accelerations by tilt coordination. All these variables are assumed to be the same in the drivers head because the simulator and head referentials are parallel.

The given set of variables is the input of the vestibular system as presented in section F.2.

Tracking variables - TV The current set consists of the group of variables that are tracked by the MPC. Every sampling time, future references for the tracking variables are fed to the controller. Every time instant the controller will try to minimize the error between the estimated values for all tracking variables, given a set of control inputs, and the provided references. This set of variables consists of:

$$\left[\begin{array}{cccccccccccccc} \hat{\omega}_x & \hat{\omega}_y & \hat{\omega}_z & \hat{f}_x & \hat{f}_y & \hat{f}_z & \phi & \theta & \psi & p_x & v_x & p_y & v_y & p_z & v_z \end{array} \right]. \tag{2.64}$$

From left to right, the first three variables correspond to the rotations felt by the driver and the three following ones the felt specific forces. These six variables are the ones for which references are computed. The minimization of the error between the sensation signals in the platform and in the vehicle is the utmost goal of this work.

The last remaining variables have a constant reference which is equal to zero. This is done to guarantee that the platform returns to its neutral position, therefore allowing more freedom of displacements in future motion cues. In other words, the tracking of this variables is performed to ensure the washout of the motion platform. About this group of nine variables, from left to right, the first three refer to the Euler angles of the motion platform while the last six are positions, p, and translational velocities, v, of the motion platform referential referent to the inertial one.

All underscripts refer to the axis that the variable is represented in.

Constrained Variables - CV The set of constrained variables consists of all states that are bounded to a certain interval of values. This set is represented by:

$$\begin{bmatrix} MV & \Delta MV & \phi & \theta & \psi & p_x & v_x & p_y & v_y & p_z & v_z \end{bmatrix}. \tag{2.65}$$

MV and ΔMV refer to the manipulated variables and their variation rates, while the remaining variables, also introduced previously, can be found in the group of tracking variables. The bounds for these states can be seen in the Appendix B, tables B.1 and B.2.

As shown in the section about the CVS, the platform command is in the form of Euler angles and positions. This means that the set MV can not be sent directly to the motion platform but instead integration is necessary. The necessary values are already computed in the CV set and are applied directly as control commands.

2.7.5 Weights

Weights are used in the objective function in order to set relationships between the different variables. They are applied to tracking variables, Q matrix, inputs, S matrix, and input rates, R matrix.

The most common approach, when dealing with these kind of problems, is to scale the states and the state space matrices. The goal of this process is to make the weighting task easy. If all variables belong to the same range, 0-1, the relationship between the different weights can be translated to the variables they are weighing. For example, if two variables are weighted by the same quantity it means that they have the same importance.

In the current master thesis, the state space model had states with no physical meaning. This implies that the boundaries of the latter are not known, therefore they can not be scaled. This reason forces the application of completely different weights on the variables thus, making the relationship between the latter not so obvious.

Sensed rotations

Given the prediction model in appendix F.2, the weights for these variables need to be high when compared to the remaining ones. This is done to prevent the controller from creating false cues, by tilt coordination. These high weights have no other effect in the controlling task than the above, and guarantee a good tracking of the sensed rotations.

Felt specific forces

When weighting these variables one should be careful enough to think about the constraint hitting. If weighted harshly, the controller will do everything that is possible to track their values. This can lead to constraint hitting for the positions and angular displacements, causing the driver to feel a discontinuity in the specific force.

Euler angles, translational positions and velocities

These values have a constant reference of zero. Weighting these values is done to guarantee that the platform returns to its original position when the errors on the sensed signals start to decrease.

Inputs and input rates

Weighting the input rates is important to prevent big input variations.

Input weighting is not so obvious in this Thesis. It was used to establish a relationship between the usage of the different control inputs. For example, weights on accelerations were far smaller than the weights in the Euler angular rates. This was done to make the controller use both inputs in an even way when simulating specific forces.

2.7.6 MPC Mathematical Model

Using the linear plant defined in section 2.6.3, the optimization problem can be solved with a quadratic programming optimization technique. It is a well known optimization operation and due to the convexity of the problem it is relatively easy to achieve a global optimum.

Consider the discretization of the vestibular system presented in the preceding section.

$$\begin{aligned} \dot{x}_{vest}(k+1) &= A_{vest_d} x_{vest}(k) + B_{vest_d} u(k), \\ Y_{vest}(k+1) &= C_{vest} x_{vest}(k) + D_{vest} u(k), \end{aligned} \tag{2.66}$$

where A_{vest_d} and B_{vest_d} are the discrete time correspondent state and input matrices. More information about the discretization process is presented in Appendix C.

The plant model 2.66 expresses the plant state x_{vest} in terms of the values of the input u. However, it is desired to have the input changing rates, $\Delta u(k)$, as the optimization variables in the MPC problem. For that reason, $\Delta u(k)$ will be included in the above state space representation. Moreover, the plant model, as presented in section 2.6.3, leaves no room for constraints on the angular rate inputs, when it comes to tilt coordination.

As mentioned previously, angular displacement is going to be applied to simulate low frequency accelerations. However, one should be careful enough not to generate false rotation cues and for that reason a maximum tilting velocity needs to be specified. Because of this fact, and the inclusion of $\Delta u(k)$ in the MPC plant, the prediction model is going to differ slightly from the given vestibular model. The formulation of the prediction model is presented in appendix F.2.

Let the objective function be defined by

$$\begin{aligned} J(t) &= \sum_{i=1}^{H_p} ||y_{tr}(t+i|t) - r(t+i|t)||_Q^2 + \sum_{i=0}^{H_u-1} ||u(t+i|t)||_S^2 + \\ &\quad \sum_{i=0}^{H_u-1} ||\Delta u(t+i|t)||_R^2, \end{aligned} \tag{2.67}$$

where all states, inputs and outputs of the prediction model are distinguished from the physical system ones by the usage of term t. It refers to the instant in which they were created. For example $y_{tr}(k|t)$ refers to the output estimated for instant k at time t. The problem of having an initial state $x(k)$ at time k is assumed, then the model predictive control action $u(k+i)_{i=0}^{H_p-1}$ at time k is obtained by solving the optimization problem:

$$\min_{\Delta U(t)} \quad J(\xi(t), u(t-1), \Delta U(t)), \tag{2.68a}$$

$$\begin{aligned} &subj.\ to, \\ &\xi(t+i+1|t) = A_{aug}\xi(t+i|t) + B_{aug}\delta u(t+i|t), && i = 0,\dots,H_p && \text{(2.68b)} \\ &\tilde{y}(t+i|t) = C_{aug}\xi(t+i|t) + D_{aug}\delta u(t+i|t), && i = 0,\dots,H_p && \text{(2.68c)} \\ &u(t+i|t) = \Delta u(t+i|t) + u(t+i-1|t), && i = 0,\dots,H_p-1 && \text{(2.68d)} \\ &\Delta u(t+i|t) = 0, && i = H_u,\dots,H_p-1 && \text{(2.68e)} \\ &y_{min} \le y(t+i|t) \le y_{max}, && i = 0,\dots,H_c-1 && \text{(2.68f)} \\ &u_{min} \le u(t+i|t) \le u_{max}, && i = 0,\dots,H_c-1 && \text{(2.68g)} \\ &\Delta u_{min} \le \Delta u(t+i|t) \le \Delta u_{max}, && i = 0,\dots,H_c-1 && \text{(2.68h)} \\ &\xi(t|t) = \xi(t), && && \text{(2.68i)} \\ &u(t-1|t) = u(t-1). && && \text{(2.68j)} \end{aligned}$$

The meaning of the introduced constrains is explained in the next set of steps. The given indices $b, c, ..., j$ are equivalent to the ones found in equation 2.68.

b)- The next prediction state is computed calculated accordingly to the system dynamics.

c)- Like the next state, the predicted output is a function of the system dynamics.

d)- The input in instant t is a sum of the input rate at same instant with the input for $t-1$.

e)- For all prediction instants above $H_u - 1$ the input rate is zero.

f)- All outputs must be within their bounds. y_{max} and y_{min} are vectors that contain upper and lower bounds for the output.

g)- All inputs must be within their bounds. u_{max} and u_{min} contain upper and lower bounds for the input.

h)- All input rates must be within their bounds. Δu_{max} and Δu_{min} contain upper and lower bounds for the input changes.

i)- Prediction states at time t are equal to the system state at time t.

j)- Predicted input at time $t-1$ is equal to the applied input to the system at time $t-1$.

Chapter 3

Experimental Results

In this section the results obtained for offline and real time simulations of the implemented controllers will be presented. of the bicycle model is discussed as well as issues with the discretization of the MPC plant.

3.1 Testing Scenarios

The testing scenarios should contemplate the majority of inputs that can be found when driving a vehicle. Bearing this in mind, a combination of maneuvers was used in all presented simulations; lane changing and posterior braking. This maneuver was chosen due to its combination of lateral and longitudinal dynamics excitation, rendering a very complete test scenario.

For testing the MPC in the motion platform the maneuver was augmented. In this case, the car starts from a stopped position and accelerates until a speed of about $100Km/h$ prior to performing the actions mentioned above.

3.2 Experimental Setup

All offline simulations and tests were performed with Matlab R2007a, Version 7.4.0.287. All applications, regarding the proposed controllers, were developed using Simulink applying the built in blocks as well as some user C-coded S-functions. Tuning and validation of the bicycle model was done recurring to a series of user defined functions in m-files.

Real time tests were performed recurring to Real Time Workshop, RTW, present in Matlab Version 7.1.0.246, R14. The Target computer, working in real time, was a Pentium 4 with $2.4GHz$ and $512mB$ of available RAM. All real time applications are developed in Simulink just like in the previous case.

The developed controller was C-coded, and the optimization problem was solved with the use of a package named LSSOL that solves quadratic programming problems. LSSOL consists of a series of optimization routines, and for more information consult [11].

3.3 Tuning of the Vehicle Model

In order to completely define the bicycle model of section 2.2.3, there are four parameters that were needed to be defined. These are the longitudinal and lateral stiffness coefficients for the front and rear wheels. They were optimized, via genetic algorithms, by minimization of the error between estimated states by the bicycle model and states extracted from the vehicle model in the driving simulator.

The bicycle model was tuned based on two different sets of data. One was focused on the longitudinal stiffness coefficients, breaking maneuver, while the other focused on lateral stiffness coefficients, lane changing maneuver. One hundred iterations with genetic algorithms, appendix E, proved to be enough to achieve a group of parameters that guarantee a good performance in simulation. Nevertheless, when the model was validated with the driving scenario presented in section 3.1, it showed a very bad performance when tracking the velocity about the X-axis. It was discovered that the slip ratio values would push the tire forces away from the linear region, thus compromising the estimations of the tuned model.

We decided that the values for the longitudinal forces would be directly extracted from the vehicle model. This reduced the number of optimization parameters to two, front and rear lateral stiffness coefficients.

With a new set of data and the previously mentioned modification, the model was tuned once again. Like in the first tuning phase, one hundred generations resulted in very satisfactory set of results for the bicycle model states evolution. At the bottom of this section one can find a batch of figures displaying the states path for test scenario described above.

The values for the lateral stiffness coefficients, resulting from the optimization process, are presented in the next table.

Table 3.1: Lateral stiffness coefficients

Front (N)	Rear (N)
46679	52971

After analyzing the results, it is possible to conclude that there is a good tracking, particularly in the cases of yaw angular displacement and rate as well as estimated position in the inertial frame. In what the concerns the velocity about

the X-axis one can observe the good state estimation, even though a small tracking error is present.

Relatively to the estimated values for the velocity about the Y-axis, it is possible to conclude that the evolution of the bicycle model and the real vehicle signal are similar. Nevertheless, one can observe a difference in the signals magnitude. This might be due to nonlinear behavior of the tires that is not reproduced by the bicycle model.

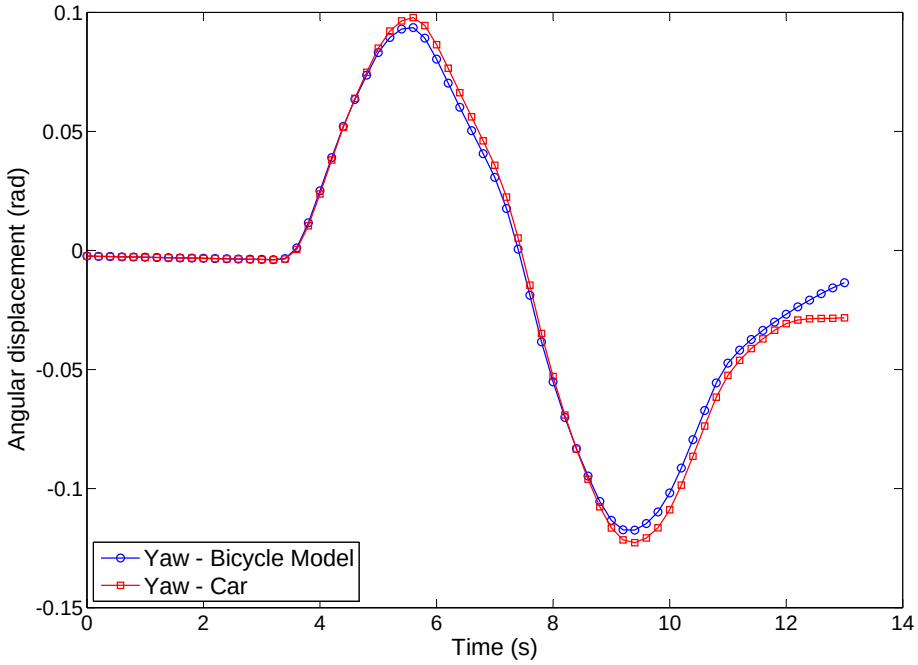

Figure 3.1: Bicycle model estimation of Yaw displacement

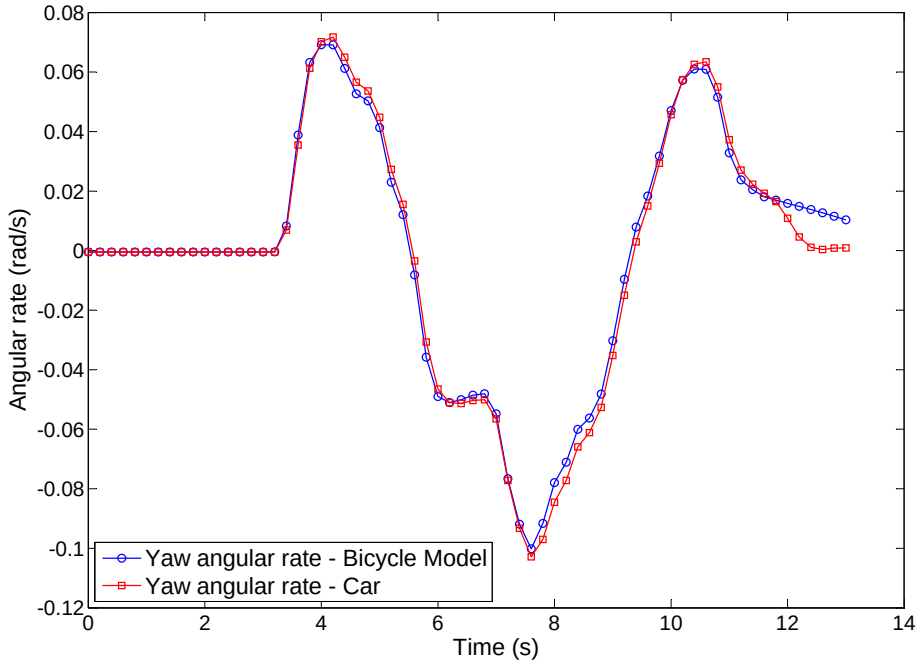

Figure 3.2: Bicycle model estimation of Yaw rate

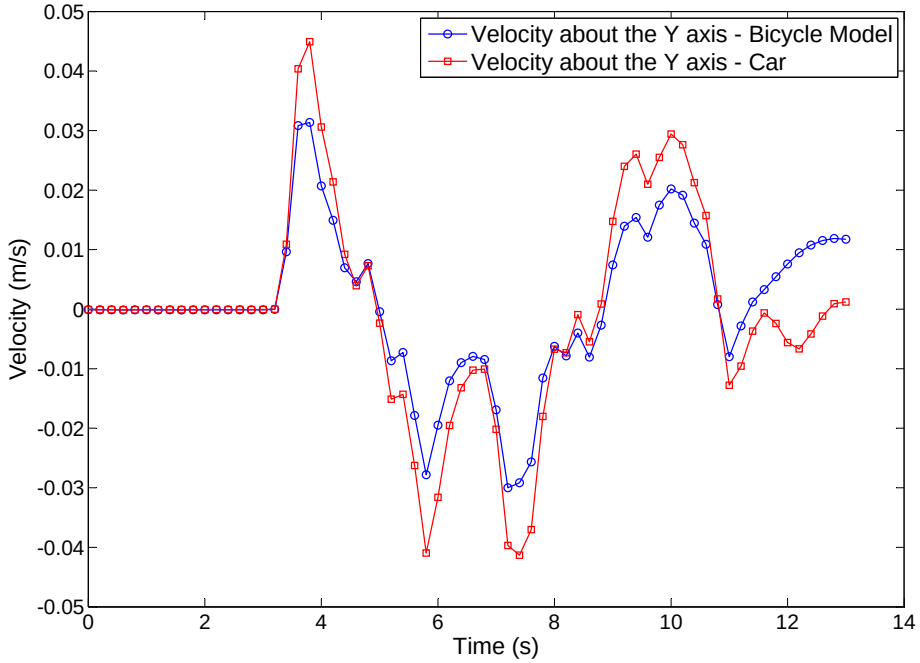

Figure 3.3: Bicycle model estimation of the velocity about the Y-axis

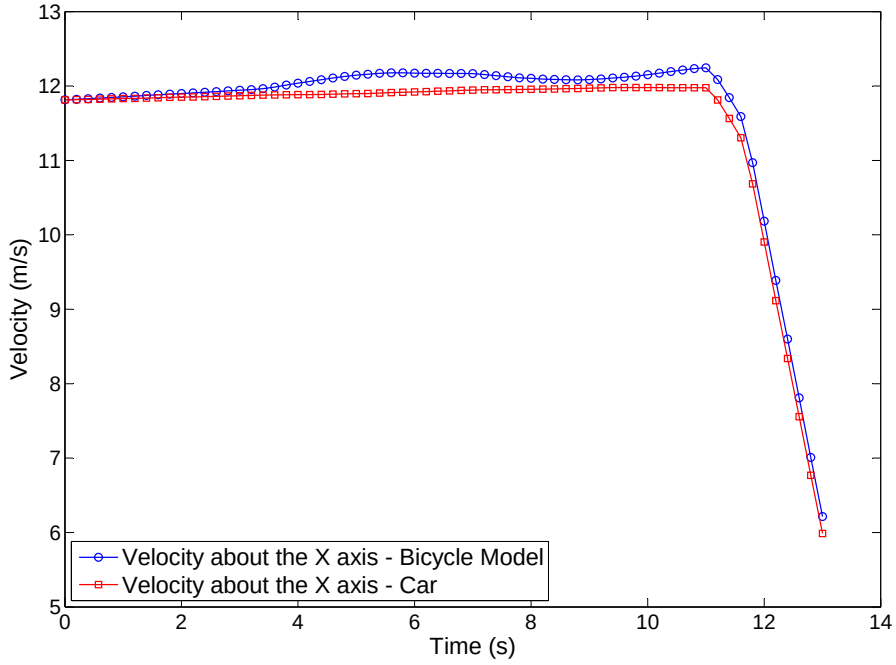

Figure 3.4: Bicycle model estimation of the velocity about the X-axis

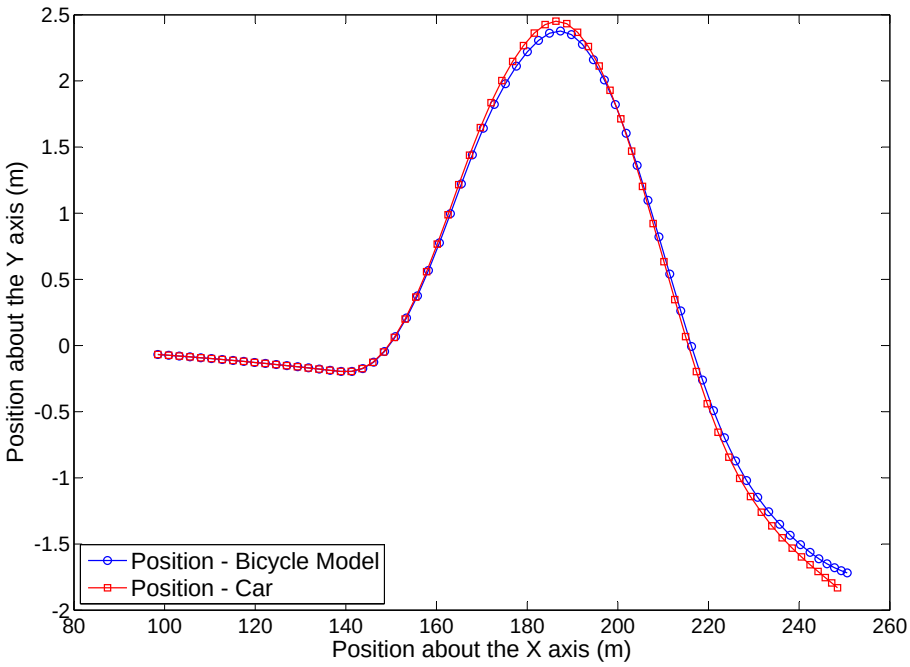

Figure 3.5: Bicycle model estimation of the position in the inertial referencial

3.4 Choosing the sampling period

The sampling period is an important feature of the presented controller because it is directly connected with the computational time. The sampling period dictates the size of the required parameters of the MPC, like prediction horizon, therefore the dimension of the optimization problem. The choice of the sampling rate is done so that stability is achieved and the dynamical behavior of the plant is replicated.

In appendix C, was shown that a minimal sampling period of $0.2s$ should be selected in order to obtain a stable discrete plant. In what concerns the reproduction of plant dynamics, selecting a sampling rate equal or less than $1/10$ of the rising time of the fastest pole usually ensures that all dynamics are captured. Figure 3.6 shows that the latter is around $0.025s$.

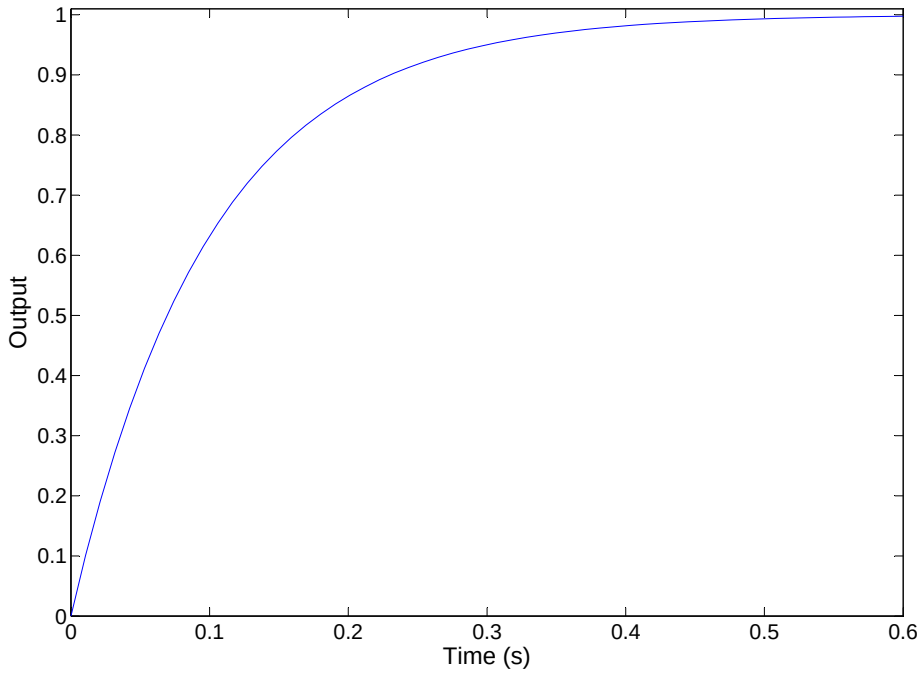

Figure 3.6: Step response of the first order filter whose pole is the fastest in the vestibular system

The selection of the sampling period in this problem would be done taking in consideration the rate at which information is sent to the platform, $0.01s$. This is the minimal sending rate that ensures smooth movements by the motion platform, according to the user manual. Nevertheless, testing showed that the above value requires large horizons, thus high computational weights. It was confirmed in real

time that it is impossible to implement the controller using the referred period of $0.01s$, because the optimization problem is to complex to be solved in the required time.

The last conclusion forces a change in the approach. Either the simulation period is increased allowing more time to perform the necessary computations for the optimization or the sampling time of the MPC plant is changed.

The simulation period is $0.001s$. This value was selected given the fast dynamics of the vehicle model that is supposed to be simulated, so it should not be modified. This result implies that the sampling period of the MPC plant will be changed.

Offline simulations showed that $0.05s$ represents a good trade off between results quality and optimization problem size.

The chosen sampling period does not respect the criterion specified above for dynamic behavior reproduction. Nevertheless, given that they do not differ much, the difference will be ignored and the model dynamics will still be considered as reproduced.

3.5 MPC selection

This section contains a comparison between the results of the three developed linear MPC controllers. The testing data was extracted from the vehicle model, in driving simulator, and comprises of the maneuver introduced in section 3.1. Let the different controllers be defined as follows:

- **Controller 1** - This controller assumes that the vehicle accelerations and rotations are constant for all the prediction horizon. References in terms of tracking variables are computed by filtering the latter data with the vestibular system.

- **Controller 2** - This controller assumes that the bicycle model inputs are constant through the prediction horizon. With this information accelerations and rotations are computed for all time instants in the prediction horizon. The latter are filtered by the vestibular system thus, creating references in terms of tracking variables.

- **Controller 3** - This control approach uses the bicycle model to generate references in terms of accelerations and rotations. Instead of assuming that the inputs are constant, like the previous one, the input derivative is assumed constant. Generation of references for tracking variables is done in the same way as for the other controllers.

Figures 3.7 to 3.12 show the achieved results. The tests were performed with a prediction horizon of 18 time instants, control and constraint horizon of 6 time instants. These were selected by trial and error, and it has verified that applying bigger constraint and control horizons would not affect considerably the results.

The weights of objective function variables are given in A.5.

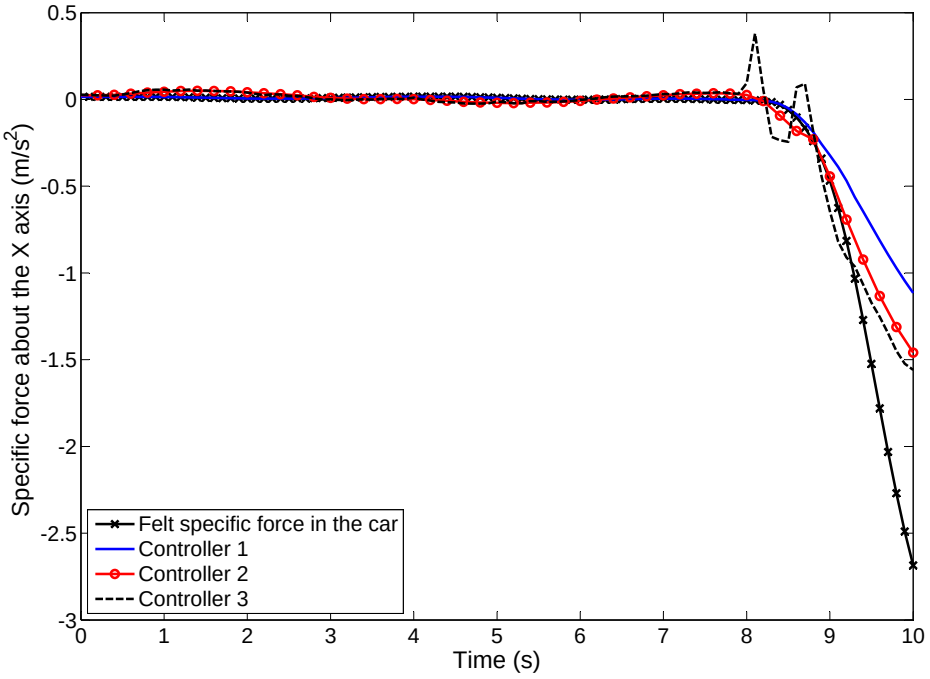

Figure 3.7: Felt specific force about the X-axis

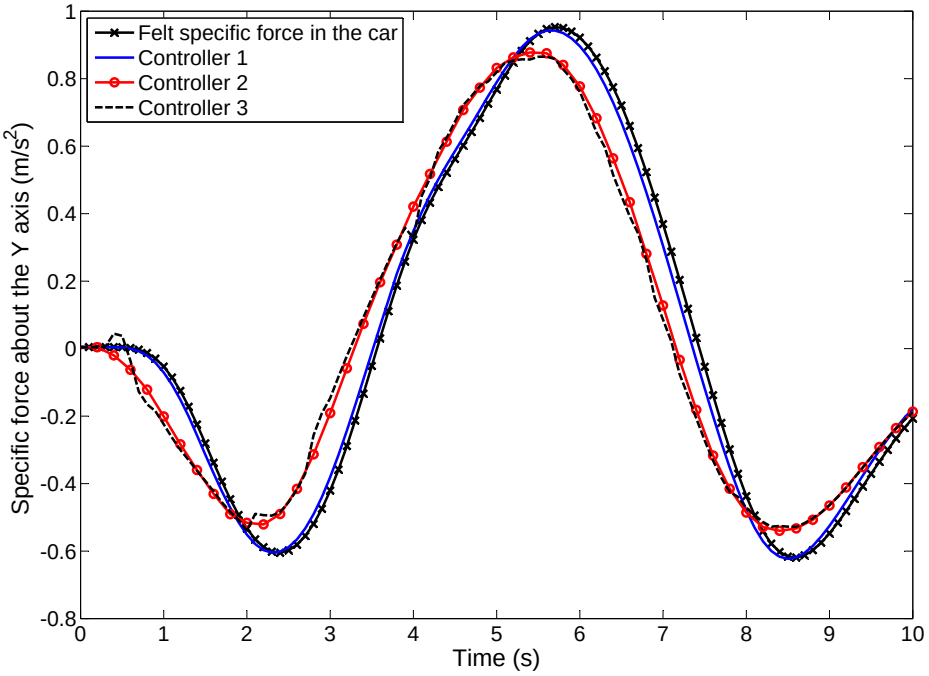

Figure 3.8: Felt specific force about the Y-axis

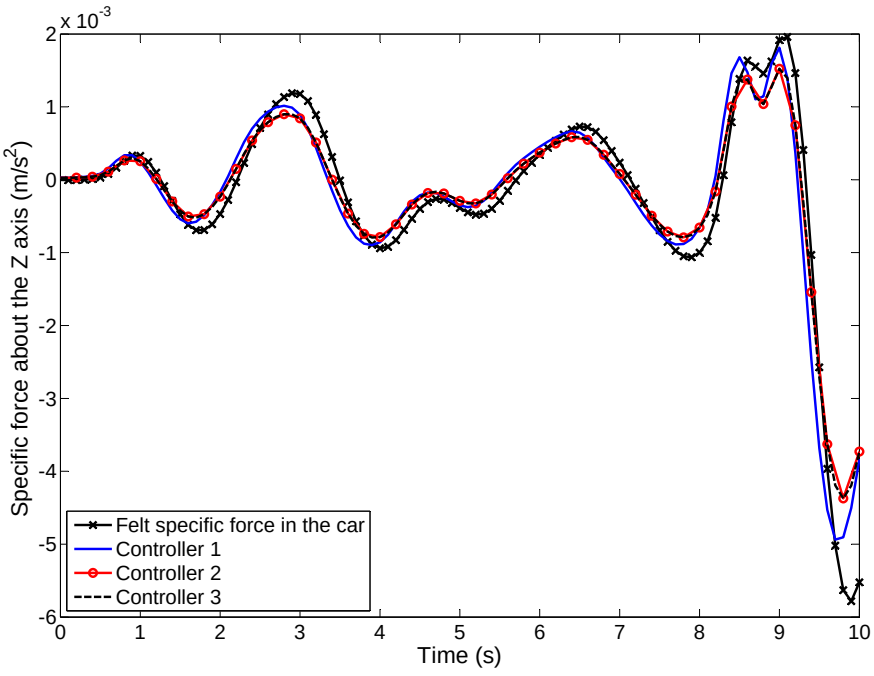

Figure 3.9: Felt specific force about the Z-axis

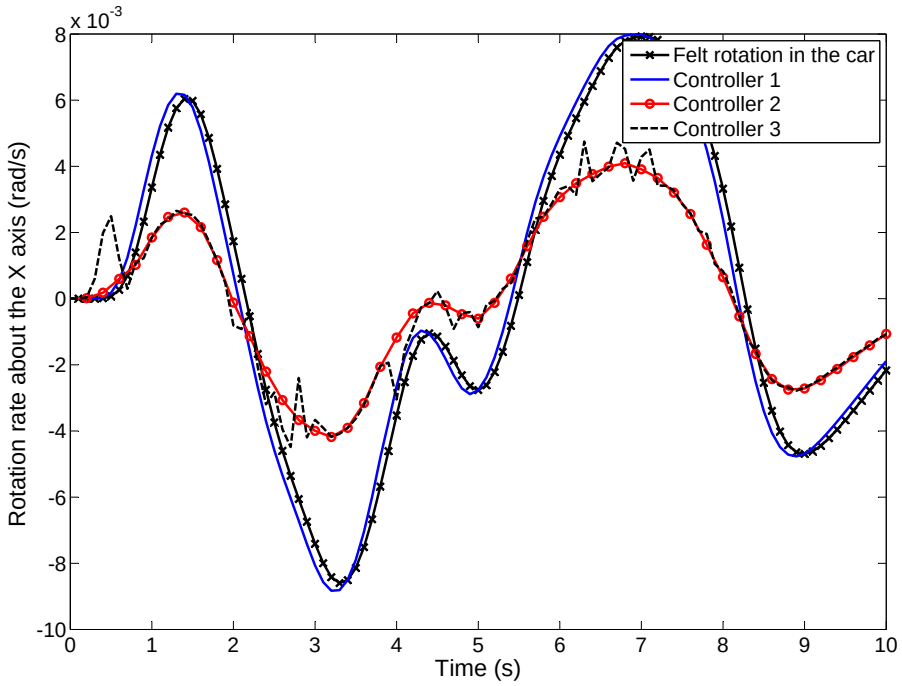

Figure 3.10: Felt rotation about the X-axis

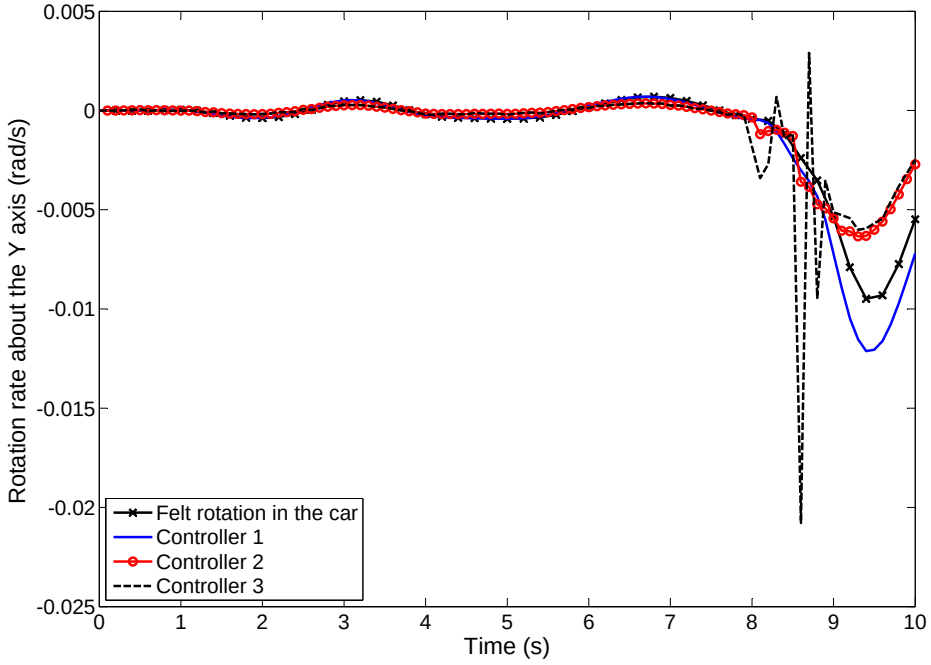

Figure 3.11: Felt rotation about the Y-axis

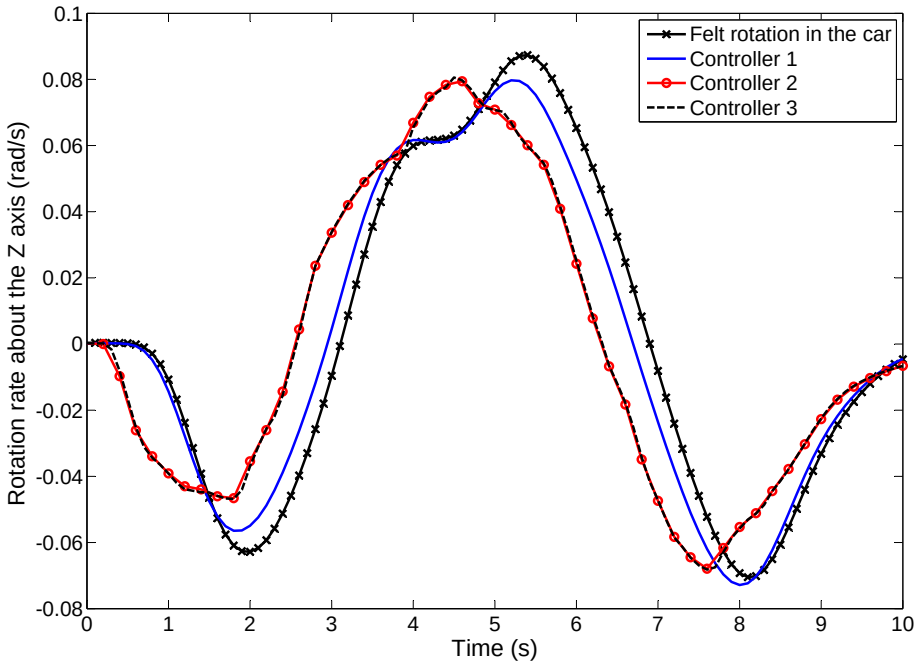

Figure 3.12: Felt rotation about the Z-axis

In what concerns the specific force about the X-axis, figure 3.7 shows that both controllers 1 and 2 behave in the same fashion but the second one has a lower tracking error. The remaining controller, 3, shows a different behavior, showing two major deviations in the results. This is possibly due to the generated references for the specific force about X-axis. The derivative of the bicycle model inputs is considered constant for the prediction horizon. Recalling that the derivatives are computed with a backwards method, this may be the reason behind the lack of quality of achieved results, since there could be very big gaps between reality and predictions.

Figure 3.8 represents the perceived specific force about the Y-axis. It is possible to observe a lower tracking error from controller 1. For the remaining two controllers, the signals evolution is very similar while the results appear somewhat shifted in the time axis. Given that it affects both controllers, 2 and 3, one can safely assume that the mentioned shifting is connected with the reference generation since it is the only feature common to both and not present in controller 1.

The sensed specific force about the Z-axis was well tracked by the three controllers, figure 3.8. There is a very small and very similar tracking error for all the different approaches.

Both the felt rotations about the X and Y axes present identical patterns. Controller 1 is the one with lower tracking error. The remaining two have similar behavior while controller 3 has bigger error fluctuations. Moreover, one can add that controllers 2 and 3 clearly track the reference variations, nevertheless there exists a gap in magnitude.

The felt rotation about the Z-axis can be commented like the specific force about the Y-axis. Controller 1 holds the smaller tracking error, while the other two have a matching behavior and what looks like a shift on the time axis.

Finally, based on the presented results, the controller chosen for implementation on the motion platform was number 1 because it originates the smaller overall tracking error, from all the three approaches.

3.6 Comparison of MPC with Washout Filter

Tests were done to compare the performance of controller 1 with the behavior of the washout filter. After reading [27] it was concluded that, from the controllers implemented in the driving simulator, the classical washout filter was the one with better performance. The testing scenario was once again the maneuver introduced in 3.1.

Even though the same driving scenario was used, there is a noticeable difference in the reference signals between these results and the ones from the previous section, 3.5. This is due to data filtration.

There is a considerable amount of noise on the signals coming from the motion platform. Thus, filters were created in order to counter its effects. The washout filters were developed taking in to account such filters and without them the results are not the anticipated ones. Then, to obtain a just comparison between the performance of both controllers, the data sent to the MPC needed to be filtered too, resulting in the difference observed in results mentioned in the previous paragraph.

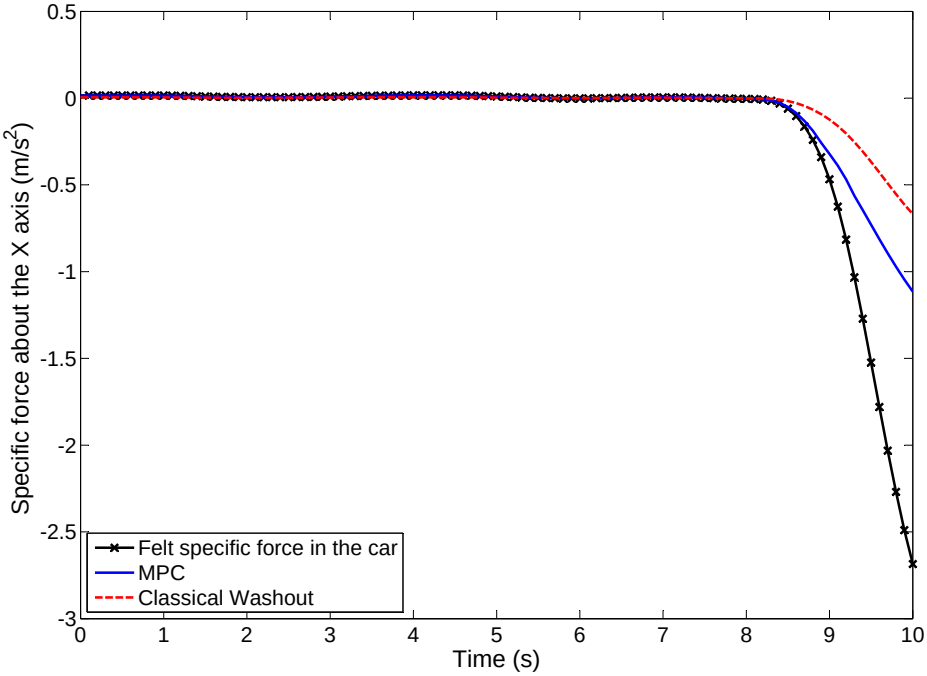

Figure 3.13: Specific force about the X-axis, MPC vs Washout

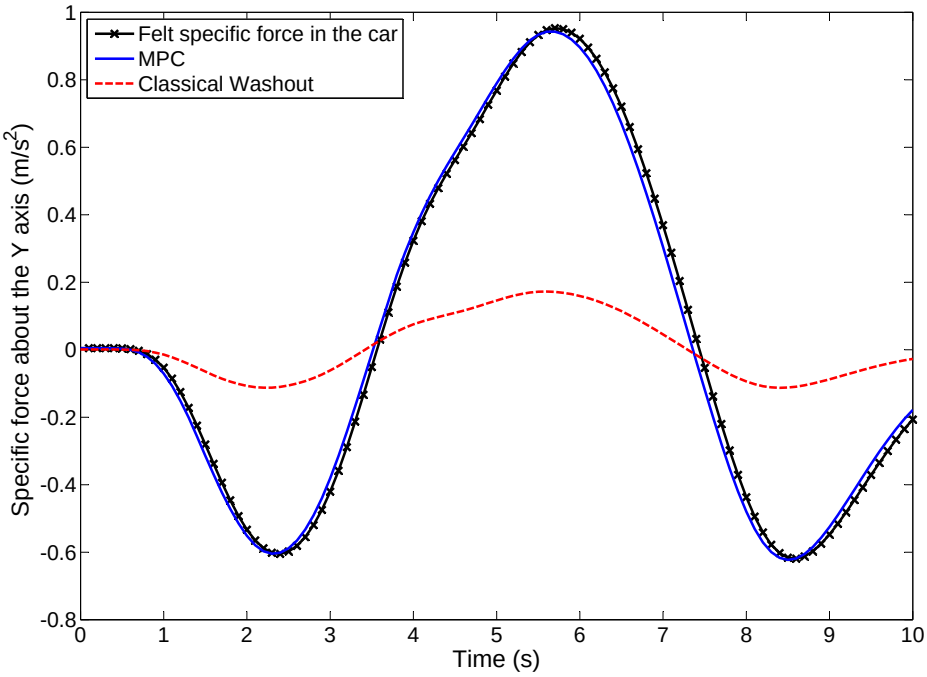

Figure 3.14: Specific force about the Y-axis, MPC vs Washout

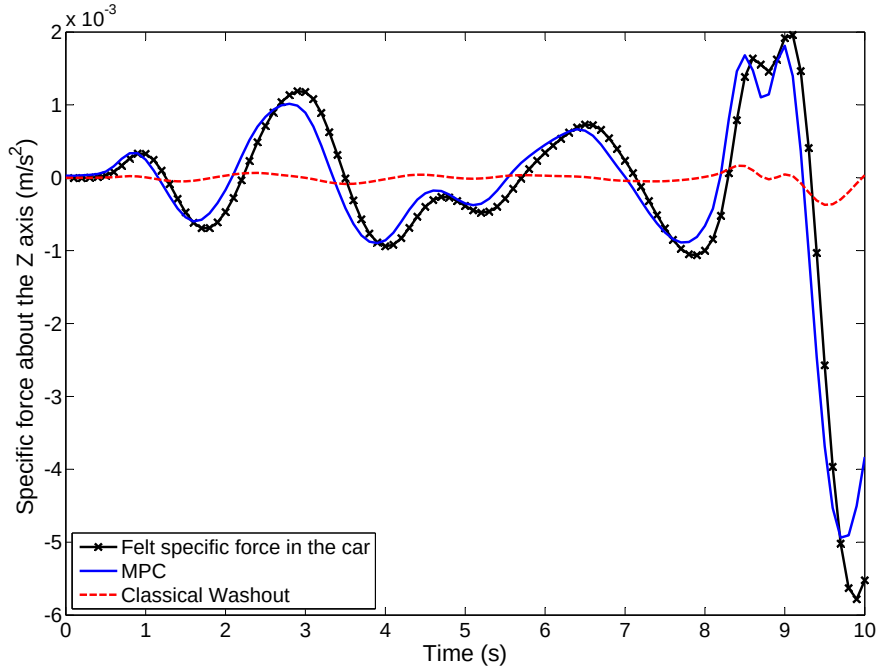

Figure 3.15: Specific force about the *Z*-axis, MPC vs Washout

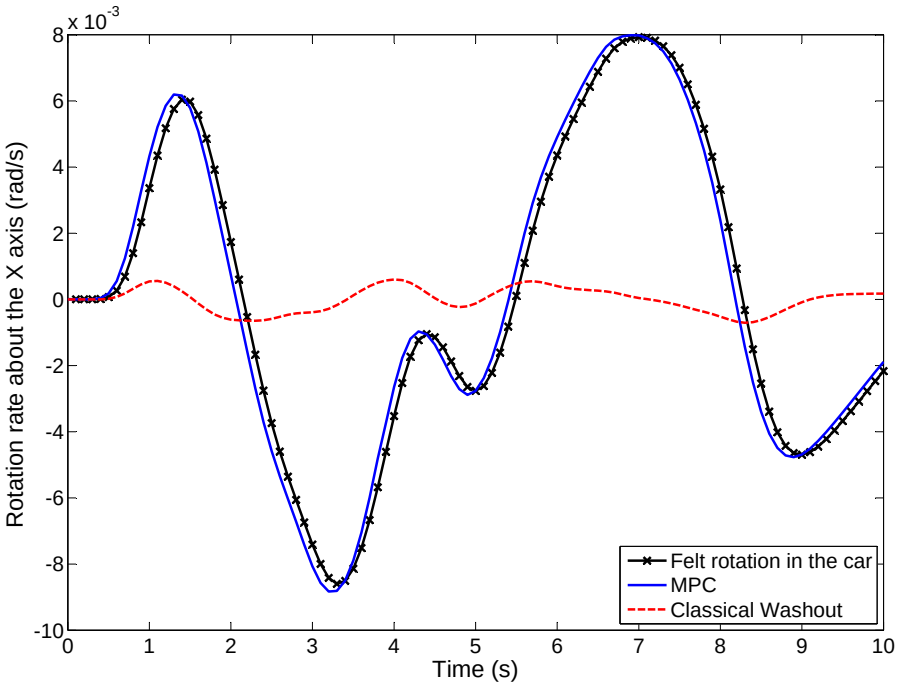

Figure 3.16: Felt rotation about the *X*-axis, MPC vs Washout

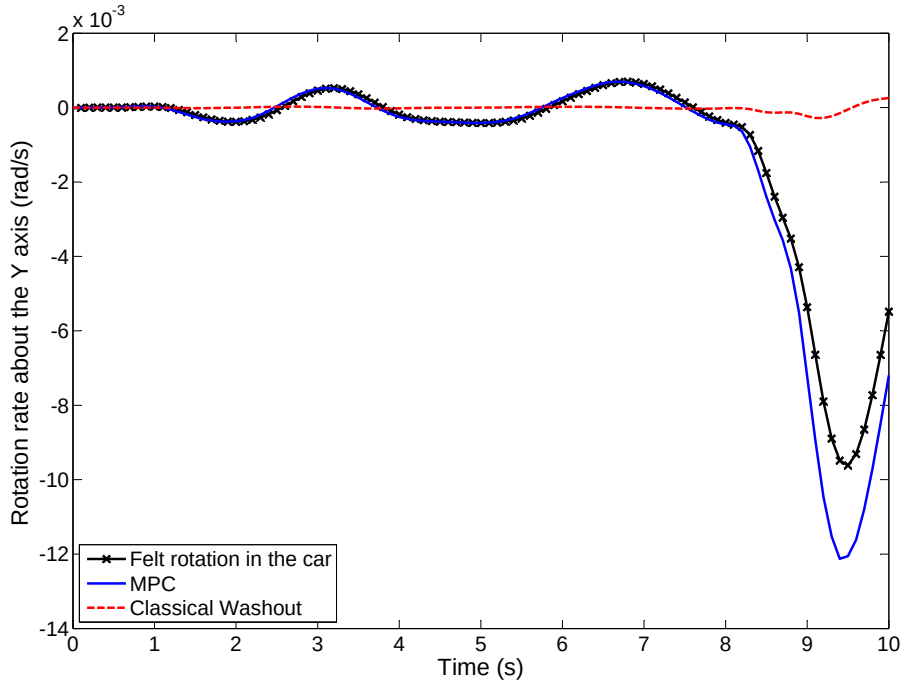

Figure 3.17: Felt rotation about the *Y*-axis, MPC vs Washout

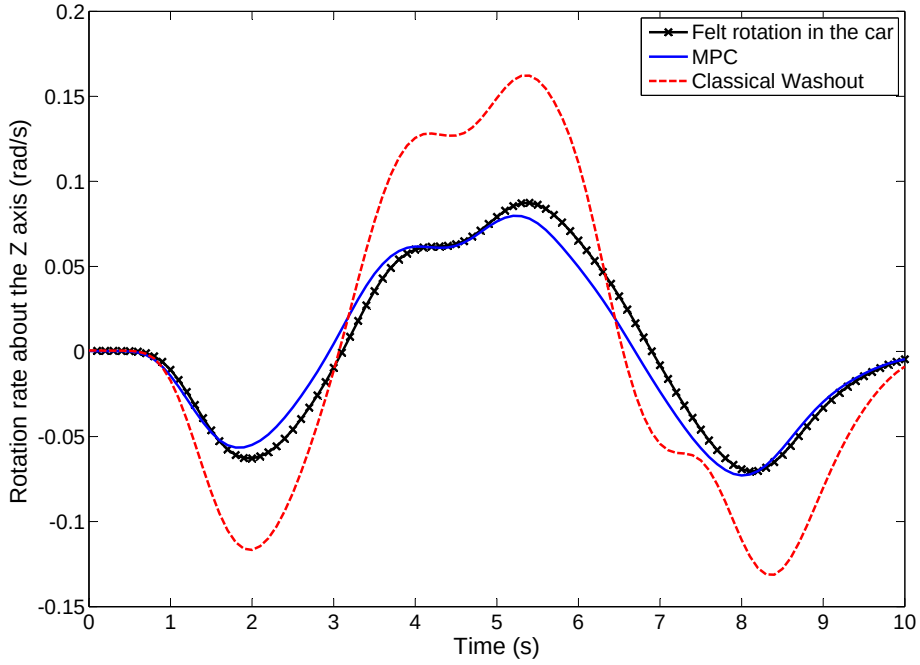

Figure 3.18: Felt rotation about the *Z*-axis, MPC vs Washout

A general remark, for the given set of figures, is that the MPC clearly follows the references better than the classical washout filter. The difference in results observed in figures 3.13 and 3.14 is most certainly due to the scale and limiting in the washout filter.

Figures 3.15, 3.16 and 3.17 have the most relevant differences between the washout filter results and the computed references. The washout filter results present a similar evolution to the one given by the reference signal, nevertheless, the signals magnitude do not match the referenced ones. This might be explained, once again, by the scale and limiting implemented in the washout filter. However, given the magnitude of the signals, the latter is not expected to interfere with the washout results.

Finally, figure 3.18 shows that both washout and MPC present analogous progression for the tracked variables. Contrary to the last three discussed cases, the washout output has a bigger magnitude than the one from the MPC. To the current date, no justification was found for these results.

In section A, the commands sent to the motion platform are presented. These are of interest because they allow a comparison between the utilization of the workspace. As expected, it is possible to see a bigger exploitation of the workspace by the MPC, except in figure A.6. This is a connected to the previously discussed problem of tracking the felt rotations about the Z-axis.

3.7 Driving simulator testing

The current section contains the results of a simulation done in the motion platform. First the implementation of the controller in the real time environment will be discussed followed by a presentation of the data extracted from the driving simulator.

3.7.1 Implementation of the controller in the CVS

The MPC was included in the existing Simulink model for motion cueing. The controller generated a control action every $0.05s$ and the motion commands were sent to the platform at the rate of $0.01s$, the minimal sending rate that ensures smooth displacements. The motion commands were assumed constant in the interval between two different MPC commands. This method resulted in displacements that were not smooth and the user could feel the difference between consecutive control inputs.

To solve this issue the hypothesis of sending different commands to the platform at every $0.01s$ was considered. The idea was to extract the derivatives of

the motion commands from the MPC at every $0.05s$. Then, for every $0.01s$ in between consecutive MPC commands, compute a different control action based on the available derivatives. This was first tested in offline simulation and only then implemented in the platform.

Section A.4 holds a comparison between the commands originating from the MPC and the ones actually sent to the motion platform. As an overall overview, one can say that when the accelerations, angular or translational, go through big variations there is a difference in the evolution of the two motion command signals. This was expected since the assumption of a constant first derivative will present big errors in situations where the second derivative has big variations, even if for short time periods.

Despite all, the latter implementation provides smooth displacements of the motion platform and it is better than the first approach.

This method should only be considered for normal driving situations. In harsh changing lane maneuvers, one can feel the platform changing its lateral and roll displacements severely. This is prone to cause a discontinuity in the perceived specific forces.

Big variations on the accelerations, lead to bigger gaps between the current MPC motion command and the previous command originating from the developed algorithm (consult figure A.20 to see these gaps). This requires higher velocities, from the platform, to comply with the consecutive control inputs. It is obvious that these velocities are not in accordance with the ones predicted by the MPC, thus creating the mentioned discontinuity in specific forces. Until the date of delivery of the current report no more solutions arose for this problem.

3.7.2 Results

The platform displacements were compared against its motion commands. Section A.3 shows these results. All the displayed plots confirm that the low level controller, installed in the motion platform, has a very good performance, because the error between the presented signals is very low.

The last statement validates the following assumption: the predicted perceived signals by the MPC are an almost perfect match to the ones generated by the platform displacements.

This is very convenient because there is not the possibility of extracting the platform accelerations and angular rates. If necessary, these signals would have to be numerically differentiated from the extracted displacements. Also, filtering would have been required to eliminate the noise. These two processes would distort the actual signal derivative and they were avoided thanks to the conclusion drawn in the previous paragraph.

Based on these conclusions the felt signals in the platform will be represented by the felt signals predicted by the MPC.

The shown plots represent the evolution of the perceived signals against the reference signals from vehicle car model. Moreover, the classical washout filter results are shown. These will not be addressed (check section 3.6) and are only presented to recall the relationship between the two controllers.

The testing maneuver is described in section 3.1.

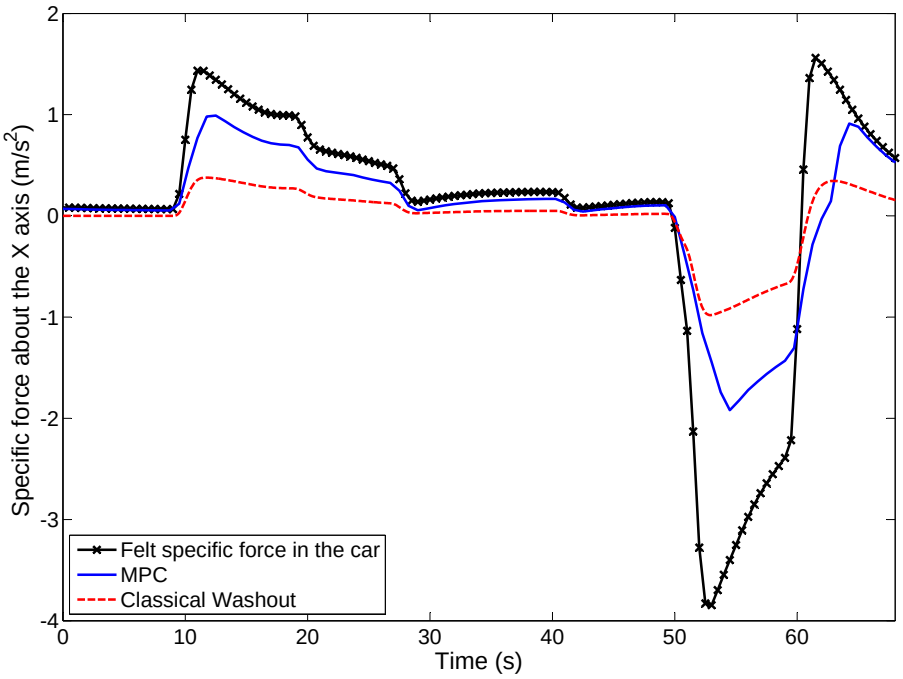

Figure 3.19: Felt specifc force about the X-axis

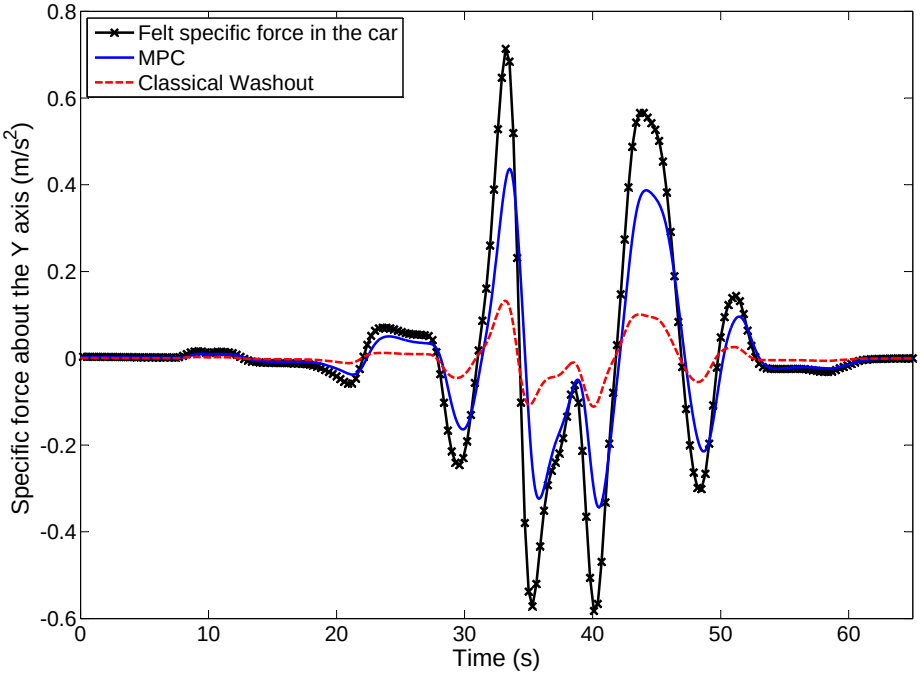

Figure 3.20: Felt specific force about the Y-axis

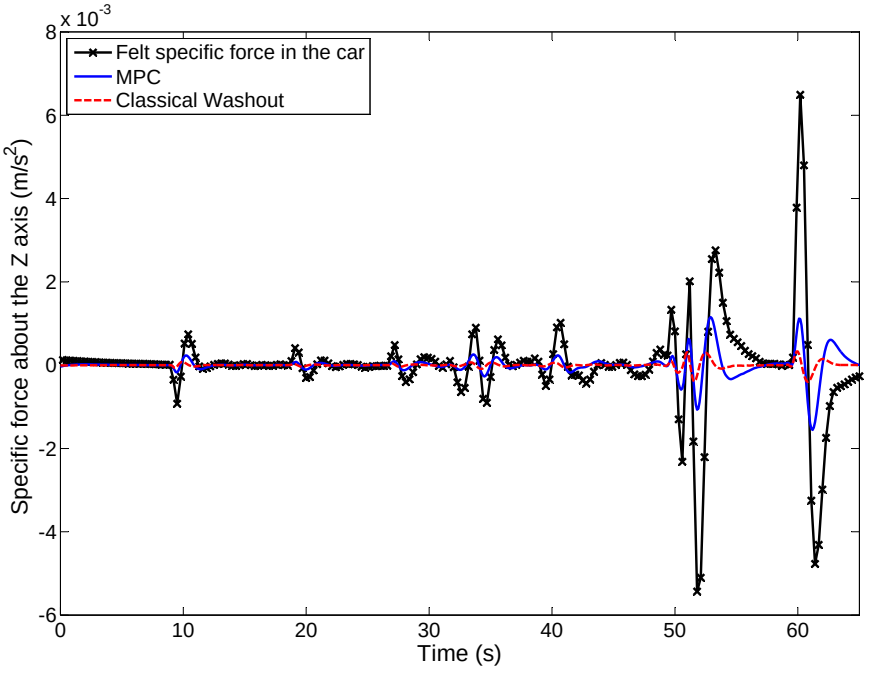

Figure 3.21: Felt specific force about the Z-axis

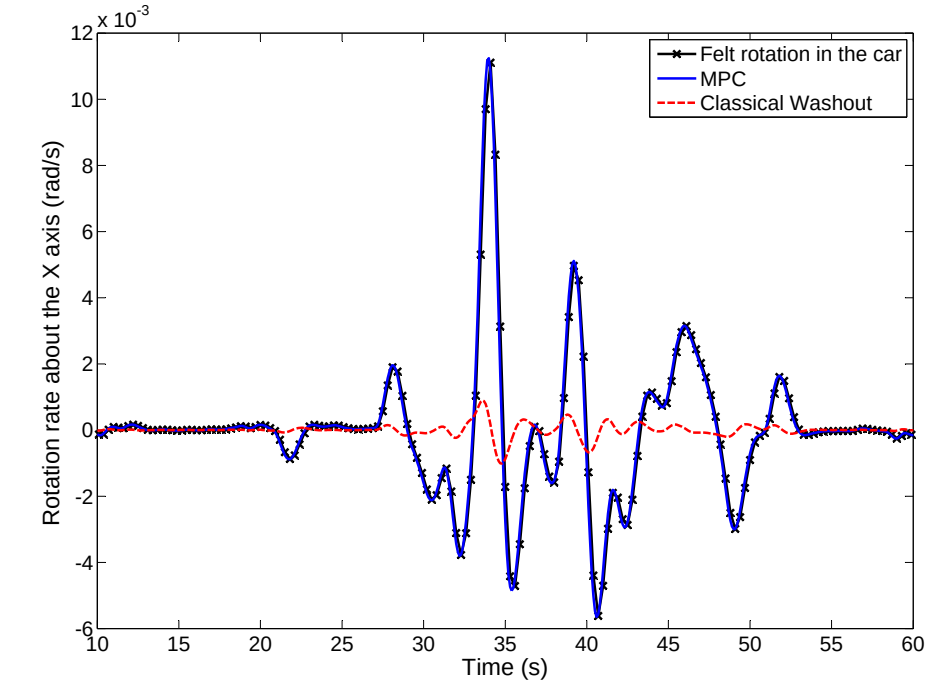

Figure 3.22: Felt rotation about the X-axis

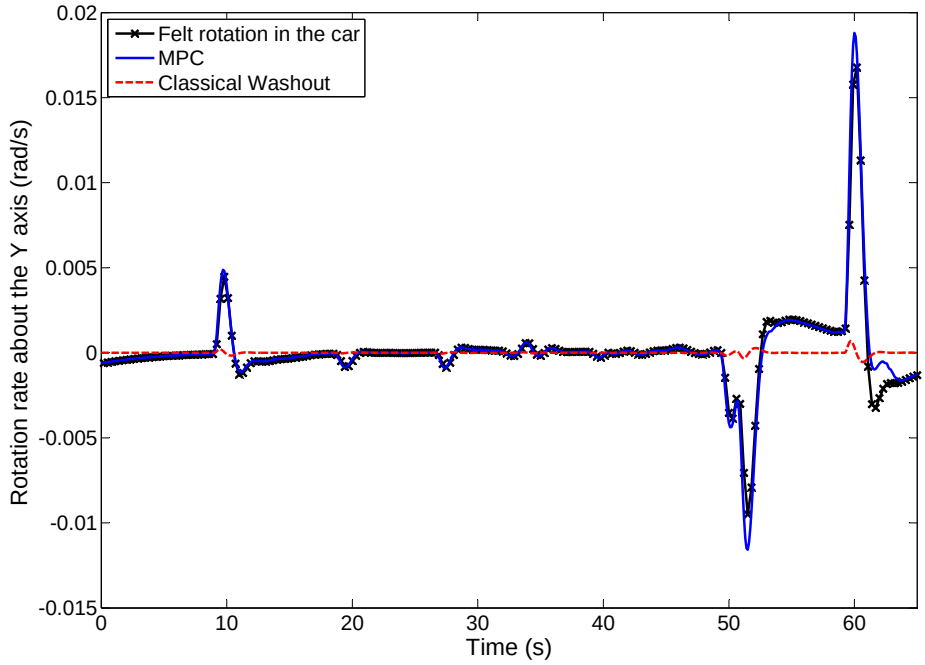

Figure 3.23: Felt rotation about the Y-axis

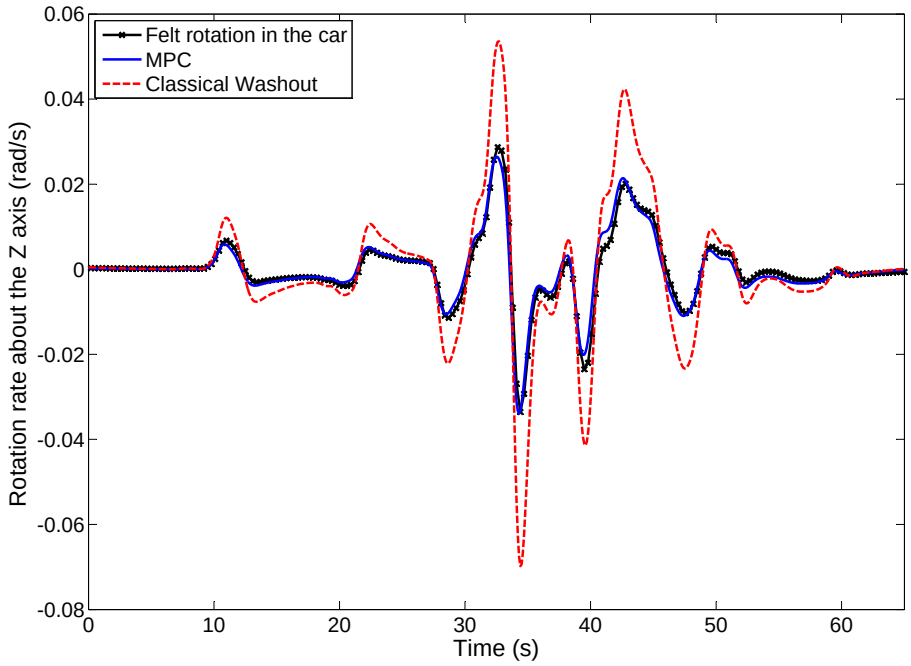

Figure 3.24: Felt rotation about the Z-axis

Analyzing the evolution of the specific force about the X-axis, figure 3.19, one can conclude that the controller performs a good tracking. The difference in amplitudes is due to the weight applied on the evolution of this variable. It is small to avoid the platform from going against the boundaries causing a discontinuity in the tracking task.

In what concerns the specific force about the Y-axis, figure 3.20, similar remarks can be made. Furthermore, it is possible to see that the tracking error for the offline simulation was lower. This is due to the application of a smaller weight on this tracking variable (consult sections A.5 and A.6).

According to driver remarks the perceived specific force about the Y-axis was to irrealistic. To reduce the reproduction of this signal, its weight on the objective function was lowered resulting in a bigger tracking error.

The controller presents a good tracking of the reference for felt specific force about the Z-axis. Nevertheless, like in the previous case, the tracking error is bigger than the one achieved with the offline simulations. This is due to the same problem affecting the specific force about the Y-axis; irrealistic results led to a smaller weight which is responsible for a bigger tracking error.

In what concerns perceived rotations, it is possible to conclude that the con-

troller behaves very well when it comes to sensed angular rates about the X and Y axes, figures 3.22 and 3.23 respectively. There is a very small tracking error, nevertheless one should pay attention to the small amplitude of the signals which are, most likely, not felt by the driver.

The sensed rotation about the Z-axis is given in figure 3.24. It has a bigger tracking error when compared to the remaining felt rotations, but the overall performance is good.

In section A.2, a comparison between the motion commands computed by both MPC and washout filter is presented. Like in the offline tests, the MPC shows a better exploitation of the motion platform displacements. However one should refer that the position about the Y-axis has smaller displacements on the MPC case. This happens because the controller has a lower priority on tracking the specific force about that axis, thus using less displacement.

In what concerns the Yaw case, higher displacements by the washout filter are connect with its behavior when tracking the perceived rotations about the Z-axis, figure 3.24. These are not according to the reference or the expected outcome of the washout filter, thus no comparison can be performed.

Furthermore, there is one interesting result to be highlighted in section A.2. Analyzing figure A.7 one can see that the MPC is operating very close to the boundaries. As expected, the constraints were respected.

Chapter 4

Conclusion

This thesis proposed a new motion cueing approach, the MPC. It was motivated by the possibility to enforce the displacements constraints of the platform, while providing optimal motion commands.

The control goal was to reproduce sensed specific forces and rotation rates, in the driving simulator. This reason prompt the study of the vestibular system that acted as the MPC plant.

Furthermore, three different methods were considered for reference generation. Two of them used a simplified vehicle model, bicycle model, for prediction of the vehicle future behavior. By application of the bicycle model, the reference generation methods were expected to compute more reliable information, in what concerns future vehicle states.

The methods for reference generation were tested. The one that resulted in a controller with better performance, was compared with the conventional washout strategies.

Finally, the controller was implemented in the Chalmers Vehicle simulator and its application evaluated.

4.1 Summary of Results

The bicycle model was used for reference generation. It was defined according to section 2.2.3 and its parameters were tuned by genetic algorithms. As it is possible to conclude from the figures in section 3.3, the tuned parameters establish the bicycle model as a good estimator for the vehicle behavior.

References for the MPC were generated in three ways, section 2.2. The three given methods defined three different controllers. These controllers were tested and their performance judged in terms of tracking capabilities. Contrary to what was initially expected, controllers working with the bicycle model, did not have

better results than the controller who did not. The figures in section 3.5, show that controller one (not using references generated by the bicycle model) has smaller or similar tracking error when compared to the remaining controllers. Particularly one should mention the results obtained with controller three, because of the large variations in the tracking errors. This may be justified by the way in which the bicycle model inputs derivative is computed, a backward approach.

The best MPC , controller one, was compared against the classical washout filter. Studying the attained results, section 3.6, was possible to conclude that the developed controller has smaller tracking errors than the washout filter.

Appendix A.1, shows the motion commands generated by washout filter and the MPC. One can understand that all the given plots (except Yaw displacement) show a better exploitation of the motion platform capabilities by the MPC, as expected.

The low level controller, in the motion platform, has a very good performance. All motion commands by the MPC were reproduced almost flawlessly by the latter. Section A.3 illustrates the drawn conclusion.

The implementation of the controller in the platform showed that tuning the weights can not be carried out while only considering the computed tracking error. In an offline simulation, what appears to be a good tracking action, is considered to harsh by the driver in the platform, section 3.7. This was reported for the specific forces about the Y and Z axes.

Due to the way in which motion commands are sent to the platform, section 3.7.1, it was shown that the current implementation should be avoided in rough driving conditions. As explained earlier big variations in translational acceleration will cause discontinuities on the felt perceived forces.

Through the performed tests and given results, the developed controller proved to be a good alternative for the implemented washout filter, nevertheless, there is still work to be done.

4.2 Future Work

The noise in the data, originating from the vehicle model, should be addressed. Its presence requires filtering before any kind of data handling. The filtering process not only removes the noise but also affects the properties of the filtered signal and adds delay to the motion commands. To solve this issue, a new car model could be developed for the CVS.

Feedback of the platform states could be used to improve the MPC performance. For instance, if a rotation and acceleration sensor is placed at the driver head, real sensed specific forces and rotations could be computed. This would provide the MPC with more reliable states of the vestibular system, which would reflect in a better tracking performance.

The current thesis showed that the MPC can be a good alternative to the motion cueing algorithms installed in the CVS. This can be improved by utilization of different and more recent vestibular models, like the ones given in [35]. The used vestibular model was only applied in order to perform a even comparison with the washout filter, since the latter was tuned with the same model.

The washout function in the developed MPC is performed thanks to tracking of the positions and angular displacements to zero. These variables need to be weighted and the washout is triggered when the errors in the tracking perceived signals are small. This feature has its disadvantages because it penalizes the usage of displacements even when they are necessary. Also, it is possible to ascertain, specially in the specific forces about the X and Y axes, that aggressive weights will lead the platform against its constraints very fast, disturbing the tracking task. This implies that "worst case scenario" weight tuning is required, compromising the exploitation of the platform workspace.

Both problems in the last paragraph could be addressed by a similar solution. The weights in all the referred variables could be adapted according to the situations. For instance, considering the first problem, when the accelerations remain constant for long time periods, the weight in the position of the motion platform frame could increase. In the second problem, the weights increase as the specific force decreases and vice versa.

This is a very simple way to describe these solutions. Most probably they would be implemented AS optimization problems, functions of the current vestibular states and inputs.

Stability analyzes were not performed for the controller developed in this thesis. It could be interesting to address this issue in future works.

The method used to send motion commands to the platform should be revised. The current implementation prevents users from doing harsh maneuvering, section 3.7.1.

For example, a study could be done to determine the possibility of using a more powerful processor in the real time computer. This study would focus on the implementation of a controller with smaller sampling time, as close to the platform receiving rate as possible.

In the developed thesis, the boundaries of the motion platform displacements

were extracted from the user manual. These consisted on a set of limitations for the degrees of freedom based in combined motion scenarios. Was one can understand, the referred values are computed in a conservative fashion, and do not match with the boundaries for single D.O.F. displacement. This fact limits the utilization of the platform, compromising an optimal usage of the available workspace.

To fix this issue, an algorithm could be built that computes the real constraints in real time. This algorithm would make use of the platform model and, given the current actuators length, update the motion boundaries for the MPC.

Another interesting proposal of future work is connected with the vestibular system itself. The idea is to implement the nonlinear model as the MPC plant. The purpose of such an approach would be to improve the driving experience by a better exploitation of the motion hardware capabilities.

Using the nonlinear model as the plant, the MPC would be aware of the human motion thresholds. Using these, the controller would know how to displace the platform without any kind of perceived signal by the driver. This is useful because the control approach would be able to displace the platform to a set of D.O.F. that enable a better dynamical reproduction of the car dynamics; all without generation of false cues.

Bibliography

[1] Felipe Jiménez Alonso. Experimental analysis of vehicle dynamics. validation of a novel vehicle mathematical model using advanced instrumentation. Technical report, INSIA - Polytecnic University of Madrid (Spain), 2005.

[2] Pieter Van Balen. Development of washout filters for the chalmers vehicle simulator. Master's thesis, Chalmers University of Technology, 2005.

[3] E.F Camancho and C. Bordons. *Model Predictive Control*. Springer, 2006.

[4] F. Colombet, M. Dagdelen, G. Reymond, C. Pere, F. Merienne, and A. Kemeny. Motion cueing: what's the impact on the driver's behaviour? Technical report, 2008.

[5] Mehmet Dagdelen, Gilles Reymond, Andras Kemeny, Marc Bordier, and NadiaMaýzi. Model-based predictive motion cueing strategy for vehicle driving simulators. *Control Engineering Practice*, 2009.

[6] Brian L. Day and Richard C. Fitzpatrick. The vestibular system. *Current Biology*, Vol.15(No.15):583–586, 2005.

[7] Niclas Andrésso Anton Evgrafov and Michael Patriksson. *An Introduction to Continuous Optimization*. Studentlitteratur, 2005.

[8] Paolo Falcone. *Nonlinear Model Predictive Control for Autonomous Vehicles*. PhD thesis, University of Sannio, Benevento, 2007.

[9] Rolf Findeisen and Frank Allgöwer. An introduction to nonlinear model predictive control. Technical report, Institute for Systems Theory in Engineering, University of Stuttgart, 2002.

[10] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms & Engineering Design*. John Wiley & Sons, Inc, 1997.

[11] P. Gill, S.J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright. User's guide for lssol (version 1.0): A fortran package for constrained linear least-squares and convex quadratic programming. Technical report, 1986.

[12] Myung-Chul Han, Hyung-Sang Lee, Suk Lee, and Man Hyung Lee. Optimal motion cueing algorithm using the human body model. *JSME International Journal*, Vol.45(No. 2):487–491, 2002.

[13] Randy L. Haupt and Sue Ellen Haupt. *Practical Genetic Algorithms*. John Wiley & Sons, Inc, 2004.

[14] Alice F. Heally and Robest W. Proctor. *Handbook of Psychology, Volume four: Experimental Psychology*. John Wiley & Sons, Inc.

[15] Ruud Hosman, Sunjoo Advani, and Nils Haeck. Integrated design of flight simulator motion cueing systems. pages 1–12, London, May 2002. Royal Aeronautic Society.

[16] Chin-I Huang and Li-Chen Fu. Human vestibular based (hvb) senseless maneuver optimal washout filter design for vr-based motion simulator. pages 4451–4458, Taipei, Taiwan, October 2006. IEEE International Conference on Systems, Man, and Cybernetics.

[17] I.Siegler, G.Reymond, A.Kemeny, and A.Berthoz. Sensorimotor integration in a driving simulator: contributions of motion cueing in elementary driving tasks. *???*, ???(???), September 2001.

[18] Dean Karnopp. *Enginnereing Applications of dynamics*. Hohn Wiley and Sons, Inc, 2004.

[19] Dean Karnopp. *Vehicle stability*. Marcel Dekker, 2004.

[20] Andras Kemeny and Francesco Panerai. Evaluating perception in driving simulation experiments. *TRENDS in Cognitive Sciences*, Vol. 7(No. 1):31–37, January 2003.

[21] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2000.

[22] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P.O.M. Scokaert. Constrained model predicitive control: Stability and optimality. *Automatica, 36*, pages 789–814, 2000.

[23] Jacob L. Meiry. *The Vestibular System and Human Dynamic Space Orientation*. PhD thesis, Massachusetts Institute of Technology, 1965.

[24] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[25] M.L.Saidi, A. Debbeh, H.Arioui, M.S. Kermiche, and H.A. Abbassi. Predictive control of motion platform in driving simulator. *Asian journal of information technology*, Vol.5(No.2):133–138, 2006.

[26] Manfred Morari and Jay H. Lee. Model predictive control: Past, present and future. *Computers and Chemical Engineering*, 23:667–682, 1997.

[27] Nikolce Murgovski. Vehicle modelling and washout filter tuning for the chalmers vehicle simulator. Master's thesis, Chalmers University of Technology, Goteborg - Sweden, 2007.

[28] L. Nehaoua, H. Arioui, S. Espie, and H. Mohellebi. Motion cueing algorithms for small driving simulator. Orlando, Florida, May 2006. IEEE International Conference on Robotics and Automation.

[29] Lamri Nehaoua, Hakim Mohellebi, Ali Amouri, Hichem Arioui, Stéphane Espié, and Abderrahmane Kheddar. Design and control of a small - clearance driving simulator. *IEEE Transactions on Vehicular Technology*, Vol. 57(No. 2):736–746, March 2008.

[30] S.Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, pages 733–764, 2003.

[31] L. D. Reid and M.A.Nahon. Fligth simulation motion-base drive algorithms: Part 1 - developing and testing the equations. Technical Report ,UTIAS No.296, University of Toronto, 1985.

[32] Awanand Sahasrabuddhe. Insights into implementing genetic algorithm based production schedulers. *Infosys Website*, 2008.

[33] Raphael Sivan, Jehuda Ish-Shalom, and Jen-Kuang Huang. An optimal control approach to the design of moving flight simulators. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-12(No. 6):818–827, November/December 1982.

[34] Jae-Bok Song, Ui-Jung Jung, and Hee-Dong Ko. Washout algorithm with fuzzy-based tuning for a motion simulator. *KSME International Journal*, Vol. 17(No. 2):221–229.

[35] Robert J. Telban and Frank M. Cardullo. Motion cueing algorithm development: Human-centered linear and nonlinear approaches. Technical Report CR-2005-213747, Nasa, May 2005.

[36] Robert J. Telban, Frank M. Cardullo, and Lon C. Kelly. Motion cueing algorithm development: Piloted performance testing of the cueing algorithms. Technical Report CR-2005-213748, Nasa, May 2005.

[37] Andrey Alexandrovich Tyagunov. *High-Performance Model Predictive Control for Process Industry*. PhD thesis, Technische Universiteit Eindhoven, 2004.

[38] http://en.wikipedia.org/wiki/Charles_Darwin#cite_note 0, 2009.

[39] http://en.wikipedia.org/wiki/Vestibular_system , 2009.

[40] http://www.britannica.com/EBchecked/topic/434748, 2009.

[41] http://www.causeof.org/neuro_vest.html, 2009.

[42] http://www.pacontrol.com/MPC.html, 2009.

[43] www.unmc.edu/physiology/Mann/mann9.html, 2009.

[44] Su-Chiun Wang and Li-Chen Fu. Predictive washout filter design for vr-based motion simulator. pages 6291–6295. IEEE International Conference on Systems, Man and Cybernetics, 2004.

Appendix A

Results

A.1 Commands for the Motion Platform: MPC vs Washout - offline simulations

This section presents the comparisons between the commands sent to the platform by the washout filter and the MPC. Information about these figures can be found in section 3.6.

Notice that these values are all constrained. Upper and lower bounds are only shown if the signals are close to the latter. The boundaries can be found in B.1.

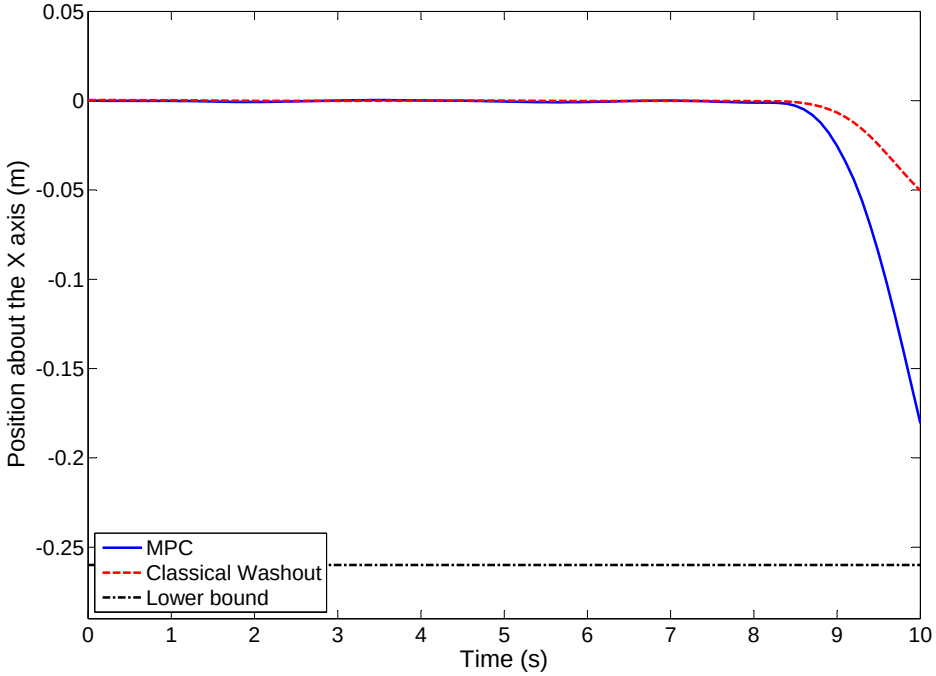

Figure A.1: Position about the X-axis

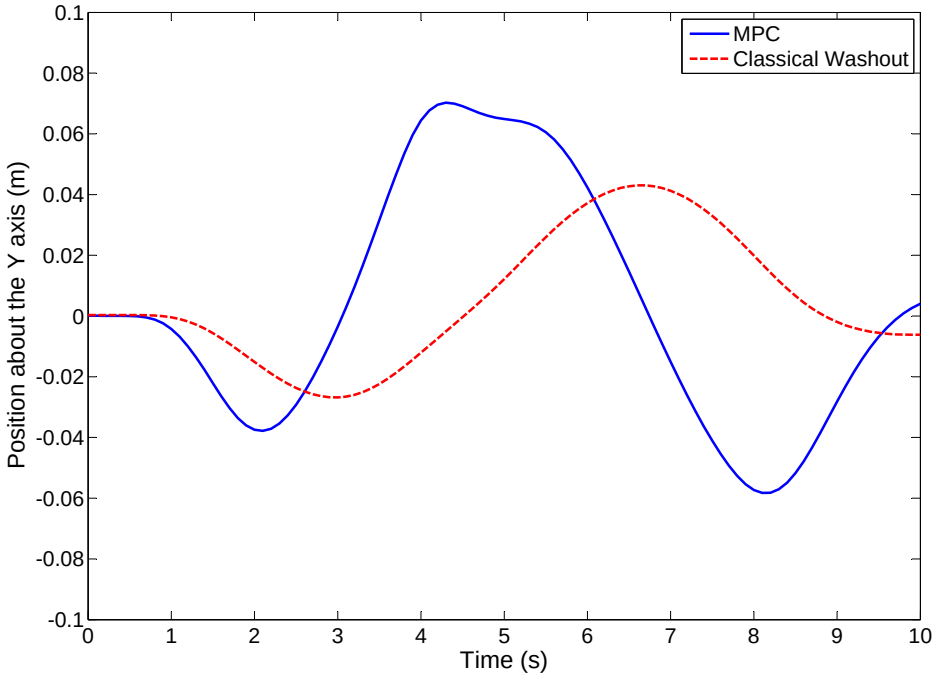

Figure A.2: Position about the Y-axis

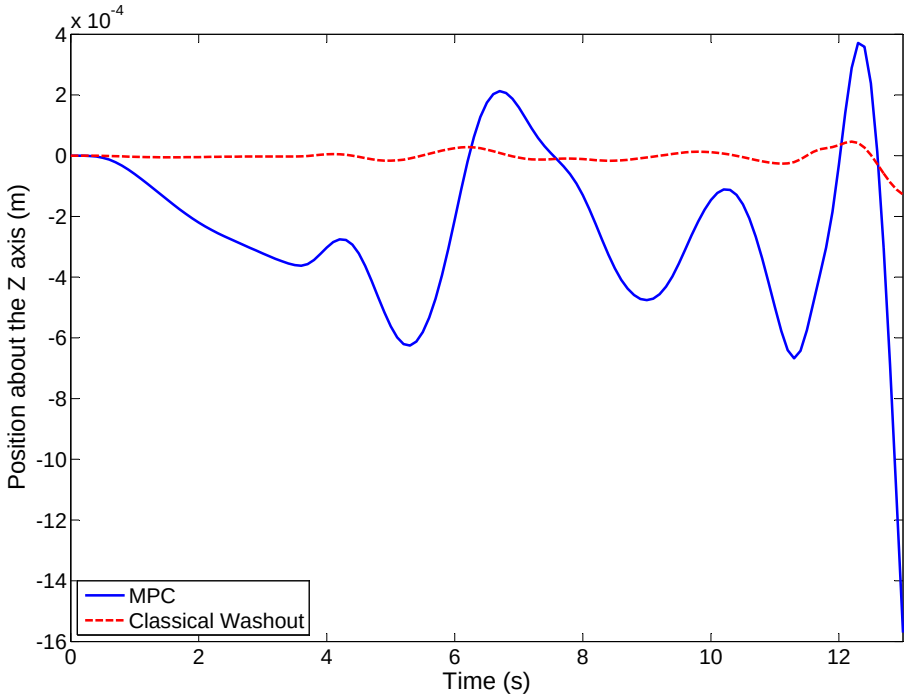

Figure A.3: Position about the Z-axis

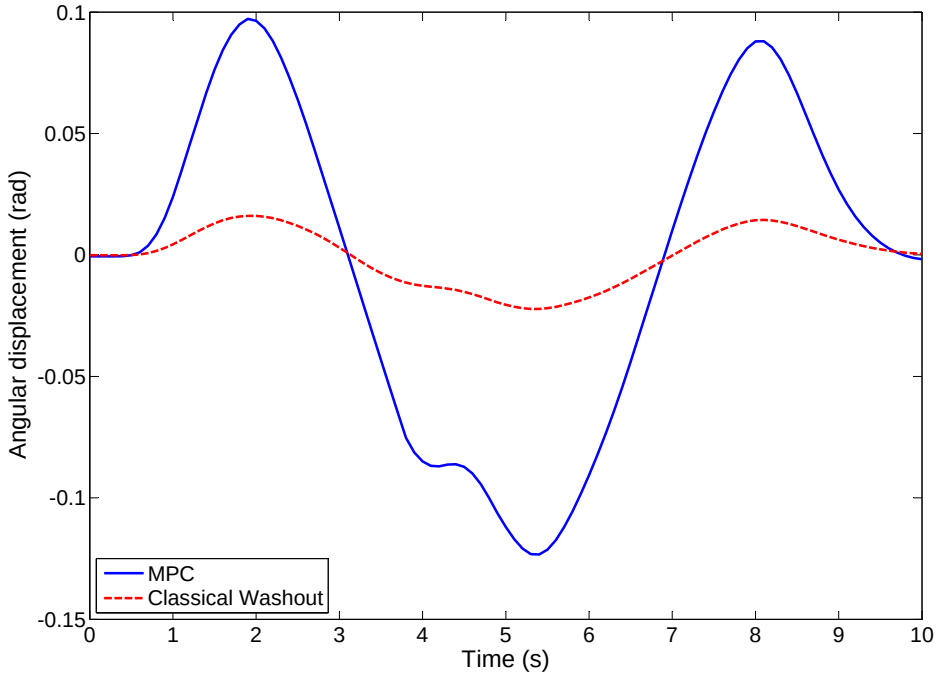

Figure A.4: Angular displacement - Roll

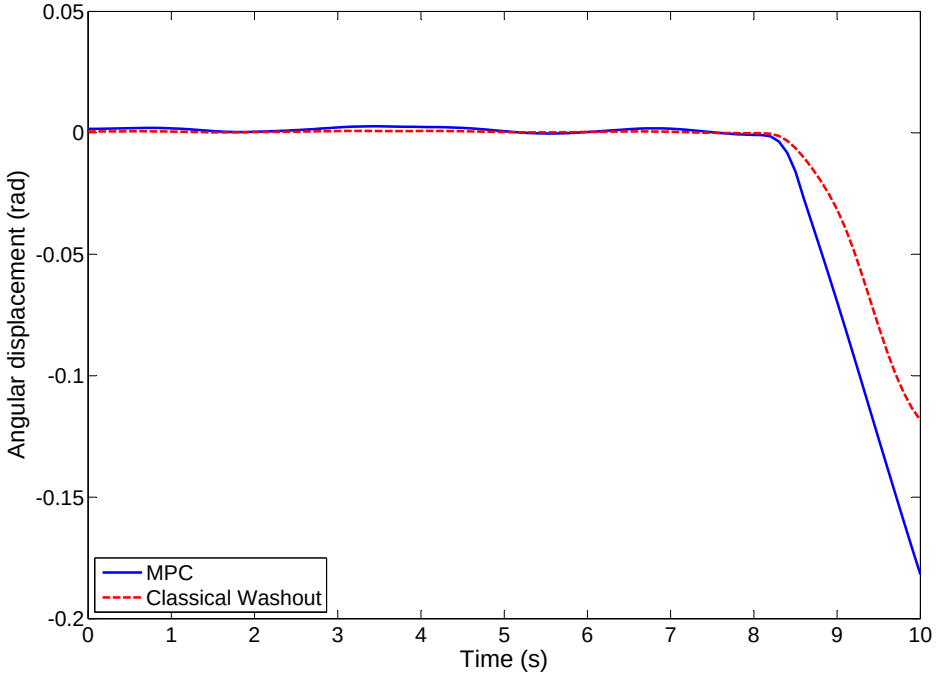

Figure A.5: Angular displacement - Pitch

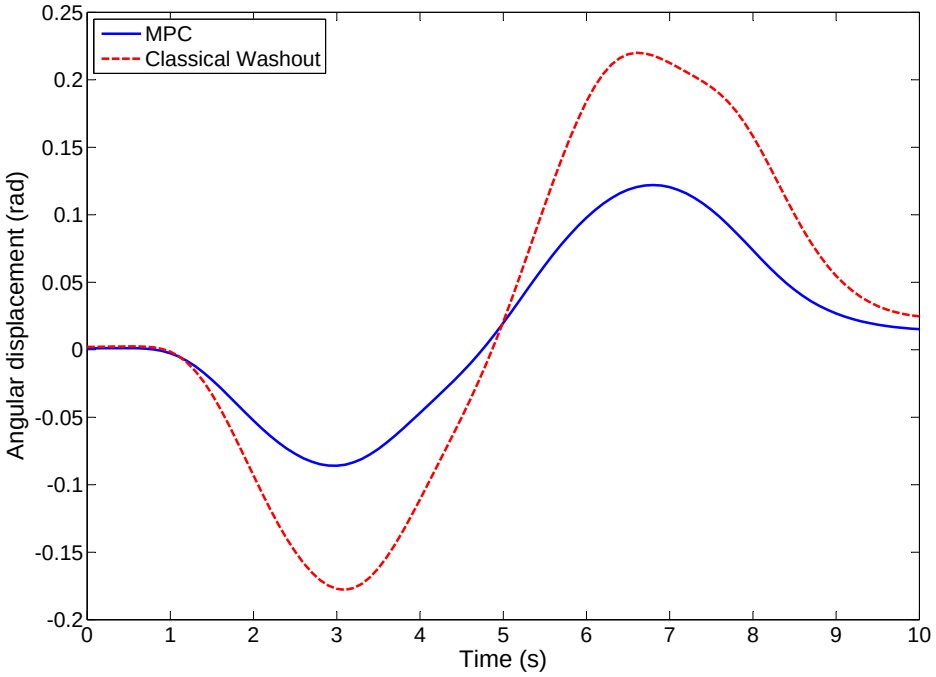

Figure A.6: Angular displacement - Yaw

A.2 Commands for the Motion Platform: MPC vs Washout - online simulations

This section presents the comparisons between the commands sent to the platform by the washout filter and the MPC. Information about these figures can be found in section 3.7.

Notice that these values are all constrained. Upper and lower bounds are only shown if the signals are close to the latter. The boundaries can be found in B.1.

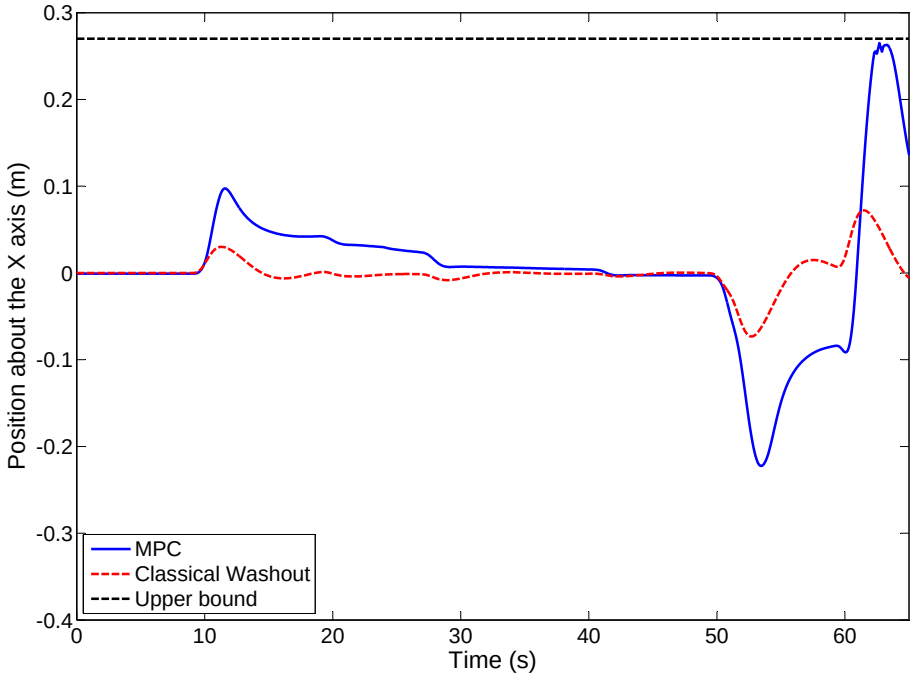

Figure A.7: Position about the X-axis

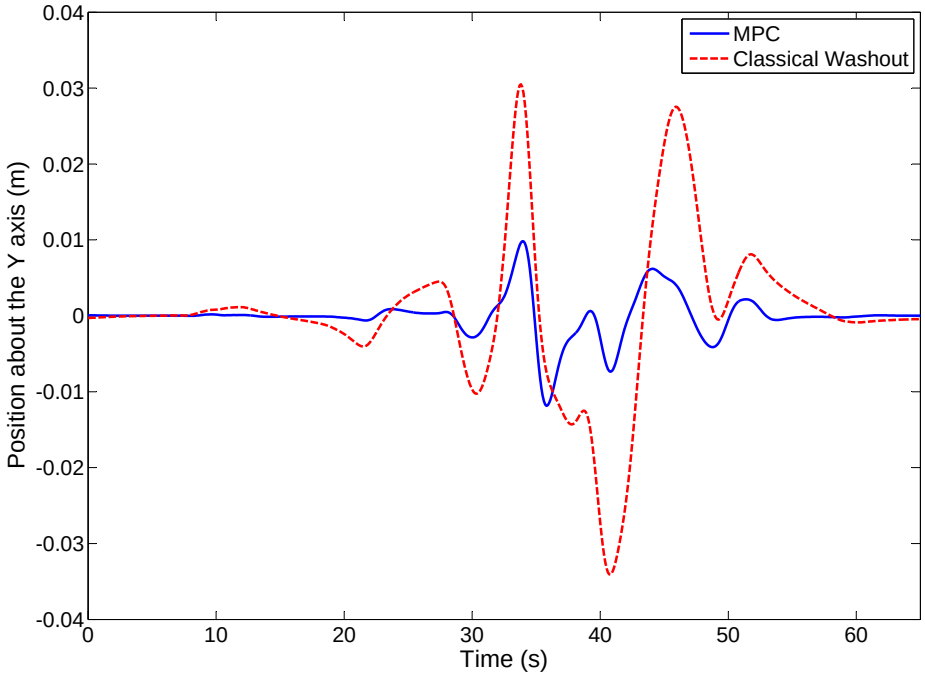

Figure A.8: Position about the Y-axis

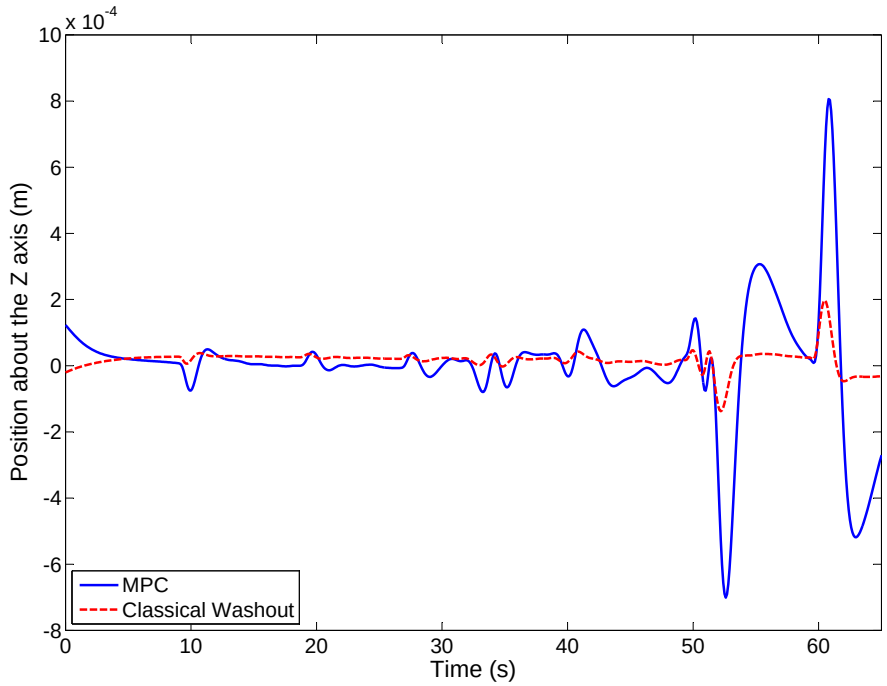

Figure A.9: Position about the Z-axis

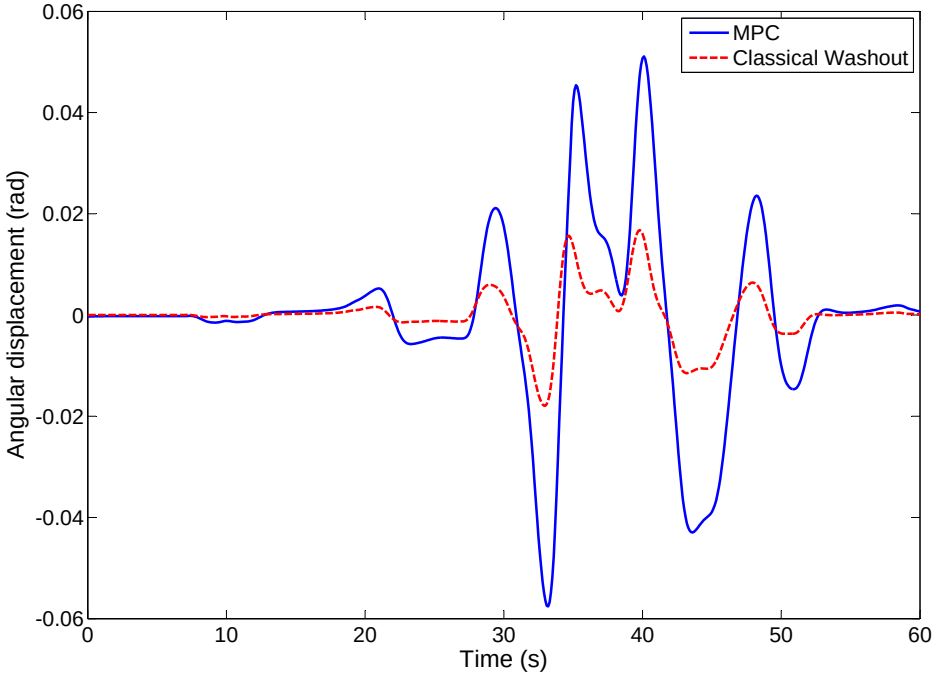

Figure A.10: Angular displacement - Roll

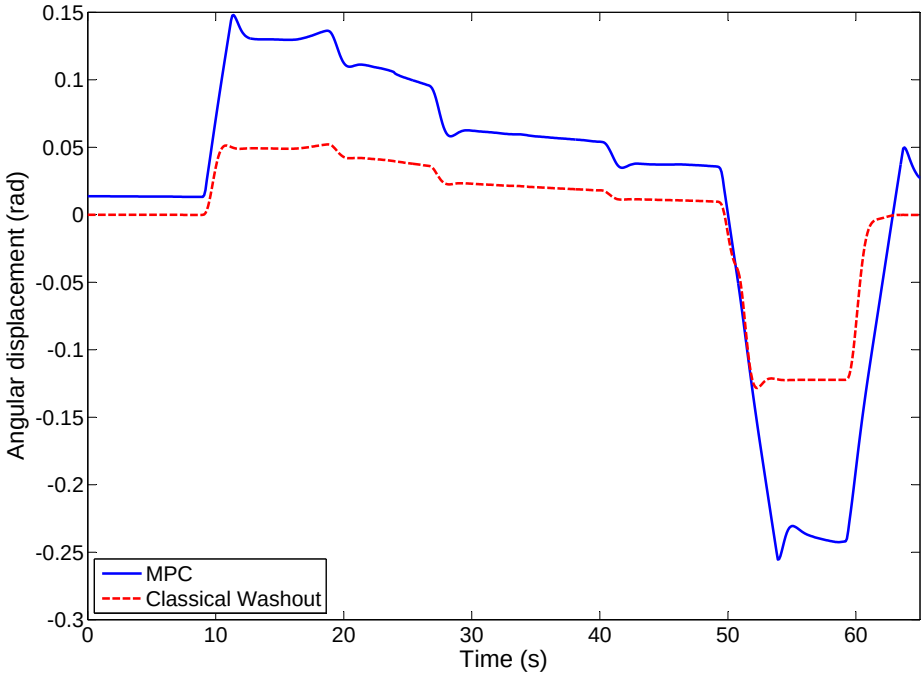

Figure A.11: Angular displacement - Pitch

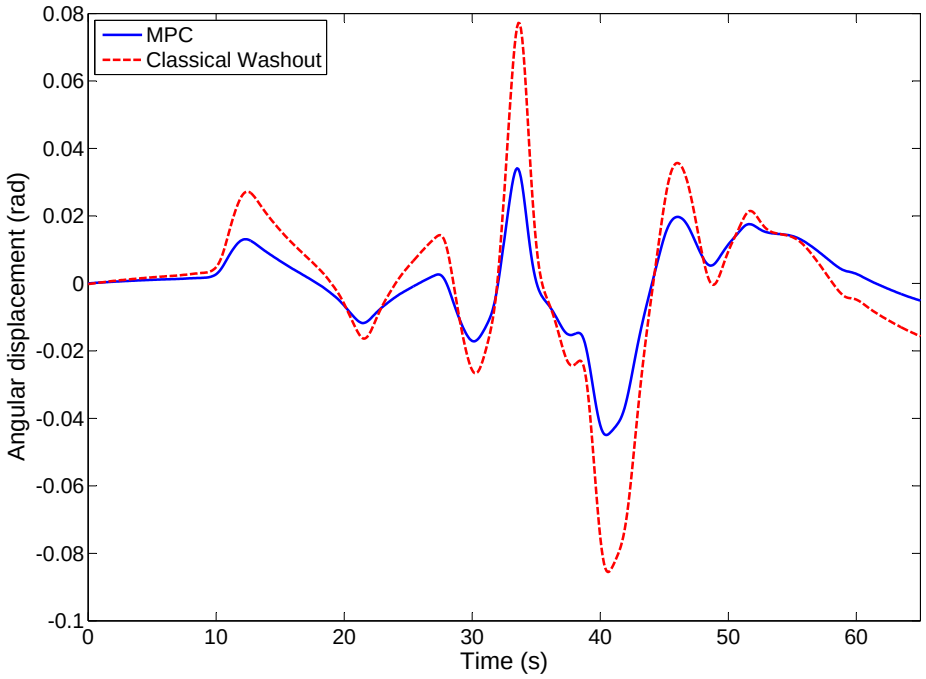

Figure A.12: Angular displacement - Yaw

A.3 Commands for the Motion Platform: Motion commands vs Platform displacements

This section holds the comparison between the results sent to the motion platform and the displacements it executes. More information about these figures can be found in section 3.7.

Notice that these values are all constrained. Upper and lower bounds are only shown if the signals are close to the latter. The boundaries can be found in B.1.

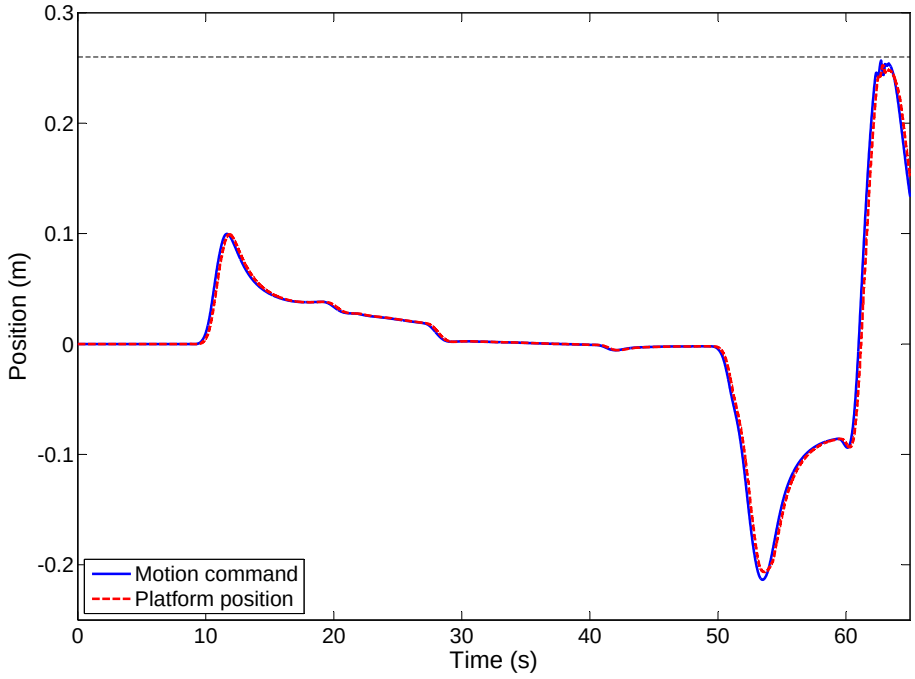

Figure A.13: Position about the X-axis

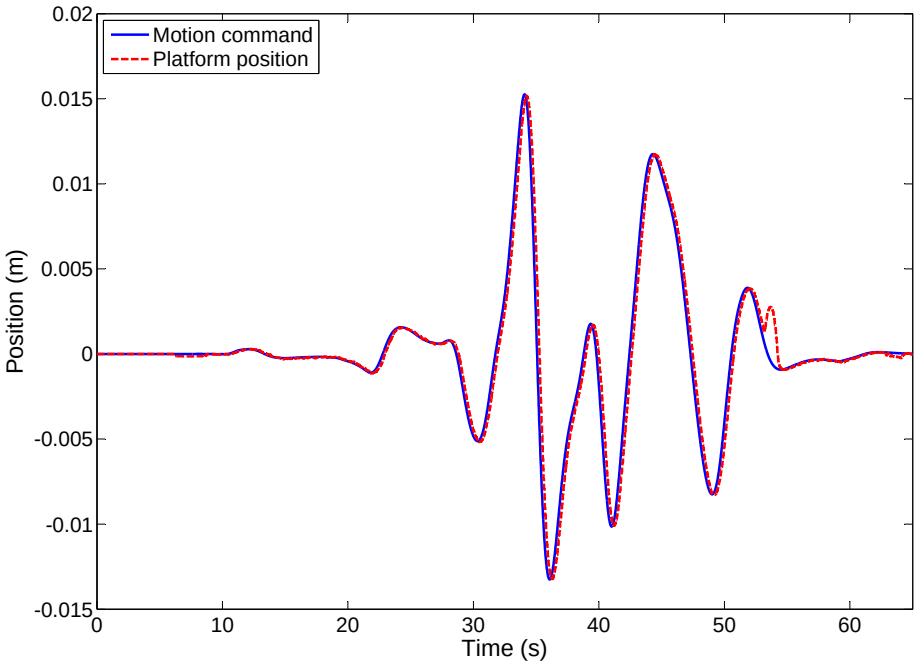

Figure A.14: Position about the Y-axis

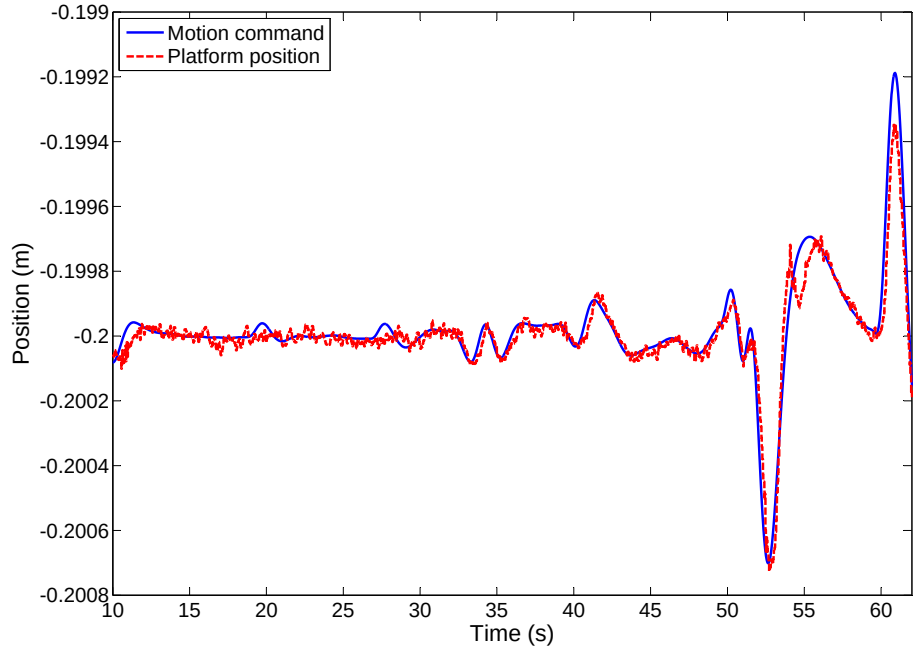

Figure A.15: Position about the Z-axis

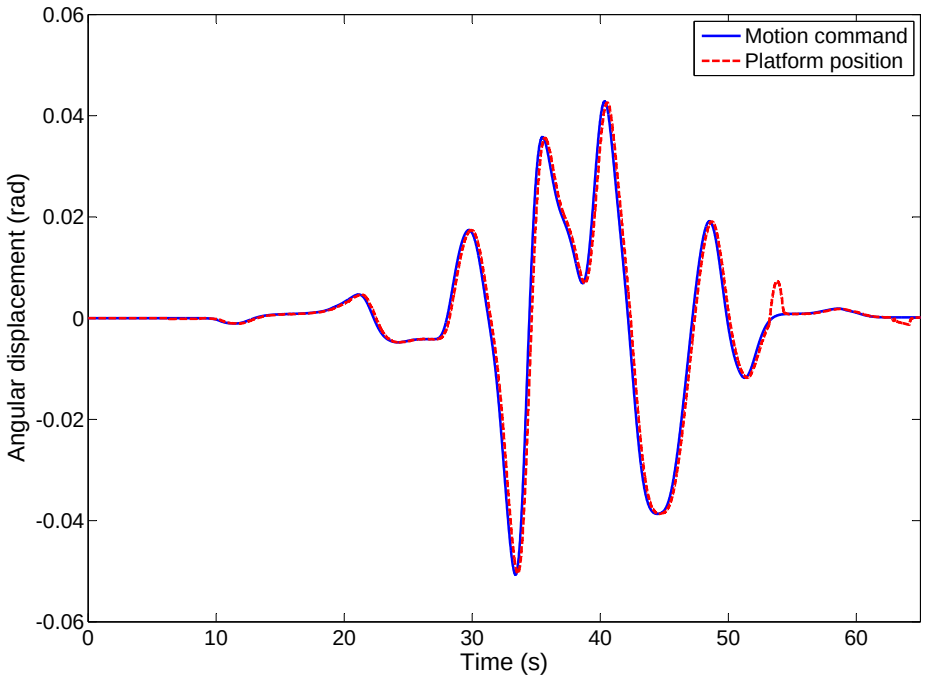

Figure A.16: Angular displacement - Roll

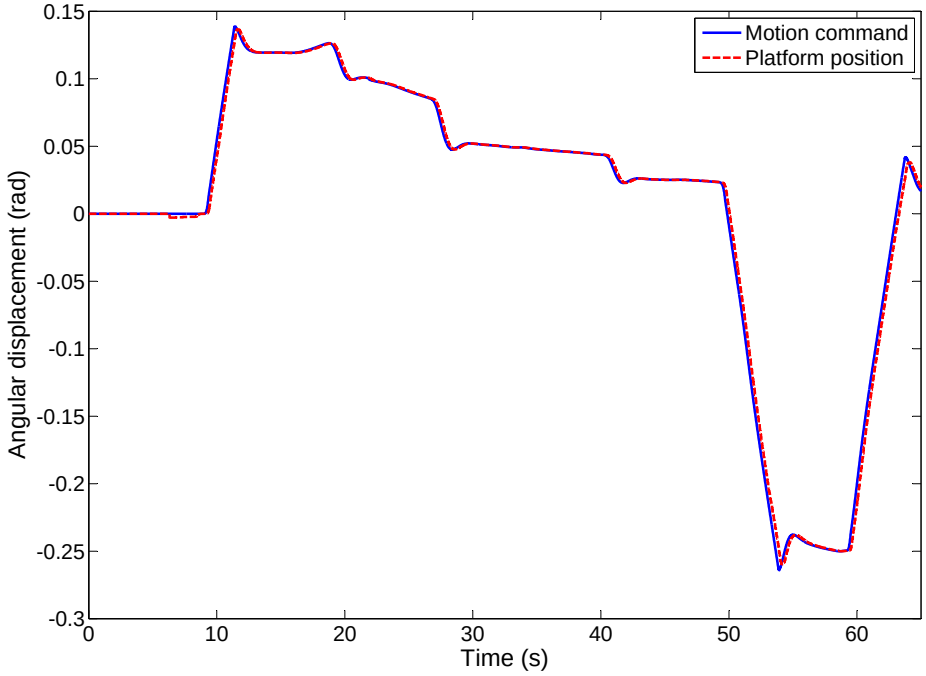

Figure A.17: Angular displacement - Pitch

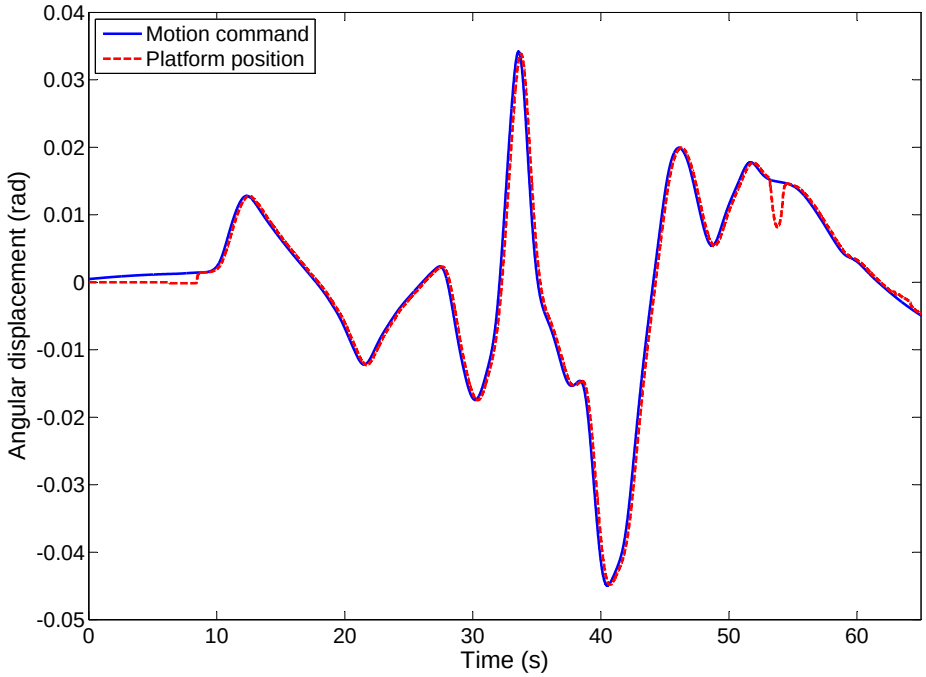

Figure A.18: Angular displacement - Yaw

A.4 Commands for the Motion Platform: Sending rate

This section contains a comparison between the commands computed by the MPC and the ones actually sent to the platform. For more information on this issue consult section 3.7. These results were extracted from a test done with the Controller 1, section 3.5 with the maneuver presented in section 3.1.

Notice that these values are all constrained. Upper and lower bounds are only shown if the signals are close to the latter. The boundaries can be found in B.1.

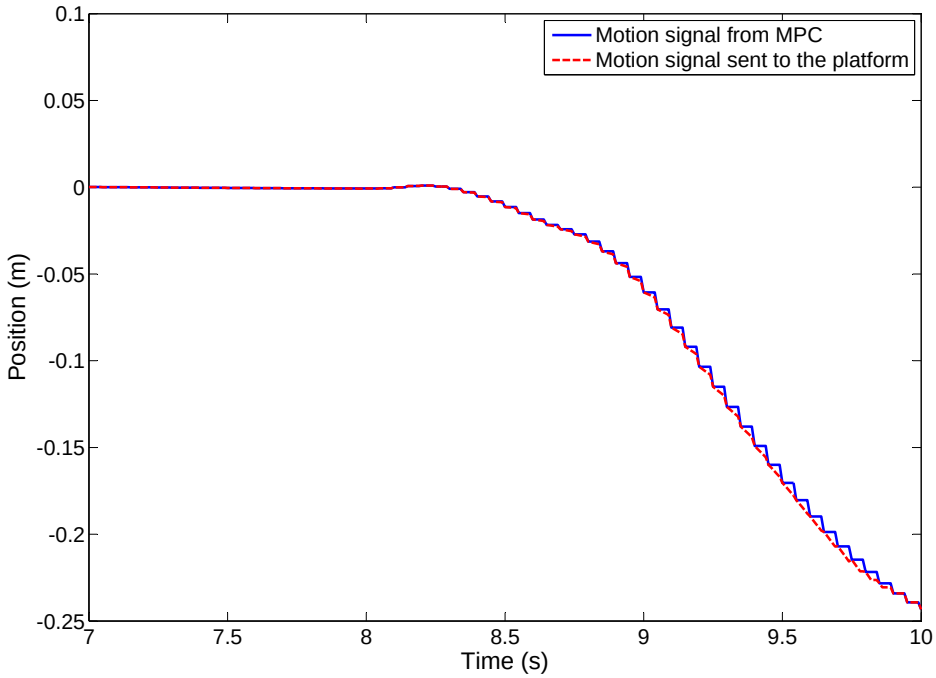

Figure A.19: Position about the X-axis

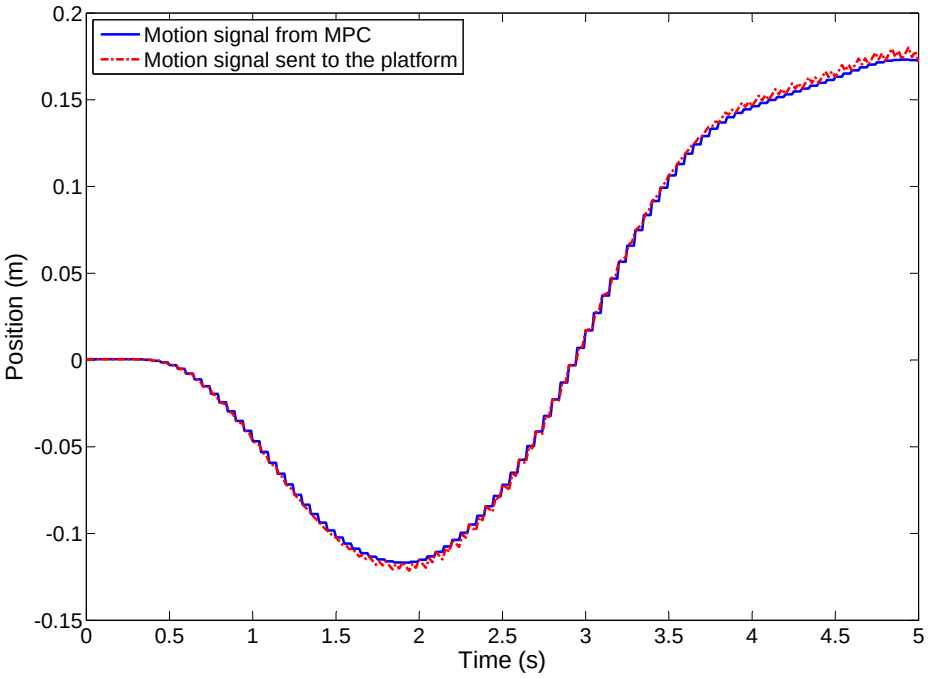

Figure A.20: Position about the Y-axis

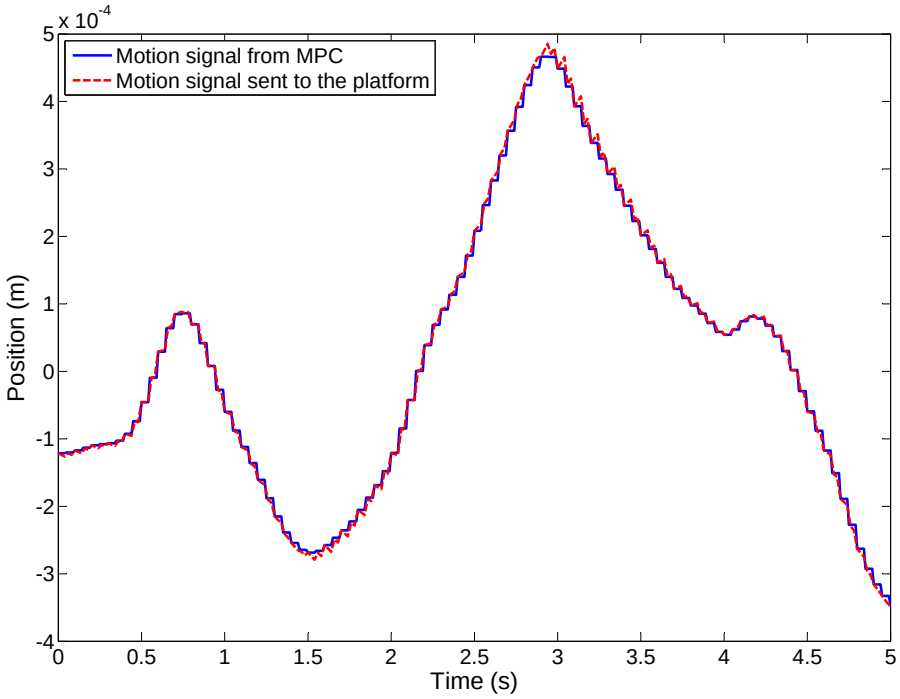

Figure A.21: Position about the Z-axis

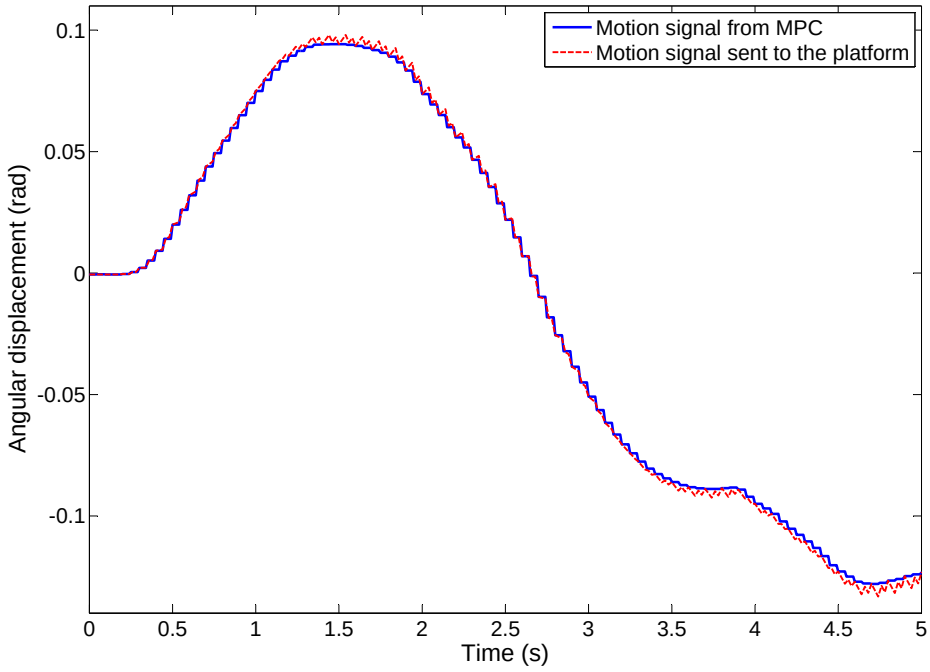

Figure A.22: Angular displacement - Roll

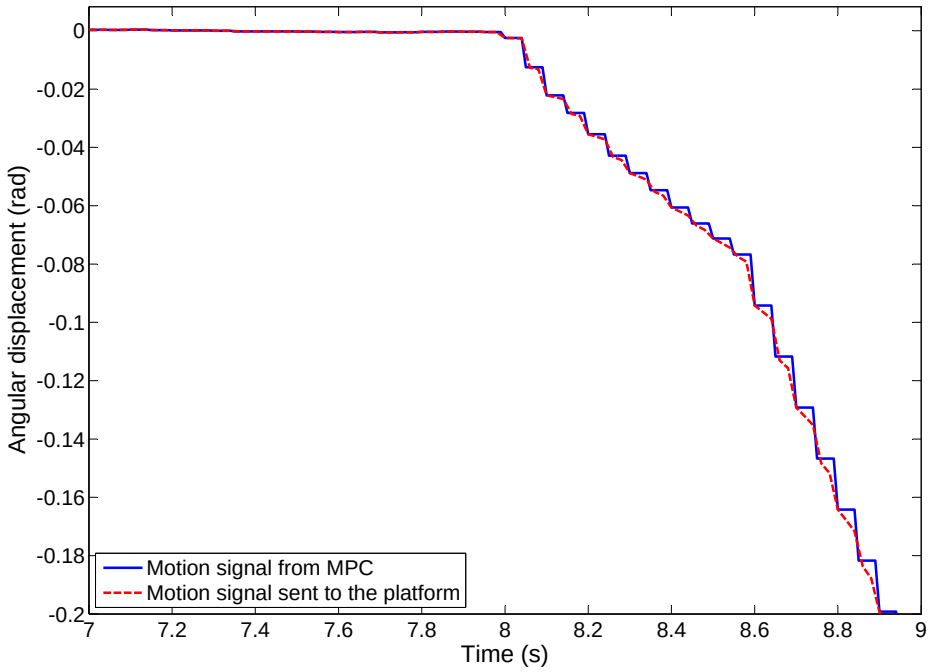

Figure A.23: Angular displacement - Pitch

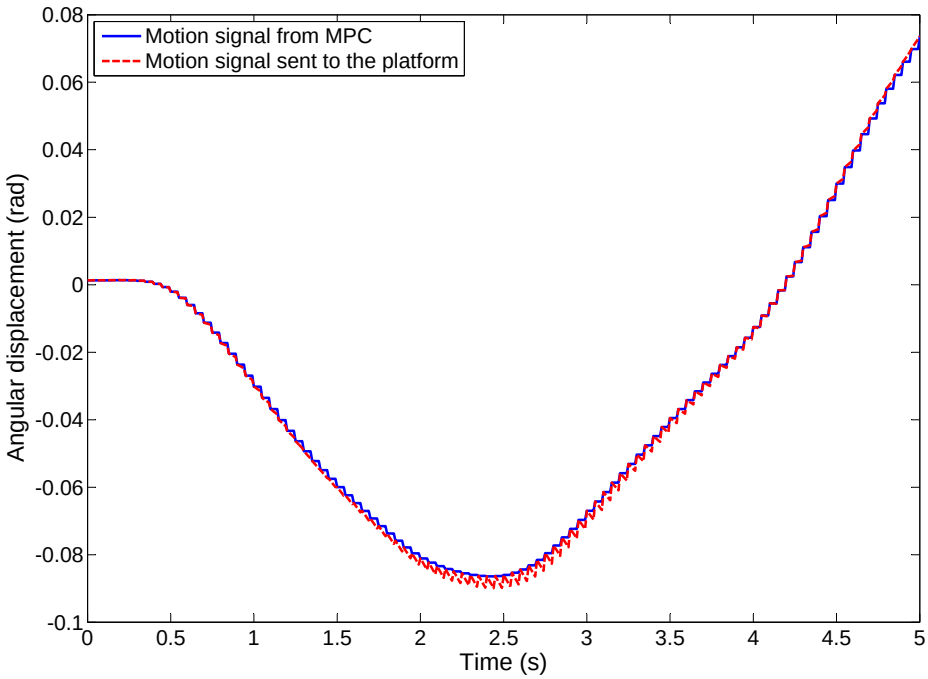

Figure A.24: Angular displacement - Yaw

A.5 Weights of the MPC - Offline Simulations

The weights used for the MPC formulation are presented in the tables A.1, A.2 and, A.4. They were chosen by a trial and error. For more information about weights and its relationships consult section 2.7.5.

Table A.1: Weighs on the tracking variables

	w_x	w_y	w_z	f_x	f_y	f_z
Weights	10000	10000	60	9	10	80

Table A.2: Weighs on the tracking variables cont.

	ϕ	θ	ψ	p_x	v_x	p_y	v_y	p_z	v_z
Weights	20	20	5	10	15	10	15	5	9

Table A.3: Weighs on the control inputs

	$\dot{\phi}_r$	$\dot{\theta}_r$	$\dot{\psi}$	a_x	a_y	a_z	$\dot{\phi}_a$	$\dot{\theta}_a$
Weights	1	1	1	1	1	1	20	20

Table A.4: Weighs on the control input rates

	$\Delta\dot{\phi}_r$	$\Delta\dot{\theta}_r$	$\Delta\dot{\psi}$	Δa_x	Δa_y	Δa_z	$\Delta\dot{\phi}_a$	$\Delta\dot{\theta}_a$
Weights	1	1	1	0.01	0.01	0.01	8	8

A.6 Weights of the MPC - Online Simulations

The weights used for the MPC formulation are presented in the tables A.5, A.6 and, A.8. They were tuned by trial and error, according to the drivers tips about the perceived signals. For more information about weights and its relationships consult section 2.7.5. Commentaries about this specific set of weights can be consulted in section 3.7.

Table A.5: Weighs on the tracking variables

	w_x	w_y	w_z	f_x	f_y	f_z
Weights	20000	20000	300	6	3	10

Table A.6: Weighs on the tracking variables cont.

	ϕ	θ	ψ	p_x	v_x	p_y	v_y	p_z	v_z
Weights	60	60	25	20	20	20	20	30	15

Table A.7: Weighs on the control inputs

	$\dot{\phi}_r$	$\dot{\theta}_r$	$\dot{\psi}$	a_x	a_y	a_z	$\dot{\phi}_a$	$\dot{\theta}_a$
Weights	1	1	1	1	1	1	20	20

Table A.8: Weighs on the control input rates

	$\Delta\dot{\phi}_r$	$\Delta\dot{\theta}_r$	$\Delta\dot{\psi}$	Δa_x	Δa_y	Δa_z	$\Delta\dot{\phi}_a$	$\Delta\dot{\theta}_a$
Weights	1	1	1	0.1	0.1	0.1	20	20

Appendix B

Chalmers Vehicle Simulator - Communications

This section explains in detail the communication between the driving simulator computers. A scheme of connections is shown and a brief explanation of the computers is given.

All communications between platform and control computer as well as between all computers are presented in figure B.1.

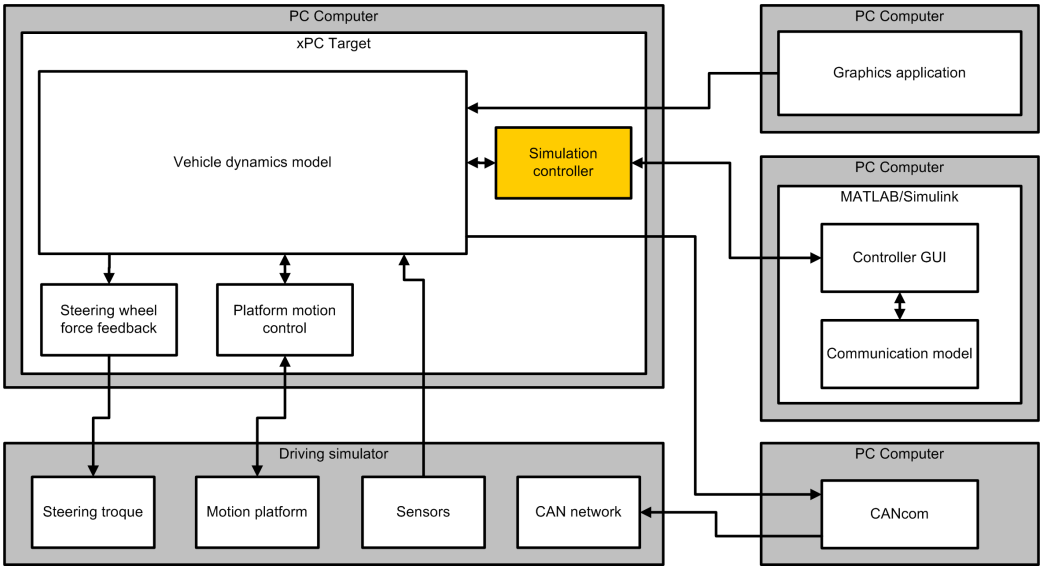

Figure B.1: Communications layout

- Control Computer - In this computer the Simulink control scheme is developed, compiled and sent to the real time environment. In the scheme this computer is represented bellow the computer with the graphics application.

- xPc Target Computer - This computer is responsible for all real time calculations. Inside it is possible to find the vehicle model, and all the motion cueing algorithms that generate control inputs for the Platform.
- Platform Computer - Located in the motion platform base, this computer is responsible for communicating to the platform all desired displacements so that all control commands from xPc Target Computer are met. Its command inputs are in the form of Euler and positions about the X, Y and Z axes. In the given scheme, it is included in the driving simulator, between the xPc-target computer and the motion platform.
- Graphical Computer - This computer function is to create all visual cues displayed in the drivers screen.

B.1 Bounds on Platform States

Table B.1: Bounds for Euler angles

	ϕ	θ	ψ
$displacement(deg)$	± 22	$+25/-23$	± 23
$rate(deg/s)$	± 30	± 30	± 40
$acceleration(deg/s^2)$	± 500	± 500	± 400

Table B.2: Bounds for translational movement

	x	y	z
$position(m)$	± 0.18	± 0.27	± 0.26
$velocity(m/s)$	± 0.3	± 0.5	± 0.25
$acceleration(m/s^2)$	$\pm 0.6g$	$\pm 0.6g$	$\pm 0.6g$

Bounds for the roll and pitch angular rates used to simulate rotations, are the ones presented in the above tables. The same quantities, when used to simulate accelerations, are bounded by the values for threshold, in table 2.1.

Appendix C

Discretization

The pole map illustrated in figure C.1 holds the poles for the system represented by the state space model 2.55.

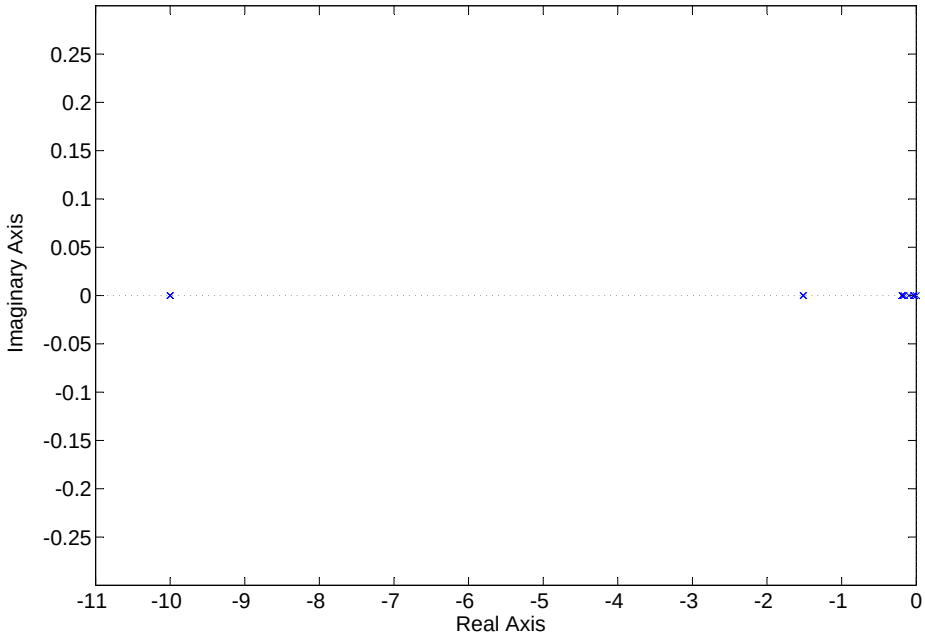

Figure C.1: Pole Zero Map

It is possible to conclude that the system has slow dynamics given that the furthest pole, from the origin, is situated at -10 units. This result enables the utilization of the forward Euler method. Let T_{sm} be the maximum sampling period that guarantees stability for the discrete system.

The Euler forward method does not necessarily guarantee stability in the Z plain even if that is a property of the continuous time system. In this transformation only the values contained in the circle presented in the S plain, figure C.2, are mapped to the unit circle in the Z plain, thus being stable in the discrete time form.

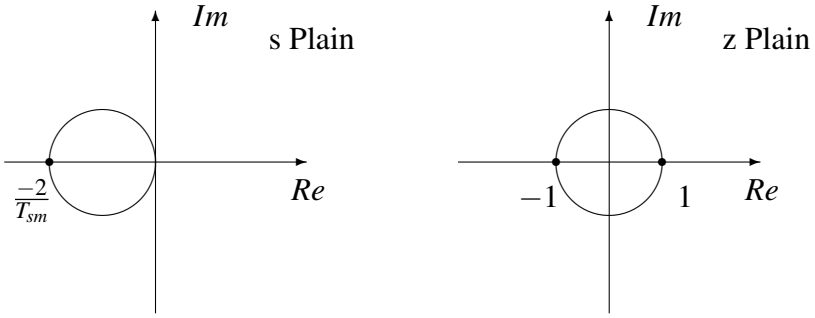

Figure C.2: Forward Euler Method stability zone

Given that the most distant pole from the origin is a located at a distance of 10 units, a minimum sampling period of $0.2s$ is required to enforce a stable discretized system.

The discretized state space model has the following format:

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k), \\ y(k) &= C_d x(k) + D_d u(k), \end{aligned} \tag{C.1}$$

where

$$A_d = AT_{samp} + I, \quad B_d = BT_{samp}, \quad C_d = C, \quad D_d = D. \tag{C.2}$$

Appendix D

Bicycle Vehicle Model Data

The vehicle model simulates the dynamic motion of a Volvo XC90. The data given in this section was extracted from the latter. It will be applied to bicycle model, section 2.2.1.

Vehicle Data		
Parameters	Values	Units (SI)
mass(m)	2081	Kg
Inertia(I)	4512.6	$Kg \times m^2$
Distance from the CoG to the front axles (d_f)	1.3242	m
Distance from the CoG to the rear axles(d_r)	1.5328	m

Table D.1: Vehicle model constants

Appendix E

Genetic Algorithms

Genetic algorithm (GA) is an optimization method developed by John Holland in early 1970's, who based his thoughts on the Darwin's theory. Establishing that all species of life have evolved over time from common ancestors, through the process he called natural selection [38]. The initial idea came from observing how the evolution of biological creatures derives from their constituent DNA and chromosomes. In this sense a simple analogy can be made with a mathematical problem. Genetic algorithms are used to solve several different problems, for instance, Optimization, Automatic programming and Economics [10].

The field of Genetic optimization is quite wide, this work only explains the main points in order to help the reader to understand the work developed. For a deeper insight into the genetic algorithm the reader should consult the following books [24] and [10].

E.1 Main Concepts of GA

The following list of important concepts of Genetic algorithm can be established:

- *Population*: Group formed by many individual solutions, each of them with its own characteristics.
- *Individual*: Can be represented in a decimal or binary base. It is usually referred to as chromosome.
- *Fitness*: Objective function value for a determined individual. It is directly proportional to the quality of the solution presented by the latter.
- *Crossover*: This genetic operator is responsible for the creation of new offspring. Several types of crossover exist, [10], where single point crossover is one of the most common one and simplest techniques, [24].

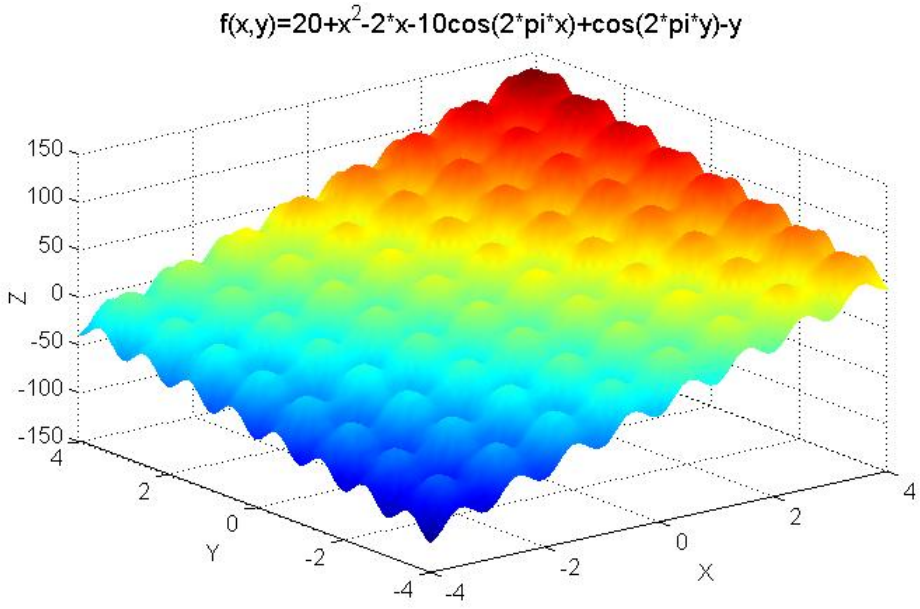

Figure E.1: Set of feasible solutions with many local optima

- *Mutation*: Introduced in this algorithm to reproduce the natural possibility of mutation, this operator is the second way for GA to explore the cost surface [13]. According to a specified probability mutation may occur swapping one bit to its antagonic value.
- *Elitism*: Top individuals are conserved without any kind of alteration for the following generation. This guarantees that the best fitness value is at least the same as in the previous iteration.

Genetic Algorithms are population based optimization algorithm. The fitness of each individual in a current population is evaluated in order to establish the quality of the respective solution. Individuals within the population will mate with each other generating offspring, according to a previously defined crossover rule. The offspring will be subjected to the possibility of mutation before the new generation is completely defined. This process establishes a new population to be processed by the algorithm once again, and it will be iterated until some stopping criterion is fulfilled.

This method is usually implemented when the set of feasible solutions is so large that typical brute force optimization algorithms would take a lot of time to check all the possible combinations and achieve an optimal solution. One disadvantage of this method is that it does not guarantee a global optimum. However, a wide solution set is tested which increases the probability of finding a global optimum or at least a close solution.

E.2 Outline of the Genetic Algorithm

1. An initial population of N individuals is created.
2. Evaluate the fitness of all individuals according to an objective function.
3. Convert the population from decimal to binary representation. By performing this transformation the problem of formulating offspring as well as performing eventual mutations will be easily implemented.
4. Create offspring by mating individuals according to a crossover technique.
5. A probability of mutation is defined. Each gene of all offspring is subjected to this possibility of change. If mutation occurs, the gene value will change to its antagonic value.
6. A new population is formed by the offspring. Usually, the individuals with higher fitness are kept from population to population ensuring that the best fitness value in the new iteration is at least the highest value of the previous generation (Elitism).
7. Convert the population from binary to decimal. Then, check the stop criteria. If it is verified the algorithms stops else, it goes to step 2.

E.3 Experimental Setup

The genetic algorithm was implemented as a m-file function. One point crossover was applied and twenty iterations without improvement was the used stopping criterion. Elitism was implemented and the two top individuals of each iteration passed to the next generation. Populations consisted of twenty individuals.

The fitness value for each individual was the mean square error between the bicycle model estimated states and the actual ones.

Appendix F

Optimization problem formulation

F.1 Optimization Algorithm for QP Problems

A quadratic programming (QP) problem has a quadratic objective function of the optimization variables. A QP problem may be constrained.

A quadratic problem can be formulated as follow:

$$\textit{Minimize} \qquad f(x) = c^T x + \frac{1}{2} x^T A x, \tag{F.1a}$$

$$\text{subj. to} \qquad A_c x \leq b \quad \textit{and} \quad x \geq 0, \tag{F.1b}$$

where $c \in \Re^n$, and $A \in \Re^{n \times n}$ describe the coefficients of the linear and quadratic terms in the cost function, respectively. Let x is defined as the decision/optimization variable. If A is a positive semidefinite matrix, $f(x)$ is a convex function, on the open and convex set $S \subseteq \Re^n$, and every local minimum is also a global one. Furthermore, if A is positive definite then, only a unique local minimum exists which is also the global one [7]. There are several methods to solve QP problems, for instance, a modified version Simplex method, the Frank-Wolfe algorithm or the Simplicial Decomposition algorithm [7].

The solver used in this work to compute the optimization problem is named LSSOL. The latter uses a two-phase (primal) quadratic programming [11].

- Feasible Phase (I) - Finds an initial feasible point by minimizing the sum of infeasibilities.
- Optimality Phase (II) - Minimizes the objective function within the the feasible region.

At each new iteration the variable values x_{k+1} are found by applying the following equation:

$$x_{k+1} = x_k + \alpha_k p_k, \tag{F.2}$$

where, α_k and p_k are a positive scalar step length and the search direction respectively.

An important feature regarding the LSSOL algorithm is the computation of the search direction. It is performed in a way that, if one or several variables are close (within a Feasibility Tolerance defined by the user) to its constraints, the referred search direction is found so that the value of those variables remain unaltered.

Detailed information regarding LSOOL package can be found in [11].

F.2 Augmented Model

Consider the linearized discrete-time state space model 2.66. It was already discussed that the MPC prediction model needs different inputs for constraint purposes. Let us then introduce the changes.

The state matrix is defined as

$$A_{pred} = \begin{bmatrix} A_{vest} & \iota_A \\ 0_{2\times 24} & 0_{2\times 2} \end{bmatrix}, \tag{F.3}$$

where

$$\iota_A = \begin{bmatrix} 0_{12\times 1} & 0_{12\times 1} \\ 0 & \frac{K\tau_a}{\tau_l \tau_s} \\ 0 & \frac{K}{\tau_l \tau_s} \\ -\frac{K\tau_a}{\tau_l \tau_s} & 0 \\ -\frac{K}{\tau_l \tau_s} & 0 \\ 0_{10\times 1} & 0_{10\times 1} \end{bmatrix}. \tag{F.4}$$

The input matrix is defined like:

$$B_{pred} = \begin{bmatrix} B_{vest} & 0_{24\times 2} \\ 0_{2\times 6} & I_{2\times 2} \end{bmatrix}. \tag{F.5}$$

The output matrix is given by:

$$C_{pred} = \begin{bmatrix} C_{vest} & \iota_C \end{bmatrix}, \tag{F.6}$$

where

$$\iota_C = \begin{bmatrix} 0_{6\times1} & 0_{6\times1} \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0_{5\times1} & 0_{5\times1} \end{bmatrix}. \tag{F.7}$$

Finally, the direct feedtrough matrix, D_{pred}, has two more columns of zeros than D_{vest} and the same number of rows.

The state vector is present as follows:

$$x_{pred} = \begin{bmatrix} x_{scc} \\ x_{oto_x} \\ x_{oto_y} \\ x_{oto_z} \\ \phi_r \\ \theta_r \\ \phi \\ p_x \\ v_x \\ p_y \\ v_y \\ p_z \\ v_z \\ \phi_a \\ \theta_a \end{bmatrix}, \tag{F.8}$$

where ϕ_r and θ_r are the Euler angles generated to simulate rotations and, ϕ_a, θ_a the Euler angles used for tilt coordination.

This formulation allows the tilt coordination rates to be constrained because they are now a part of the problem variables. It also allows the MPC to understand that every rotation cue will also affect the felt specific force cues. The contrary does not happen because we are dealing with angular velocities undetected by the driver.

Since the MPC sees that angular velocities, used to generate sensed rotations, also affect the specific force, it may be temped to use those inputs for the purpose of tracking specific forces. Using a big weight on felt rotations this situation is prevented. For more information on this issue consult section 2.7.5

The formulation applied by this work defines the input variation rate $\Delta u(k)$ as the optimized variable, thus the state space model needs to be modified in order to include the latter as input.

Let us first define the augmented state as

$$\xi(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}. \tag{F.9}$$

This alteration to the state vector enables the definition of the following augmented model

$$\begin{aligned} \xi(k+1) &= A_{aug}\xi(k) + B_{aug}\Delta u(k), \\ Z(k|k) &= C_{aug}\xi(k) + D_{aug}\Delta u(k), \end{aligned} \tag{F.10}$$

where

$$A_{aug} = \begin{bmatrix} A_{pred} & B_{pred} \\ 0_{8\times 26} & I_8 \end{bmatrix}, B_{aug} = \begin{bmatrix} B_{pred} \\ I_8 \end{bmatrix}, C_{aug} = \begin{bmatrix} C_{pred} & D_{pred} \end{bmatrix}, D_{aug} = \begin{bmatrix} D_{pred} \end{bmatrix}. \tag{F.11}$$

F.3 Problem Formulation

F.3.1 Prediction Model

The current section is based on the work of Paolo Falcone [8]. Taking in account the augmented system model defined in the last section, the prediction model can be defined as follows

$$\begin{aligned} \xi(k+1|t) &= A_{aug}\xi(k|t) + B_{aug}\Delta u(k|t), \\ y(k|t) &= C_{aug}\xi(k|t) + D_{aug}\Delta u(k|t). \end{aligned} \tag{F.12}$$

Let $\lambda(t)$ be a vector $p \times H_p$, where p is the number of outputs. $\lambda(t)$ contains the predictions $[y(t|t), ..., y(t+H_p|t)]^T$ generated by the sequence of input rates, $\Delta U(t) = [\Delta u(t|t), ..., \Delta u(t+H_p|t)]^T$. Then, the relationship between both quantities is presented as,

$$\lambda(t) = \psi\xi(t|t) + \Theta\Delta U(t), \tag{F.13}$$

where, dropping the underscript $()_{aug}$ for simplicity,

$$\begin{bmatrix} y(t+1|t) \\ y(t+2|t) \\ \vdots \\ y(t+H_u|t) \\ y(t+H_u+1|t) \\ \vdots \\ y(t+H_p|t) \end{bmatrix} = \underbrace{\begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{H_u} \\ CA^{H_u+1} \\ \vdots \\ CA^{H_p} \end{bmatrix}}_{\Psi} \xi(t|t) \; + \;$$

$$\underbrace{\begin{bmatrix} CB & 0_{p\times n_{in}} & \cdots & 0_{p\times n_{in}} \\ CAB & CB & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots \\ CA^{H_u-1}B & CA^{H_u-2}B & \cdots & CB \\ CA^{H_u}B & CA^{H_u-1}B & \cdots & CAB \\ \vdots & \cdots & \ddots & \vdots \\ CA^{H_p-1}B & CA^{H_p-2}B & \cdots & CA^{H_p-H_u}B \end{bmatrix}}_{\Theta} \begin{bmatrix} \Delta u(t|t) \\ \Delta u(t+1|t) \\ \vdots \\ \Delta u(t+H_u-1|t) \end{bmatrix} \tag{F.14}$$

Note that in the system F.14 it is assumed that $\Delta u(t+H_u|t) = \Delta u(t+H_u+1|t) = \ldots = \Delta u(t+H_p-1|t) = 0$. n_{in} is the number of inputs.

The output vector of the system $\lambda(t)$ can be divided in two parts, $\lambda(t) = [\lambda(t)_{tr}, \lambda(t)c]$, where $\lambda(t)_{tr}$ are the outputs that need to be tracked, and $\lambda(t)_c$ are the outputs to be hard constrained.

Since in this problem all the outputs are tracked, the following conclusion is immediate,

$$\lambda(t)_{tr} = \psi\xi(t|t) + \Theta\Delta U(t) = \lambda(t). \tag{F.15}$$

Nevertheless, it is not the same case for the constrained variables. Let υ be a $p_c \times H_p$ by $p \times H_p$ matrix . If

$$\lambda(t)_c = \upsilon\lambda(t), \tag{F.16}$$

then, υ takes the form

$$\upsilon = \begin{bmatrix} 0_{p_c\times(p-p_c)} & I_{p_c\times p_c} & \cdots & 0_{p_c\times(p-p_c)} & 0_{p_c\times p_c} \\ 0_{p_c\times(p-p_c)} & 0_{p_c\times p_c} & \cdots & 0_{p_c\times(p-p_c)} & 0_{p_c\times p_c} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0_{p_c\times(p-p_c)} & 0_{p_c\times p_c} & \cdots & 0_{p_c\times(p-p_c)} & I_{p_c\times p_c} \end{bmatrix} \tag{F.17}$$

because the constrained outputs are the last p_c in the output vector, $y(k|t)$. With this result, it is possible to define

$$\lambda(t)_c = \psi_c\xi(t|t) + \Theta_c\Delta U(t), \tag{F.18}$$

where

$$\psi_c = \upsilon\psi, \quad \Theta_c = \upsilon\Theta. \tag{F.19}$$

F.3.2 Objective Function Formulation

Just like in the previous function the current one is based on the work presented in [8]. Let equation 2.67 be redefined such that

$$J(t) = ||\lambda(t)_{tr} - ref(t)||^2_Q + ||U(t)||^2_S + ||\Delta U(t)||^2_R, \tag{F.20}$$

where $\lambda_{tr}(t)$ and $\Delta U(t)$ were introduced in the previous section, while $ref(t) = [r(t+1,t),\ldots,r(t+H_p,t)]$ and $U(t)$ was defined previously in section 2.7.2.

$U(t)$ can be defined as a function of $\Delta U(t)$ and $u(t-1)$, like shown in equation F.21.

$$U(t) = \kappa\Delta U(t) + U_t, \tag{F.21}$$

where U_t is the product $u(t-1)\otimes 1_{H_u}$ for 1_{H_u} defined as a vector of ones with H_c rows and $\otimes$ defined as the Kronecker product. κ is established as the Kronecker product between a lower triangular matrix of ones $H_c \times H_c$ and an identity matrix $n_{in}\times n_{in}$.

Let us define λ_{tr}, when $\Delta U(t) = [0,\ldots,0]$, as the free response of the system, and $E(t)$ as the difference between the free response and $ref(t)$. Then, the cost function F.20, can be defined as

$$\begin{aligned} J(t) \;=\; & E(t)^T QE(t) + U_t^T SU_t + \\ & 2(E^T(t)Q\Theta + U_t^T S\kappa)\Delta U(t)^T + \\ & \Delta U(t)^T[\Theta^T Q\Theta + R + \kappa^T S\kappa]\Delta U(t). \end{aligned} \tag{F.22}$$

Finally, the optimization problem can be described as:

$$\min_{\Delta U(t)} \quad J(\Delta U(t), u(t-1), \xi(t)), \tag{F.23a}$$

$$subj.\ to,$$

$$Y_{min} \leq \psi_c \xi(t|t) + \Theta_c \Delta U(t), \leq Y_{max} \tag{F.23b}$$

$$U_{min} - U_t \leq \kappa \Delta U(t) \leq U_{max} - U_t, \tag{F.23c}$$

$$\Delta U_{min} \leq \Delta U(t) \leq \Delta U_{max}. \tag{F.23d}$$

The presented constraint vectors are a function of the ones presented in section 2.7 and are defined as follows

$$Y_{min} = y_{min} \otimes 1_{H_c}, \quad U_{min} = u_{min} \otimes 1_{H_u}, \quad U_{min} = u_{min} \otimes 1_{H_u}. \tag{F.24}$$

where 1_{H_*} is defined as a column vector of ones with size H_*.

In order to be applied to the algorithm for optimization presented in F.1, the optimization problem F.23 needs to be changed. As it is possible to conclude, the objective function, equation F.22, is already presented in the desired format but the set of constrains is not. With all the gathered information one can define the matrices A_c and b in equation F.1b

$$A_c = \begin{bmatrix} \kappa \\ -\kappa \\ \Theta_c \\ -\Theta_c \end{bmatrix}, b = \begin{bmatrix} U_{max} - U_t \\ -U_{min} + U_t \\ Y_{max} - \psi_c \xi(t|t) \\ -Y_{min} + \psi_c \xi(t|t) \end{bmatrix}. \tag{F.25}$$

Leaving the problem completely defined for application of the optimization algorithm.