MPC-based Motion Cueing Algorithm for a 6 DOF Driving Simulator with Actuator Constraints

Y.R. Khusro

Delft University of Technology

TU Delft

MPC-Based Motion Cueing Algorithm For A 6 DOF Driving Simulator With Actuator Constraints

by

Yash Raj Khusro

For the degree of Master of Science in Mechanical Engineering
(Vehicle Engineering) at the Delft University of Technology

August 21, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3ME) · Delft University of Technology

Keywords: Driving Simulator, Motion Cueing Algorithm, Model Predictive Control, Nonlinear, Actuator Constraints

PREFACE

This thesis is submitted for the Masters of Science degree in Mechanical Engineering at the Delft University of Technology. The aim of this research was to investigate various cueing principles used for the motion cueing of driving simulators and propose an efficient algorithm for high fidelity motion cueing.

The research was divided into two parts. The first part consisted of the literature review. In this, the focus was to understand the motion cueing process and review various motion cueing algorithms (MCA) present in the literature. In the study, it was found out that the most commonly used cueing algorithms are classical washout filter, optimal washout filter and adaptive washout filter. However, their inability to impose constraints on the simulator states results in producing low fidelity cues and sub-optimal workspace utilization. The literature review pointed out that the paradigm has recently shifted to advanced control techniques like Model Predictive Control (MPC). It was noted that MPC offers an elegant solution to the problem as it allows to impose explicit constraints on the state and output variables. The research showed that the Linear MPC-based MCA produced superior results compared to the conventional filter-based MCAs. Further, it was noted that most MPC-based MCAs derived in the literature used linear models ignoring the non-linearities of the motion platform, leading to conservative results. Therefore, it was concluded that in order to improve the workspace utilization and increase the fidelity of the produced cues, a non-linear MPC-based MCA with actuator-based constraints is required.

The second part of this research aimed at designing a nonlinear MPC-based motion cueing algorithm incorporating the human vestibular system model and the non-linear kinematic model of the motion platform. The proposed algorithm was then tested thoroughly and was compared with the classical washout filter and linear MPC-based MCA. The results showed superior performance of the proposed algorithm in terms of reference tracking and workspace utilization.

The output of this research is presented in the form of a scientific article which can be found in Chapter 1 of this report. The supplementary information about the fundamentals of motion cueing, filter-based algorithms and Linear MPC-based MCA is presented in Appendices A, B and C. Further, the details about the non-linear MPC solution techniques used are presented in the Appendix D. Lastly, extensive results, in supplement to the ones provided in the chapter 1 can be found in Appendix E.

Yash Raj Khusro
Delft, August 2020

ACKNOWLEDGEMENT

Starting from the decision to pursue masters to finally concluding this thesis, it has been a fun, eventful and educative journey. I am glad that I took this journey and met some of the most brilliant and supportive people. This acknowledgement is a word of thanks to not only those whom I learned from but also those who supported and helped me become a better version of myself.

Firstly, I would like to thank Dr. Barys Shyrokau for believing in me and giving me this challenging project. He has been a constant pillar of support and I couldn't have wished for a better supervisor. He has been enthusiastic and patient enough, allowing me to grow at my own pace. Apart from imparting the knowledge about the subject, he has also taught me about the approach one should take for their research. Moreover, I would like to thank Mr. Yanggu Zheng, Mr. Marco Grottoli and Dr. Riender Happee for their valuable insights in this thesis.

Next, I would like to thank my parents and sister for their wise counsel and belief, not just in my masters but throughout my education. I would also like to thank Aprajita for her support and sympathetic ear. Also, a word goes out to my extended family members who have been a part of this journey during different phases. Thank you for always being there with me.

Lastly, I would like to thank all my friends and housemates for always being there for me and for inspiring me.

Yash Raj Khusro
Delft, August 2020

Contents

List of Figures

List of Tables

1
Scientific Article

Article

MPC-based motion cueing algorithm for a 6 DOF driving simulator with actuator constraints

Abstract: Driving simulators are widely used for understanding Human-Machine Interaction, driver behavior and driver training. The effectiveness of such simulators in this process depends largely on their ability to generate realistic motion cues. Though the conventional filter-based motion cueing strategies have provided reasonable results, these methods result in poor workspace management. To address this issue, linear MPC-based strategies have been applied in the past. However, since the kinematics of the motion platform itself is non-linear and the required motion varies with the driving conditions, this approach tends to produce sub-optimal results. In this paper, a nonlinear MPC-based algorithm is presented which incorporates the non-linear kinematics of the Stewart platform within the MPC algorithm to increase the effectiveness and utilize maximum workspace. Further, adaptive weights-based tuning is used to smoothen the movement of the platform near its physical limits. Full-track simulations were carried out and performance indicators were defined to objectively compare the response of the proposed algorithm with classical washout filter and linear MPC-based algorithms. The results indicate a better reference tracking with lower root mean square error and higher shape correlation for the proposed algorithm. Lastly, the effect of the adaptive weights-based tuning was also observed in the form of smoother actuator movements and better workspace utilization.

Keywords: driving simulator, motion cueing algorithm, model predictive control, nonlinear, actuator constraints

---

1. Introduction

With the increasing demand for advanced driver assistance systems, driving simulators hold the potential to transform the research and development of the intelligent vehicles. They can reduce the cost and time incurred in the vehicle development process and also help in designing robust and intelligent solutions. Further, these simulators are increasingly being used for other purposes, such as human machine interface studies, understanding driver behavior, and training of drivers in a safe environment.

The effectiveness of such driving simulators is measured by their ability to generate realistic motion cues i.e. the driver or the passenger, sitting inside the simulator should perceived similar motion cues that s/he would perceived while sitting in a real vehicle performing the same maneuver. However due to its limited workspace, the driving simulator cannot be directly subjected to the vehicle motion as then the platform would quickly reach its physical limits and no motion cues could be provided to the driver any further. To overcome this limitation, motion cueing algorithms have been developed. A Motion Cueing Algorithm (MCA) is the strategy that governs the process of producing motion cues while keeping the motion platform within its physical limits. Thus, the main objectives of an MCA are the following:

- Providing realistic motion cues to the driver or passenger sitting inside the simulator.
- Keeping the motion platform within its physical boundaries.

A detailed description about the working of an MCA and the fundamentals of motion cueing is given in Appendix A. The classical approach of designing an MCA is done by using the classical washout filters. The algorithm is a combination of linear filters (as shown in Figure 1). The translational accelerations are high-pass filtered to extract fast dynamics. The resulting signal is integrated to calculate the translational displacement. The slow dynamics or sustained accelerations are extracted by filtering the translational accelerations using a low pass filter. The resulting signals are reproduced by tilting the platform in order to exploit the acceleration due to gravity (Tilt Coordination). The angular velocities are also high pass filtered and then integrated to calculate the angular displacement. The signals from the tilt channel and rotational channel are added to calculate the total angular displacement of the motion platform.

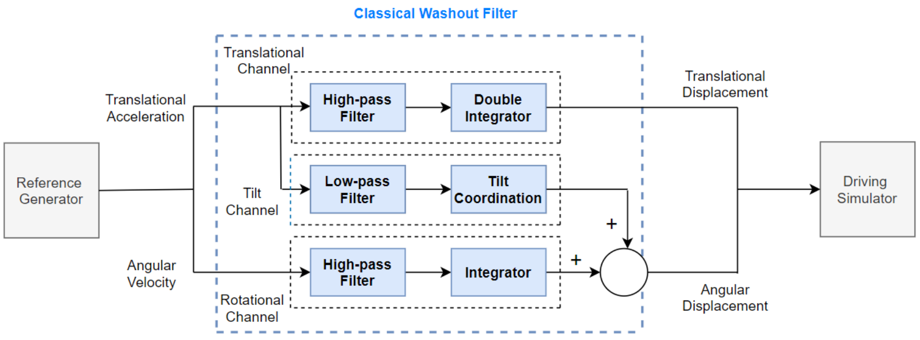

Figure 1. Scheme of classical washout filter-based MCA

As per Nahon et al. [1], the major advantage of using such algorithms is that its design is simple and computationally cheap. However, these algorithms have the following shortcomings as well:

- Because the parameters of the filters are fixed, they must be designed for the worst-case maneuver. As a result, the algorithm doesn't utilize the available workspace for gentle maneuvers and produces minimum motion.

- Tuning the filters is a complex task as the user has to change the filter coefficients instead of changing weights on meaningful physical quantities.
- Since there is no provision for incorporating the physical limits of the motion platform within the algorithm, the filters have to be tuned for each maneuver to ensure that the motion platform remains within its physical limit.

To overcome these limitations, adaptive washout filter-based MCA was developed. It tends to produce more realistic cues when the simulator is near the neutral position and only reduces the fidelity when the simulator is near its physical limits. The algorithm is based on minimizing a cost function comprising of penalties on the tracking error and the platform states. Generally, the optimization is performed by using the steepest descent method. The control scheme of a typical adaptive washout filter is shown in Figure 2. Further, similar to the adaptive washout filter, another scheme called the optimal washout filter is also often used for the motion cueing application. The difference between this approach is that instead of using a gradient descent to minimize a cost, it uses the solution of the algebraic riccati equation to derive the optimal gain. A detailed description of the filter-based algorithms is presented in Appendix B.
Although, the adaptive and the optimal washout-based filters provide a better solution than the classical washout filters [1], the optimization problem is still solved without imposing any constraints on the physical states of the simulator, resulting in sub-optimal utilization of the workspace.

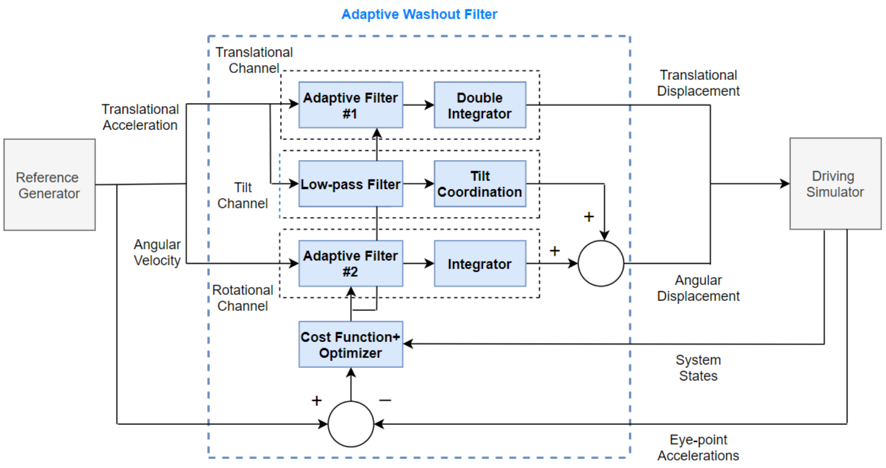

Figure 2. Scheme of adaptive washout filter-based MCA

To address this issue, Model Predictive Control (MPC) technique has been recently applied to design an MCA. Its ability to handle constraints on system states and usage of the prediction model to regulate the current state makes it a well-suited contender for this application. It has been shown that besides producing realistic motion cues, undesired effects like the occurrence of motion sickness are also lowered when using MPC-based MCA compared to the conventional filter-based approaches [2]. Recently, various approaches to MPC-based MCA have been explored and the superiority of this method compared to the conventional approach has been established for offline simulation and passive driving [2,3]. The scheme of a typical MPC-based MCA is shown in figure 3. A detailed derivation of the Linear MPC-based MCA is presented in Appendix C.

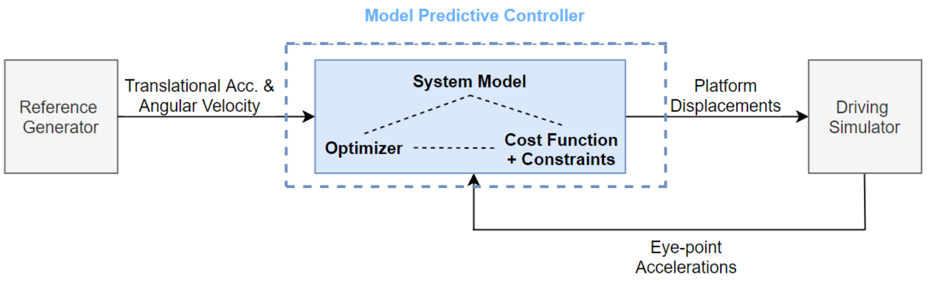

Figure 3. Scheme of MPC-based MCA

To keep the problem linear, the algorithms designed in the past ([4–6]) apply the constraints on the position and velocity of the driver's eye-point. In order to find the available workspace for the eye-point displacement, the forward kinematic relations have to be used which is concerned with determining the displacement of the platform given the position of all the actuators. However for a six DOF motion platform, there are many solutions to the forward kinematic problem [7] and only one of them corresponds to the actual pose of the platform. Generally, Newton-Raphson method is used to iteratively solve the forward kinematics problem. To reduce the computational effort, a conservatively chosen constant space is often used as the workspace for driver's eye-point. However, this tends to produce sub-optimal results. An efficient way to manage the workspace is to use actuator-based constraints instead of the eye-point displacements. Garret et al. [8] derived an MPC-based MCA which uses actuator-based constraints. However, linear approximations were applied to the constraints on the actuator lengths. This simplification also affected constraint handling as the inverse kinematics of the motion platform is nonlinear in nature. Degdelen et al. [9], implemented the MPC-based MCA in the Renault ULTIMATE Simulator. The study was done for a single DOF cueing problem (surge acceleration) and tilt coordination was demonstrated as an extension to the basic algorithm. Taking it further, in [10], an explicit MPC-based concept for the Renault ULTIMATE Simulator was presented. The control problem was decoupled into four separate cases (pitch-surge, roll-sway, heave, and yaw) and a stability condition was determined. The algorithm works in real-time. However, fast degradation in the computational effort was seen as the problem was extended to higher dimensions.

Katliar et al. [11,12] implemented an MPC-based MCA which included the motion platform actuation for a Cable-Robot-based motion simulator. Their main finding was that with proper software and numerical methods, it is possible to run an MPC-based MCA with complex model in real-time.
In this paper, a new MPC-based MCA has been designed that incorporates the non-linear kinematics of the Stewart platform. Inverse kinematics relations are used to calculate the length and the velocity of the actuators, which are included as states within the MPC framework. Moreover, the human vestibular system model is included within the MPC formulation to increase the fidelity of the produced motion cues. To manage the workspace efficiently, constraints are imposed on the actuator displacements and state-dependent adaptive weights are used to tune the MPC algorithm. The formulated non-linear optimization problem is solved using the Real-Time Iteration (RTI) method [13] in order to increase the computational efficiency of the algorithm. Thus, a distinctive feature of this approach is that it aims at developing an efficient algorithm which produces high fidelity motion cueing by using a non-linear MPC-based controller with actuator-based constraints and state-dependent adaptive weights.
This paper is structured as follows. Section 2 describes the motion platform and the frames of references associated with it. The system model used within the MPC controller is derived in Section 3. Section 4 presents the details of the MPC formulation, including the objective function, constraints, reference generation, tuning and the optimization problem. In Section 5, several performance indicators are described. The simulation results and discussion are presented in Section 6. Finally, the conclusions and the recommendations for future work are presented in Section 7.

2. Motion Platform

The motion platform generally used in driving simulators is a Stewart platform, which is a parallel manipulator that is controlled by six actuators. In this paper, the following three frames of reference with respect to the motion platform are used (the same is shown in Figure 4).

1. Inertial Frame (IF) - is fixed to the ground and does not move with the motion platform. The origin coincides with the centroid of the fixed base of the platform (Point O in figure 4). The positive x-axis points forward, in the direction of drive. The positive y-axis points to the right, while the positive z-axis points vertically downwards.
2. Platform Frame (PF) - is fixed to the motion platform and moves with it. The origin coincides with the centroid of the moving plate (Point P_0 in figure 4). Similar to the IF, the positive x-axis points forward, the positive y-axis points to the right, while the positive z-axis points in the downwards direction. Since the HF is body-fixed, its axes are only aligned with that of the IF when the platform has a zero roll, pitch and yaw angle.

3. Driver Frame (DF) - is fixed to the driver's head and moves with it. The origin coincides with the eyepoint of the driver (Point D_0 in figure 4). The positive x-axis points forward, the positive y-axis points to the right, while the positive z-axis points in the downwards direction. Since DF is fixed to the driver's eyepoint, its axes are only aligned with the IF when the platform has a zero roll, pitch and yaw angle.

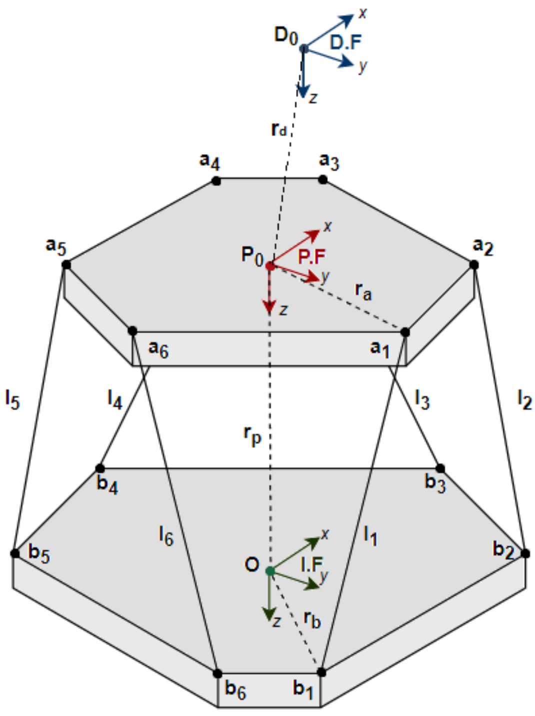

Figure 4. Motion Platform and reference frames used in the MCA

It should be noted that throughout this paper, all the physical quantities are mentioned in IF unless specified by a superscript. Moreover, the translational acceleration vector is represented by the symbol a and consists of components in all the three canonical directions, i.e. $[a_x\ a_y\ a_z]^T$. Similarly, the translational velocity vector is represented by the symbol v and the translational displacement is represented by the symbol r. Further, the angular acceleration vector is divided into rotational and tilt components. This is done to impose constraints on the tilt component without affecting the rotational component. The total angular acceleration is the sum of both the components and is represented by the symbol α. The vector consists of angular accelerations in three canonical directions, i.e. $[\alpha_\phi\ \alpha_\theta\ \alpha_\psi]^T$. Similarly, the angular velocity vector is represented by the symbol ω and the angular displacement is represented by the symbol β.

3. System Model

Model Predictive Control is a model-based optimal control strategy that computes the control input by solving an optimization problem. This is done to obtain the best possible reference tracking performance by predicting future states using the system model. The system model for the MPC algorithm is divided into two sub-parts.

1. Vestibular system model: Responsible for providing realistic motion cues.
2. Motion platform model: Responsible for managing the workspace.

In the following subsections, the vestibular model and motion platform model used in this paper are derived.

3.1. Vestibular System Model

The vestibular system located inside the human ear is primarily responsible for providing cues which we use to perceive motion in space. Within the vestibular system, there are two parts - the semi-circular canals, responsible for sensing the rotational accelerations and otolith organs, responsible for sensing the translations accelerations [5].
Although extensive research had been conducted in the past for modeling the vestibular system mathematically (in [14–18]), reliable linear models have been derived only recently due to the fact that every person has slightly different perception and in general it isn't a linear process [19].

3.1.1. Semi-circular canals

In their research, Telban et al. [19] derived the linear transfer function of the semi-circular canal as follows:

$$\frac{\hat{\omega}_i(s)}{\omega_{i,\ rot}(s)} = 5.73\frac{80s^2}{(1+80s)(1+5.73s)} \tag{1}$$

where $\omega_{i,\ rot}$ is the angular velocity to which the passenger is subjected and $\hat{\omega}_i$ is the perceived angular velocity in one of the three rotational degrees of freedom. In this study, it has been assumed that the parameters of the model used here are the same in all the three rotational degrees of freedom. This assumption is based on the physiological results based on the afferent responses of the semi-circular canals as mentioned by Telban et al. [19]. From here forward both are expressed at the driver's eyepoint (D_0) in DF. Representing Equation 1 in its observable canonical state-space form, we get:

$$\begin{aligned} \dot{x}_{scc} &= A_{scc} \cdot x_{scc} + B_{scc} \cdot u_{scc} \\ y_{scc} &= C_{scc} \cdot x_{scc} + D_{scc} \cdot u_{scc} \end{aligned} \tag{2}$$

where $y_{scc} = \hat{\omega}_i$ and $u_{scc} = \omega_{i,\ rot}$ in one of the three canonical directions. Since this model has to be adopted for each rotational degree of freedom (roll ϕ, pitch θ and yaw ψ) individually, the complete model for semi-circular canals is given as follows:

$$\begin{aligned} \dot{x}_s &= A_s \cdot x_s + B_s \cdot u_s \\ y_s &= C_s \cdot x_s + D_s \cdot u_s \end{aligned} \tag{3}$$

where

$$\begin{aligned} A_s &= \begin{bmatrix} A_{scc_\phi} & 0_{2x2} & 0_{2x2} \\ 0_{2x2} & A_{scc_\theta} & 0_{2x2} \\ 0_{2x2} & 0_{2x2} & A_{scc_\psi} \end{bmatrix} \quad B_s = \begin{bmatrix} B_{scc_\phi} & 0_{2x1} & 0_{2x1} \\ 0_{2x1} & B_{scc_\theta} & 0_{2x1} \\ 0_{2x1} & 0_{2x1} & B_{scc_\psi} \end{bmatrix} \\ C_s &= \begin{bmatrix} C_{scc_\phi} & 0_{1x2} & 0_{1x2} \\ 0_{1x2} & C_{scc_\theta} & 0_{1x2} \\ 0_{1x2} & 0_{1x2} & C_{scc_\psi} \end{bmatrix} \quad D_s = \begin{bmatrix} D_{scc_\phi} & 0 & 0 \\ 0 & D_{scc_\theta} & 0 \\ 0 & 0 & D_{scc_\psi} \end{bmatrix} \end{aligned} \tag{4}$$

and the input and the output signals are shown in equation 5 and 6. It must be noted that both of these quantities are expressed at the driver eyepoint (D_0) in DF.

$$u_s = \omega^{DF}_{D_0,\ rot} = [\omega_{\phi,\ rot}\ \omega_{\theta,\ rot}\ \omega_{\psi,\ rot}]^T \tag{5}$$

$$y_s = \hat{\omega}^{DF}_{D_0} = [\hat{\omega}_\phi\ \hat{\omega}_\theta\ \hat{\omega}_\psi]^T \tag{6}$$

3.1.2. Otolith Organ

As per the results of Telban et al. [19], the linear transfer function for the otolith organ is given as follows:

$$\frac{\hat{a}_i(s)}{a_i(s)} = 0.4 \frac{(1+10s)}{(1+5s)(1+0.016s)} \tag{7}$$

where a_i is the specific acceleration to which the passenger is subjected and $\hat{a}_i$ is the perceived specific acceleration in one of the three translational degrees of freedom. Similar to the semi-circular canals, it has been assumed that the parameters of the otolith model used here are also same in all the three translational degrees of freedom. Moreover, both the input and output are to be specified at the driver's eyepoint (D_0) in DF. Representing Equation 7 in its observable canonical state-space form, we get:

$$\begin{aligned} \dot{x}_{oth} &= A_{oth} \cdot x_{oth} + B_{oth} \cdot u_{oth} \\ y_{oth} &= C_{oth} \cdot x_{oth} + D_{oth} \cdot u_{oth} \end{aligned} \tag{8}$$

where $y_{oth} = \hat{a}_i$ and $u_{oth} = a_i$, in one of the three canonical directions (surge x, sway y or heave z). Therefore, the complete model for otolith organs is as follows:

$$\begin{aligned} \dot{x}_o &= A_o \cdot x_o + B_o \cdot u_o \\ y_o &= C_o \cdot x_o + D_o \cdot u_o \end{aligned} \tag{9}$$

where

$$\begin{gathered} A_o = \begin{bmatrix} A_{oth_x} & 0_{2x2} & 0_{2x2} \\ 0_{2x2} & A_{oth_y} & 0_{2x2} \\ 0_{2x2} & 0_{2x2} & A_{oth_z} \end{bmatrix} \quad B_o = \begin{bmatrix} B_{oth_x} & 0_{2x1} & 0_{2x1} \\ 0_{2x1} & B_{oth_y} & 0_{2x1} \\ 0_{2x1} & 0_{2x1} & B_{oth_z} \end{bmatrix} \\ C_o = \begin{bmatrix} C_{oth_x} & 0_{1x2} & 0_{1x2} \\ 0_{1x2} & C_{oth_y} & 0_{1x2} \\ 0_{1x2} & 0_{1x2} & C_{oth_z} \end{bmatrix} \quad D_o = \begin{bmatrix} D_{oth_x} & 0 & 0 \\ 0 & D_{oth_y} & 0 \\ 0 & 0 & D_{oth_z} \end{bmatrix} \end{gathered} \tag{10}$$

and the input and the output signals are shown in equation 11 and 12. It must be noted that both of these quantities are also expressed at the driver eyepoint (D_0) in DF.

$$u_o = a_{D_0}^{DF} = [a_x \; a_y \; a_z]^T \tag{11}$$

$$y_o = \hat{a}_{D_0}^{DF} = [\hat{a}_x \; \hat{a}_y \; \hat{a}_z]^T \tag{12}$$

The complete model of otolith organ should also include the tilt coordination effects into it. Please refer to Appendix A.4 for a detailed explanation. The otolith matrix is augmented to the following:

$$\begin{gathered} A_{\bar{o}} = \left[\begin{array}{c|c} A_o & \bar{B} \\ \hline 0 & 0 \end{array}\right] \quad B_{\bar{o}} = \left[\begin{array}{c|c} B_o & 0 \\ \hline 0 & I_3 \end{array}\right] \\ C_{\bar{o}} = \left[\begin{array}{c|c} C_o & 0 \end{array}\right] \quad D_{\bar{o}} = \left[\begin{array}{c|c} D_o & 0 \end{array}\right] \end{gathered} \tag{13}$$

where

$$\bar{B} = B_o \cdot \begin{bmatrix} 0 & g & 0 \\ -g & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{14}$$

It must be noted that small-angle approximation has been made here and the input and the output signals expressed at the eyepoint of the driver (D_0) in DF are mentioned in the following equations:

$$u_{\bar{o}} = [a_{D_0}^{DF} \; ; \; \omega_{D_0,\, tilt}^{DF}] \tag{15}$$

$$y_{\bar{o}} = \hat{a}_{D_0}^{DF} \tag{16}$$

3.1.3. Complete Model

The complete vestibular system is further modeled by combining the state-space models of the semi-circular canals and the otolith organ, resulting in the following system:

$$\begin{aligned} \dot{x}_v &= A_v x_v + B_v u_v \\ y_v &= C_v x_v + D_v u_v \end{aligned} \tag{17}$$

where

$$\begin{aligned} A_v &= \begin{bmatrix} A_s & 0_{6x9} \\ 0_{9x6} & A_{\bar{o}} \end{bmatrix} \quad B_v = \begin{bmatrix} 0_{6x6} & B_s \\ B_{\bar{o}} & 0_{9x3} \end{bmatrix} \\ C_v &= \begin{bmatrix} C_s & 0_{3x9} \\ 0_{3x6} & C_{\bar{o}} \end{bmatrix} \quad D_v = \begin{bmatrix} 0_{3x6} & D_s \\ D_{\bar{o}} & 0_{3x6} \end{bmatrix} \end{aligned} \tag{18}$$

and the input and the output signals expressed at the eyepoint of the driver (D_0) in DF are mentioned in the following equations:

$$u_v = [a_{D_0}^{DF} \ ; \ \omega_{D_0,\ tilt}^{DF} \ ; \ \omega_{D_0,\ rot}^{DF}] \tag{19}$$

$$y_v = [\hat{a}_{D_0}^{DF} \ ; \ \hat{\omega}_{D_0}^{DF}] \tag{20}$$

Since the motion platform has to be controlled in the IF, the inputs of the vestibular system must be converted from DF to IF. To transform the translational acceleration from DF to IF, the following relation is used:

$$a_{D_0}^{DF} = a_{P_0} + (\omega_{P_0} \times (\omega_{P_0} \times (R_{PF}^{IF} \cdot r_d^{PF}))) + (\alpha_{P_0} \times (R_{PF}^{IF} \cdot r_d^{PF})) \tag{21}$$

where r_d^{PF} is the vector from point P_0 to the driver′s eyepoint (point D_0) in PF. a_{P_0}, ω_{P_0} and α_{P_0} are the translational acceleration, total angular velocity and total angular acceleration respectively at point P_0 in IF. Further, R_{PF}^{IF} is the rotation matrix for translational acceleration from PF to IF, which can be written in terms of the total inclination angles (roll (ϕ), pitch (θ) and yaw (ψ)) of the motion platform as follows:

$$R_{PF}^{IF} = \begin{bmatrix} cos\phi \cdot cos\psi - sin\phi \cdot cos\theta \cdot sin\psi & -cos\phi \cdot sin\psi - sin\phi \cdot cos\theta \cdot cos\psi & sin\phi \cdot sin\theta \\ sin\phi \cdot cos\psi + cos\phi \cdot cos\theta \cdot sin\psi & -sin\phi \cdot sin\psi + cos\phi \cdot cos\theta \cdot cos\psi & -cos\phi \cdot sin\theta \\ sin\theta \cdot sin\psi & sin\theta \cdot cos\psi & cos\theta \end{bmatrix} \tag{22}$$

Moreover, to transform the rotational velocity from DF to IF, the following relation is used:

$$\omega_{D_0}^{DF} = T_{DF}^{IF} \cdot \omega_{P_0} \tag{23}$$

where T_{PF}^{IF} is the rotation matrix for rotational velocity from DF to IF, which can be written in terms of the total inclination angles (roll (ϕ), pitch (θ) and yaw (ψ)) of the motion platform as follows:

$$T_{DF}^{IF} = \begin{bmatrix} 0 & cos\phi & sin\phi \cdot sin\theta \\ 0 & sin\phi & -cos\phi \cdot sin\theta \\ 1 & 0 & cos\theta \end{bmatrix} \tag{24}$$

3.2. Motion Platform Model

The motion platform model is used by the MPC algorithm to manage the workspace. The workspace of a 6-DOF motion platform is defined as a six-dimensional complex-shaped body where the system is free to move without violating its actuator limitations. The boundaries of the workspace are formed due to the excursion limitations of one or more actuators. The motion-space of the platform is defined as space where the system is free to move in the future as per the current state of the system. Since Stewart platforms are synergistic systems, the movement in a single DOF requires contribution from all the actuators. As a consequence, the available motion-space in one DOF depends on the excursions in other DOFs as well. The following approaches can be used to manage the workspace:

- **Limiting motion workspace** - Forward kinematics is used to calculate the motion-space (as per the current actuator position) in terms of the translational and angular displacement of the point P_0. Further, the constraints are applied based on the current motion-state and the same should be updated at each time-step. The resulting motion-space is a 6-dimensional complex body.
- **Limiting actuator workspace** - Inverse kinematics is used to determine the motion-space directly in terms of the actuator positions. Subsequently, fixed constraints are added based on the permissible actuator length.

Limiting the actuator workspace results in simpler relations. This is because the limit on each actuator is independent of that of the other actuators while the degrees of freedom of the point P_0 are coupled with each other. For example, the available workspace for the surge motion of point P_0 would depend on the current state of the other degrees of freedom (sway, heave, roll, pitch and yaw). Therefore, the bound on the surge motion will be state-dependent and has to be calculated at every time step. Meanwhile, the bound on each actuator will be constant (its excursion or retraction limit) and independent of the other actuators. Therefore in this paper, limiting the actuator workspace is used as the strategy for workspace management.

3.2.1. Actuator Kinematics

The inverse kinematic relations of the Stewart platform can be used to implement the constraints on the actuator length and velocity. The actuator kinematic relations are derived in [20]. As per Figure 4, the following relation for actuator length vector can be derived using vector arithmetic:

$$\vec{L}_i = \vec{r}_p + R_{PF}^{IF} \cdot \vec{r}_a^{\,PF} - \vec{r}_b \tag{25}$$

The actuator length can be written as:

$$l_i = \sqrt{\vec{L}_i \cdot \vec{L}_i} \tag{26}$$

Moreover, the unit vector along the length vector can be written as:

$$\vec{n}_i = \vec{L}_i / l_i \tag{27}$$

Further, the actuator velocity vector can be computed by differentiating Equation 26.

$$\dot{l}_i = Q_1^{-1} \cdot Q_2^{-1} \cdot \begin{bmatrix} v_p \\ \alpha_{P_0} \end{bmatrix} \tag{28}$$

where

$$Q_1^{-1} = \begin{bmatrix} \vec{n}_1^T & ((R_{PF}^{IF} \cdot \vec{r}_{a_1}^{\,PF}) \times \vec{n}_1)^T \\ \vdots & \vdots \\ \vec{n}_6^T & ((R_{PF}^{IF} \cdot \vec{r}_{a_6}^{\,PF}) \times \vec{n}_6)^T \end{bmatrix} \quad Q_2^{-1} = \begin{bmatrix} I_{3\times3} & O_{3\times3} \\ O_{3\times3} & T \end{bmatrix} \tag{29}$$

3.3. Combined system model

The combined system model consisting of both, the vestibular system and the motion platform model. Therefore, the combined system states can be written as following:

$$\dot{x}_c(t) = \begin{cases} \dot{\omega}_{rot,\,P_0} = \alpha_{rot,\,P_0} \\ \dot{\beta}_{rot,\,P_0} = \omega_{rot,\,P_0} \\ \dot{\omega}_{tilt,\,P_0} = \alpha_{tilt,\,P_0} \\ \dot{\beta}_{tilt,\,P_0} = \omega_{tilt,\,P_0} \\ \dot{v}_p = a_{P_0} \\ \dot{r}_p = v_p \\ \dot{x}_v = A_v x_v + B_v \,\cdot\, [a_{D_0}^{DF}\,;\,\omega_{D_0}^{DF}] \\ \dot{l}_i = Q_1^{-1} \cdot Q_2^{-1} \cdot [v_p\,;\,\alpha_{P_0}] \end{cases} \tag{30}$$

where

$$\begin{aligned}
&a_{D_0}^{DF} = a_{P_0} + (\omega_{P_0} \times (\omega_{P_0} \times (R_{PF}^{IF} \cdot r_d^{PF}))) + (\alpha_{P_0} \times (R_{PF}^{IF} \cdot r_d^{PF}))\\
&\omega_{D_0}^{DF} = T \cdot \omega_{P_0}\\
&\alpha_{P_0} = \alpha_{rot,\ P_0} + \alpha_{tilt,\ P_0}\\
&\omega_{P_0} = \omega_{rot,\ P_0} + \omega_{tilt,\ P_0}\\
&\beta_{P_0} = \beta_{rot,\ P_0} + \beta_{tilt,\ P_0}
\end{aligned} \tag{31}$$

Therefore, the state vector x_c can be written as follows:

$$x_c = [\omega_{rot,\ P_0} \quad \beta_{rot,\ P_0} \quad \omega_{tilt,\ P_0} \quad \beta_{tilt,\ P_0} \quad v_p \quad r_p \quad x_v \quad l_i]^T \tag{32}$$

and, the input vector u_c can be written as follows:

$$u_c = [a_{P_0} \quad \alpha_{tilt,\ P_0} \quad \alpha_{rot,\ P_0}]^T \tag{33}$$

This combined system can be represented as the following:

$$\dot{x}_c(t) = f\big(x_c(t), u_c(t)\big) \tag{34}$$

To discretize the system, direct multiple shooting technique is used [21]. The time horizon $[t_0, t_0 + T]$ (where $T = N_p \times Ts$) is divided into N_p sub-intervals $[t_k, t_{k+N_p}]$ and the state trajectory is computed on each sub-interval independently. Further, matching constraints are added to ensure continuity of the optimal state trajectory on the whole horizon [22]. After discretization, the obtained system can be represented as follows:

$$x_c(k+1) = f_d\big(x_c(k), u_c(k)\big) \tag{35}$$

4. MPC Formulation

The MPC controller uses the system model and the current state of the system to predicts the evolution of the future state over a finite prediction horizon (N_p). Using this, the optimal control action is derived over a control horizon (N_c). Then, only the first control input is applied to the real system and the same process is repeated for the next time-step. Therefore, the MPC input can be regarded as a nonlinear state feedback control input, obtained online by repeatedly solving the optimization problem - minimizing an objective function while adhering to the system dynamics and fulfilling the given constraints at every time-step.

4.1. Objective Function

The standard objective function for MPC consists of quadratic functions of both the tracking error and the control action along the prediction horizon. In this paper, the objective function is divided into two parts, namely the stage cost $(\ell(x_c(k), u_c(k)))$ and the terminal cost $(V(x_c(N_p)))$. The total objective function is given as:

$$J(x_c, u_c) = \sum_{k=0}^{N_p-1} \ell(x_c(k), u_c(k)) + V(x_c(N)) \tag{36}$$

The expression for stage cost is shown below:

$$\ell(x_c(k), u_c(k)) = \|x_{ref}(k) - x_c(k)\|_Q + \|u_{ref}(k) - u_c(k)\|_R \tag{37}$$

where Q and R are the positive semi-definite weight matrices for a penalty on tracking error and control input respectively. The stage cost function is defined such that it satisfies the following conditions:

$$\begin{aligned} &\ell(0,0) = 0 \\ &\ell(x_c(k), u_c(k)) > 0,\ \forall x(k) \in \mathbb{X},\ x(k) \neq x_{ref}(k) \end{aligned} \tag{38}$$

The expression for the terminal cost is shown below:

$$V(x_c(N_p)) = \|x_{ref}(N_p) - x_c(N_p)\|_P \tag{39}$$

where P is the positive semi-definite weight matrix for a penalty on the tracking error at the terminal stage of the prediction horizon.
For finite prediction horizon problems, stability can be guaranteed by choosing a suitable terminal cost (V) and terminal attractive region Ω [23–25]. Even though the conditions for asymptotic stability are clearly defined, choosing V and Ω is still an open problem [26]. It is shown in [27], that stability can be guaranteed by simply tuning the matrices Q, R, and P. Further, a longer prediction horizon (N_p) would help the algorithm to achieve convergence at the cost of a more computational demanding problem.

4.2. Constraints

For motion cueing, the following constraints are generally applied:

- Constraint on the tilt rate (ω).
- Constraints on the actuator positions (l_i).

The constraint on the tilt rate is to ensure that the tilt coordination effects are not perceived by the observer. Therefore, the tilt-rate should be limited to the threshold values for rotation. In the paper, the tilt rate constraints are imposed based on the values derived in the research of Reid et. al [28] and the same is mentioned in Table 1.

Table 1. Threshold values for rotational velocities

Degree of Freedom	Threshold Value
Roll ω_ϕ	3.0 deg/s
Pitch ω_θ	3.6 deg/s
Yaw ω_ψ	2.6 deg/s

$$\omega_{tilt,\,\min} \leq \omega_{tilt} \leq \omega_{tilt,\,\max} \tag{40}$$

Further, the constraints on the actuator positions are to ensure that the platform remains within its physical limits. This is expressed as the maximum extension (l_{max}) and retraction (l_{min}) of the actuator allowed.

$$l_{min} \leq l_i \leq l_{max} \quad i = 1, \ldots, 6 \tag{41}$$

In this paper, the set of states $x(k)$ which satisfies the aforementioned constraints is denoted by $\mathbb{X}$. Therefore, the combined constraint equations are represented as:

$$x(k) \in \mathbb{X} \tag{42}$$

4.3. *Reference Generation*

The reference vector contains the following variables:

$$x_{ref} = [\omega_{rot,\,ref} \quad \beta_{rot,\,ref} \quad \omega_{tilt,\,ref} \quad \beta_{tilt,\,ref} \quad v_{p,\,ref} \quad r_{p,\,ref} \quad x_{v,\,ref} \quad l_{i,\,ref}]^T \tag{43}$$

To ensure that the platform returns to neutral position, the reference for ω_{rot}, β_{rot}, ω_{tilt}, β_{tilt}, v_p, r_p, l_i are set to zero for the entire prediction horizon. Further, the reference for x_v contains the reference for $\hat{a}^{DF}_{D_0}$ and $\hat{\omega}^{DF}_{D_0}$. These are computed by translating the translational acceleration and angular velocities obtained from the vehicle model to the driver's eyepoint in DF and then passing them through the vestibular system model. Since the future reference is not available in advance, the current value of x_v is kept constant throughout the prediction horizon.

4.4. *Adaptive weight-based tuning*

The weight matrices used in the above formulation are as follows:

$$Q = diag([\boldsymbol{W}_{\omega_{rot}}\ \boldsymbol{W}_{\beta_{rot}}\ \boldsymbol{W}_{\omega_{tilt}}\ \boldsymbol{W}_{\beta_{tilt}}\ \boldsymbol{W}_{v_p}\ \boldsymbol{W}_{r_p}\ \boldsymbol{W}_{x_v}\ \boldsymbol{W}_{l_i}]) \tag{44a}$$

$$R = diag([\boldsymbol{W}_{a_{P_0}}\ \boldsymbol{W}_{\alpha_{tilt}}\ \boldsymbol{W}_{\alpha_{rot}}]) \tag{44b}$$

$$P = diag([\boldsymbol{w}_{\omega_{rot}}\ \boldsymbol{w}_{\beta_{rot}}\ \boldsymbol{w}_{\omega_{tilt}}\ \boldsymbol{w}_{\beta_{tilt}}\ \boldsymbol{w}_{v_p}\ \boldsymbol{w}_{r_p}\ \boldsymbol{w}_{x_v}\ \boldsymbol{w}_{l_i}]) \tag{44c}$$

In the conventional MPC scheme, the weights of these tuning matrices are fixed in advance. This approach works well if the individual states are not dependent

on each other or if the properties of the system do not change during the course of the simulation. However, since the motion cueing algorithm needs to adapt to the changing workspace, an adaptive weight-based tuning approach has been implemented in this paper. The fundamental idea of this approach is to increase the weight on the position error ($\boldsymbol{W}_{r_p}$) and velocity error ($\boldsymbol{W}_{v_p}$) non-linearly as the motion platform reaches near the actuator limit.
The adaptive weights-based tuning results in two main advantages. Firstly, for a constrained MPC problem with a short prediction horizon, the resulting output trajectories are often not smooth when the system states reach the limits of the workspace. This is because the penalty on the system states is constant irrespective of the available workspace. By varying the tuning weights-based on the available workspace, a damping action is provided which results in smooth movement of the platform near the physical limit.
Secondly, the motion cueing algorithms have a tendency to produce false or missing cues as they constantly try to perform washout. By keeping the weights on v_p and r_p low and only increasing it when the platform is near its limits, the adaptive weights-based tuning would also help to reduce the production of false or missing cues. The following weight function is used in this paper:

$$\boldsymbol{W}_{r_p} = \left(\sum_{i=1}^{6} \frac{1}{(1.1 \cdot l_{max})^2 - l_i^2} - a \right) / b \tag{45a}$$

$$\boldsymbol{W}_{v_p} = \left(\sum_{i=1}^{6} \frac{1}{(1.1 \cdot l_{max})^2 - l_i^2} - c \right) / d \tag{45b}$$

where a, b, c and d are fixed parameters. The effect of the aforementioned adaptive weight function can be seen in figure 5. Parameters a and c determine the value of the function at point m in Figure 5, while parameters b and d determine the value of the function at point n_1 and n_2 in Figure 5.

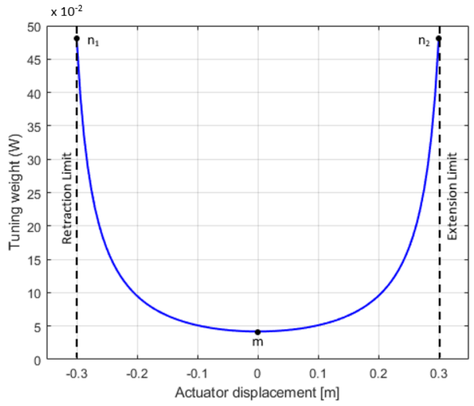

Figure 5. The effect of actuator displacement on the tuning weight

Table 2. Constant tuning parameters

Parameter	$W_{\omega_{rot}}$	$W_{\beta_{rot}}$	$W_{\omega_{tilt}}$	$W_{\beta_{tilt}}$	$W_{y_v,\hat{a}}$	$W_{x_v,\hat{\omega}}$	W_{l_i}
Weight	$0.1e^{-2}$	$0.1e^{-2}$	$0.1e^{-2}$	$0.1e^{-2}$	$2e^{-2}$	$10e^{-2}$	$0.1e^{-2}$

It should be noted that the other tuning weights (except $\boldsymbol{W_{r_p}}$ and $\boldsymbol{W_{v_p}}$) in Equation 44 are kept constant and are shown in Table 2. Since the magnitude of the perceived angular velocity is much smaller than that of the perceived translational acceleration, a high weight for it is chosen (i.e. $W_{x_v,\hat{a}} \ll W_{x_v,\hat{\omega}}$). This scaling of weights is important so that the errors on both the quantities are given equal weightage and the MPC algorithm seeks to track both the quantities equally. Moreover, since the actuator displacements are already constrained and within the available workspace, free movement of the actuators is desired, a small weight is selected for the actuator displacement (W_{l_i}). Similarly, since the tilt rate is already constrained and free rotatory movement of the hexapod is desired, a low weight on the hexapod inclination angle ($W_{\beta_{rot}}$ and $W_{\beta_{tilt}}$) and inclination velocity ($W_{\omega_{rot}}$ and $W_{\omega_{tilt}}$) is chosen.

4.5. Optimization Problem

Model Predictive Control calculates the optimal control input by solving the following optimization problem:

$$\begin{aligned} & u(k) = argmin\ J\big(x_c(k), u_c(k)\big) \\ \text{s.t} & \\ & x_c(k+1) = f_d\big(x_c(k), u_c(k)\big) \\ & x \in \mathbb{X} \end{aligned} \tag{46}$$

To solve the aforementioned optimization problem, ACADO toolkit [29] is used which uses a real-time iteration (RTI) method to solve the nonlinear MPC problem. As mentioned before, multiple shooting technique is used to discretize the nonlinear continuous-time system. The objective function, which is arranged in the least-squares form is solved using the Sequential Quadratic Programming (SQP) technique. The RTI scheme uses the warm-start technique with shifting procedure to linearize the system, i.e the solution of the optimization problem at the previous time-step is shifted and used as the new linearization point. To reduce the number of optimization variables in the QP problem, a condensing procedure is used [21]. The resulting condensed QP problem is then passed to the qpOASES solver [30] which uses the active set method to evaluate the solution. In order to reduce the computational time and solve the problem quickly, the RTI method divides the optimization problem into two parts, i.e:

- Preparation step- The objective function is evaluated in the form of unknown state feedback x_0. The original QP problem is formulated and condensed into a smaller and denser QP.

- Feedback step- The state feedback x_0 is substituted and the QP is solved to obtain the control input.

The preparation step is performed at the previous time-step. As soon as the state feedback x_0 is obtained, it is substituted in the QP and the solution is obtained. A detailed description of the solution techniques used in this paper is given in Appendix D. Further, more about the ACADO toolkit and the real-time iteration method can also be found in [13,29]. In this paper, a sampling time of 0.01 seconds was used with a prediction horizon $N_P = 50$ and the control horizon N_C equal to N_P. The time required by the solver to solve the OCP is shown in figure 8. Further, details about the effects of the sampling time and prediction horizon on tracking performance and computational load is given in Appendix E.

5. Performance Indicators

To compare different motion cueing algorithms, specific performance indicators must be specified. Further, these indicators must be chosen to compare both, the reference tracking performance and workspace utilization of the MCA.

5.1. Indicators for reference tracking performance

Root mean square error is a good indicator of the error in reference tracking. Further, Bourke [31] defined dedicated indicator to measure the shape correlation and the delay between two signals.

Root Mean Square Error (RMSE) calculated for each time step is added and the result is normalized so that the indicator can compare short and long signals fairly. RMSE is given in Equation 47. The range of RMSE indicator is $[0, +\infty]$. A signal is close to the reference should have RMSE close to zero.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_{ref,\,i} - x_i\right)^2} \tag{47}$$

Correlation Coefficient (CC) is the shape correlation between the reference and the actual signal. CC is given in Equation 48. The range of CC indicator is $[0, +1]$. If the two signals are similar in shape, then the CC should be close to one, while it should be close to zero when there is low correlation [31]. This indicator can be particularly useful to signify if there are many missing or false cues.

$$CC(x_{ref}, x) = \frac{\sum_{i=1}^{n}\left(x_{ref,\,i} - \bar{x}_{ref}\right)\cdot\left(x_i - \bar{x}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_{ref,\,i} - \bar{x}_{ref}\right)^2}\cdot\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}} \tag{48}$$

where

$$\bar{x}_{ref} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_{ref,\,i}$$
$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \tag{49}$$

Estimated Delay (ED) calculates the magnitude of delay between the reference and the actual signal. Since both the signals are not exactly equal, actual delay cannot be calculated. Therefore it is estimated as the offset applied to the reference signal which maximizes the correlation coefficient. The range of ED indicator is $[0, +\infty]$. A signal with no delay with respect to the reference should have $ED = 0$.

5.2. Indicators for workspace utilization

Interquartile range (IQR) of the actuator length can be used to analyze how an MCA uses the available actuator workspace [32]. It is a measure of variability and is defined as the difference between the 75th and 25th percentile of the given sample. A high interquartile range denotes high usage of the actuator workspace.

6. Results and Discussion

6.1. Simulation Setup

The control scheme of the experiment is shown in figure 6.

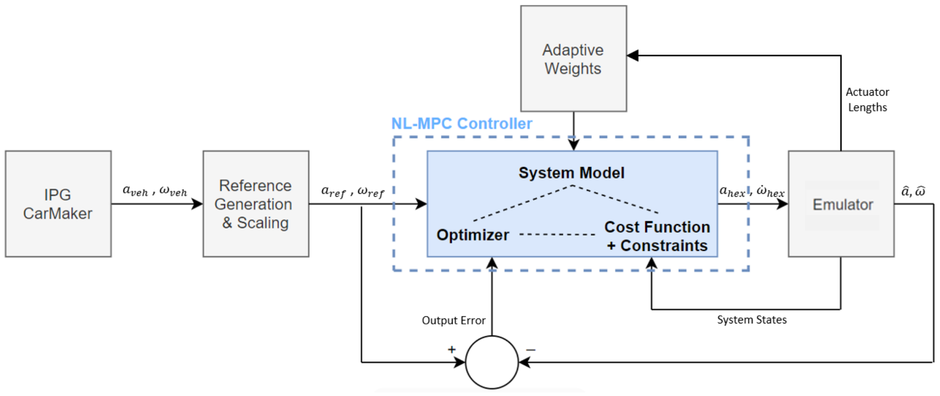
Adaptive Weights
Actuator Lengths
NL-MPC Controller
IPG CarMaker
a_{veh} , ω_{veh}
Reference Generation & Scaling
a_{ref} , ω_{ref}
System Model
Optimizer
Cost Function + Constraints
a_{hex}, $\dot{\omega}_{hex}$
Emulator
$\hat{a}, \hat{\omega}$
Output Error
System States
+
−

Figure 6. The control scheme used for full-track simulations

A four-seater hatchback car with an electric motor and a continuously variable transmission (CVT) was simulated on a digital version of the Hockenheim ring (Germany) in the IPG Carmaker software. The vehicle was simulated for a single lap on the circuit. A virtual sensor was placed on the eyepoint of the driver to record the accelerations and angular velocities. The resulting signal was recorded, passed on to the vestibular system model and the output was fed to the reference generator. Since a 1:1 reproduction of these quantities is often not possible due to limited workspace of the motion platform, a scaling factor is needed. Moreover, in their research, Grácio et al. [33] established that a 1:1 ratio of the inertial and visual cues are reported as too strong by the subjects and thus, not preferred. They reported that the optimal scaling factor, called as optimal gain depends on the amplitude and the frequency of the stimuli. It was also reported that the preferred motion gain decreases with the increase of the stimuli amplitude. Taking this and the capabilities of the motion platform into consideration, a scaling factor of 0.5 applied to the reference quantities and the resulting signals are passed to the controller as the reference signal. At every time-step, the adaptive weights are calculated based on the actuator lengths. Further, the controller receives the system states and the output error from the plant and calculates the control input using the non-linear MPC scheme. The calculated control input, i.e. the translational and rotational acceleration of the moving base centroid of the platform is passed to the platform emulator which is configured to emulate the performance of the Delft Advanced Vehicle Simulator (DAVSi) (shown in Figure 7).

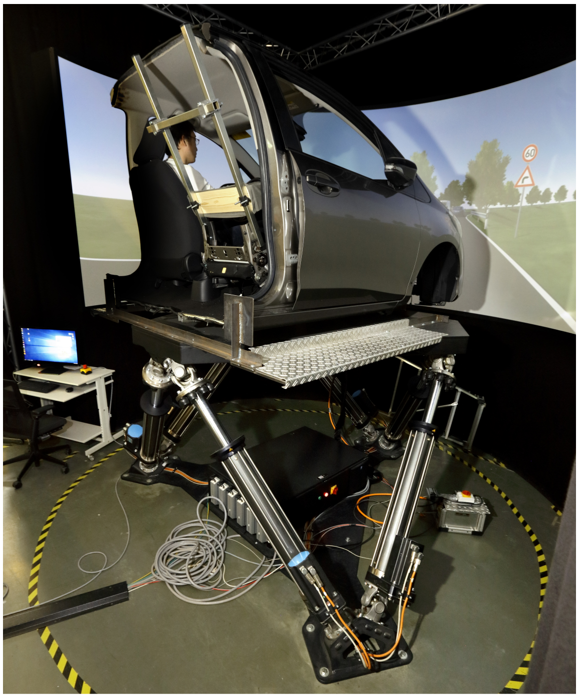

Figure 7. The Delft Advanced Vehicle Simulator

The system performance of the platform is summarized in table 3.

Table 3. System performance of the motion platform

Motion	Excursion $[m]$	Velocity $[m/s]$	Acceleration $[m/s^2]$
Surge x	-0.51 ... 0.63	$\pm$ 0.81	$\pm$ 7.1
Sway y	-0.51 ... 0.51	$\pm$ 0.81	$\pm$ 7.1
Heave z	-0.42 ... 0.42	$\pm$ 0.61	$\pm$ 10.0
Roll ϕ	$\pm$ 24.3	$\pm$ 35.0	$\pm$ 260.0
Pitch θ	-25.4 ... 28.4	$\pm$ 38.0	$\pm$ 260.0
Yaw ψ	$\pm$ 25.0	$\pm$ 41.0	$\pm$ 510.0
Actuator	-1.297 ... 1.937	-	-

The performance of the algorithm is analyzed based on the performance indicators mentioned in Section 5. The results of the proposed nonlinear MPC-based MCA (NLMPC) are compared with Linear MPC (LMPC) and the classical washout filter (CWF) based MCAs. All the algorithms were tuned to maximize the reference tracking performance while keeping the actuator positions within the physical limits. The simulations were performed on a standard Intel Core i7 2.6 GHz system with 16 GB RAM and x64 bit Windows 10 operating system. Further, to test the real-time capabilities of the algorithm, the execution time required by the ACADO solver to solve the OCP was recorded and the same is shown in Figure 8.

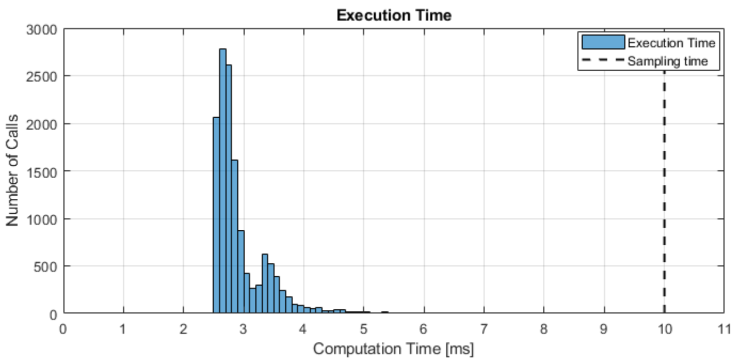

Figure 8. The execution time required by ACADO to solve the OCP

It can be inferred from the figure that the time taken by the solver at each time-step throughout the simulation is less than the sampling time (0.01 sec), making it feasible to implement in real-time.

6.2. Simulation Results

The reference tracking performance for perceived acceleration are shown in Figure 10,9,11 while the reference tracking performance for the angular velocities is shown in Figure 12,13,14. Further to analyze the workspace utilization, actuator lengths spanned during the course of this simulation are shown in Figure 15.

Reference Tracking Performance: Translational Acceleration

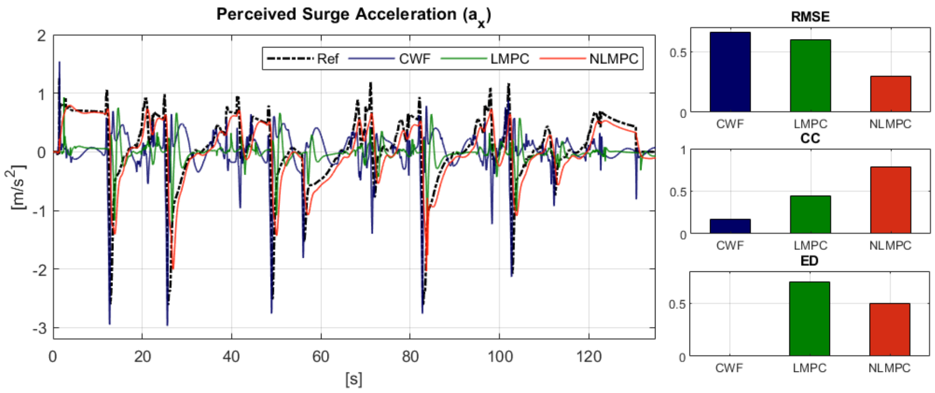

Figure 9. Reference Tracking performance: Perceived Surge Acceleration

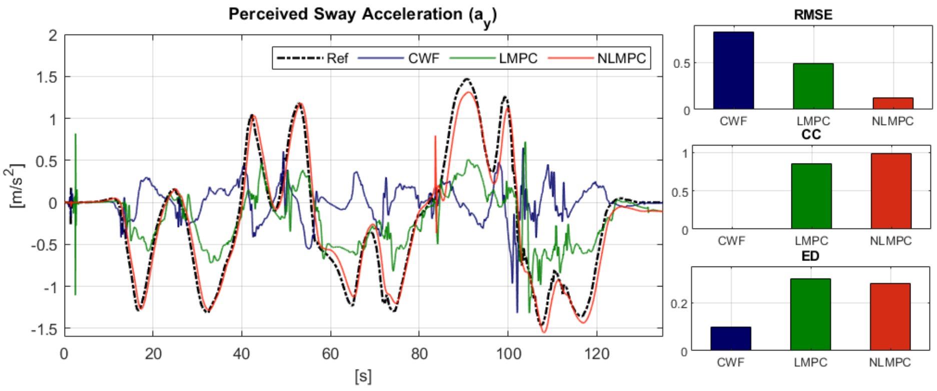

Figure 10. Reference Tracking performance: Perceived Sway Acceleration

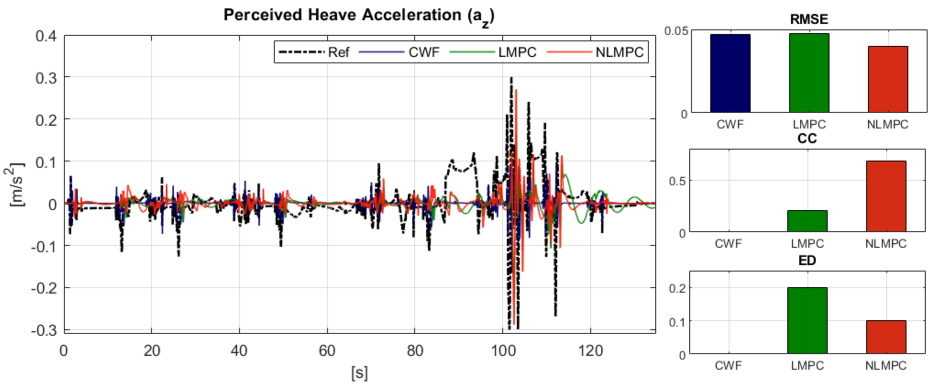

Figure 11. Reference Tracking performance: Perceived Heave Acceleration

From Figure 9, it can be inferred that the NLMPC algorithm results in a lower RMSE value compared to the other algorithms. Moreover, the high CC value indicates a high shape correlation. The RMSE and CC values for the CWF and LMPC algorithms indicate a comparatively inferior performance. While the LMPC algorithm produces better results than CWF, both the algorithms produce false or missing cues. It can also be seen that both the MPC-based algorithms result in higher ED value compared to the CWF algorithm. This behavior of the MPC-based algorithms can be improved if the reference is known a priori.
Similar conclusions can be drawn from Figure 10. The NLMPC algorithm produces produces a superior reference tracking performance, which is reflected in the low RMSE and high CC values. Although the LMPC algorithm results in significant RMSE value, its CC value is high. This is because it produces the right but scaled-down cues resulting in high shape correlation but high error as well. Moreover, the performance of CWF is again inferior compared to the other two algorithms.
From Figure 11, it can be inferred that since the value of the reference signal is small, all the three algorithms result in comparatively lower RMSE values. The NLMPC algorithms produces a high CC value compared to the other algorithms. As mentioned before, the ED value is high for both the MPC-based algorithms in all the above cases, which means that the produced cues are delayed. A reference prediction strategy can be used to improve this behavior.

Reference Tracking Performance: Rotational Velocity
The reference tracking performance for the perceived angular velocity is shown in figure 12, 13 and 14. Since most of the reference cues are below the perception threshold, the quality of the produced cues in such cases does not matter as long as it is below the threshold. Therefore, the MCA performance should be judged based on the quality of the cues above the perception threshold.

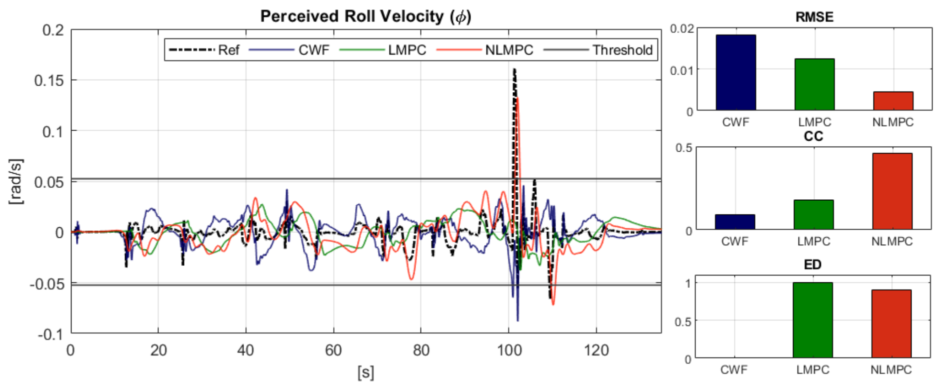

Figure 12. Reference Tracking performance: Perceived Roll Velocity

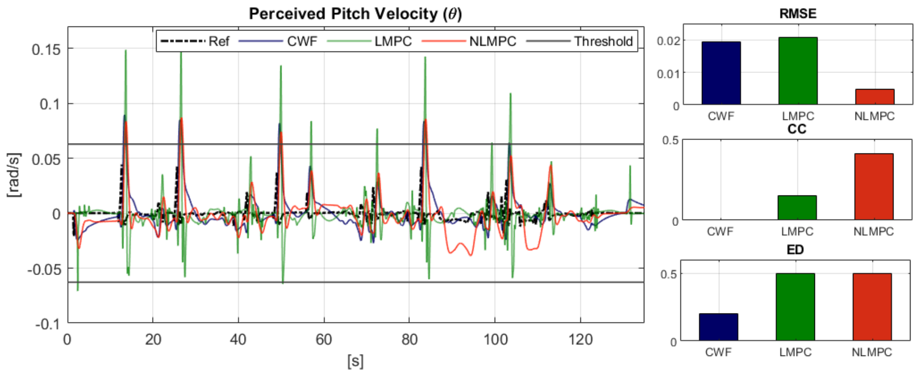

Figure 13. Reference Tracking performance: Perceived Pitch Velocity

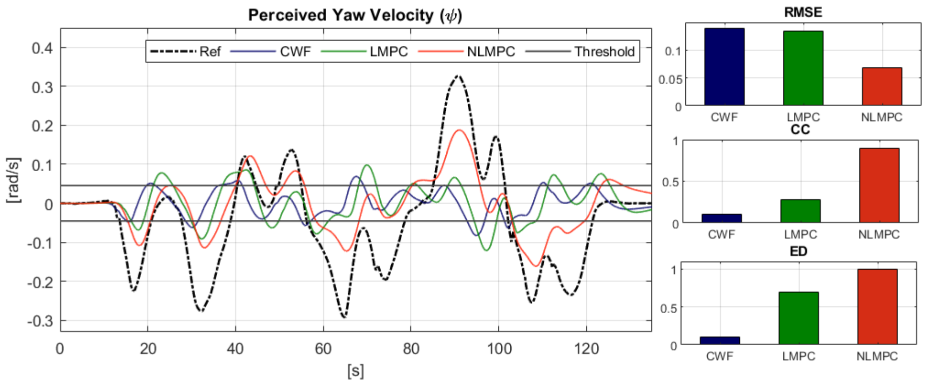

Figure 14. Reference Tracking performance: Perceived Yaw Velocity

From the above figures, it can be inferred that the NLMPC algorithm outperforms the LMPC and CWF algorithm in terms of both, the RMSE and the CC values. Moreover, a high ED value is observed for both the MPC-based algorithms.
From Figure 13, it can be seen that the LMPC algorithm produces perceivable false cues. For every peak that the algorithm tracks on the positive side, it produces an opposite peak in the negative side, resulting in a false cue. This is because after producing the cue form the high peak, the algorithm quickly tries to bring back the platform to the neutral position (washout effect), resulting in the production of the false cue. In NLMPC algorithm, this behavior is governed by the adaptive weight-based tuning scheme. The weights on the position and velocity of the hexapod is only increased when the platform is near its limits. Therefore, the washout process becomes effective only when the platform is near the limits which reduces the tendency of the algorithm to produce false cues.

Workspace Utilization: Actuator displacement

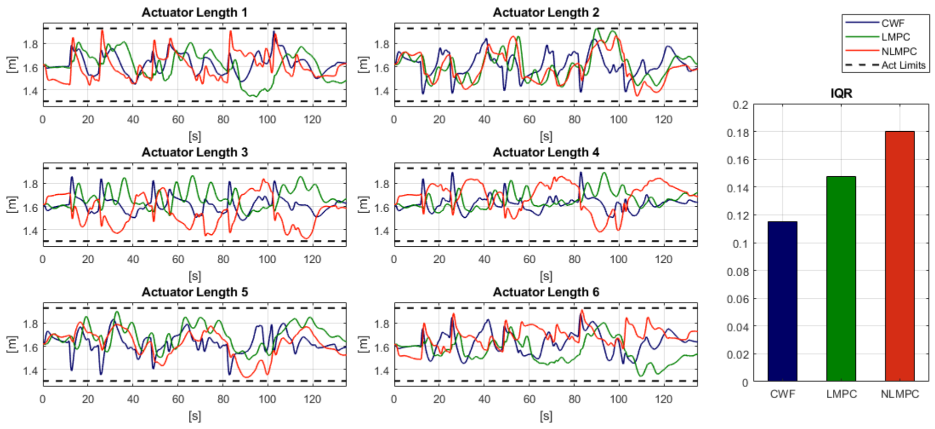

Figure 15. Workspace Utilization: Actuator Displacement and Mean IQR

From Figure 15, it can be inferred that the CWF algorithm utilizes the available workspace conservatively, as reflected by the lower IQR value. This can be attributed to the fact that the algorithm was tuned as per the limiting excursion (excursion of actuator 1 at 102 sec). Therefore, during the other parts of the simulation, the algorithm utilizes the workspace conservatively. The LMPC scheme allows overcoming this limitation as the algorithm has a better knowledge of the platform limits and the same is taken into account while optimizing at each time step to obtain the control action. This results in a higher IQR value for the LMPC algorithm. Meanwhile, the adaptive tuning scheme further allows the NLMPC algorithm to span more workspace as the washout effect becomes effective only when the platform is near its physical limits resulting in a higher IQR value.

Additionally, extensive results in supplement to the aforementioned results are presented in Appendix E.

7. Conclusion

This paper aimed at developing a new nonlinear MPC-based motion cueing algorithm that incorporates the vestibular system model and the non-linear kinematic model of the Stewart platform. The human vestibular system was modeled and the tilt coordination scheme was incorporated within it. To incorporate constraints on the rate of g-tilting without affecting the production of roll, pitch or yaw cues, the rotational velocity states were decoupled into separate rotational states (for actual rotational motion) and tilt states (for tilt coordination) and constraints were imposed only on the latter. In the motion platform model, the actuator positions and velocities were modeled by using the nonlinear inverse kinematics of the Stewart platform

and these were included as states within the MPC framework. Further, the actuator displacements were constrained. Lastly, to manage the actuator workspace efficiently and attain smoother movement of the platform, an adaptive weight-based tuning methodology was proposed which changes the tuning weights on the platform displacement and velocities as per the available actuator motion-space.
To test the proposed algorithm, full-track simulations were performed and the performance of the proposed algorithm was compared to the classical washout filter and linear MPC-based MCA. Based on the literature, several performance indicators were defined to objectively evaluate and compare the reference tracking and workspace utilization performance of different MCAs.
The results showed superior performance of the proposed algorithm in terms of reference tracking when compared with the Linear MPC and CWF-based algorithms. It was further noted that the algorithm produced less false or missing cues compared to the classical washout filter and linear MPC-based MCA which might reduce the chances of motion sickness. Lastly, the proposed algorithm showed better workspace utilization when compared to the Linear MPC and CWF-based algorithms.
The controller frequency achieved in this paper is 100 Hz with a prediction horizon of 50. It is further shown that the optimization at each step requires less than 0.01 seconds which makes it feasible to implement it in real-time. As future work, the algorithm can be tested on a real-time system to evaluate it further and validate the findings of this paper. Other future work could include a subjective evaluation of the MCA in active and passive driving conditions.

References

1. Nahon, M.; Reid, L. Simulator motion-drive algorithms - A designer's perspective. *Journal of Guidance, Control, and Dynamics*, **1990**, *13*, 356–362.
2. Lamprecht, A.; Steffen, D.; Haecker, J.; Graichen, K. Comparision between a filter and an MPC-based MCA in an offline simulator study. In proceedings of Driving Simulation Conference (DSC), Europe. **2019**, 101-–107.
3. Venrooij, J.; Cleij, D.; Katliar, M.; Pretto, P.; Bülthoff, H.; Steffen, D.; Hoffmeyer, F.; Schoener, H. Comparison between filter- and optimization-based motion cueing in the Daimler Driving Simulator. In proceedings of Driving Simulation Conference and Exhbition, Paris, France. **2016**.
4. Baseggio, M.; Bruschetta, M.; Maran, F.; Beghi, A. An MPC approach to the design of motion cueing algorithms for driving simulators.In proceedings of 14th IEEE international conference on Intelligent Transportation Systems (ITSC), **2011**, 692–697.
5. Augusto,B.; Loureiro, R. Motion cueing in the Chalmers driving simulator: a model predictive control approach. MSc. Thesis, Chalmers university of Technology, 2009.
6. Bruschetta, M.; Maran, F.; Beghi, A. A fast implementation of MPC based motion cueing algorithms for mid-size road vehicle motion simulators. *Vehicle System Dynamics*, **2017**, *51*, 802–826.
7. Husty, M. An Algorithm for Solving the Direct Kinematics of General Stewart-Gough Platform.*Mech. Mach. Theory*, **1996**, *4*, 365-—380.
8. Garrett, N.; Best, M. Model predictive driving simulator motion cueing algorithm with actuator-based constraints, *Vehicle System Dynamics*, **2013**, *51*, 1151–1172.
9. Dagdelen, M.; Reymond, G.; Kemeny, A. Model-based predictive motion cueing strategy for vehicle driving simulators. *Control Engineering Practice*, **2009**, *17*, 995–1003.

10. Fang, Z.; Kemeny, A. Motion cueing algorithms for a real-time automobile driving simulator. In proceedings of Driving Simulation Conference, Paris, **2012**.
11. Katliar, M.; Fischer, J; Frison, G.; Diehl, M.; Teufel, H.; Bülthoff, H. Nonlinear Model Predictive Control of a Cable-Robot-Based Motion simulator. IFAC-PapersOnLine, **2017**, *50(1)*, 9833-9839.
12. Katliar, M.; Olivari, M; Drop, F.; Nooij, S.; Diehl, M.; Bülthoff, H. Offline motion simulation framework: Optimizing motion simulator trajectories and parameters. Transportation Research Part F: Traffic Psychology and Behaviour, **2019**, *66*, 29-46.
13. Gros, S.; Zanon, M.; Quirynen, R.; Bemporad, A.; Diehl, M. From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control*, **2011**, 1–19.
14. Fernandez, C.; Goldberg, J. Physiology of peripheral neurons innervating otolith organs of the squirrel monkey. III. response dynamics. *Journal of neuro-physiology* **1976**, *39*, 996—1008.
15. Mayne, R. A Systems Concept of the Vestibular Organs. In *Vestibular System Part 2: Psychophysics, Applied Aspects and General Interpretations. Handbook of Sensory Physiology*; Kornhuber, H.H., Ed.; Springer, Berlin, Heidelberg, Germany, **1974**.
16. Young, L.; Oman, C. Model for vestibular adaptation to horizontal rotation. *Journal of Aerospace medicine*, **1969**, *40*, 1076-–1077.
17. Grant, W.;Best, W. Otolith-organ mechanics: lumped parameter model and dynamic response. *Aviation, Space and Environmental Medicine*, **1987**, *58*, 970-–976.
18. Ormsby, C. Model of human dynamic orientation. PhD thesis, Massachusetts Institute of Technology, USA, 1974.
19. Telban, R. J.; Cardullo, F. M. Motion cueing algorithm development: Human-centered linear and nonlinear approaches. NASA Tech Report CR-2005-213747, **2005**, Available online: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20050180246.pdf (accessed on 8 May 2020).
20. Harib, K.; Srinivasan, K. Kinematic and dynamic analysis of Stewart platform-based machine tool structures. *Robotica*, **2003**, *21*, 541–554.
21. Bock, H.; Plitt, K. A multiple shooting algorithm for direct solution of optimal control problems. In Proceedings 9th IFAC World Congress Budapest, **1984**, 243-–247.
22. Vukov, M.; Domahidi, A.; Ferreau, H.; Morari, M.; Diehl, M. Auto-generated algorithms for nonlinear model predictive control on long and on short horizons. In proceedings of 52nd IEEE Conference on Decision and Control, Florence, Italy. **2013**, 5113–5118.
23. Magni, L.; Nicolao, G.; Magnani, L.; Scattolini, R. A stabilizing model-based predictive control algorithm for nonlinear systems. *Automatica*, **2001**, *37*, 1351-–1362.
24. Mayne, D.; Rawlings, J.; Rao, C.; Scokaert, P. Constrained model predictive control: Stability and optimality. *Automatica*, **2000**, 789—814.
25. Chen H.; Allgower, F. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, **1998**, *34*, 1205-–1217.
26. Abdelaal, M.; Franzle, M.; Hahn, A. Nonlinear Model Predictive Control for Tracking of Underactuated Vessels under Input Constraints. In proceedings of 2015 IEEE European Modelling Symposium, Madrid, **2015**, 313–318.
27. Grune, L.; Pannek, J. Stability and Suboptimality Without Stabilizing Terminal Conditions. In *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer, Switzerland, **2011**.
28. Reid, L.; Nahon, M. Flight simulation motion-base drive algorithms: Part 1 - developing and testing the equations. UTIAS Report No. 296, CN ISSN0082-5255, Institute for Aerospace Studies, University of Toronto, Toronto,Canada, **1985**.
29. Houska, B.; Ferreau, H.; Diehl, M. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, **2011**, *32*, 298–312.
30. qpOASES Homepage. Available online: http://www.qpoases.org (accessed on 8 May 2020).
31. Bourke P. Cross Correlation - 2D Pattern Identification, Available online: http://paulbourke.net/miscellaneous/correlate/ (accessed on 8 August 2020)
32. Grottoli, M.; Cleij, D.; Pretto, P.; Lemmens, Y.; Happee, R.; Bülthoff, H. H. Objective evaluation of prediction strategies for optimization-based motion cueing. *Simulation*, **2018**, *95*, 707-–724.

33. Grácio, B.; van Paassen, M.; Mulder, M.; Wentink, M. Tuning of the lateral specific force gain based on human motion perception in the Desdemona simulator. In proceedings of AIAA Modeling and Simulation Technologies Conference, Toronto, Canada, **2010**.

A
Background

This appendix gives an overview of the fundamentals of motion cueing. It is imperative to understand these fundamentals in order to develop a profound understanding of the motion cueing process. Section A.1 gives an overview of the general framework of motion cueing. In section A.2, a description of the Stewart platform commonly used in driving simulators is provided. In section A.3, the fundamentals of motion perception in humans are described. Lastly, in section A.4, the tilt coordination technique is explained. An extensive description of these topics can also be found in the literature review report associated this thesis.

A.1. GENERAL FRAMEWORK OF MOTION CUEING

Motion cueing deals with the basic problem of providing the passenger sitting in the simulator with the same sensory stimuli which s/he feels in the real vehicle while keeping the motion platform within its physical boundaries. Figure A.1 shows the flow of information in a typical motion cueing scheme.

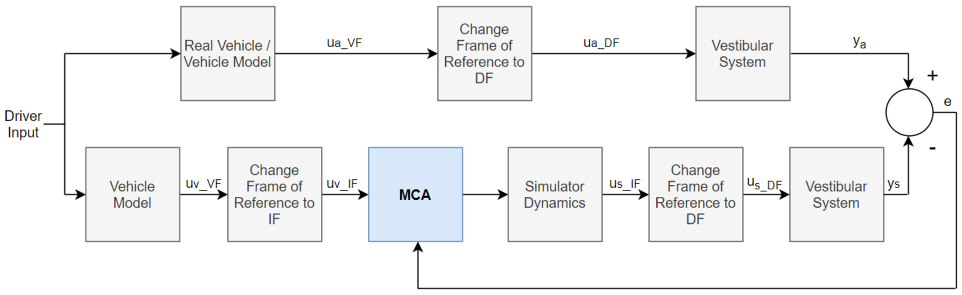
Driver Input
Real Vehicle / Vehicle Model
ua_VF
Change Frame of Reference to DF
ua_DF
Vestibular System
ya
+
e
-
Vehicle Model
uv_VF
Change Frame of Reference to IF
uv_IF
MCA
Simulator Dynamics
us_IF
Change Frame of Reference to DF
us_DF
Vestibular System
ys

Figure A.1: Typical motion cueing scheme

The upper branch provides the reference signals for the Motion Cueing Algorithm (MCA). The input command from the driver (steering input and throttle or brake input) is passed to the real vehicle or vehicle model and the resultant accelerations and rotational velocities (u_{a_VF}) are measured at the Centre of Gravity (CoG) of the vehicle in the Vehicle frame of reference (VF). The signals are then translated to the driver's eyepoint (u_{a_DF}) in the Driver frame of reference (DF) and passed to the vestibular system model. The output from the vestibular model (y_a) is considered as the motion cues that the passenger would perceive and the same is to be tracked by the MCA.

The lower branch in the figure refers to the motion cueing scheme. The same driver input is applied to the vehicle model and the resulting signals (u_{v_VF}) are transformed from VF to the Inertial frame of reference (IF) giving (u_{v_IF}). These signals are then passed to the motion cueing algorithm which gives the motion commands in terms of actuator movement to the simulator in IF. The resulting acceleration and angular velocities (u_{s_IF}) are converted from IF to DF (u_{s_DF}) and passed on to the vestibular system. The output (y_s) represents the signals perceived by the observer sitting inside the simulator and the resulting error (e) with respect to the reference (y_a) is calculated and fed back to the MCA.

In order to implement this scheme, a Motion Cueing Algorithm should achieve the following objectives:

- Ensuring good simulator fidelity by reducing the error between the actual cues that the passenger perceives while sitting inside the real vehicle and the provided cues from the simulator.
- Keeping the motion platform within its physical boundaries and utilizing the available workspace efficiently.
- Returning the platform state to the neutral position once the desired cue is provided.
- Washing out cues below the driver's perception threshold.

A.2. MOTION PLATFORM

The motion platform commonly used for driving simulator applications is a Gough-Stewart platform, more commonly known as the Hexapod. This type of platform can have a motion in all the 6 degrees of freedom (DOF), i.e. motion in three principle translational and three principle rotational directions. Recently, advanced motion platforms with 9-DOF are also being used for the motion cueing application. Such systems generally have a hexapod mounted on the top of a tripod. The tripod is used for large translational displacements, while the hexapod is used for the rotational and smaller translational movements. The main advantage of such a system is that for a hexapod of similar size, this system provides a larger workspace compared to a regular hexapod based 6-DOF motion system [1]. Moreover, recently Cable-robot-based motion simulators are also being increasingly used. However, it must be noted that the focus of this thesis is limited to the hexapod or the Stewart platform.

A.2.1. STRUCTURE

The motion platform is a type of parallel structured robot manipulator that is controlled by six actuators [2]. These actuators are attached in pairs at three positions to the triangular shaped top plate called the moving base and at three positions to the fixed base plate at the bottom, as shown in figure A.2. Both electric and hydraulic actuators are used in motion simulator systems [3]. However, electrically actuated systems are generally preferred for this application as they offer better perception fidelity and larger bandwidth compared to hydraulically actuated systems. Further, due to the nonlinear nature of the hydraulic systems, precise control is also difficult [4]. All the six actuators can be controlled individually. Torque is provided to the spindles by servo motors which results in extension of the actuators. In the case of the driving simulator, the vehicle cockpit is attached on top of the moving base. Since the motion platform can have a motion in three translational directions (surge x, sway y and heave z) and three rotational directions (roll ϕ, pitch θ and yaw ψ), it can imitate the motion of a freely suspended body.

The frames of reference generally defined for the motion platform are the same as described in Chapter 1.

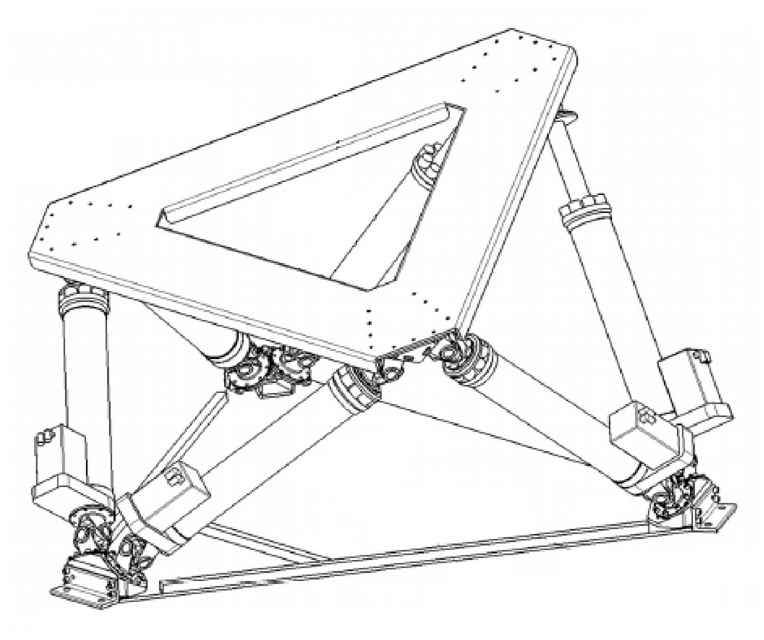

Figure A.2: A 6-DOF Motion Platform [5]

High-end motion systems are able to carry up to 14 tons of payload and reaching accelerations of typically 8 to 10 m/s^2 with actuator strokes of around 1.5 m. The bandwidth of these motion systems is in the order of 5-15 Hz, depending on the desired direction of control and the mechanical design of the system. Although the motion control of these systems is beyond the scope of this research, but its effects on the real-time performance are to be noted. Therefore, it is important to remark that these controllers are called at a sampling frequency in the range of 100-1000 Hz in order to minimize the latency of the cueing algorithm [6]. Further, it should also be noted that control frequency of 100 Hz or higher is recommended to synchronise the platform motions with the graphics [1].

A.2.2. WORKSPACE

The workspace of a 6-DOF motion platform is defined as a six dimensional complex shaped body where the system is free to move without violating its actuator limitations. The boundaries of the workspace are formed due to excursion limitations of one or more actuators. As mentioned in Chapter 1, the motion-space of the platform is defined as space where the system is free to move in the future as per the current state of the system. Since, Stewart platforms are synergistic systems, movement in a single DOF requires contribution from all the actuators. As a consequence, the available workspace (motion-space) in one DOF depends on excursions in other degrees.

Since the driving simulator is supposed to reproduce cues while keeping the platform within its physical limits, workspace management become essential for the effectiveness of the motion cueing process. In order to compute and manage the workspace, it is important to model the motion platform accurately. Detailed explanation about the workspace management strategies considered and used in this report is given in Chapter 1. The kinematics of the Stewart platform (with respect to figure A.3) used for the workspace management are described below.

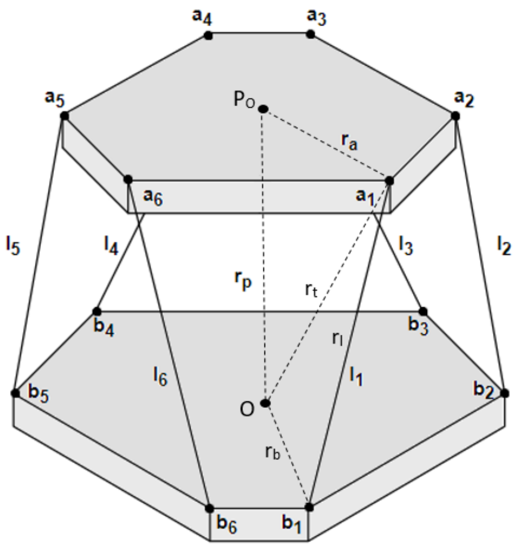

Figure A.3: Geometrical model of the Stewart platform

INVERSE KINEMATICS

The inverse kinematics of the platform is concerned with determining the lengths and the velocities of the six links or actuators (l_i and $\dot{l}_i$, $i = 1...6$) of the platform given the translational and rotational displacements and velocities of the moving base centroid of the platform (point P_0).
From figure A.3, the relations given in equation A.1 and A.2 can be written using the vector algebra.

$$\mathbf{r}_t = \mathbf{r}_p + R_{PF}^{IF}\,\mathbf{r}_a^{PF} \tag{A.1}$$

$$\mathbf{r}_t = \mathbf{r}_l + \mathbf{r}_b \tag{A.2}$$

Where, $\mathbf{r}_a^{PF}$ is expressed in the platform frame (PF) while all the other quantities are expressed in the inertial frame (IF). Further, R_{PF}^{IF} is the rotation matrix from PF to IF. Combining equation A.1 and A.2, the expression for the actuator length vector (equation A.3) can be derived.

$$\mathbf{r}_l = \mathbf{r}_p + R_{PF}^{IF}\,\mathbf{r}_a^{PF} - \mathbf{r}_b \tag{A.3}$$

Similar relations can be derived for the other actuators as well.

$$\mathbf{l}_i = \mathbf{r}_p + R_{PF}^{IF}\,\mathbf{r}_a^{PF} - \mathbf{r}_b \tag{A.4}$$

The magnitude of the actuator length (l_i) and the unit vector along the direction of the actuator ($\mathbf{n}_i$) are given by:

$$l_i = \sqrt{\mathbf{l}_i \cdot \mathbf{l}_i} \tag{A.5}$$

$$\mathbf{n}_i = \frac{\mathbf{l}_i}{l_i} \tag{A.6}$$

In order to find the velocity of the actuator length, equation A.3 is differentiated with respect to time.

$$\frac{d}{dt}(\mathbf{r}_l) = \frac{d}{dt}\left(\mathbf{r}_p + R_{PF}^{IF}\,\mathbf{r}_a^{PF} - \mathbf{r}_b\right) \tag{A.7}$$

$$\Rightarrow \dot{\mathbf{r}}_l = \dot{\mathbf{r}}_p + \omega \times (R_{PF}^{IF}\,\mathbf{r}_a^{PF}) \tag{A.8}$$

where ω is the angular velocity of the centroid of the moving base (point P_0). Since we are interested in the component of the velocity vector in the direction of the actuator, equation A.8 is multiplied by the unit vector along the actuator length ($\mathbf{n}_i$).

$$\dot{\mathbf{l}}_i = \dot{\mathbf{r}}_p \cdot \mathbf{n}_i + \omega \times (R_{PF}^{IF}\,\mathbf{r}_a^{PF}) \cdot \mathbf{n}_i \tag{A.9}$$

Using the vector calculus identities, the product $a \times b \cdot c$ can be written as $a \cdot b \times c$.

$$\Rightarrow \dot{\mathbf{l}}_i = \dot{\mathbf{r}}_p \cdot \mathbf{n}_i + \omega \cdot (R_{PF}^{IF}\,\mathbf{r}_a^{PF}) \times \mathbf{n}_i \tag{A.10}$$

The magnitude of the actuator velocity is given by:

$$\dot{l}_i = \sqrt{\dot{\mathbf{l}}_i \cdot \dot{\mathbf{l}}_i} \tag{A.11}$$

To summarize, the inverse kinematics relations for actuator lengths and velocities can be written from equation A.4 and A.10 in the form of the following matrices:

$$\begin{bmatrix} \mathbf{l}_1 \\ \vdots \\ \mathbf{l}_6 \end{bmatrix} = \mathbf{r}_p + \begin{bmatrix} R_{PF}^{IF}\,\mathbf{r}_{b,\,1}^{PF} - \mathbf{r}_{b,\,1} \\ \vdots \\ R_{PF}^{IF}\,\mathbf{r}_{b,\,6}^{PF} - \mathbf{r}_{b,\,6} \end{bmatrix} \tag{A.12}$$

$$\begin{bmatrix} \dot{\mathbf{l}}_1 \\ \vdots \\ \dot{\mathbf{l}}_6 \end{bmatrix} = \begin{bmatrix} \mathbf{n}_1^T & ((R_{PF}^{IF}\,\mathbf{r}_{a_1}^{PF}) \times \mathbf{n}_1)^T \\ \vdots & \vdots \\ \mathbf{n}_6^T & ((R_{PF}^{IF}\,\mathbf{r}_{a_6}^{PF} \times \mathbf{n}_6)^T \end{bmatrix} \begin{bmatrix} \dot{\mathbf{r}}_p \\ \omega \end{bmatrix} \tag{A.13}$$

FORWARD KINEMATICS

The forward kinematics of the platform is concerned with determining the translational and rotational displacements of the moving base centroid of the platform given the lengths of all the links or actuators. Unlike the inverse kinematic problem, there are many solutions to the forward kinematic problem [7]. As per Wampler [8], a maximum of 40 solutions are possible for the forward kinematic problem. However, Only one solution out of them corresponds to the actual pose of the platform. Generally, Newton-Raphson method is used to iteratively solve the forward kinematics problem [9]. The following relation is used to solve the problem:

$$\mathbf{q}(k+1) = \mathbf{q}(k) - \mathrm{J}^{-1}(\mathbf{l}_m - \mathbf{l}_c) \tag{A.14}$$

where, $\mathbf{q}(k+1)$ and $\mathbf{q}(k)$ are the predicted and the current platform states. $\mathbf{l}_m$ is the measured actuator length while $\mathbf{l}_c$ is the actuator length analytically computed using equation A.4. The jacobian J describes the local linearization of the relationship between actuator displacements and the platform displacement.

$$\mathrm{J}^{-1} = \left(\frac{\delta\,(l_m - l_c)}{\delta q}\right)^{-1} \tag{A.15}$$

However, this method strongly relies on the initial guess which is to be specified by the user. Since there are many possible solutions, to obtain the correct solution, the initial guess should be a good estimate of the actual position of the platform . Such a guess can be either obtained from the known desired state, or the state at the previous time step if small sampling time is used.

A.3. MOTION PERCEPTION

Motion cueing aims at providing the driver with the best possible virtual environment which is closest to the reality. Thus, it is important to understand the cues that the driver is presented and how the driver perceives these cues. Presenting the right cues is the primary task of a motion cueing algorithm. However in order to completely understand the cueing process, one needs to take a closer look into the human motion perception system. Human beings identify their current position and motion in space by using the following perception systems:

- the visual system
- the auditory system
- the somatosensory system
- the vestibular system.

The information arriving from these sensory channels goes into the brain, where it is processed. If the information stemming from two channels is contradictory, generally the stronger signal is given priority over the weaker one. From the motion cueing perspective, this means that not all the cues have to be perfect, but the overall picture presented should be fitting, when compared to the real picture.

However in certain conditions, when both the signals are sufficiently strong and contradicting, we experience motion sickness [10]. Thus, it is important that the contradicting information is handled carefully and does not lead to a state of motion sickness or orientation issues. To tackle such an issue, one has to understand different perception systems within the human body.

A.3.1. VISUAL PERCEPTION SYSTEM

This visual information refers to the information originating from the human eye. The visual perception system is responsible for providing an accurate position estimate to the brain. The position estimate stemming from this channel is strong and should not contradict the information coming from other channels, especially the vestibular system [11]. Continuous change in the position also creates an impression of movement in the human mind, leading to the information about the velocity estimate. However, this information is weaker in nature compared to the position estimate. An example often used in literature for demonstrating this is that of a moving train. While sitting in a stationary train and looking at an adjacent moving train, the observer cannot determine which train is moving. Thus, a reference point is always needed by the visual system to identify motion. This channel also generates information about the acceleration, however, the information has a large time delay and is considered as weak [12]. Thus, this information has to be supported by other sensory channels.

A.3.2. AUDITORY PERCEPTION SYSTEM

The acoustic information is sensed by the human ear. Acoustic cues such as engine noise, road noise, etc. act as a support for assessing information about velocity. However, the signals coming from this channel are generally considered weak and thus, only act as supporting signals inside the brain [13].

A.3.3. SOMATOSENSORY SYSTEM

The somatosensory organs include all the proprioceptive receptors (skin, muscles and joints, etc.). These organs are responsible for providing haptic feedback to the brain. The sensory organs located below the skin surface sense pressure changes and thus, contain implicit information about the forces acting on the body [14]. The changes in the position can also be inferred through these receptors. It must be noted that the information coming from these sensors is moderate in nature and when assisted by the information from the vestibular system, it helps in estimating the translational and angular accelerations [14].

A.3.4. VESTIBULAR SYSTEM

The vestibular system is responsible for providing information about the motion by sensing the translational accelerations and rotational velocities. It also contributes to compensatory eye movements during motion, postural control of the human body and spatial orientation in space [15, 16]. It also enables humans to have a perception of illusory self-tilt and illusory self-motion, which confirms a strong visual-vestibular interaction [17]. Further, this type of sensory organ is an essential instrument to train skill-based control behavior in driving or flight simulation [18]. Since this channel provides the primary information about the movement in space, the vestibular channel is of utmost interest from the motion cueing perspective.

The human vestibular system is located in the human inner ear and is composed of many different components. However, for motion cueing application, there are two most important parts: the semi-circular canals, responsible for sensing the rotational velocities and otolith organs, responsible for sensing the translations accelerations [19]. The human vestibular system is shown in figure A.4.

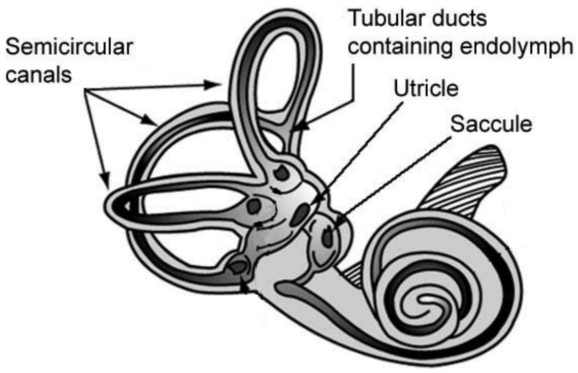

Figure A.4: The Human Vestibular system [20]

SEMICIRCULAR CANALS

The semicircular canals are responsible for detecting the rotational movements. The organ consist of three orthogonally arranged canals, filled with a fluid called endolymph. When the head is rotating about one plain, the endolymph moves due to inertia in the direction opposite to that of the head movement, pressing against the walls of the organ. This displaces the sensory cells in the organ and the angular accelerations are sensed [3, 21]. It must be noted that these organs exhibit strongly damped high-pass behavior and cannot sense information at lower rotational velocities (lower than the rotational threshold) [22].

OTOLITH ORGANS

The otolith organs are responsible for detecting the translational movements. The utricle organs sense the accelerations in the horizontal plane and the saccule organs sense the motion in the vertical plane [3]. The otolith organ senses the specific acceleration, i.e. the acceleration sensed is the superposition of translational acceleration and acceleration due to gravity. Thus, by definition, the sensed accelerations should be zero in a free fall, that is:

$$a_f = a - g \tag{A.16}$$

where a_f is the sensed specific acceleration, a is the actual translational acceleration of the body and g is the acceleration due to gravity acting upon the body. Since the gravitational force is sensed in the same way as the translational force, the otolith organ cannot distinguish between the two. This weakness is exploited by motion cueing algorithms in the form of tilt coordination (explained in section A.4).

MODELING THE VESTIBULAR SYSTEM

Although extensive research has been conducted in the past for modeling the vestibular system [23–29], majority of them avoid to implement perception thresholds into these models. The reason behind this choice is the appearance of non-linear relations due to the thresholds. Perception thresholds are beneficial as the cueing algorithm can avoid the cues which are below the threshold and will not be sensed by the driver. Reid et al. [30] presented a non-linear dynamical model of the vestibular system which captures these non-linearities as well. These non-linear dynamical models are shown in figure A.5 and A.6. Here, ω is the angular velocity, $\hat{\omega}$ is the sensed angular velocity, a is the specific acceleration and $\hat{a}$ is the sensed specific acceleration, all specified at the driver's eyepoint in the DF.

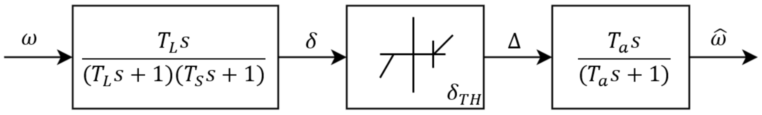

Figure A.5: Non-linear model of the Semi-circular canals

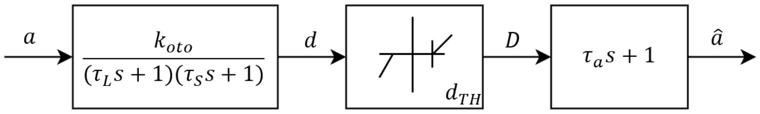

Figure A.6: Non-linear model of the Otolith organ

It should be noted that the non-linear dynamical models are a combination of linear transfer functions and non-linear motion thresholds. As a consequence of incorporating the these non-linear models, the computational time could increase and lead to a poor real-time performance. To counteract this, reliable linear models have been derived in recent times [3]. In their research, Telban et al. [3] derived the linear transfer function of the semi-circular canal as follows:

$$H_{\text{scc}}(s) = \frac{\hat{\omega}_i(s)}{\omega(s)} = \frac{T_{\text{L}} T_a s^2}{(T_{\text{L}} s+1)(T_{\text{S}} s+1)(T_a s+1)} \tag{A.17}$$

Further, the linear transfer function for the otolith organ is given as follows:

$$H_{\text{oto}}(s) = \frac{\hat{a}(s)}{a(s)} = \frac{k_{\text{oto}}(\tau_{\text{a}} s+1)}{(\tau_{\text{L}} s+1)(\tau_{\text{S}} s+1)} \tag{A.18}$$

PARAMETER SELECTION

In the scheme of motion cueing, the vestibular model determines which cues are important and should be presented. The parameters of the model directly affect the filtering process of the onset cues and thus influence the motion perception.

Ormsby et al. [28], Reid et al. [30] and Telban et al. [3] have specified the model parameters related to the linear transfer functions for the semicircular canals. The same is mentioned in the table A.1. The analysis done to select amongst the given models is shown in figure A.7.

Table A.1: Model parameters for Semicircular canals

Parameter	**Ormsby** [28]	**Reid** [30]	**Telban** [3]
$T_a(s)$	30	30	80
$T_L(s)$	18	6.1, 5.3, 10.1	5.73
$T_S(s)$	0.0	0.1	0.0

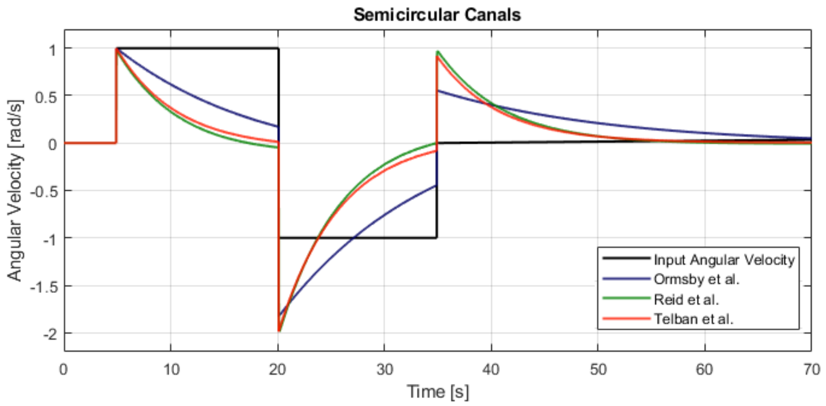

Figure A.7: Response of the Semicircular canal models with different parameters

To analyze the effect of different parameters on the onset cues, response to the step input is generally considered. Along with the step input, it is also important to check the response to a large variation in the input signal. Considering both, the response to the signal shown in figure A.7 is considered. The input signal is a unit step from 5 *sec* to 20 *sec* after which, there is a sudden drop from magnitude 1 to -1.

It can be observed that unlike the other two models, the Ormsby model has a smaller decay and thus doesn't quickly converges to zero. It can also be observed that the other two models show more high-pass behavior compared to the Ormsby model. Since the vestibular system is characterized by its high-pass behavior, it can be inferred that the other two models capture this behavior more effectively. It can also be observed that the models given by Reid et al. and Telban et al. show a very similar response. In this report, the model given by Telban et al. [3] has been used for modeling the semicircular canals.

Ormsby et al. [28], Young et al. [29] and Telban et al. [3] have specified the model parameters related to the linear transfer functions for the otolith organs. The same is mentioned in the table A.2. An analysis similar to the one performed for the semicircular canals is done to analyze the effect of different parameters of the otolith organ model. The response to the input signal is shown in figure A.8.

Table A.2: Model parameters for Otolith organ

Parameter	**Ormsby** [28]	**Young** [29]	**Telban** [3]
k_{oto}	0.4	0.4	0.4
$\tau_a(s)$	10.1	13.2	10.0
$\tau_L(s)$	7.5	5.3	5.0
$\tau_S(s)$	0.51	0.66	0.016

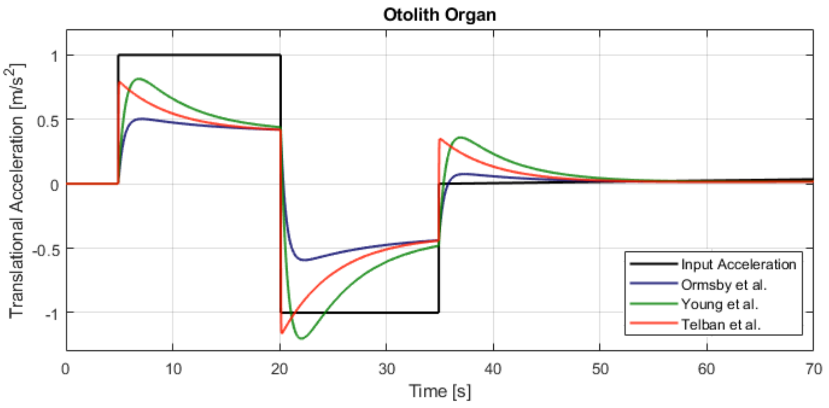

Figure A.8: Response of the Otolith organ models with different parameters

It can be observed that the Telban model shows higher bandwidth and has lower delay compared to the other two models. This would help to avoid the formation of high frequency false cues. Therefore in this report, the model given by Telban et al. [3] has been used for modeling the otolith organs.

Further, Reid et al. [30] have mentioned the threshold parameters related to the translational accelerations and the rotational velocities in their research. The same is mentioned in table A.3. It should further be noted that this model is to be adopted for each degree of freedom separately.

Table A.3: Threshold values as per Reid [30]

Threshold d_{TH}/δ_{TH}	**Value**
Surge x (m/s^2)	0.17
Sway y (m/s^2)	0.17
Heave z (m/s^2)	0.28
Roll ϕ (deg/s)	3.0
Pitch θ (deg/s)	3.6
Yaw ψ (deg/s)	2.6

DISCRETIZATION

The aforementioned continuous-time models are converted to a discrete-time model by using the bilinear transform (also known as the Tustin's method). This method has been selected because of the following property - A stable continuous-time systems is always mapped to a stable discrete-time system. A detailed explanation of the method can be found in [31].

The selection of sampling time (or sampling frequency) is an engineering decision that has to be made keeping in mind the minimum frequency required to preserve the system dynamics and the maximum frequency that wouldn't create excessive computational burden. If a very small sampling frequency is chosen, the fast dynamics of the system will be missed and also, aliasing can occur. On the other hand, a very large sampling frequency would require excessive computational effort. The following rule of thumb is often used to determine the sampling frequency:

$$10\omega_b \leq \omega_s \leq 30\omega_b \tag{A.19}$$

where ω_b is the highest frequency in the bandwidth of the system and ω_s is the sampling frequency of the system. Further, as per Shannon's sampling theorem, to prevent aliasing, ω_s should be at-least higher than $2 \times \omega_b$.

Looking at the transfer function of the semi-circular canals and otolith organs with the parameters selected as per Telban et al. [3], it can be seen that the farthest pole is located at 62.5 s^{-1}. Therefore as per Shannon's sampling theorem, the minimum sampling period should be 125 Hz while as per the rule of thumb (equation A.19), the minimum sampling period should be 625 Hz. However, it should be noted that in most of the MPC-based motion cueing problems, the computational effort required to run the system is the limiting factor with respect to the sampling frequency and therefore, the final selection of the sampling time is often done as per the available computational resources.

A.4. TILT COORDINATION

The acceleration that a vehicle is subjected to while negotiating any maneuver can be divided into two categories: accelerations with fast dynamics (high-frequency accelerations) and accelerations with slow dynamics (sustained accelerations). The sustained accelerations tend to prolong the actuators continuously for a long period of time, thus pushing the platform to its physical limit. Once the limit is reached, these components of the acceleration cannot be reproduced any further.

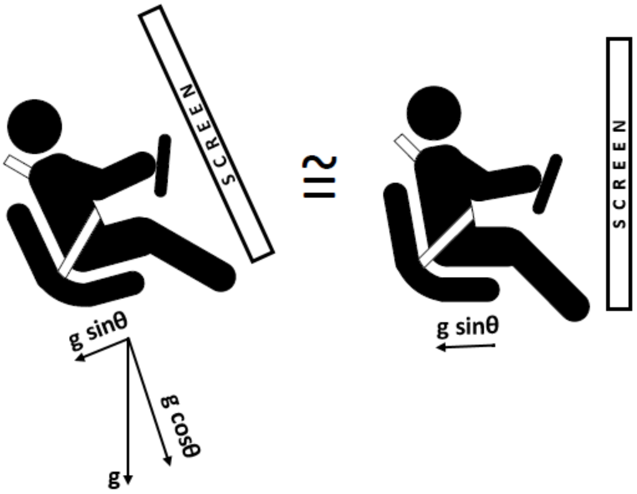

Figure A.9: Tilt coordination: exploiting gravity to represent sustained accelerations

A commonly used technique to reproduce the sustained acceleration is tilt coordination, which uses a component of the acceleration due to gravity (g) to represent the sustained acceleration in the lateral or longitudinal direction by tilting the platform (as shown in figure A.9). The idea behind this technique is that since the otolith organs sense the acceleration due to gravity in the same way as they sense translational accelerations, the observer sitting inside the simulator would not be able to distinguish between the two, given that the visual information is also rotated accordingly. Further, to avoid any conflict in perception, the tilt rate has to be kept below the perception threshold (mentioned in table A.3).

Tilt coordination contributes to the longitudinal acceleration due to the pitch angle (θ) and to the lateral acceleration due to the roll angle (ϕ). The gravity vector in the non-inertial driver reference frame is given by the following vector:

$$g_{tilt} = R_y(\phi) \cdot R_x(\theta) \cdot \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} -g\ sin\theta \\ g\ cos\theta\ sin\phi \\ g\ cos\theta\ cos\phi \end{bmatrix} \approx \begin{bmatrix} -g\theta \\ g\phi \\ g \end{bmatrix} \tag{A.20}$$

It must be noted that the small-angle approximation has been applied here. Also, it must be observed that the yaw angle (ψ) does not affect the tilt coordination. Due to the high-pass filtering behavior of the otolith organs, the constant acceleration (g) in the vertical direction (third component) is not perceived. Thus, only the first two components have to be considered and added to the translational acceleration vector, as follows:

$$a_{total} = a_{translation} + g_{tilt} = \begin{bmatrix} a_x - g\theta \\ a_y + g\phi \\ a_z \end{bmatrix} \tag{A.21}$$

B

Conventional Motion Cueing Algorithms

This appendix presents a detailed overview of the conventional strategies used for motion cueing applications. The conventional approach of designing an MCA is done by using different types of filters. The following strategies are generally used:

- Classical Washout Filter (CWF)
- Optimal Washout Filter (OWF)
- Adaptive Washout Filter (AWF)

In the case of the classical washout filter, the design is simple and computationally cheap but a major shortcoming of this technique is that the tuning procedure is complicated as the user needs to change the filter coefficients. Further, re-tuning is required even if the reference or the conditions are changed slightly [32]. To overcome these problems, approaches like optimal washout filters and adaptive washout filters have been used and found effective. In this appendix, the aforementioned filter-based MCAs have been explained.

B.1. CLASSICAL WASHOUT FILTER

The classical washout filter scheme was initially developed for flight simulators and was later applied to the vehicle simulators. Several versions of CWF-based MCAs were introduced according to different platforms and requirements. Grant et al. [33] derived a method for the 3-DOF platform, while Nahon et al. [32] derived the same for a 6-DOF hexapod. Taking it further, Fischer et al. [34] and Chapron et al. [35] derived the MCA for an 8-DOF platform. This section aims at introducing the traditional classical washout algorithm for the hexapod platform based on the work of Nahon et al. [32].

The classical washout method is characterized by the combination of linear high-pass and low-pass filters used together to extract the required dynamics. These signals are further integrated to calculate the translational and angular displacement of the motion platform. Flowchart of typical classical washout filter is shown in figure B.1.

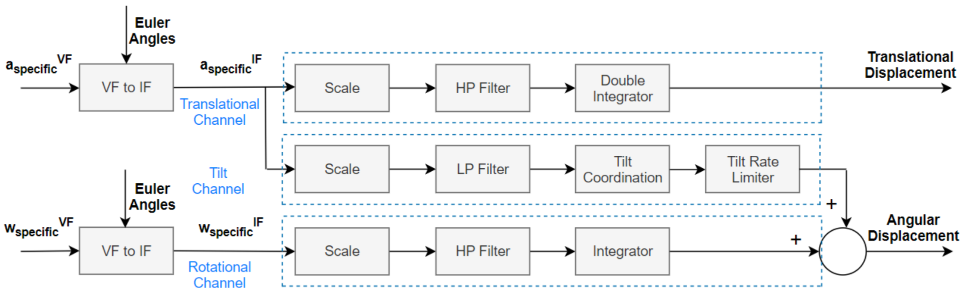

Figure B.1: Typical classical washout filter

It must be noted that typical inputs to the motion cueing algorithm are specific accelerations (a) and angular velocities (ω) at the driver's eyepoint in VF. However, they must be converted to IF before passing them to the MCA as shown in figure B.1.

The input signals often have a high amplitude and their generation in the simulator is not possible. Thus, the input translational accelerations and angular velocities are scaled to a reproducible magnitude. The entire scheme is divided into three channels, namely translational channel, tilt channel and rotational channel. In the translational channel, the accelerations are high-pass filtered to extract the fast dynamics (i.e. to reject the sustained accelerations). The resultant filtered accelerations are integrated to calculate the translational displacement of the simulator. The low-frequency components (sustained accelerations) are perceived through the tilt channel by tilting the motion platform to a calculated angle as explained in section A.4. Lastly, the angular velocities are high-pass filtered to remove the components which are below the perception threshold. The resulting filtered angular velocities are then integrated to calculate the angular displacement. The angular displacement from the tilt and the rotational channel is added and the total angular displacement of the motion platform is obtained. A detailed description of all the components shown in figure B.1 is given in the following sub-sections.

B.1.1. Scaling

As mentioned earlier, the input signals often have high amplitude and their generation through the motion platform is often not possible. Lower the value of scaling, higher is the possibility to reproduce the cue while keeping the motion platform within its physical limit. Meanwhile, a very low value might also result in lower simulator fidelity. Thus, the scaling factor has to be optimized to meet both the requirements. The following equation is used to determine the scaled inputs to the MCA:

$$u_{scaled} = K \cdot u_{actual} \tag{B.1}$$

where K is the scaling factor. Comprehensive studies have been done to find out the range of acceptable scaling factor. In their research, Grácio et al. [36] established that a 1:1 ratio of the inertial and visual cues are reported as too strong by the subjects and thus, not preferred. They reported that the optimal scaling factor, called as optimal gain depends on the amplitude and the frequency of the stimuli. It was also reported that the preferred motion gain decreases with the increase of the stimuli amplitude. Boer et al. [37] compared the effects of different scaling factors on controllability and coherence of perceived cues and found the scaling factor of 0.5 to be optimum. Other maneuver specific studies have also been done and concluded that a scaling value of 0.5 yields optimal results for braking [38], cornering [39] and slalom maneuvers [40]. In a detailed study, Grant et al. [41] compared the classical washout scheme for different parameters with variable scaling. The results were similar to the previously mentioned work and suggested a scaling factor between 0.4 to 0.5. However, it should be noted that there have been studies like Kading et al. [42] which indicate that any scaling factor below 0.8 results in poor simulator fidelity.

It should also be noted that in all the above-mentioned studies, a uniform scaling factor was applied to all three channels. Sammet [43] in his study compared different scaling and uniform scaling for cornering maneuvers and the uniform scaling scheme was rated as the best. The reason stated for this was that while the performance of both the schemes were similar, uniform scaling made the MCA easy to tune.

Lastly, one must not scale the gravitational component included within the total acceleration. For example, if the total input acceleration is given as:

$$a_{total} = a_{translation} - g \tag{B.2}$$

Then the scaled acceleration should be given by:

$$a_{total,scaled} = \left(K \cdot a_{translation}\right) - g \tag{B.3}$$

B.1.2. Translation Channel

The specific accelerations in the translational channel are high-pass filtered to extract the high-frequency components and reject the low-frequency components as they would drive the simulator to its physical limits. Another role of the high-pass filter is to return the motion platform to its neutral position once the desired cue is provided, i.e. the displacement of the motion platform should tend to zero as time progresses. This factor plays a vital role in deciding the order of the high-pass filter.

High-pass Filter

The order can be decided by looking into the convergence of step response of filters of different orders. The general expression for high-pass filters is as follows:

$$H(s) = \frac{s^n N(s)}{D(s)} \tag{B.4}$$

where,

$$N(s) = \sum_{i=0}^{k} a_j s^i$$
$$D(s) = \sum_{j=0}^{m} b_j s^j$$
$$m \geq n + k$$
$$a_0, b_0 \neq 0$$

Further, the platform displacement can be written in terms of the filtered acceleration as follows:

$$x(t) = \int\int a(t)dt = \frac{1}{s^2} A(s) \cdot H(s) \tag{B.5}$$

Since the actuator displacement should tend to zero as time progresses, using Final Value Theorem (FVT):

$$\lim_{t\to\infty} x(t) = 0 \Rightarrow \lim_{t\to\infty} \int\int a(t)dt = 0 \Rightarrow \lim_{s\to 0} \frac{1}{s^2} A(s) \cdot H(s) = 0 \tag{B.6}$$

where, $a(t)$ is the filtered acceleration in the time domain and its Laplace transform is $A(s) \cdot H(s)$. Substituting equation B.4 into equation B.6, we get:

$$\lim_{s \to 0} \tilde{x}(s) = \lim_{s \to 0} A(s) \cdot \frac{s^{n-2} N(s)}{D(s)} = 0 \tag{B.7}$$

From the above equation, it can be concluded that $n \geq 3$, or $H(s)$ should be a 3rd order high-pass filter in order to implement washout. The same analysis is summarized in table B.1.

Table B.1: Convergence of step response of filters of different orders

Filter Order	**H(s)**	**a(t)**	**v(t)**	**d(t)**
1	$\frac{s}{s+\omega_0}$	0	Constant	∞
2	$\frac{s^2}{s^2+2D\omega_0+\omega_0^2}$	0	0	Constant
3	$\frac{s^2}{s^2+2D\omega_0+\omega_0^2} \cdot \frac{s}{s+\omega_n}$	0	0	0

It should be noted that a 2nd order high-pass filter is used to extract the accelerations with fast dynamics and a 1st order filter is used to implement washout, thus giving the following 3rd order filter:

$$H(s) = \frac{s^2}{s^2 + 2D_{hp} \cdot \omega_{hp} \cdot s + \omega_{hp}^2} \cdot \frac{s}{s + \omega_w} \tag{B.8}$$

Moreover, ω_{hp} is the break frequency which determines the cut-off frequency below which the translational accelerations will be rejected. ω_w is the washout frequency which determines the rapidity of return of the motion platform, back to its neutral position. Further, D_{hp} is the damping factor. The values for the natural frequencies and the damping coefficients are either tuned manually for each maneuver to get the desired results or they are optimized using genetic algorithms as mentioned in section B.1.5.

B.1.3. TILT CHANNEL

As mentioned in section A.4, sustained accelerations tend to prolong the actuators. Thus to reproduce these low-frequency components, the tilt coordination method is used. In this scheme, the motion platform is tilted and the driver perceives a component of the acceleration due to gravity as lateral or longitudinal acceleration. Therefore, one needs to first extract the sustained accelerations (either longitudinal, lateral or both) using a low-pass filter, then transform these into the Euler angles (either pitch, roll or both). Lastly, the rate of tilting should be limited to a value below the driver's perception threshold.

LOW-PASS FILTER

In the translational channel, a 2nd order high-pass filter is used to extract the fast dynamics. To extract all the rejected accelerations, Nahon et al. [32] recommend the use of a 2nd order low-pass filter. The general expression for a 2nd order low-pass filter is given as follows:

$$L(s) = \frac{\omega_{lp}^2}{s^2 + 2D_{lp} \cdot \omega_{lp} \cdot s + \omega_{lp}^2} \tag{B.9}$$

where, ω_{lp} is the break frequency which determines the cut-off frequency above which the translational accelerations will be rejected. Further, D_{lp} is the damping factor.

In order to extract all the signals rejected by the translational channel, Sammet [43] recommends using the same break frequencies as for the rotational channel, i.e.:

$$\omega_{lp} = \omega_{hp} \tag{B.10}$$

This provides the ideal overall transfer function:

$$H(s) + L(s) = 1 \tag{B.11}$$

However, when using this scheme the response to the input near the break frequency is often abrupt. Nahon et al. [32] recommend the following relationship as this provides a good transition behavior:

$$\omega_{lp} = 2 \cdot \omega_{hp} \tag{B.12}$$

TILT COORDINATION AND TILT RATE LIMITER

Once the low-pass filtered accelerations (a_{lp}) are extracted, they are to be transformed into the Euler angles. This is done by using the relations shown in equation B.13. By solving the above equations, values for roll angle (ϕ) and pitch angle (θ) are determined.

$$\begin{bmatrix} a_{x,lp} \\ a_{y,lp} \\ a_{z,lp} \end{bmatrix} = R_y(\phi) \cdot R_x(\theta) \cdot \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} -g\ sin\theta \\ g\ cos\theta\ sin\phi \\ g\ cos\theta\ cos\phi \end{bmatrix} \approx \begin{bmatrix} g\theta \\ g\phi \\ g \end{bmatrix} \tag{B.13}$$

Lastly, the tilt rate should be limited below the driver's perception threshold so that the driver does not feel the rotation. Tilt-rate limiting introduces non-linearities in the system which help in coping with special situations [32]. Since tilt-coordination is a technique used to deceive the passenger, it is important that s/he does not perceive the angular velocities associated with this. Thus, an effort is made to limit the tilt rate below the driver's threshold. The perception threshold is mentioned in table A.3. Rate limiting is applied by differentiating the angular displacements and limiting the resultant velocities. These signals are integrated to calculate the rate-limited angles. A combined scheme for tilt coordination and rate limiting is shown in the figure B.2.

B.1.4. ROTATIONAL CHANNEL

A high-pass filter is used in the rotational channel to reject the unrepresentable rotational frequencies and to perform washout once the rotational cues are provided.

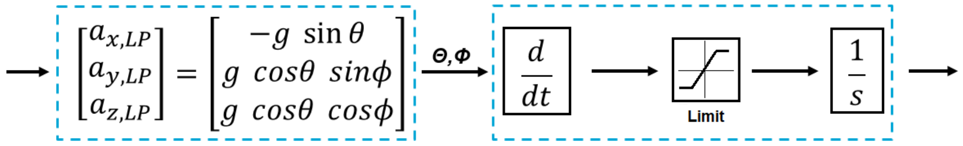
$\begin{bmatrix} a_{x,LP} \\ a_{y,LP} \\ a_{z,LP} \end{bmatrix} = \begin{bmatrix} -g \sin\theta \\ g \cos\theta \sin\phi \\ g \cos\theta \cos\phi \end{bmatrix}$
Θ,Φ
$\frac{d}{dt}$
Limit
$\frac{1}{s}$

Figure B.2: Tilt coordination and tilt rate limiting

High-pass Filter

To reject the unrepresentable frequencies, a 1st order filter could suffice. However, a filter of higher order will provide a better roll-off rate and smaller transition band. Further, in order to bring the angular displacement to zero once the angular velocity cues have been provided, a filter of at least 2nd order would be necessary (analysis similar to that of the high-pass filters could be performed for the rotational channel). Therefore, a 2nd order high-pass filter is generally used. The general expression for a 2nd order high-pass filter is given as follows:

$$R(s) = \frac{s^2}{s^2 + 2D_r \cdot \omega_r \cdot s + \omega_r^2} \tag{B.14}$$

where, ω_r is the break frequency which determines the cut-off frequency below which the rotational velocities will be rejected. Further, D_r is the damping factor.

B.1.5. Parameter Tuning

There are 6 parameters per canonical direction ($\omega_{hp}, D_{hp}, \omega_w, D_{lp}, \omega_r, D_r$) which are to be tuned. ω_{lp} is generally chosen based on the empirical relationships described in equations B.10 and B.12.

There are two methods generally followed to tune the parameters. First, is by looking at the step input response. The step response provides important information about the response characteristics of the system. A detailed tuning methodology based on step input and frequency response is described by Grant et al. [44].

Another tuning methodology is by using genetic algorithms to minimize the error between the motion perception of the driver sitting in the real vehicle and the simulator (shown in figure A.1). A detailed procedure of the same is given by Murgovski [45].

Nahon et al. [32] stated that the principal advantage of using the classical washout algorithm is that the method is mathematically and computationally simple, and therefore computationally cheap. Additionally, the method is quite transparent from the designer's perspective. On the other hand, the main disadvantage of this method is that it is essentially a feed-forward control technique and therefore it does not effectively exploit the simulator capabilities. The algorithm is difficult to tune and has to be re-tuned if there is a considerable change in the inputs.

B.2. OPTIMAL WASHOUT FILTER

In the case of classical washout filters, the fixed filter coefficients make it inflexible and unsuitable for producing cues for different maneuvers that a vehicle is subjected to. Further, CWF is essentially a feed-forward control scheme. To address this inflexibility and incorporate a feed-back scheme, the Optimal Washout Filter (OWF) scheme was designed.

The OWF scheme was initially developed by Sivan et al. [46] with four main assumptions:

- In the human perception system, the vestibular system dominates the perception of motion cues.
- The deviation between the motion cues in the real vehicle and the driving simulator can be estimated by the mean-square value of the vestibular error.
- The actual vehicle motion can be modeled as a random process with a rational spectrum.
- The vestibular system can be represented accurately by a linearized model.

The OWF algorithm calculates the error and minimizes a predefined cost by solving the Algebraic Riccati equation (A.R.E). Further, the algorithm incorporates a vestibular system model within the scheme to increase the fidelity. Sivan et al. [46] designed an optimal control algorithm that solves the Riccati equation in real-time. Further to optimize the workspace utilization, Chen et al. [47] included motion platform states into the controller. In all the implementations, the MCA problem is decoupled into four subproblems, i.e. pitch-surge, roll-sway, heave, and yaw. The algorithm solves these subproblem separately.

Figure B.3 shows the general scheme of the optimal washout filter, where the transfer function $W(s)$ describes the optimal filter gain, u_A is the output from the real vehicle or vehicle model, while u_S is the input to the simulator.

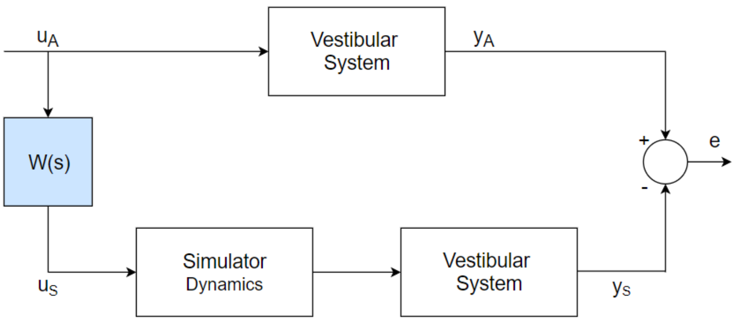

Figure B.3: General scheme of optimal washout filter

In the following sections, the linear optimal washout filter is derived based on the work of Sivan et al. [46].

B.2.1. MODELLING VESTIBULAR SYSTEM

The linear vestibular system model used in OWF is derived in this section.

SEMI-CIRCULAR CANALS

Representing the transfer function of the semi-circular canal (equation A.17) in its observable canonical state-space form, we get:

$$\dot{x}_s = A_s x_s + B_s u_s \tag{B.15}$$
$$y_s = C_s x_s + D_s u_s \tag{B.16}$$

where the input and the output signal is:

$$\begin{aligned} u_s &= \omega \\ y_s &= \hat{\omega} \end{aligned} \tag{B.17}$$

OTOLITH ORGANS

Representing the otolith transfer function (equation A.18) in its observable canonical state-space form, we get:

$$\dot{x}_o = A_o x_o + B_o u_o \tag{B.18}$$
$$y_o = C_o x_o + D_o u_o \tag{B.19}$$

where the input and the output signal is:

$$\begin{aligned} u_o &= a \\ y_o &= \hat{a} \end{aligned} \tag{B.20}$$

In order to implement the tilt coordination, the otolith matrices are augmented as follows:

$$\begin{aligned} A_{\bar{o}} &= \left[\begin{array}{c|c} A_o & \bar{B} \\ \hline 0 & 0 \end{array}\right] \quad B_{\bar{o}} = \left[\begin{array}{c|c} B_o & 0 \\ \hline 0 & 1 \end{array}\right] \\ C_{\bar{o}} &= \left[\begin{array}{c|c} C_o & 0 \end{array}\right] \quad D_{\bar{o}} = \left[\begin{array}{c|c} D_o & 0 \end{array}\right] \end{aligned} \tag{B.21}$$

where

$$\bar{B} = B_o \cdot g \quad \text{for surge acceleration } (x\text{-direction}) \tag{B.22}$$

and

$$\bar{B} = B_o \cdot -g \quad \text{for sway acceleration } (y\text{-direction}) \tag{B.23}$$

The input and the output signals are:

$$\begin{aligned} u_{\bar{o}} &= [a;\omega] \\ y_{\bar{o}} &= \hat{a} \end{aligned} \tag{B.24}$$

COMPLETE VESTIBULAR SYSTEM

The individual vestibular models can be modeled as the following complete vestibular system:

$$\begin{aligned} \dot{x}_s &= A_v \cdot x_s + B_v \cdot u_s \\ y_s &= C_v \cdot x_s + D_v \cdot u_s \end{aligned} \tag{B.25}$$

$$A_v = \left[\begin{array}{cc} A_s & 0 \\ 0 & A_{\bar{o}} \end{array}\right], B_v = \left[\begin{array}{cc} 0 & B_s \\ B_{\bar{o}} & 0 \end{array}\right], C_v = \left[\begin{array}{cc} C_s & 0 \\ 0 & C_{\bar{o}} \end{array}\right], D_v = \left[\begin{array}{c} D_s \\ D_{\bar{o}} \end{array}\right] \tag{B.26}$$

where, the input and output signals are:

$$\begin{aligned} u_s &= [a\ \omega]^T \\ y_{\bar{o}} &= \hat{a} \end{aligned} \tag{B.27}$$

B.2.2. ERROR DYNAMICS

Let the vestibular dynamics of the real vehicle be defined as follows:

$$\dot{x}_r = A_v x_r + B_v u_r \tag{B.28}$$

Then the error, x_e can be defined as $x_e = x_s - x_r$, where x_r and x_s are vestibular states for real vehicle and the simulator respectively. The error dynamics can be written as follows:

$$\begin{aligned} \dot{x}_e &= A_v \cdot x_e + B_v \cdot u_s - B_v u_a \\ e &= C_v x_e + D_v u_s - D_v u_r \end{aligned} \tag{B.29}$$

B.2.3. INTEGRATING SIMULATOR STATES

The platform displacement, velocity and rotation angle are also included in the state-space model to integrate the simulator states.

$$\dot{x}_i = A \cdot x_i + B \cdot u_s, \text{ where: } A = \left[\begin{array}{ccc} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right], B = \left[\begin{array}{cc} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{array}\right] \tag{B.30}$$

The state matrix x_i is:

$$x_i = [p\ v\ \theta]^T \tag{B.31}$$

where, p, v and θ are the platform displacement, velocity and rotation angle expressed at the Drivers eye-point in DF.

The vehicle input u_a consists of filtered noise and can be represented as:

$$\begin{aligned} \dot{x}_n &= A_n x_n + B_n w \\ u_A &= x_n \end{aligned} \tag{B.32}$$

where x_n is the filtered white noise state, w is the white noise. Further,

$$A_n = \left[\begin{array}{cc} -\beta_1 & 0 \\ 0 & -\beta_2 \end{array}\right], B_n = \left[\begin{array}{c} \beta_1 \\ \beta_2 \end{array}\right] \tag{B.33}$$

where β_1 and β_2 are the first order break frequencies for each degree-of-freedom as explained in [47].

Thus, combining equations B.29, B.30 and B.32, the total state-space system can be written as follows:

$$\begin{aligned} \dot{X} &= AX + Bu_s + Hw \\ Y &= CX + Du_s \end{aligned} \tag{B.34}$$

where, $Y = [e \quad x_i]$ and $X = \begin{bmatrix} x_e & x_i & x_n \end{bmatrix}^T$, and:

$$A = \begin{bmatrix} A_v & 0 & -B_v \\ 0 & A_i & 0 \\ 0 & 0 & A_n \end{bmatrix}, B = \begin{bmatrix} B_v \\ B_v \\ 0 \end{bmatrix}, H = \begin{bmatrix} 0 \\ 0 \\ B_n \end{bmatrix}$$
$$C = \begin{bmatrix} C_v & 0 & -D_v \\ 0 & I & 0 \end{bmatrix}, D = \begin{bmatrix} D_v \\ 0 \end{bmatrix} \tag{B.35}$$

B.2.4. DERIVING OPTIMAL GAIN

The quadratic cost function to be used for deriving the optimal gain is described as following:

$$J = \left\{ \int_{t_0}^{t_1} \left(e^T Q e + x_i^T R_c x_i + u_s^T R u_s\right) dt \right\} \tag{B.36}$$

Here, Q, R_c and R are positive semi-definite tuning matrices corresponding to the sensation error e, platform states x_i and the control input u_s respectively. The above-mentioned cost function is minimized by solving the Algebraic Riccati Equation (A.R.E) as mentioned by Bellon [48]. By solving the A.R.E, the optimal gain (K_1, K_2 and K_3) is obtained. Thus giving the following control law:

$$u_s = -\begin{bmatrix} K_1 & K_2 & K_3 \end{bmatrix} \begin{bmatrix} x_e \\ x_i \\ x_n \end{bmatrix} \tag{B.37}$$

Since, $x_n = u_a$, by substituting equation B.37 in equation B.34, we get:

$$\begin{bmatrix} \dot{x}_e \\ \dot{x}_i \end{bmatrix} = \begin{bmatrix} A_v - B_v K_1 & -B_v K_2 \\ -B_i K_1 & A_i - B_i K_2 \end{bmatrix} \begin{bmatrix} x_e \\ x_i \end{bmatrix} + \begin{bmatrix} -B_v (I + K_3) \\ -B_i K_3 \end{bmatrix} u_A \tag{B.38}$$

From B.38 and B.29, we can write:

$$u_s(s) = W(s) u_A(s) \tag{B.39}$$

where,

$$W(s) = \begin{bmatrix} K_1 \\ K_2 \end{bmatrix}^T \begin{bmatrix} sI - A_v + B_v K_1 & B_v K_2 \\ B_i K_1 & sI - A_i + B_i K_2 \end{bmatrix}^{-1} \begin{bmatrix} B_v (I + K_3) \\ B_i K_3 \end{bmatrix} - K_3 \tag{B.40}$$

The obtained optimal gain (K_1, K_2 and K_3) is substituted in equation B.40, to obtain the matrix $W(s)$. This matrix is then multiplied to the system input (u_A) to obtain the control input (u_s) as given in equation B.39.

Because of the fixed parameters, the optimal washout filters must be tuned for worst-case maneuvers and thus, they often generate minimal motion for gentle maneuvers [32]. In their work, Cardullo et al. [49] and Ish-Shalom [50] suggested using a nonlinear optimal filter to overcome these difficulties. The approach combines the ideas of the adaptive and optimal washout filters to maximize the simulator fidelity. Based on this, Telban et al. [11] proposed a nonlinear optimal washout filter that updates the filter coefficients at each time step using a feedback loop. However, in order to achieve this, a real-time Riccati Equation Solver needs to be implemented, which increases the computational effort.

B.3. ADAPTIVE WASHOUT FILTERS

As mentioned before, the classical washout and optimal filters are generally tuned for the worst-case scenario and thus, produce minimum motion for gentle maneuvers. To overcome this limitation, the adaptive washout filters were designed. The AWF control scheme tends to give more realistic cues when the simulator is near its neutral position and only reduces the fidelity when the simulator is at limits.

AWF-based MCA was first introduced by Parrish et al. [51] and later developed by Reid and Nahon [30] and Telban et al. [52]. Further, Naseri et al. [53] developed an actuator state-based adaptive algorithm which included actuator states within the cost function which is to be minimized. Later, using a similar approach Nehaoua et al. [54] derived this algorithm for a vehicle simulator.

Similar to the optimal control MCA, the adaptive filter algorithm is also based on minimizing a cost function comprising of penalties on sensation error and platform states. Generally non-linear filters are used and thus, the optimization is performed by using the steepest descent method. Similar to the optimal washout algorithm, the adaptive filter algorithm also uses decoupled degrees of freedom (Pitch-surge mode, Roll-sway mode, Yaw mode and Heave mode) and solves for each case separately. In the following section, the adaptive filter is derived for the pitch-surge mode based on the work of Telban et al. [52].

B.3.1. FILTER EQUATIONS

The non-linear filter equation is given by the following equation:

$$\begin{aligned}\ddot{x}_s &= \lambda \ddot{x}_v - d\dot{x}_s - e x_s \\ \dot{\theta}_s &= \gamma \ddot{x}_v + \delta \dot{\theta}_v\end{aligned} \tag{B.41}$$

where, $\ddot{x}_v$ and $\dot{\theta}_v$ are vehicle translational accelerations and rotational velocities, $\ddot{x}_s$, $\ddot{\theta}_s$, $\dot{x}_s$, $\dot{\theta}_s$ and x_s are the platform translational acceleration, rotational acceleration, platform translational velocity, rotational velocity and translational position respectively. d, e and γ are fixed parameters for pitch-surge mode. Further, λ and δ are the adaptive parameters which are continuously adjusted to minimize the cost function given below.

B.3.2. COST FUNCTION

The cost function generally used is a sum of quadratic functions of perception error and simulator states. The adaptation law drives to minimize this cost function.

$$J_x = \frac{1}{2}\left[(\ddot{x}_v - \ddot{x}_s)^T Q_a (\ddot{x}_v - \ddot{x}_s) + (\dot{\theta}_v - \dot{\theta}_s)^T Q_r (\dot{\theta}_v - \dot{\theta}_s) + \dot{x}_s^T Q_v \dot{x}_s + x_s^T Q_d x_s\right] \tag{B.42}$$

where, Q_a, Q_r, Q_v and Q_d are weights of the penalties on translational acceleration error, rotational velocity error, simulator velocity and simulator displacement respectively.

B.3.3. ADAPTIVE LAW

The adaptive parameters are determined by solving this optimization problem via steepest descent method, giving the following adaptive law:

$$\begin{aligned}\dot{\lambda} &= -K_\lambda \frac{\partial J_x}{\partial \lambda} + K_{i\lambda}(\lambda_0 - \lambda)\\ \dot{\delta}_x &= -K_\delta \frac{\partial J_x}{\partial \delta} + K_{i\delta}(\delta_0 - \delta)\end{aligned} \tag{B.43}$$

where, $K_\lambda, K_{i\lambda}, K_\delta$ and $K_{i\delta}$ are constants adaptation parameters. The first term of RHS changes the adaptive parameters in order to minimize the cost function. Meanwhile, the second term drives to restrain the deviation of λ and δ from their original values. The adaptive filter scheme derived above is summarized in figure B.4.

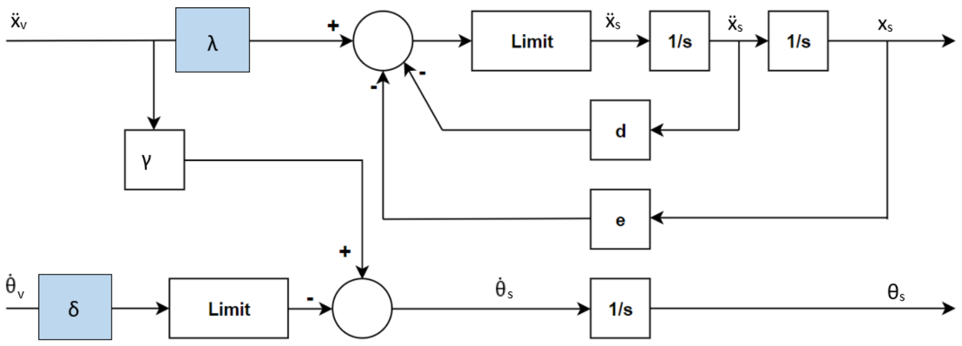

Figure B.4: General scheme of adaptive filter based MCA

It must be noted that the convergence of the optimization via steepest descent method strongly depends on the adaptation parameter, which also defines the convergence speed of algorithm.

As stated by Nahon et al. [32], the adaptive filter tends to give more realistic cues when the simulator is near its neutral position and only reduces the fidelity when simulator is at limits. However, the behavior of the simulator is extremely complicated and difficult to predict. Lastly, the absence of explicit constraints makes it hard for the designer to achieve a common tuning for varying maneuvers.

B.4. CONCLUSION

This appendix presented an overview of the motion cueing algorithms used conventionally. While CWF offers a computationally simple solution to the motion cueing problem, it has no control over the physical states of the simulator. Further, re-tuning is required if the maneuver is change considerably. Although the OWF and AWF-based MCA address this issue by allowing the user to put penalties on the simulator states, the optimization problem is still solved without any explicit constraints on these states. This lack of constraints in these approaches results in sub-optimal workspace utilization [32]. In conclusion, the main disadvantage of using the conventional methods comes from the inability to impose explicit constraints on the physical states of the motion platform.

Thus, in order to compute the optimal solution within the physical limitations of the motion platform, an optimization-based algorithm that can incorporate explicit constraints within the algorithm is required.

C

Linear MPC-Based Motion Cueing

The limitations of the conventional motion cueing approach (mentioned in appendix B), can be addressed by using Model Predictive Control (MPC). MPC is an optimal control strategy that uses a predefined system model and computes the control input by solving an optimization problem over a prediction horizon. The key advantage of using this technique is its ability to handle explicit constraints on both inputs and outputs. Further, the usage of the system model to predict and regulate future states helps to efficiently utilize the workspace and produce good reference tracking performance. Recent studies [19, 55–57] have shown that the MPC algorithm produces a better reference tracking and workspace utilization performance when compared to the filter-based approaches. It has also been shown that undesired effects like the occurrence of motion sickness are also lowered when using MPC [58].

In this appendix, the theoretical aspects pertaining to model predictive control are explained in section C.1. In section C.2, the linear MPC-based MCA is derived. Section C.3 presents a brief review of the existing research on MPC-based motion cueing and highlights the need for a non-linear MPC-based MCA.

C.1. Model Predictive Control: Theory

MPC is an advanced control strategy that is used to control a system while satisfying a set of constraints on the inputs and outputs. As mentioned earlier, the main advantage of this scheme is that it computes the control input by optimizing the system states and control effort over a finite time-horizon while satisfying the given constraints. Figure C.1 depicts the framework of a typical MPC controller. At each sampling instant, the reference (up to a finite prediction horizon) and the current system state (system output) are fed to the controller. The controller uses the prediction model (system model) and the current state to predict the evolution of the future state (in terms of the control input) for the entire prediction horizon. Using these predicted states and the reference, the error in terms of the control input is computed. The cost function is formulated using the obtained error values and the optimization problem is solved which adhering to the specified constraints. The first control input out of the obtained array of control inputs is fed to the system and the same process is repeated at every time instant.

C.1.1. Discretization

In the general MPC scheme, the continuous-time problem is discretized with a sampling time (T_s). The sampling frequency is given as $1/T_s$. Choosing the correct sampling frequency is extremely important for implementing any control scheme. While a discrete system with a large sampling frequency results in a high computation load, a system with a smaller sampling frequency tends to miss the dynamics of the plant which is to be controlled. The following rule of thumb is often used to determine the sampling frequency:

$$10\omega_b \leq \omega_s \leq 30\omega_b \tag{C.1}$$

Further, as per Shannon's sampling theorem, any frequency information above half of the sampling frequency emerging from the plant is either lost or results in aliasing. While these rules are to be kept in mind, the real-time performance and the computational load are often the determining factors for the MPC-based controllers.

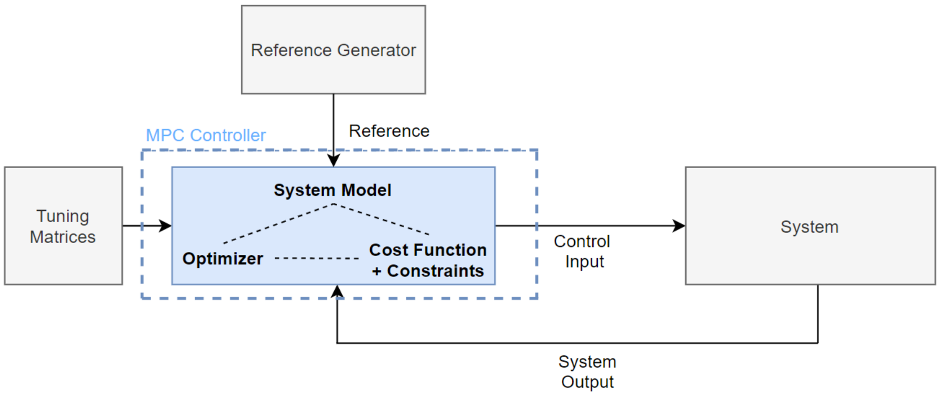

Figure C.1: MPC controller in control loop

C.1.2. PREDICTION

MPC uses a model of the plant (system model) to predict the future evolution of the system and compute the optimal control input. At each sampling interval, the problem is optimized over a window of samples known as the prediction horizon while being subjected to the constraints. A set of optimal control inputs over the whole prediction horizon is computed, but only the first control input is applied. After this, the whole problem and the prediction horizon shifts to the next stage by one sample. This is called the receding horizon policy.

The predicted window has N_p samples where N_p is called the prediction horizon length or simply, the prediction horizon. The prediction time (T_p), is defined in terms of sampling time (T_s) and prediction horizon (N_p) as follows:

$$T_p = N_p \cdot T_s \tag{C.2}$$

In each optimization step, an array of optimal control inputs is computed based on the control horizon (N_c). Lastly, the constraints are applied to the problem for a defined constraint horizon (N_{cons}). While choosing the control and constraint horizon, the following relations must be adhered:

$$\begin{aligned} N_c &\leq N_p \\ N_{cons} &\leq N_p \end{aligned} \tag{C.3}$$

As seen in figure C.2, the control inputs are calculated for the entire control horizon, after which it is assumed to be constant over the remaining prediction horizon. To simplify the computation, MPC computes the change in control input (Δu_c) instead of computing the control input (u_c) itself. If $N_c < N_p$, Δu_c is assumed to be zero for samples between N_c and N_p. As stated earlier, at each sampling time only the first control input $u_c(k|k)$ is applied to the system.

The prediction horizon is closely related to the time and computational complexity of the MPC problem and high values of N_p increase the computational effort and time

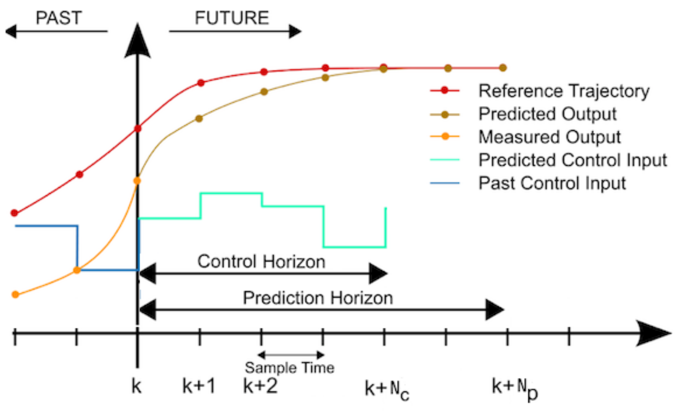

Figure C.2: Prediction and control horizon in MPC [59]

tremendously. Moreover, for systems with high modeling uncertainties, the solution may diverge from the optimal value if a larger prediction horizon is used. However, using a short prediction horizon often results in system instability and poor reference tracking performance. Therefore, it is extremely important to choose a prediction horizon which is long enough to make the system stable while keeping the effect of modeling uncertainties and computational effort in mind.

The effects of the control horizon length are similar to that of the prediction horizon. A very short control horizon (with respect to the prediction horizon) may result in sub-optimal results or instability while a longer one results in increase of the computational effort. Often, in order to simplify the problem, the control and constraint horizons are kept equal to the prediction horizon.

C.1.3. COST FUNCTION

The standard cost function for MPC consists of quadratic functions of both the tracking error and the control action along the prediction horizon. As stated earlier, the control action at each time instant is the first element of the input sequence that minimizes the cost function while satisfying the constraints. The system state is then updated using the estimates and the procedure is repeated. The total cost function is often divided into two parts, namely the stage cost and the terminal cost, as shown in equation C.4.

$$J(x(k),u(k)) = \sum_{k=1}^{N_p-1} \ell\big(x(k),u(k)\big) + V\big(x(N_p)\big) \tag{C.4}$$

The expression for stage cost is shown below:

$$\ell\big(x(k),u(k)\big) = \|x_{ref}(k) - x(k)\|_Q + \|u_{ref}(k) - u(k)\|_R \tag{C.5}$$

where Q and R are the positive semi-definite weight matrices for a penalty on tracking error and control input respectively. The stage cost function is defined such that it satisfies the following conditions:

$$\begin{aligned}&\ell(0,0) = 0\\&\ell\big(x(k),u(k)\big) > 0,\ \forall x(k) \in \mathbb{X},\ x(k) \neq x_{ref}(k)\end{aligned} \tag{C.6}$$

where $\mathbb{X}$ represents the set of states $x(k)$ which satisfies the system constraints. Further, the expression for the terminal cost is shown below:

$$V\big(x(N_p)\big) = \|x_{ref}(N_p) - x(N_p)\|_P \tag{C.7}$$

where P is the positive semi-definite weight matrix for a penalty on the tracking error at the terminal stage of the prediction horizon.
For finite prediction horizon problems, stability can be guaranteed by choosing a suitable terminal cost (V) and terminal attractive region Ω [60–62]. Even though the conditions for asymptotic stability are clearly defined, choosing V and Ω is still an open problem [63]. It is shown in [64], that stability can be guaranteed by simply tuning the matrices Q, R, and P. Further, a longer prediction horizon (N_p) would help the algorithm to achieve convergence [65].

C.1.4. SYSTEM CONSTRAINTS

MPC allows to apply explicit constraints on the system output and the control inputs. Typically, these constraints are represented as follows:

$$\begin{aligned}&\boldsymbol{x}_{\min} \leq \boldsymbol{x}(k) \leq x_{\max}, && i = 1\ldots N_{cons}\\&\boldsymbol{u}_{\min} \leq \boldsymbol{u}(k) \leq \boldsymbol{u}_{\max}, && i = 1\ldots N_{cons}\\&\Delta\boldsymbol{u}_{\min} \leq \Delta\boldsymbol{u}(k) \leq \Delta\boldsymbol{u}_{\max}, && i = 1\ldots N_{cons}\end{aligned} \tag{C.8}$$

It must be noted that the system model is also included as a constraint within the MPC formulation. Further, depending on the nature of the system, objective function and constraints, the MPC problem is classified as linear or non-linear.

C.1.5. OPTIMIZATION PROBLEM

Finally, the problem is converted to a constrained optimization problem which minimizes the sum of stage cost and terminal cost while satisfying the constraints and system dynamics to compute the optimal input. Therefore the optimization problem can be written as:

$$\begin{aligned}&u(k) = \text{argmin } J(x(k),u(k))\\&\textbf{s.t}\\&x(k+1) = f\Big(x(k),u(k)\Big)\\&\boldsymbol{u}_{\min} \leq \boldsymbol{u}(k,i) \leq \boldsymbol{u}_{\max}, && i = 1\ldots N_{cons}\\&\Delta\boldsymbol{u}_{\min} \leq \Delta\boldsymbol{u}(k,i) \leq \Delta\boldsymbol{u}_{\max}, && i = 1\ldots N_{cons}\\&\boldsymbol{x}_{\min} \leq \boldsymbol{x}(k,i) \leq x_{\max}, && i = 1\ldots N_{cons}\end{aligned} \tag{C.9}$$

C.1.6. EXPLICIT MPC

Real-time implementation of complex MPC problem is difficult as the optimization is computationally expensive. To overcome this problem, Bemporad et al. [66] used the parametric programming technique to derive the explicit MPC scheme which computes the solution to the QP problem offline as a function of initial states. This solution is in the form of a piece-wise affine function (PWA) and is stored as the sum of coefficients of the PWA for each control region of the state space, as well as coefficients of parametric representations of all the regions. It is also stated that there are finite number of solutions that need to be computed and stored. The offline solution is given as follows:

$$\Delta u^{*}(x) = C \cdot x + d \tag{C.10}$$

where C and d are the solutions for the corresponding control region.

Though in low dimensional problems, explicit MPC offers the advantage of low computational time over the implicit scheme, it suffers high degradation as the problem is extended to the higher dimensions [67].

C.1.7. Input Blocking Strategies

As stated earlier, the computational burden of an MPC problem depends on the number of free control inputs (u). In this method, the optimal problem is reduced from u to u' inputs by clustering the inputs into groups and binding the inputs of each group with each other.

$$\begin{aligned} &\text{Conventional input:} && \boldsymbol{u} = \left[\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N_c-1}\right]^{\top} \\ &\text{Input blocking input:} && \boldsymbol{u}' = \left[\boldsymbol{u}'_0, \boldsymbol{u}'_1, \ldots, \boldsymbol{u}'_{N'_c-1}\right]^{\top} \end{aligned}$$

where, N_c is the control horizon, N'_c is the input blocking horizon and $N_c < N_c$. The new control input u' is given by:

$$\boldsymbol{u}' = \left(\boldsymbol{T} \otimes \boldsymbol{I}_{N'_c}\right)\boldsymbol{u} \tag{C.11}$$

where $\otimes$ is the Kronecker product and $T \in R^{N_c \ddot{O} N_c}$ is called input blocking matrix. At each row of this the blocking matrix (T), there exist only one element which is unity and the rest of the elements are zero. An example of the matrix T, used by Bruschetta et al. [68] is shown below:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.12}$$

Usually, the matrix T is set to block the inputs near the end of the control horizon. This is because it affects the control problem to a lesser extent when compared to blocking the initial inputs.

C.2. Linear MPC-Based MCA

Recently, linear MPC-based algorithms have been implemented in the domain of motion cueing. The general scheme of a typical lineal MPC-based MCA is shown in figure C.3. The translational acceleration and angular velocity at the vehicle CoG in VF are transformed to the driver's eye-point in DF. The resulting signals are passed through the

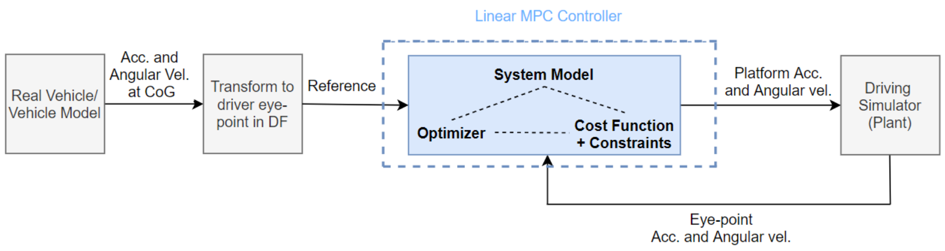
Linear MPC Controller
Real Vehicle/ Vehicle Model
Acc. and Angular Vel. at CoG
Transform to driver eye-point in DF
Reference
System Model
Optimizer
Cost Function + Constraints
Platform Acc. and Angular vel.
Driving Simulator (Plant)
Eye-point Acc. and Angular vel.

Figure C.3: Linear MPC-based motion cueing algorithm

vestibular system model and then passed to the MPC controller as the reference signal. The MPC controller incorporates a linear system model to predict the evolution of future states of the system. The system model used is the linear vestibular system model, which is augmented to include the simulator states. Based on the current system state and future estimates, the cost function is formulated. The control input, i.e the platform acceleration and angular velocity is then derived based on the optimization of this cost function subjected to the system and input constraints. The first control input is the control action and the same is sent to the plant. The plant sends the perceived eye-point acceleration and angular velocity as feedback to the controller.

In the following subsections, the linear MPC problem is derived and formulated.

C.2.1. SYSTEM MODEL

SEMI-CIRCULAR CANALS

Representing the transfer function of the semi-circular canal (equation A.17) in its observable canonical state-space form, we get:

$$\dot{x}_{scc} = A_{scc}x_{scc} + B_{scc}u_s \tag{C.13}$$

$$y_{scc} = C_{scc}x_{scc} + D_{scc}u_s \tag{C.14}$$

where $y_{scc} = \hat{\omega}_i$ and $u_s = \omega_i$ in one of the three canonical directions. Therefore, the complete model for semi-circular canals, in all three canocical directions combined is given as follows:

$$A_s = \begin{bmatrix} A_{scc_x} & 0_{2x2} & 0_{2x2} \\ 0_{2x2} & A_{scc_y} & 0_{2x2} \\ 0_{2x2} & 0_{2x2} & A_{scc_z} \end{bmatrix} \quad B_s = \begin{bmatrix} B_{scc_x} & 0_{2x1} & 0_{2x1} \\ 0_{2x1} & B_{scc_y} & 0_{2x1} \\ 0_{2x1} & 0_{2x1} & B_{scc_z} \end{bmatrix}$$
$$C_s = \begin{bmatrix} C_{scc_x} & 0_{1x2} & 0_{1x2} \\ 0_{1x2} & C_{scc_y} & 0_{1x2} \\ 0_{1x2} & 0_{1x2} & C_{scc_z} \end{bmatrix} \quad D_s = \begin{bmatrix} D_{scc_x} & 0 & 0 \\ 0 & D_{scc_y} & 0 \\ 0 & 0 & D_{scc_z} \end{bmatrix} \tag{C.15}$$

and the input and the output signals are shown in equation C.16 and C.17. It must be noted that both of these quantities are expressed at the driver eyepoint (D_0) in DF.

$$u_s = \omega_{D_0}^{DF} = [\omega_\phi \; \omega_\theta \; \omega_\psi]^T \tag{C.16}$$

$$y_s = \hat{\omega}_{D_0}^{DF} = [\hat{\omega}_\phi \; \hat{\omega}_\theta \; \hat{\omega}_\psi]^T \tag{C.17}$$

OTOLITH ORGANS

Representing the otolith transfer functions (equation A.18) in its observable canonical state-space form, we get:

$$\dot{x}_{oth} = A_{oth} x_{oth} + B_{oth} u_o \tag{C.18}$$

$$y_{oth} = C_{oth} x_{oth} + D_{oth} u_o \tag{C.19}$$

where $y_{oth} = \hat{a}_i$ and $u_o = a_i$, in one of the three canonical directions. Therefore, the complete model for otolith organs is given as follows:

$$\begin{gathered} A_o = \begin{bmatrix} A_{oth_x} & 0_{2x2} & 0_{2x2} \\ 0_{2x2} & A_{oth_y} & 0_{2x2} \\ 0_{2x2} & 0_{2x2} & A_{oth_z} \end{bmatrix} \quad B_o = \begin{bmatrix} B_{oth_x} & 0_{2x1} & 0_{2x1} \\ 0_{2x1} & B_{oth_y} & 0_{2x1} \\ 0_{2x1} & 0_{2x1} & B_{oth_z} \end{bmatrix} \\ C_o = \begin{bmatrix} C_{oth_x} & 0_{1x2} & 0_{1x2} \\ 0_{1x2} & C_{oth_y} & 0_{1x2} \\ 0_{1x2} & 0_{1x2} & C_{oth_z} \end{bmatrix} \quad D_o = \begin{bmatrix} D_{oth_x} & 0 & 0 \\ 0 & D_{oth_y} & 0 \\ 0 & 0 & D_{oth_z} \end{bmatrix} \end{gathered} \tag{C.20}$$

and the input and the output signals are shown in equation C.21 and C.22. It must be noted that both of these quantities are also expressed at the driver eyepoint (D_0) in DF.

$$u_o = a_{D_0}^{DF} = [a_x \; a_y \; a_z]^T \tag{C.21}$$

$$y_o = \hat{a}_{D_0}^{DF} = [\hat{a}_x \; \hat{a}_y \; \hat{a}_z]^T \tag{C.22}$$

INCORPORATING TILT COORDINATION

The complete model of transitional acceleration should include the tilt coordination effects into it. Thus, the otolith matrix is augmented to the following:

$$A_{\bar{o}} = \left[\begin{array}{c|c} A_o & \bar{B} \\ \hline 0 & 0 \end{array}\right], B_{\bar{o}} = \left[\begin{array}{c|c} B_o & 0 \\ \hline 0 & I_3 \end{array}\right], \; C_{\bar{o}} = \left[\begin{array}{c|c} C_o & 0 \end{array}\right] \text{ and } D_{\bar{o}} = \left[\begin{array}{c|c} D_o & 0 \end{array}\right] \tag{C.23}$$

where

$$\bar{B} = B_o \cdot \begin{bmatrix} 0 & g & 0 \\ -g & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{C.24}$$

and the input and the output signals are mentioned in equation C.25 and C.26.

$$u_{\bar{o}} = [a_{D_0}^{DF} \; ; \; \omega_{D_0}^{DF}] \tag{C.25}$$

$$y_{\bar{o}} = \hat{a}_{D_0}^{DF} \tag{C.26}$$

INTEGRATING SIMULATOR STATES

The positions and velocities of the driver's eyepoint in DF can also be included in the optimization problem in order to control these states and implement the washout action. These states are obtained by using the following second-order integral system:

$$\dot{x}_l = A_l x_l + B_l u \tag{C.27}$$

where

$$A_l = \begin{bmatrix} A_i & 0 & 0 \\ 0 & A_i & 0 \\ 0 & 0 & A_i \end{bmatrix} \text{with } A_i = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$
$$B_l = \begin{bmatrix} B_i & 0 & 0 \\ 0 & B_i & 0 \\ 0 & 0 & B_i \end{bmatrix} \text{with } B_i = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{C.28}$$

and

$$x_l = [v^{DF}_{D_0,\,x} \quad p^{DF}_{D_0,\,x} \quad v^{DF}_{D_0,\,y} \quad p^{DF}_{D_0,\,y} \quad v^{DF}_{D_0,\,z} \quad p^{DF}_{D_0,\,z}]^T \tag{C.29}$$

$$u = [a^{DF}_{D_0}\ ;\ \omega^{DF}_{D_0}] \tag{C.30}$$

THE COMPLETE SYSTEM MODEL

The complete system model is derived by combining the vestibular system model and the simulator states as follows:

$$\begin{aligned} \dot{x}_v &= A_v x_v + B_v u_v \\ y_v &= C_v x_v + D_v u_v \end{aligned} \tag{C.31}$$

where

$$A_v = \begin{bmatrix} A_s & 0 & 0 \\ 0 & A_{\bar{o}} & 0 \\ 0 & 0 & A_l \end{bmatrix}, B_v = \begin{bmatrix} 0 & B_s \\ & B_{\bar{o}} \\ & B_l \end{bmatrix}, C_v = \begin{bmatrix} C_s & 0 & 0 \\ 0 & C_{\bar{o}} & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{bmatrix} \text{and } D_v = \begin{bmatrix} 0 & D_s \\ & D_{\bar{o}} \\ & 0 \\ 0 & I \end{bmatrix} \tag{C.32}$$

and the input and output signals are:

$$u_v = [a^{DF}_{D_0}\ ;\ \omega^{DF}_{D_0}] \tag{C.33}$$

$$y_v = [\hat{a}^{DF}_{D_0}\ ;\ \hat{\omega}^{DF}_{D_0}\ ;\ \alpha^{DF}_{D_0}\ ;\ v^{DF}_{D_0}\ ;\ p^{DF}_{D_0}\ ;\ \dot{\alpha}^{DF}_{D_0}] \tag{C.34}$$

Here α and $\dot{\alpha}$ are the angular displacement and the angular velocity vector respectively.

Finally, the system is discretized by using bilinear transform (also known as Tustin's method), thus giving the following discrete time system:

$$\begin{aligned} x_{k+1} &= A_d x_k + B_d u_k \\ y_k &= C_d x_k + D_d u_k \end{aligned} \tag{C.35}$$

C.2.2. CONSTRAINTS

In the framework of this linear MPC-based MCA, the following constraints are applied:

- Constraints on tilt rate ($\dot{\alpha}^{DF}_{D_0}$) to ensure that the tilt coordination effects are not perceived by the observer.

- Constraints on the displacement and velocity of the driver's eyepoint ($v^{DF}_{D_0}$ and $p^{DF}_{D_0}$) to ensure that the motion platform does not exceed its physical limit.

C.2.3. REFERENCE

The reference vector contains the following variables:

$$y_v = [\hat{a}_{ref}\ \hat{\omega}_{ref}\ \alpha_{ref}\ v_{ref}\ p_{ref}\ \dot{\alpha}_{ref}]^T \tag{C.36}$$

To ensure that the platform returns to the neutral position, the references for α_{ref} v_{ref} p_{ref} and $\dot{\alpha}_{ref}$ are given as zero. Further, $\hat{a}_{ref}$ and $\hat{\omega}_{ref}$ are computed by translating the translational and angular acceleration obtained from the vehicle model to the driver's eyepoint in DF and then passing them through the vestibular system model.

C.2.4. OPTIMIZATION PROBLEM

Model Predictive Control calculates the control input by solving the following linear optimization problem:

$$\begin{aligned}
&u(k) = \text{argmin } J(x(k), u(k)) \\
&\textbf{s.t} \\
&x(k+1) = A_d \cdot x(k) + B_d \cdot u(k) \\
&y(k) = C_d \cdot x(k) + D_d \cdot u(k) \\
&\dot{\alpha}_{\min} \le \dot{\alpha} \le \dot{\alpha}_{\max} \\
&v^{DF}_{D_0,\, min} \le v^{DF}_{D_0} \le v^{DF}_{D_0,\, \max} \\
&p^{DF}_{D_0,\, min} \le p^{DF}_{D_0} \le p^{DF}_{D_0,\, \max}
\end{aligned} \tag{C.37}$$

The above formulated MPC problem has the following sets of variables:

- The control variables ($a^{DF}_{D_0}$ and $\omega^{DF}_{D_0}$), that are used to control the plant. These are the linear accelerations and angular velocities respectively.

- The tracking variables ($\hat{a}^{DF}_{D_0}$, $\hat{\omega}^{DF}_{D_0}$, $\alpha^{DF}_{D_0}$, $v^{DF}_{D_0}$, $p^{DF}_{D_0}$ and $\dot{\alpha}^{DF}_{D_0}$), that are the output variable tracked by the MPC controller. These are the sensed specific forces, sensed angular velocities, inclination angles, eye-point positions, eye-point velocities and inclination velocities respectively.

- The constrained variables ($\dot{\alpha}^{DF}_{D_0}$, $v^{DF}_{D_0}$ and $p^{DF}_{D_0}$), that are the variable on which the constraints are imposed. These are the inclination velocities, eye-point positions and eye-point velocities respectively.

C.3. REVIEW OF THE PAST WORK

MPC-based motion cueing has been an active area of research lately. Various linear and non-linear modeling approaches have been explored and the superiority of this method compared to the conventional approach has been established. The most relevant research related to MPC based motion cueing is presented below.

Augusto et al. [19] implemented MPC-based motion cueing on the Chalmers University simulator. A linear model of the human vestibular system was used and tilt coordination was incorporated within the system. The error between the reference and the model output was calculated and a linear MPC problem was formulated. For the reference input, the following three hypotheses were considered and compared:

1. Assuming that the input to simulator from the vehicle is constant.
2. Assuming that the input to the vehicle is constant.
3. Assuming that the derivative of the input to the vehicle is constant.

From the results, it was concluded that the third assumption lead to the worst results while the first assumption lead to the best results. Further, prediction, control and constraint horizons were chosen by trial and error. The controller frequency was chosen as 20 Hz. As future work, the use of a more recent version of the vestibular system model and performing stability analysis was highlighted. It should be noted that in this work, the tilt rate limits were not imposed.

Baseggio et al. [69] used the recent vestibular model derived by Telban et al. [3] in the MPC formulation. It was assumed that the reference signal is known 2 seconds prior and the result showed improvement over the constant reference case. Assuming this availability of reference limited the research to the open-loop (no driver-in-loop) scenarios.

Bruschetta et al. [68] implemented motion cueing with linear MPC by using a more recent version of the linearized vestibular model with tilt coordination. A quadratic cost function was used and inequality constraints were applied on the eye-point displacement and velocity. The constrained optimization was performed using the active set method in the qpOASES toolbox. Further, the authors mentioned that if the future reference is available, the longest prediction and control horizon produces the best results. However, because of the unavailability of such a future reference, the horizon length is limited. The computation time also increases exponentially with the length of the control horizon. Hence, in their work, input-blocking strategies were applied to overcome real-time computational limitations. The algorithm worked at the control frequency of 100 Hz. In order to reduce the computational effort, in [1], the authors decoupled the problem into four groups (Pitch-surge mode, Roll-sway mode, Yaw mode and Heave mode) and solved separately. The optimization was carried out for the decoupled groups using parallel computing. The constraints imposed on the driver's eyepoint in the driver frame of reference ensure the problem remains linear in nature. A 200 Hz control frequency was achieved.

Garrett et al. [55] implemented an MPC-based cueing algorithm which uses the actuator positions and velocities as the constraints. This was done to improve workspace utilization while making sure that the platform never exceeds its physical limitations. However, this makes the problem non-linear in nature. In order to make the problem linear, approximations were applied to the constraints on the actuator lengths. Although this simplification affected constraint handling, it was mentioned that this was not an issue for the range of maneuvers considered in the paper. The optimizer used is the primal barrier method [70]. The MCA performed better than conventional MCAs. A control frequency of 40 Hz was achieved. It must be noted that although linearized hexapod kinematics were included in the model, the inputs were not transformed to the inertial frame of reference. As an approximation of the same, the inputs in the driver frame of reference were provided to the algorithm.

Grottoli et al. [71] designed a non-linear MPC-based MCA which used the dynamic model of the motion platform while the vestibular system model was not used in this study. Constraints were imposed on the actuator lengths using the linearized inverse kinematics. The control frequency of 10 Hz was chosen. The paper focused on the prediction strategies used to compute the reference. Two prediction strategies were considered, namely Oracle, which is the ideal prediction strategy that knows the exact future reference, and Constant, a prediction strategy that keeps the current linear accelerations and angular velocities constant for the entire prediction horizon. Indicators such as correlation, delay and Inter-quartile range for workspace utilization were used to compare the two strategies. The study indicated that while the constant strategy provided reasonable results, the results for oracle strategy show reduced delay, improved correlation with the reference and better workspace utilization.

Degdelen et al. [72] implemented the MPC-based MCA in the Renault ULTIMATE Simulator. The study was done for a single DOF cueing problem (surge acceleration) and tilt coordination was demonstrated as an extension to the basic algorithm. The control frequency was chosen as 100 Hz. Taking it further, in [67], an explicit MPC-based concept for the Renault ULTIMATE Simulator was presented. This concept was based on the Multi-Parametric Toolbox (MPT) Matlab toolbox. The control problem was decoupled into four separate cases (pitch-surge, roll-sway, heave, and yaw) and a stability criterion was determined. The control frequency was 100 Hz and the algorithm works in real-time. However, fast degradation in the computational effort was seen as the problem was extended to higher dimensions. In [73], Fang et al. presented an implicit MPC-based MCA approach. The new approach significantly improved efficiency. Compared to their previous work with explicit MPC, the implicit algorithm showed similar results but was superior since it used a more complex system model. The algorithm was implemented in real-time using qpOASES solver. Taking their work a step further, in [74] the authors developed a fast MPC-based MCA to improve the algorithm's real-time performance. The optimization problem was solved in two steps: first, finding the QP solution by a computation without constraints, then checking the solution in an accessible limit. The developed algorithm was tested for different scenarios on the Renault ULTIMATE simulator. It was reported that the new algorithm was about 5 – 10 times faster than the conventional algorithms. Further, the authors expected that the saved CPU resources could be used for reference prediction leading to a real-time driver-in-loop (DIL) MCA.

Mohammadi [75] mentioned three main challenges in designing the MPC based motion cueing algorithm: first, no definitive methods for tuning the weight matrices. Second, no definitive methods for horizon selection and third, unavailability of future reference. To address the first concern, the author used a multi-objective evolutionary solution to solve the problem of MPC tuning. The proposed method was shown to provide an effective technique to tune the MPC based optimization. Similar to this, to address the second concern, a genetic algorithm was applied to minimize the human sensation error and displacement. Lastly, to address the third concern, an artificial neural network approach is proposed to improve the reference signals. The simulation results show significant improvement when this technique is applied.

C.4. CONCLUSION

This appendix presented an overview of the use of MPC for motion cueing. The main advantage of using an MPC-based algorithm for motion cueing applications is that the physical states of the motion platform can be incorporated as explicit constraints within the problem. Further, the model-based approach of MPC also allows incorporating the human vestibular model within the problem to achieve high fidelity motion cueing. Lastly, the ability of MPC to handle multivariate systems makes it a potential candidate for the motion cueing problem. The superiority of this method compared to the conventional approach has been already established in the literature [55, 58, 76].

Although the linear MPC-based approach provides better results than the conventional filter-based motion cueing strategies, applying constraints on the displacement and velocity of driver's eyepoint results in sub-optimality. This is because the constraints are imposed on the driver's eye-point and there is no explicit permissible limit for the driver's eyepoint. Further, this point is expressed in the non-inertial driver frame of reference, which makes it very hard for the designer to determine the available workspace. In order to find the available workspace, the forward kinematic relations have to be used which increases the computational effort and makes the problem non-linear. To keep the problem linear, a conservatively chosen constant space is often used as the workspace for driver's eye-point. However, this tends to produce sub-optimal results.

This problem can be solved by using the inverse kinematics of the motion platform and transforming all the quantities in the inertial frame of reference. However, it must be noted that this would make the problem non-linear. By using this approach, it would be possible to incorporate explicit constraints on the lengths of the hexapod actuators, limits for which are explicitly defined. Further, the control input can also be derived at the centroid of the moving base of the platform in IF instead of the driver's eye-point in DF. Therefore, an efficient non-linear MPC approach that can incorporate the non-linear inverse kinematics of the 6-DOF hexapod within the MPC controller is required to increase the effectiveness of the produced cues and utilize maximum workspace.

D
Non-linear MPC

As mentioned in Appendix C, Linear Model Predictive Control (L-MPC) is a well-suited strategy for motion cueing. Although its superiority over the conventional MCAs is well established, the results produced are sub-optimal. The sub-optimality stems from the linear approximations are made to model the non-linear motion platform system. Therefore, in principle, a non-linear MPC algorithm that can incorporate the non-linear inverse kinematics of the 6-DOF hexapod can achieve the desired optimal results.

Non-linear Model Predictive Control (NL-MPC) is an effective way of tackling the problems with nonlinear system dynamics and constraints. However, a major limitation of using a non-linear optimization-based algorithms is the high computational time and cost which is required to obtain the solution. Recent developments in the digital world along with the algorithmic developments have significantly sped up the computational capacity of the modern computers, allowing for the deployment of non-linear MPC-based controllers at an outstanding speeds [77–79].

A typical NL-MPC problem is shown in equation D.1. The formulation of the problem is similar to the L-MPC problem with the difference that the system equations and/or the constraint relations are non-linear in nature. It should be noted that in this example, the terminal cost is excluded merely for the sake of brevity.

$$\begin{aligned}
&u = \text{argmin } \textstyle\sum_{k=0}^{N_p-1} \|x_{ref}(k) - x(k)\|_Q + \|u_{ref}(k) - u(k)\|_R \\
&\textbf{s.t} \\
&x(0) = x_0 \\
&x(k+1) = f\Big(x(k), u(k)\Big) \\
&h(x(k), u(k)) \le 0
\end{aligned} \tag{D.1}$$

A fast implementation of the NL-MPC problem is done via Real-time iteration (RTI) proposed by Diehl et al. [80, 81]. In their research, it is shown that since NL-MPC requires to solve closely related Optimal Control Problem (OCP) successively, the solution of the OCP at the current time-step is very similar to the solution obtained at the previous time step. The RTI approach exploits this by achieving the convergence of the NL-MPC solution in conjunction with the evolution of the dynamics of the system. The approach relies on the fast contraction rate of Newton-type optimization techniques [81]. Further, in their research, Diehl et al. have stated that RTI exploits the similarities in the L-MPC and NL-MPC approach, by "bridging the gap" between both. It is said that because the NL-MPC problem is solved approximately by solving only one QP per sampling instant, it can be seen as a special case of linear time-varying MPC with two important features:

1. The linearization of the system dynamics should occur online and should be done at the current state and control prediction rather than on the reference trajectory.
2. The system dynamics should be simulated using a numerical integration scheme.

A detailed explanation of this can be found in [81]. The RTI approach has been formally studied and has been verified in many different deployments. One such deployment is the ACADO toolkit [82] which generates a highly efficient formulation for the NL-MPC implementation. The toolbox consists of algorithms for discretization and linearization of the nonlinear system, and an algorithm to evaluate the solution. The solution is computed by using the sequential quadratic programming (SQP) scheme. The RTI scheme performs a single SQP iteration per time step in order to quickly deliver an approximate solution to the optimization problem. In this research, the ACADO toolkit is used in to solve the NL-MPC problem. The solver settings used in ACADO are mentioned in table D.1. Further, an explanation for choosing these particular settings has been given in the following sections.

Table D.1: Solver settings used in ACADO toolkit

Parameter	**Solver setting**
Hessian Approximation	Guass Newton
Discretization Type	Multiple Shootings
Sqarse QP solution	Full Condensing
Integrator Type	Explicit Range-Kutta integrator of order 2
Number of Integrator Steps	2N
QP Solver	qpOASES
Levenberg Maquardt Regularization Parameter	$1e^{-4}$
Hotstart QP	Yes

In section D.1, the OCP for NL-MPC problem is formulated. Section D.2 presents the steps involved in SQP formulation and gives an overview of the optimization technique. Section D.3 highlights the QP solver used for solving the SQP problem. Lastly, in section D.4, the methodology of the real-time iteration approach is presented.

D.1. Optimal Control Problem Formulation

The MPC controller aims at solving the following continuous time non-linear optimization problem:

$$\begin{aligned} & u = \text{argmin} \quad \frac{1}{2} \int_{t_0}^{t_0+T} \left(\| x_{ref}(t) - x(t) \|_Q^2 + \| u(t) \|_R^2 \right) dt \\ & \textbf{s.t} \\ & x(t_0) = x_0 \\ & \dot{x}(t) = f(x(t), u(t)) \\ & h(x(t), u(t)) \leq 0 \qquad \text{for all } t \in [t_0, t_0 + T] \end{aligned} \tag{D.2}$$

Here, x_0 denotes the current state measurement. In order to convert this problem into a standard SQP problem, it must be discretized and linearized. For problems with varying reference, discretization is done before linearization as it results in a better approximation [81].

To discretize the non-linear dynamics of the system, boundary value problem-solving techniques are generally used. Commonly used approaches are single shooting method and direct multiple shootings method. The direct multiple shootings method divides the given interval ($[t_0, t_0 + T]$) into several smaller intervals ([k, k+1]) for ($k = 0 \dots N_p - 1$)

and solves an initial value problem in each of these smaller intervals. It further imposes additional matching conditions to form a solution on the whole interval. The objective function and path constraints are discretized at the same grid as the states and controls. Unlike the multiple shootings method, the single shooting method solves the boundary value problem on a single interval. Although the single shooting method is simple, it has been shown that the direct multiple shooting method shows superior performance in modeling the non-linearities and numerical stability over single shooting methods [83]. For this reason, the direct multiple shootings method is selected in this research.

The RTI scheme makes use of an integrator to linearize the non-linear system and constraints at the current prediction. Therefore, the accuracy of the linearized discrete-time model also depends on the type of integrator chosen. Therefore, it is important to choose the right integrator which delivers predictions that are accurate enough to predict the evolution of the system over time. The ACADO toolbox offers a range of implicit and explicit integrators. Since the nonlinear differential equations used in this research are not stiff, explicit integrators can be chosen for this application. In their research, Gros et al. [81] used an example OCP to demonstrate that, using 2 steps of Range-Kutta integrator yields a closed-loop behaviour which is very close to that obtained by using 30 steps of explicit Euler integrator but its preparation phase takes only about 26% of the time taken by the explicit Euler integrator. Therefore, explicit Range-Kutta integrator was chosen for this research.

Other than this, the number of integration steps and the order of the integrator is also an important parameter choice that has to be made. While larger number of steps or higher order integrators offer higher accuracy, both increase the computational effort required to prepare and solve the OCP. Based on successive trials, an explicit Range-Kutta integrator of order 2 along with 2 integrator steps per instance (i.e. 2N steps for the entire horizon) was chosen in this research.

Finally, the discrete-time OCP can be written as:

$$\begin{aligned} & u = \operatorname{argmin} \sum_{k=0}^{N_p-1} \begin{bmatrix} x_k - x_k^{\text{ref}} \\ u_k - u_k^{\text{ref}} \end{bmatrix}^\top W_k \begin{bmatrix} x_k - x_k^{\text{ref}} \\ u_k - u_k^{\text{ref}} \end{bmatrix} \\ & \textbf{s.t} \\ & x(0) = x_0 \\ & x(k+1) = f\Big(x(k), u(k)\Big) \\ & h(x(k), u(k)) \leq 0 \qquad \text{for } k = 0 \dots N_p - 1 \end{aligned} \tag{D.3}$$

This resulting OCP can be solved by using the SQP approach.

D.2. SEQUENTIAL QUADRATIC PROGRAMMING

Sequential Quadratic Programming (SQP) is a state of the art algorithm for solving constrained non-linear optimization problems. In the SQP approach, the optimization problem is sequentially approximated by quadratic sub-problems (QPs) which provide the directions in which the Newton steps are to be taken to move towards the solution starting from an available guess. The iteration is repeated by taking Newton steps until the convergence criteria is met. Given an initial guess (x^{guess}, u^{guess}), the problem given in equation D.3 is approximated by the following QP:

$$\begin{aligned}
&\text{QP}_{\text{NL-MPC}}\left(x_0, x^{\text{guess}}, u^{\text{guess}}, x^{\text{ref}}, u^{\text{ref}}\right) = \\
&(\Delta x, \Delta u) = \text{argmin} \quad \sum_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} H_k \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + J_k^T \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \\
&\textbf{s.t} \\
&\Delta x_0 = x_0 - x_0^{\text{guess}} \\
&\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k + r_k \\
&C_k \Delta x_k + D_k \Delta u_k + h_k \leq 0
\end{aligned} \tag{D.4}$$

where

$$\begin{aligned}
&A_k = \left.\frac{\partial f(x,u)}{\partial x}\right|_{x_k^{\text{guess}}, u_k^{\text{guess}}}, \quad B_k = \left.\frac{\partial f(x,u)}{\partial u}\right|_{x_k^{\text{guess}}, u_k^{\text{guess}}} \\
&C_k = \left.\frac{\partial h(x,u)}{\partial x}\right|_{x_k^{\text{guess}}, u_k^{\text{guess}}}, \quad D_k = \left.\frac{\partial h(x,u)}{\partial u}\right|_{x_k^{\text{guess}}, u_k^{\text{guess}}} \\
&r_k = f\left(x_k^{\text{guess}}, u_k^{\text{guess}}\right) - x_{k+1}^{\text{guess}}, \quad h_k = h\left(x_k^{\text{guess}}, u_k^{\text{guess}}\right) \\
&J_k = W_k \begin{bmatrix} x_k^{\text{guess}} - x_k^{\text{ref}} \\ u_k^{\text{guess}} - u_k^{\text{ref}} \end{bmatrix}
\end{aligned} \tag{D.5}$$

and H_k is an approximation for the Hessian. Further, the Gauss-Newton Hessian approximation given by $H_k = W_k$ is used. Also, an ill-conditioned or indefinite Hessian can be regularized by adding Levenberg-Marquardt regularization:

$$H_k = J_k^T J_k + \alpha \cdot I \tag{D.6}$$

where J_k is the Jacobian, and the Levenberg-Marquardt regularization parameter $\alpha > 0$. Essentially, the SQP algorithm replaces the non-linear objective function with a quadratic approximation and the constraint relation with a linear approximation. The details of the algorithm is presented in [80] and an overview of the same is given below:

Algorithm 1: SQP Algorithm for NL-MPC

```
Input: $x_0, x^{\text{guess}}, u^{\text{guess}}, x^{\text{ref}}, u^{\text{ref}}$
while Not Converged do
    1) Evaluate $r_k, h_k$ and $A_k, B_k, C_k, D_k, H_k, J_k$ using equation D.5
    2) Construct and solve the $\text{QP}_{\text{NL-MPC}}$ (equation D.4) to get newton direction
       $(\Delta x, \Delta u)$
    3) Compute step-size $\alpha \in [0,1]$ to guarantee descent
    4) Update $(x^{\text{guess}}, u^{\text{guess}})$ with Newton step:
        $(x^{\text{guess}}, u^{\text{guess}}) \rightarrow (x^{\text{guess}}, u^{\text{guess}}) - \alpha \cdot (\Delta x, \Delta u)$
end
Return NL-MPC solution : $(x, u) = (x^{\text{guess}}, u^{\text{guess}})$
```

It must be noted that the SQP algorithm requires the initial guess $(x^{\text{guess}}, u^{\text{guess}})$ as an input. Choosing an appropriate initial guess is essential to get a reliable and fast convergence. A good initial guess not only reduces the possibility of an infeasible exit of SQP, but it also allows the algorithm to take full Newton steps ($\alpha = 1$), hence resulting in fast convergence. As per Diehl et al. [81], a very good initial guess for the current time instant can stem from the solution obtained at the previous time instant. This strategy is called shifting and the corresponding SQP is called hot-started/warm-started SQP. ACADO uses the following shifting procedure:

$$\begin{aligned} x^{\text{guess}}_{i,k} &= x_{i-1,k+1}, && k = 0,\ldots,N-1 \\ u^{\text{guess}}_{i,k} &= u_{i-1,k+1}, && k = 0,\ldots,N-1 \\ x^{\text{guess}}_{i,N} &= f\left(x^{\text{guess}}_{i,N-1}, u^{\text{guess}}_{i,N-1}\right) \end{aligned} \tag{D.7}$$

The ACADO toolkit allows to employ condensing strategies to reduce the size of the formulated QP by eliminating the intermediary states. A detailed explanation of the condensing procedure is given in [84].

D.3. QP SOLVER

The resulting condensed QP can then be solved via several online solvers that exist for solving convex QP problems. The solvers can be categorized according to the constraint handling approaches that they use. The following two approaches are commonly used:

- Interior Point (IP): There are two variants of the IP method - the primal-dual method and the barrier method. The barrier method uses a weighted barrier function to replace the inequality constraints. The function is added to the objective function to be minimized. The barrier function is constructed such that the weight becomes very high if the constraints are violated. Once the total objective function is formulated, Newton's method is used to calculate the optimal solution. The primal-dual IP method is an extension of the barrier method where the inner and the outer loops are combined and the weight on the barrier function is reduced in each iteration of Newton's method.

- Active Set (AS): This method proceeds by finding a working set of active constraints based on the current state of the system and then solves the resulting QP problem with equality constraints. The working set is then updated repeatedly until the optimal solution is found.

While the AS method requires more iterations than the IP method, each iteration of AS is computationally cheaper than that of IP [78, 85]. Ferreau et al. [78] have also reported that the AS method when coupled with a hot-start strategy, can lead to substantial speed-ups. Because of these advantages, the AS method is chosen as the constraint handling strategy in this research. A commonly used QP solver that employs the active-set method for constraint handling is qpOASES [78]. In their research, Adhau et al. [86] compared various QP solvers based on their real-time performance for NL-MPC application and found that the performance of the qpOASES solver was superior in terms of computational time. Owing to these advantages, qpOASES solver with an active-set strategy is selected for this research.

D.4. REAL-TIME ITERATION

The fundamental idea behind the RTI approach is to divide the whole solution into two phases - namely the preparation phase and the feedback phase. To reduce the computational time, the preparation phase for time i is performed at time $i-1$ and as soon as the measurement at time i is available, the feedback phase is performed.

In the preparation phase, the QP for time = i is pre-formulated at time = $i-1$ and written in terms of the measurement $\hat{x}_i$ before it is available. The QP is also conditioned (condensed and factorized) in this phase. As soon as the measurement $\hat{x}_i$ is available, the feedback phase starts. The measurement is plugged into the conditioned QP and the next control action is determined. It should be noted that in contrast to a regular SQP scheme, the RTI scheme performs a single full Newton step as the hot-start strategy employed allows the RTI problem to converge. A detailed explanation of the RTI scheme can be found in [81] and an overview of the same is shown in the following algorithm.

Algorithm 2: The RTI scheme at time i

```
Preparation phase - performed over time interval $[t_{i-1}, t_i]$
Input: $(x_{i-1}, u_{i-1}), (x_{i-1}^{ref}, u_{i-1}^{ref})$
1) Shift $(x_{i-1}, u_{i-1})$ to get $(x_i^{guess}, u_i^{guess})$
2) Evaluate $r_k, h_k$ and $A_k, B_k, C_k, D_k, H_k, J_k$ using equation D.5
3) Form $\text{QP}_{\text{NL-MPC}}$ (equation D.4) with condensing.
return $\text{QP}_{\text{NL-MPC}}$
Feedback phase - performed at time $t_i$ upon availability of $\hat{x}_i$
Input: $\hat{x}_i$
4) Solve the $\text{QP}_{\text{NL-MPC}}$ (equation D.4) to get newton direction $(\Delta x, \Delta u)$
4) Apply the full Newton step - $(x_i, u_i) \rightarrow (x_i^{\text{guess}}, u_i^{\text{guess}}) - (\Delta x_i, \Delta u_i)$
Return NL-MPC solution : $(x_i, u_i)$
```

D.5. CONCLUSION

While NL-MPC algorithm is a good potential candidate for motion cueing applications, the high computational time required by this approach is a drawback. To reduce the computational time, the RTI scheme can be used. The scheme uses the SQP optimization technique to solve the NL-MPC problem. Further, it relies on the fast convergence of Newton-type optimization and hot-start strategy. The algorithm performs a single full Newton step at every time instance. To reduce the computational effort, the total solution is divided into two phases - namely, the preparation phase and the feedback phase. This allows to minimize the delay between obtaining the measurement and applying the control action. The ACADO toolkit provides an easy to implement package of this scheme and can be used effectively to implement non-linear MPC controllers in real-time. The same has been used in this research to solve the non-linear OCP.

E
Extensive Results

In Chapter 1, a non-linear MPC-based MCA was designed which included the non-linear inverse kinematics of the motion platform and human vestibular model. The algorithm was tested with full-track simulations using the E2M emulator and the results were compared based on several performance indicators.

In this Appendix, the results in supplement to those provided in Chapter 1 are presented. The Appendix is divided into two parts, namely - Case Studies and MCA Comparison. Section E.1 presents case studies related to the effects of sampling time, prediction horizon, adaptive weights and prediction strategies. Further, section E.2 presents the performance comparison of the proposed MCA with L-MPC and CWF algorithms in step response and slalom tests. Further, it should be noted that the test setup and the tuning of the MPC-based algorithms in the following tests was same as described in Chapter 1. Meanwhile, the CWF algorithm was re-tuned for every maneuver to achieve the best possible results.

E.1. CASE STUDIES

In this section, the case studies performed to analyze the effects of sampling time, prediction horizon, etc. are presented.

To evaluate the performance of the proposed MCA, the slalom test was considered for all the case studies presented in this section. Slalom is an important test from the vehicle dynamics point of view. From the motion cueing perspective, it gives important insights such as roll-sway coupling in tracking a sinusoidal reference. In the test considered here, a four-seater hatchback car with an electric motor and a continuously variable transmission (CVT) was simulated in the IPG Carmaker software. A virtual sensor was placed on the eyepoint of the driver to record the perceived acceleration and perceived angular velocity. The resulting signal was recorded, passed through the vestibular system and further sent as the reference to the controller. It should be noted that adaptive weights-based tuning was applied in all the cases unless explicitly stated otherwise.

E.1.1. EFFECT OF SAMPLING TIME

In MPC-based control systems, a continuous-time plant is usually controlled by a discrete-time controller which is called at a certain frequency. In sub-section C.1.1, the importance of choosing a correct sampling frequency (or sampling time) was highlighted. The proposed controller caters to two systems - the human vestibular system and the motion platform. In sub-section A.3.4 it was stated that as per the rule of thumb, the minimum sampling frequency required to include all the dynamics of the vestibular system model was 625 Hz. In sub-section A.2.1, it was further highlighted that the controllers of motion platform are generally called at a sampling frequency of 100-1000 Hz. Taking the computational complexity of the problem into consideration, a very high sampling frequency is not feasible if the controller has to be implemented in real-time. It should further be noted that in the literature, a sampling frequency of 10-100 Hz is used (in [19, 55, 67, 68, 71, 72]), with an exception of [1], where 200 Hz frequency was used. However, it must be noted that the problem considered in this research was linear in nature.

To test the effect of the controller frequency (or sampling time) on the computational effort, the execution Time taken by ACADO Solver for solving the optimal control problem with a prediction horizon of 50 was plotted. The same is shown in figure E.1.

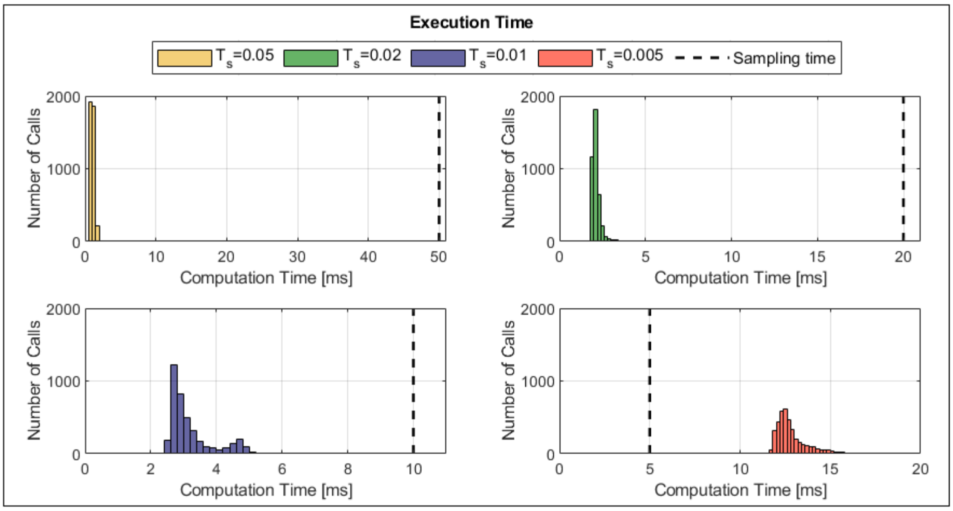

Figure E.1: Effect of sampling time: Execution Time taken by ACADO Solver

It can be seen that the computational time required to solve the OCP at 20 Hz, 50 Hz and 100 Hz is lower than the respective sampling time. However, at 200 Hz, the computational time exceeds the sampling time. To analyze the effect of sampling time on the reference tracking performance, the algorithm was tested for the same frequency range. The results are shown in figure E.2 and E.3.

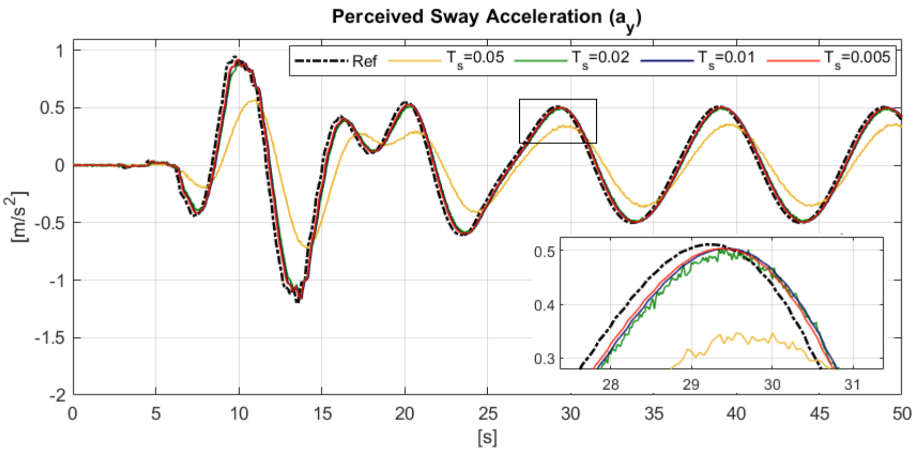

Figure E.2: Effect of sampling time: Acceleration tracking performance

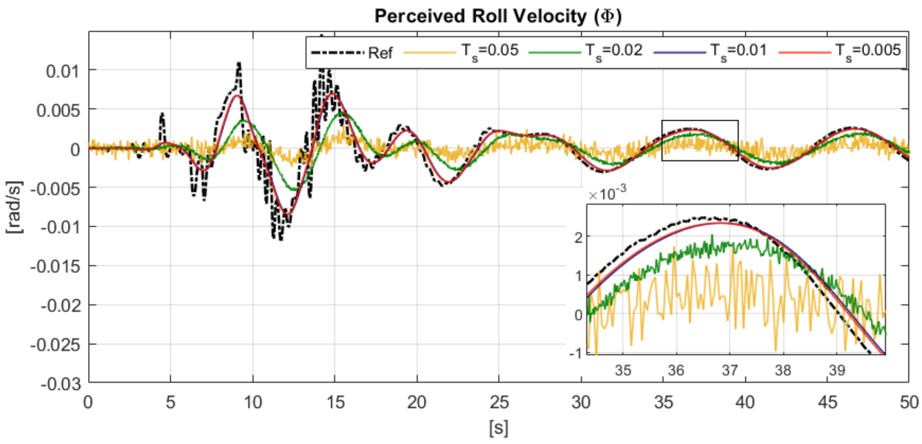

Figure E.3: Effect of sampling time: Roll velocity tracking performance

It can be seen that the performance of the controller at 200 and 100 Hz is superior to the other cases in terms of reference tracking and oscillations. While there is major degradation in the performance of the controller at 20 Hz, the performance at 50 Hz is comparable to that at higher frequencies. However on close inspection, it can be seen that the system output is not smooth as certain oscillations are observed. Therefore, taking the computational cost and the reference tracking performance into consideration, a controller frequency of 100 Hz was chosen for this research.

E.1.2. Effect of Prediction Horizon

While a longer prediction horizon generally leads to better results [68], it increases the computational burden of the MPC problem. Further, the length of the prediction horizon also determines the look-ahead time, i.e. the time for which the future reference should be known. Often, knowing the future reference for a long period of time is impractical. Therefore, the prediction horizon must be chosen while taking all these factors into consideration.

To analyze the effect of the prediction horizon on the computational effort, the execution time required by the ACADO solver to solve the OCP at 100 Hz was plotted. The same is shown in figure E.4. It can be inferred that for $N_p = 20$ and 50 the execution time is well below the sampling time. Further, for $N_p = 200$, the execution time exceeds the sampling time making it infeasible to implement in real-time. Lastly, it should also be noted that for $N_p = 100$, the time required on majority of the instances is below the sampling time but the same exceeds the sampling time on a few instances.

To evaluate the effect of the length of the prediction horizon on the controller performance, the following three cases were considered - look ahead time of 0.2 sec ($N_p = 20$), 0.5 sec ($N_p = 50$), 1 sec ($N_p = 100$) and 2 sec ($N_p = 200$). The effect of the prediction horizon on the performance is shown in figure E.5 and E.6.

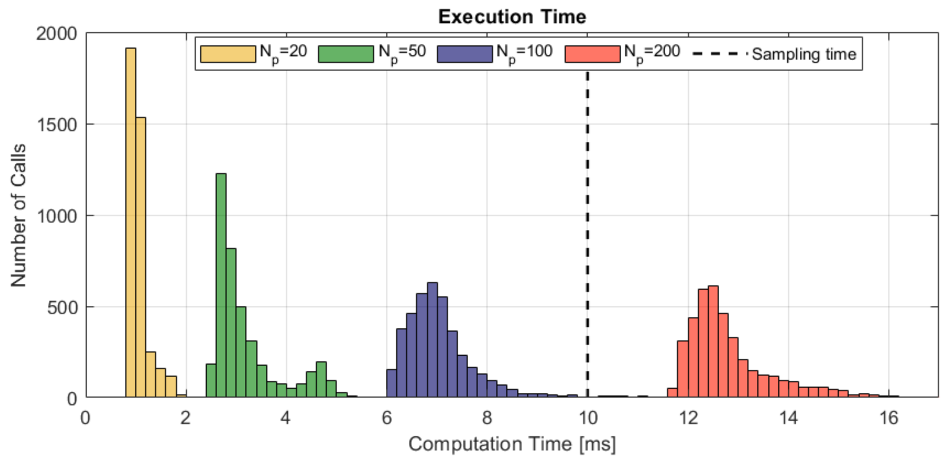

Figure E.4: Effect of prediction horizon: Execution Time taken by ACADO Solver

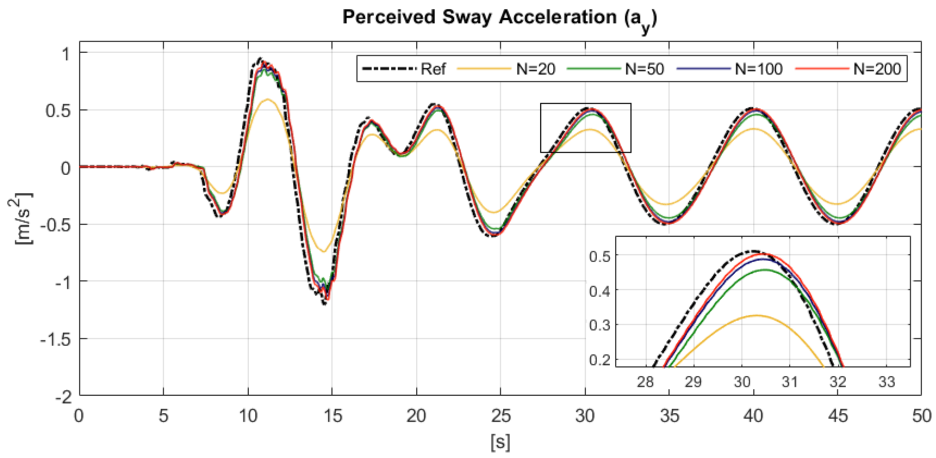

Figure E.5: Effect of prediction horizon: Acceleration tracking performance

In figure E.5, it can be seen that the reference tracking performance of the controller for $N_p = 50$, 100 and 200 is similar. While at $N_p = 20$, there is major degradation in the reference tracking performance. Further, the perceived roll velocity tracking performance (figure E.6) in all the cases was found to be similar. On close inspection, it can be seen that the performance slightly improves for a longer horizon. Taking the tracking performance and the execution time into consideration, the length of the prediction horizon was chosen as 50.

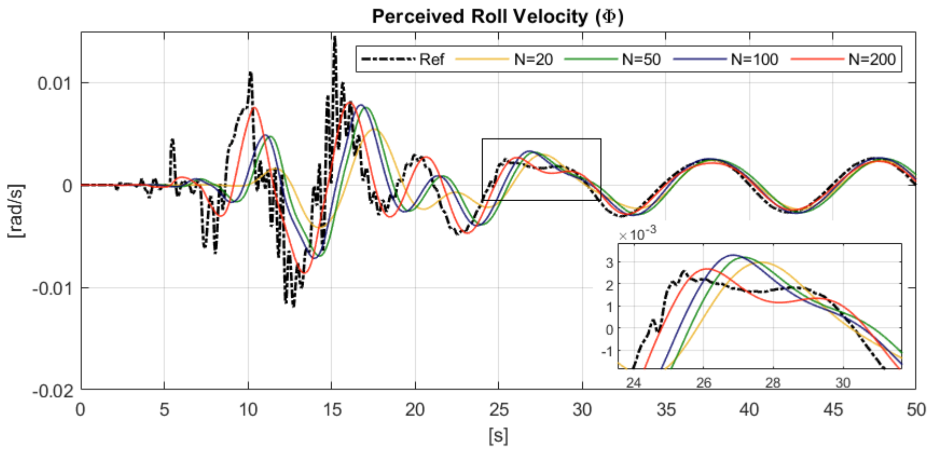

Figure E.6: Effect of prediction horizon: Roll Velocity tracking performance

E.1.3. EFFECT OF PREDICTION STRATEGY

In order to solve the optimization problem, MPC-based MCA requires the knowledge of future reference up to the prediction horizon. In most cases, either of the following two prediction strategies is used to estimate the future reference. First, the ideal prediction strategy that knows the future reference completely up to the prediction horizon (referred to as *Known* in this research). The second prediction strategy that takes the current linear accelerations and angular velocities at every sampling instance and assumes it to be constant for the entire prediction horizon (referred to as *Constant* in this research). The first strategy is ideal and although it can be used for passive driving simulations, such a predefined reference is not available for the active driving or driver-in-loop (DIL) simulations. Meanwhile, the second strategy does not utilize the MPC controller effectively due to the assumptions made about the future reference. In this section, both the prediction strategies are compared based on the slalom test. The results are shown in figure E.7 and E.8.

It can be inferred that the *Constant* strategy results in minor degradation when compared to the *Known* case. It can also be seen that the assumption made in the *Constant* case introduces a delay in the reference tracking performance. The small magnitude of the difference in the performance can be attributed to the small prediction horizon used. The effect of the prediction strategy and the prediction horizon on the reference tracking performance is shown in figure E.9 and E.10.

It can be seen that in both the cases, if the prediction horizon is increased, the root mean square error (RMSE) decreases. However, the decrease in the *Known* case is more when compared to that in the *constant* case. Therefore, the difference in the outputs would be more pronounced if a longer prediction horizon is used. For the prediction horizon chosen in this research ($N_p = 50$), the difference is relatively small. Throughout this research, the *constant* strategy is used.

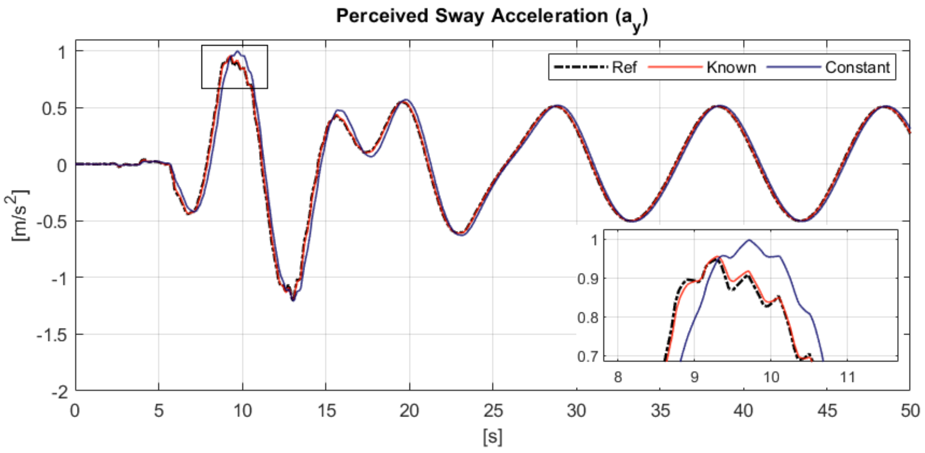

Figure E.7: Effect of prediction strategy: Acceleration tracking performance

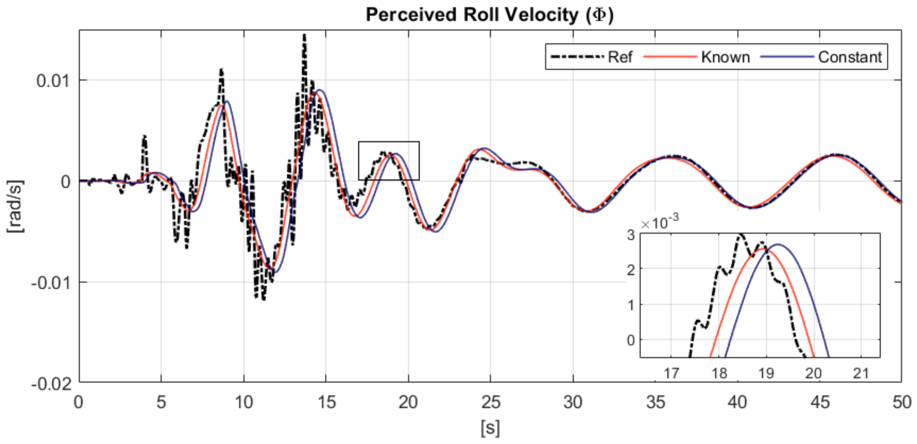

Figure E.8: Effect of prediction strategy: Roll Velocity tracking performance

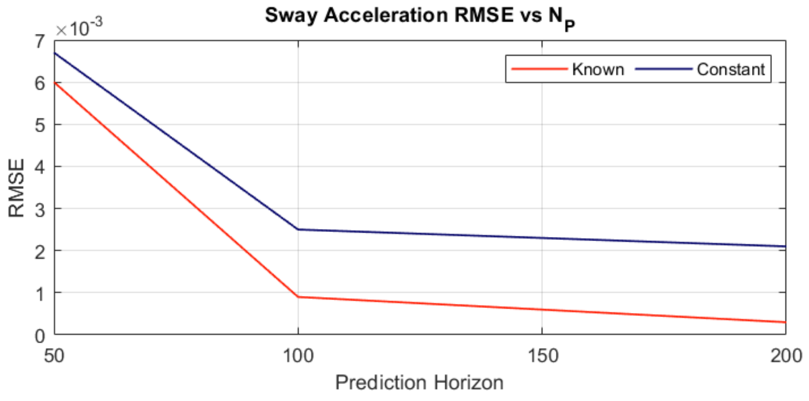

Figure E.9: Effect of prediction strategy: RMSE for acceleration tracking

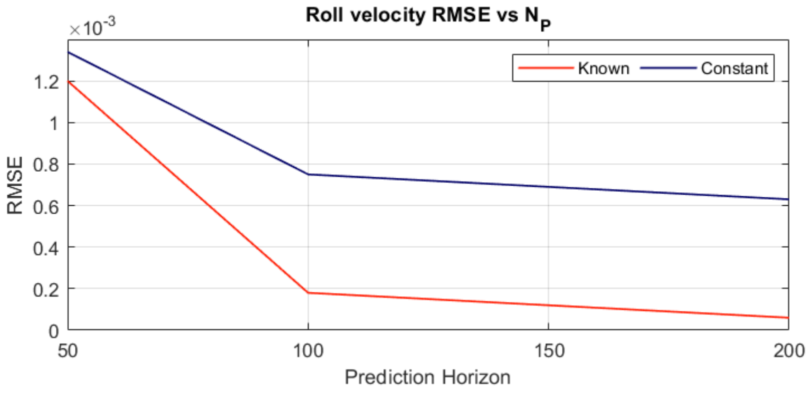

Figure E.10: Effect of prediction strategy: RMSE for roll velocity tracking

E.1.4. EFFECT OF ADAPTIVE WEIGHTS

As mentioned in Chapter 1, state-dependent adaptive weights-based tuning scheme has been employed in this research. This is done in order to provide a damping action and to smoothen the movement of the platform near the physical limit. In this scheme, the weight on the position and velocity of the moving base centroid is increased as the actuators reach the limit.

To analyze the effect of adaptive weights, the slalom test is considered again but the amplitude is increased by a factor of 2. This is done so that the platform reaches its maximum limit. In this case, actuator 2 reaches the physical limit at ~13 *sec*. The perceived surge acceleration and actuator length is shown in figure E.11 and E.12.

On close inspection, it can be seen that the controller with adaptive weights results in a smoother output when the platform is near its physical limits. In figure E.12 it can be seen that the actuator movement near the limits is also smooth. Further, it can also be inferred that when the platform is not near the limits, the adaptive weights allow more free movement of the actuators when compared to the constant weights.

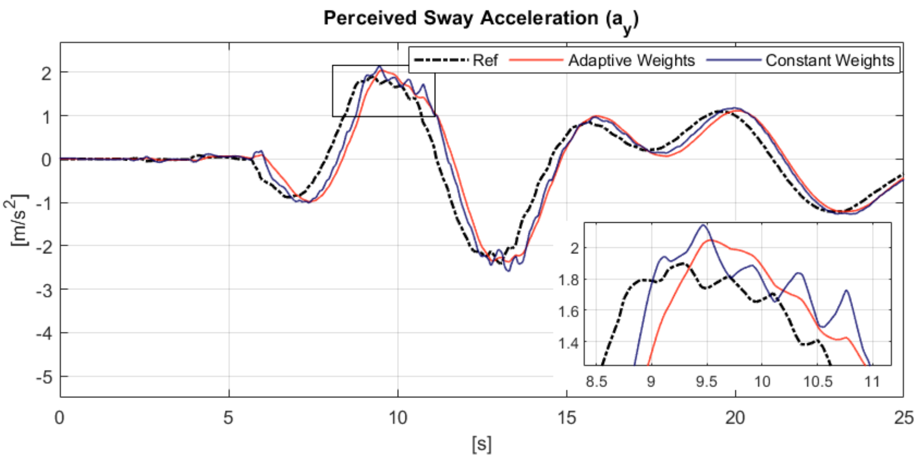

Figure E.11: Effect of adaptive weights: Acceleration tracking performance

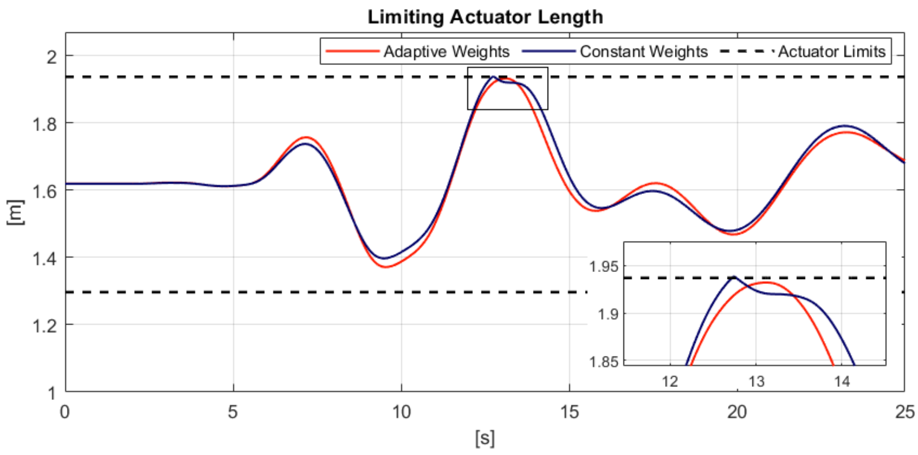

Figure E.12: Effect of adaptive weights: Limiting actuator displacement

E.2. MCA Comparison

In this section, a performance comparison of different motion cueing algorithms is presented. The comparison is based on the reference tracking and workspace utilization performance of different algorithms in the step input tracking test and slalom test. While the step input tracking test helps to analyze the tilt coordination performance of the algorithm, the slalom test gives crucial insight into the coupled DOF. In this section, the following algorithms are considered for the comparison:

- Classical Washout Filter (CWF)
- Linear MPC-based MCA (L-MPC)
- Non-linear MPC-based MCA (NL-MPC)

While actuator-based constraints were imposed in the NL-MPC algorithm, forward kinematics was used to determine a constant workspace for the L-MPC algorithm. Further, the CWF algorithm was tuned to maximize the tracking performance within the allowed workspace. Moreover, for both the MPC-based MCAs, the constant prediction strategy is used for reference generation.

E.2.1. Step Response

The step input tracking test gives useful insights into the sustained acceleration tracking performance (the tilt coordination effects) of the MCA. Further, it can also help to analyze the ability of an MCA to generate sudden high acceleration cues. In this test, a step acceleration input is provided as the acceleration reference for a fixed time, while a zero reference is provided for the angular velocity. The MCAs are then compared based on their reference tracking and workspace utilization performance. As this test utilizes only two DOFs, the pitch-surge mode was considered for all the MCAs in this study. The algorithms are expected to use the translation motion to track the initial rising acceleration, while it is expected to use the tilt-coordination technique to sustain this acceleration.

Translational Acceleration Tracking

Figure E.13 shows the perceived surge acceleration tracking performance. All the algorithms use the translational DOF of the platform to produce the initial rising acceleration and then use the tilt coordination to produce the sustained acceleration. It should be noted that the algorithms produce significant error between time 4-7 seconds because of the constraints on the tilt rate which prohibits the algorithm to increase the tilt angle to the required value with a perceivable tilt rate. The tilt rate of the hexapod is shown in figure E.14. It can be seen that the tilt rate used by all the algorithms is lower than the threshold value. While the constraints are embedded in the MPC-based algorithms, the tilt rate is externally saturated in CWF. Further, it can be seen that the NL-MPC controller results in the lowest RMSE and the highest CC values when compared to the other two controllers. In the case of the L-MPC algorithm, an overshoot in the response is observed. This is because the algorithm uses an approximated linear model of the motion platform which results in difference in the outputs of the system model used by the controller and that of the plant. Lastly, a large error is observed in the response of the CWF

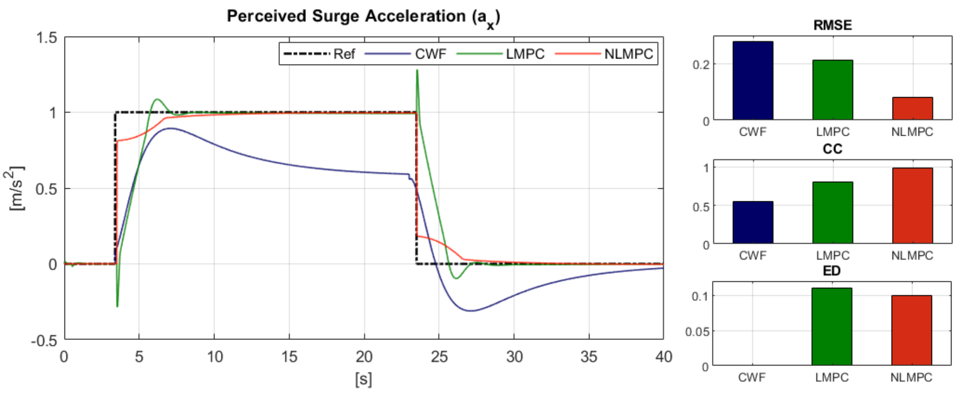

Figure E.13: Step Response Test: Surge Acceleration tracking performance

algorithm. The algorithm fails to reduce this error as it is merely a feed-forward controller.

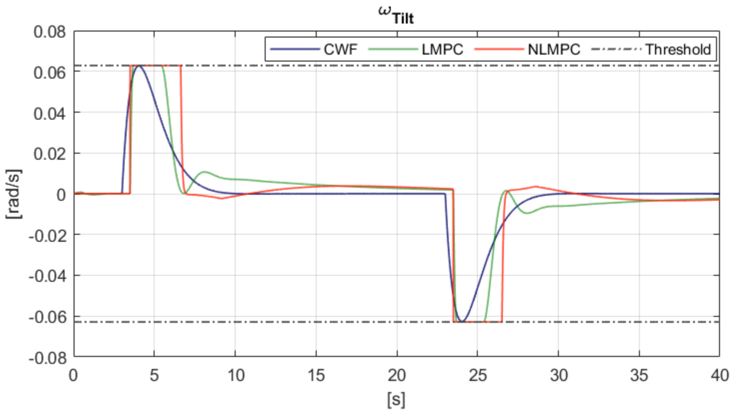

Figure E.14: Step Response Test: Hexapod Tilt rate

Rotational Velocity Tracking

Figure E.15 shows the angular velocity tracking performance. It should be noted that, for a large surge acceleration, there should be some pitch velocity associated with it. However, a zero reference was given in this case to test the algorithm's ability to produce translational acceleration without producing perceivable rotational velocity. As mentioned in Chapter 1, the pitch rate threshold is ~ 0.063 rad/s. It can be seen that all the algorithms produce pitch velocity lower than the threshold. Further, the NL-MPC algorithm produces the lowest RMSE and highest CC value.

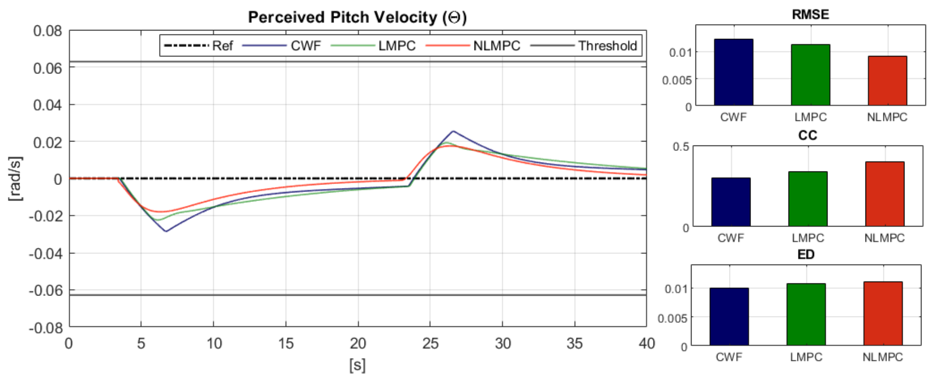

Figure E.15: Step Response Test: Roll Velocity tracking performance

WORKSPACE UTILIZATION

Figure E.16 shows the workspace utilization in terms of the pitch motion and the surge motion of the platform. In step input tracking, ideally the platform should increase the surge motion to produce the initial acceleration and then increase tilt inclination to produce the sustained acceleration. Once it reaches the desired tilt inclination, the platform should washout the surge motion. In Figure E.16, the behavior of the NLMPC algorithm is similar to the ideal behavior. Meanwhile, the L-MPC algorithm increases the inclination of the platform beyond the desired inclination and then reduces it. Lastly, it can be seen that the CWF algorithm uses a conservatively chosen workspace.

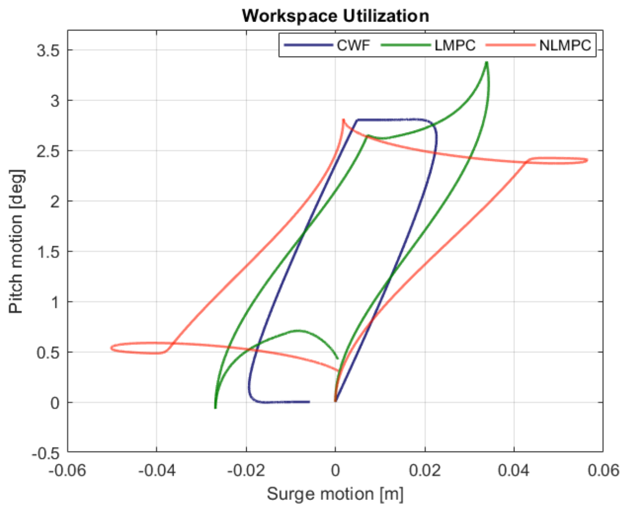

Figure E.16: Step Response Test: Workspace Utilization

E.2.2. SLALOM TEST

As mentioned before, the slalom test provides useful insights into the roll-sway coupling. The test considered in this section is similar to that in section E.1. The MCAs are then compared based on their reference tracking and workspace utilization performance.

TRANSLATIONAL ACCELERATION TRACKING

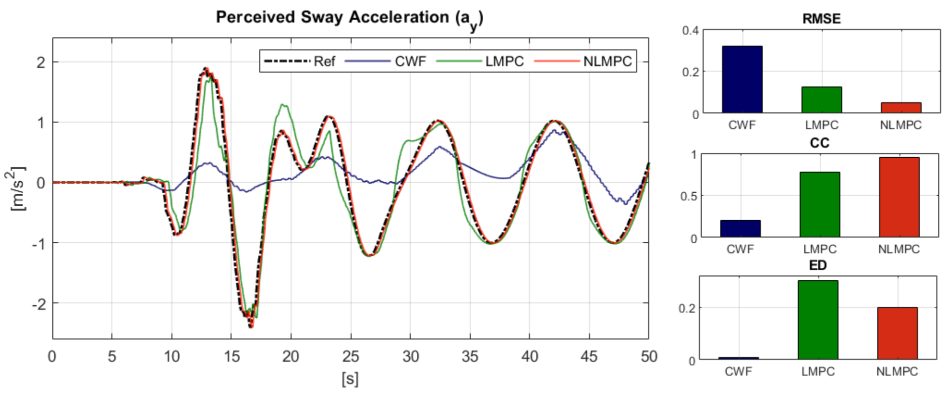

Figure E.17: Slalom Test: Surge Acceleration tracking performance

Figure E.17 shows the perceived surge acceleration tracking performance.

It can be inferred that the NL-MPC algorithm performs considerably better than CWF and L-MPC algorithms. The same is also reflected in the very low RMSE and very high CC values. Further, the superiority of the L-MPC algorithm when compared to the CWF algorithm in terms of reference tracking is also established from this figure. However, the CWF algorithm results in the lowest ED value while both the MPC-based algorithms produce a delay in response. This behavior in MPC-based algorithms can be improved by using the *Known* prediction strategy instead of *Constant*.

ROTATIONAL VELOCITY TRACKING

The rotational velocity tracking performance is shown in figure E.18. It should be noted that in this case, the reference tracking of the angular velocity is important as the same is above the threshold.

Similar conclusions can be drawn from figure E.18. The NL-MPC algorithm shows superior performance than the L-MPC and CWF algorithm as it produces low RMSE and high CC values. A delay in response of both the MPC-based algorithms is observed in the perceived angular velocity tracking performance as well which is reflected in the ED values. It should also be noted that unlike the other algorithms, the NL-MPC algorithm does not produce any false or missing cue. Further, in the case of L-MPC and CWF, this performance can be improved by adjusting the tuning, however, this will come at the cost of the lateral acceleration tracking.

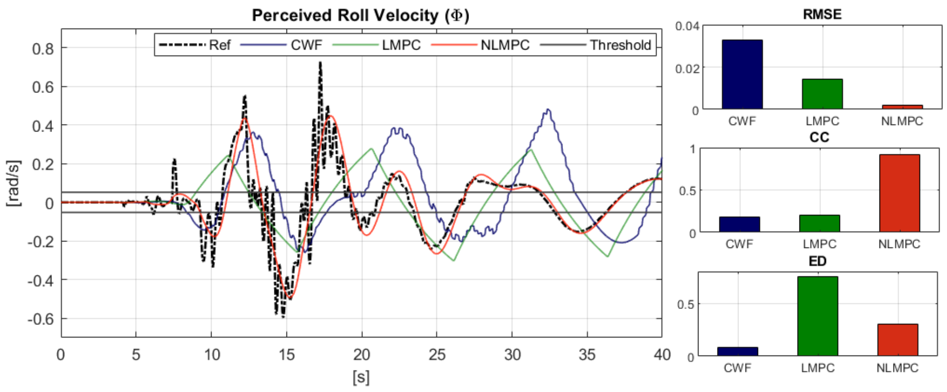

Figure E.18: Slalom Test: Roll Velocity tracking performance

WORKSPACE UTILIZATION

In figure E.19, it can be seen that the MPC-based algorithms show a good coupling performance as they use motion-space in both the directions (roll and sway) effectively. Meanwhile, the CWF algorithm shows poor coupling performance as it uses most of the available workspace in the roll direction while using limited space in the sway direction. Further, it can be seen that the actuator based constraints allow the NL-MPC algorithm to spans most of the available workspace while the L-MPC algorithm spans a smaller workspace owing to the constraints based on a conservatively chosen workspace.

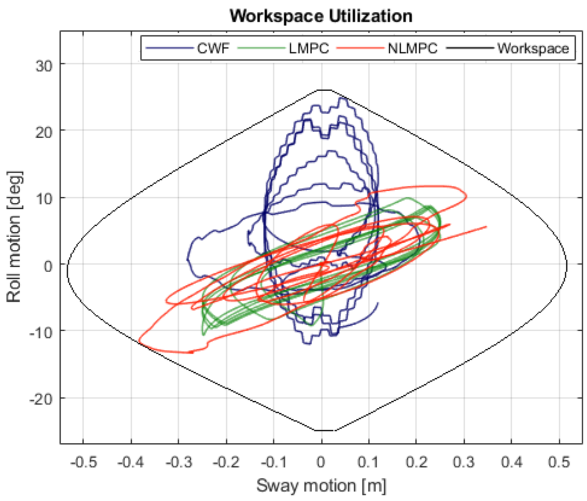

Figure E.19: Slalom Test: Workspace Utilization

E.3. CONCLUSION

In section E.1, the effects of the sampling time, prediction horizon and prediction strategy were presented. Further, the advantages of using an adaptive tuning scheme were also presented in this section.

In section E.2, the NL-MPC algorithm was compared with the L-MPC and CWF algorithms. From the results, it can be concluded that the performance of MPC-based methods is superior to that of the CWF algorithm in terms of both, the reference tracking performance and the workspace utilization. It can also be inferred that the L-MPC algorithm uses a conservatively chosen workspace and the system model used inside the controller ignores the non-linearities of the plant which produces in sub-optimal results. In this section it was validated that the NL-MPC algorithm offers an elegant solution to the motion cueing problem and performs better than the L-MPC and CWF-based MCAs.

References

[1] M. Bruschetta, F. Maran, and A. Beghi. A nonlinear, MPC-based motion cueing algorithm for a high-performance, nine-DOF dynamic simulator platform. *IEEE Transactions on Control Systems Technology*, 25(2):686–694, 2017.

[2] D. Stewart. A platform with six degrees of freedom. *In proceedings of the Institution of Mechanical Engineers*, 180(1):371–386, 1965.

[3] R. Telban and F. Cardullo. Motion cueing algorithm development: Human-centered linear and nonlinear approaches. *Technical report, NASA Langley Research Center*, 2005.

[4] H. Merritt. Hydraulic control systems. *Wiley, New York*, 1967.

[5] E2M Technologies B.V. E2M 6-DOF motion platform. *eMoveRT controller manual.*

[6] O. Ulucay. Design and control of stewart platform. *MSc. thesis, Graduate School of Engineering and Natural Sciences, Sabancı University, Turkey*, 2006.

[7] M. Husty. An algorithm for solvmg the direct kinematics of general stewart-gough platform. *Mech. Mach. Theory 31*, 4:365–380, 1996.

[8] C. Wampler. Forward displacement analysis of general Six-in-Parallel SPS (Stewart) platform manipulators using Soma coordinates. *Mech. Mach. Theory 31*, 3:331–337, 1996.

[9] C. Wilson and J. Sadler. Kinematics and dynamics of machinery. *Harper Collins College Publishers, New York*, 1993.

[10] T. Gable and B. Walker. Driving simulator sickness screening: Efficient and effective new protocol. *In Proceedings of the Fifth International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 101–102, 2013.

[11] R. Telban. A nonlinear motion cueing algorithm with a human perception model. *Research report, AIAA (American Institute of Aeronautics and Astronautics)*, 2002.

[12] W. Warren. Self motion- visual perception and visual control. *Academic Press, Perception of Space and Motion. San Diego*, pages 263–325, 1995.

[13] A. Väljamäe, P. Larsson, D. Västfjäll, and M. Kleiner. Travelling without moving: Auditory scene cues for translational self-motion. *In Proceedings of ICAD 05 - Eleventh Meeting of the International Conference on Auditory Display*, 2005.

[14] G. Tiesler. Motion perception in vehicle simulators. *Research Report, No. 12, astrophysical research society, ResearchIinstitute for Anthropotechnik. Meckenheim*, 1973.

[15] A. Berthoz. The brain's sense of movements. *Harvard University Press*, 2000.

[16] A. Kemeny and F. Panerai. Evaluating perception in driving simulation experiments. *Trends in cognitive sciences*, 7:31–37, 2003.

[17] E. Groen, I. Howard, and B. Cheung. Influence of body roll on visually induced sensations of self-tilt and rotation. *Perception*, 28:287–297, 1999.

[18] S. Kockebakker. Model based control of flight simulator motion systems. *PhD. dissertation, Delft University of Technology*, 2001.

[19] B. Augusto and R. Loureiro. Motion cueing in the chalmers driving simulator: a model predictive control approach. *MSc. Thesis, Chalmers university of Technology, Sweden*, 2009.

[20] NASA. Drawing of the inner ear. *The Effects of Space Flight on the Human Vestibular System.*

[21] A. Benson, E. Hutt, and S. Brown. Thresholds for the perception of whole body angular movement about a vertical axis. *Aviation, Space and Environmental Medicine, vol. 60*, pages 205–213, 1989.

[22] Obrist D. Fluid mechanics of the inner ear. *Habilitation treatise in Fluid Mechanics at the Department of Mechanical and Process Engineering of ETH Zurich*, pages 40–41, 2011.

[23] C. Fernandez and J. Goldberg. Physiology of peripheral neurons innervating otolith organs of the squirrel monkey. iii. response dynamics. *Journal of neuro-physiology, 39(5)*, pages 996–1008, 1976.

[24] R. Mayne. A systems concept of the vestibular organs. *Vestibular System Part 2: Psychophysics, Applied Aspects and General Interpretations, Springer*, page 493–580, 1974.

[25] A. Van Egmond, J. Groen, and L. Jongkees. The mechanics of the semicircular canal. *Journal of physiology, 110(1-2)*, pages 1–17, 1949.

[26] L. Young and C. Oman. Model for vestibular adaptation to horizontal rotation. *Journal of Aerospace medicine, 40(10)*, pages 1076–1077, 1969.

[27] W. Grant and W. Best. Otolith-organ mechanics: lumped parameter model and dynamic response. *Aviation, Space and Environmental Medicine, vol. 58(10)*, pages 970–976, 1987.

[28] C. Ormsby. Model of human dynamic orientation. *PhD thesis, Massachusetts Institute of Technology*, 1974.

[29] L. Young and J. Meiry. A revised dynamic otolith model. *Journal of Aerospace medicine, 39(6)*, pages 606–608, 1968.

[30] L. Reid and M. Nahon. Flight simulation motion-base drive algorithms: Part 1 - developing and testing the equations. *UTIAS Report No. 296, CN ISSN 0082-5255, Institute for Aerospace Studies, University of Toronto, Toronto, Canada*, page 3.8, 1985.

[31] Y. Jiang, X. Hu, and S. Wu. Transformation matrix for time discretization based on Tustin's method. *Mathematical Problems in Engineering*, pages 1–9, 2014.

[32] M. Nahon and L. Reid. Simulator motion-drive algorithms: A designer's perspective. *Journal of Guidance, Control, and Dynamics*, 13(2):356–362, 1990.

[33] P. Grant, M. Blommer, B. Artz, and J. Greenberg. Analysing classes of motion drive algorithms based on paired comparison techniques. *Vehicle System Dynamics*, 47(9):1075–1093, 2009.

[34] M. Fischer, H. Sehammar, and G. Palmkvist. Motion cueing for 3-, 6- and 8- degrees-of-freedom motion systems. *Actes INRETS*, page 121–134, 2010.

[35] T. Chapron and J. Colinot. The new PSA peugeot-citroen advanced driving simulator overall design and motion cue algorithm. *In Proceedings of Driving Simulation Conference*, 2007.

[36] B. J. Correia Grácio, M. M. van Paassen, M. Mulder, and M. Wentink. Tuning of the lateral specific force gain based on human motion perception in the desdemona simulator. *In proceedings of AIAA Modeling and Simulation Technologies Conference, Toronto, Canada*, 2010.

[37] E. Boer, N. Kuge, T. Yamamura, and N. Yokosuka. Affording realistic stopping behavior: A cardinal challenge for driving simulators. *In proceedings of 1st Human Centered Transportation Conference, Iowa*, 2001.

[38] T. Fortmüller and M. Meywerk. The influence of yaw movements on the rating of the subjective impression of driving. *In proceedings of DSC North America, Orlando, Florida*, 2005.

[39] N. Kuge, M. Kubota, and K. Itoh. A study on a motion algorithm based on a driver-centered approach. *In proceedings of DSC Europe, Paris*, 2002.

[40] P. Pretto, H. Nusseck, H. Teufel, and H. Bülthoff. Effect of lateral motion on drivers' performance in the MPI motion simulator. *In proceedings of DSC Europe, Monaco*, 2009.

[41] P. Grant. The development of tuning paradigm for flight simulator motion drive algorithms. *Dissertation (PhD), Department of Aerospace Science and Engineering, University of Toronto, Toronto*, 1995.

[42] W. Käding and F. Hoffmeyer. The advanced daimler-benz driving simulator. *SAE Technical Paper Series No. 950175*, 1995.

[43] T. Sammet. Motion-cueing algorithm for the driving simulator. *Dissertation (PhD), TU München, München*, 2007.

[44] P. Grant and L Reid. Motion washout filter tuning: Rules and requirements. *Journal of Aircraft, March – April*, 13(2), 1997.

[45] N. Murgovski. Vehicle modelling and washout filter tuning for the chalmers vehicle simulator. *MSc. Thesis, Department of Signals and Systems, Chalmers University of Technology, Sweden*, 2007.

[46] R. Sivan, J. Ish-Shalom, and J. Huang. An optimal control approach to the design of moving flight simulators. *IEEE Trans. On Systems, Man, and Cybernetics*, 12(6), 1982.

[47] S. Chen and L. Fu. An optimal washout filter design for a motion platform with senseless and angular scaling maneuvers. *In proceedings of the 2010 American Control Conference*, 2010.

[48] J. Bellon. Riccati equations in optimal control theory. *MSc. Thesis, Department of Mathematics and Statistics, Georgia State University*, 2008.

[49] F. Cardullo and R. Kosut. Old problem/new solutions—motion cuing algorithms revisited. *In Flight Simulation Technologies Conference*, page 1082, 1983.

[50] J. Ish-Shalom. Design of optimal motion for flight simulators. *PhD Thesis, Massachusetts Institute of Technology*, 1982.

[51] R. Parrish, J. Dieudonne, and D. Martin Jr. Coordinated adaptive washout for motion simulators. *Journal of Aircraft*, 12(1):44–50, 1975.

[52] R. Telban, W. Wu, and F. Cardullo. Motion cueing algorithm development: Initial investigation and redesign of the algorithms. *NASA /CR-2000-209863*, 2000.

[53] A. Naseri and P. Grant. An improved adaptive motion drive algorithm. *In AIAA Modeling and Simulation Technologies Conference and Exhibit*, page 6500, 2005.

[54] L. Nehaoua, H. Arioui, S. Espie, and H. Mohellebi. Motion cueing algorithms for small driving simulator. *In proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida*, 2006.

[55] N. Garrett and M. Best. Model predictive driving simulator motion cueing algorithm with actuator-based constraints. *Vehicle System Dynamics, Vol. 51, No. 8*, pages 1151–1172, 2013.

[56] D. Cleij, J. Venrooij, P. Pretto, M. Katliar, H. Bulthoff, D. Steffen, F.W Hoffmeyer, and H.P Schöner. Comparison between filter- and optimization-based motion cueing algorithms for driving simulation. *Transportation Research Part F: Traffic Psychology and Behaviour*, 2016.

[57] T. Miunske, J. Pradipta, and O. Sawodny. "model predictive motion cueing algorithm for an overdetermined stewart platform. *ASME. J. Dyn. Sys., Meas., Control.*, 2019.

[58] A. Lamprecht, D. Steffen, J. Haecker, and K. Graichen. Comparision between a filter and an mpc-based mca in an offline simulator study. *In proceedings of Driving Simulation Conference (DSC), Europe*, pages 101–107, 2019.

[59] M. Behrendt. A discrete mpc scheme. *en.wikipedia.org*, 2019.

[60] L. Magni, G. Nicolao, L. Magnani, and R. Scattolini. A stabilizing model-based predictive control algorithm for nonlinear systems. *Automatica, 37*, page 1351–1362, 2001.

[61] H. Chen and F. Allgower. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica, 34*, page 1205–1217, 1998.

[62] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, pages 789–814, 2000.

[63] M. Abdelaal, M. Franzle, and A. Hahn. Nonlinear model predictive control for tracking of underactuated vessels under input constraints. *In proceedings of 2015 IEEE European Modelling Symposium, Madrid*, pages 313–318, 2015.

[64] L. Grune and J. Pannek. Stability and suboptimality without stabilizing terminal conditions. *Nonlinear Model Predictive Control: Theory and Algorithms, Springer, Switzerland.*

[65] J. Rawlings, D. Mayne, and M. Diehl. Model predictive control: Theory, computation, and design. *Nob Hill Publishing, Cheryl M. Rawlings, publisher*, 2018.

[66] A. Bemporad, F. Borrelli, and M. Morari. Piece-wise linear optimal controllers for hybrid systems. *In Proceedings of the American Control Conference, volume 2*, pages 1190–1194, 2000.

[67] Z. Fang and A. Kemeny. Motion cueing algorithms for a real-time automobile driving simulator. *In proceedings of Driving Simulation Conference, Paris*, 2012.

[68] M. Bruschetta, F. Maran, and A. Beghi. A real-time implementation of an mpc based motion cueing strategy with time-varying prediction. *In IEEE international conference on systems,man and cybernetics (SMC)*, pages 4149–4154, 2013.

[69] M. Baseggio, M. Bruschetta, F. Maran, and A. Beghi. An mpc approach to the design of motion cueing algorithms for driving simulators. *In 14th IEEE international conference on Intelligent Transportation Systems (ITSC)*, pages 692–697, 2011.

[70] Y. Wang and S. Boyd. Fast model predictive control using online optimization. *IEEE Trans Control System Technology, 18(2)*, pages 267–278, 2010.

[71] M. Grottoli, D. Cleij, P. Pretto, Y. Lemmens, R. Happee, and H. Bulthoff. Objective evaluation of prediction strategies for optimization-based motion cueing. *Simulation, 95(8)*, pages 707–724, 2018.

[72] M. Dagdelen, G. Reymond, and A. Kemeny. Model-based predictive motion cueing strategy for vehicle driving simulators. *Control Engineering Practice, Volume 17, Issue 9*, pages 995–1003, 2009.

[73] Z. Fang and A. Kemeny. An efficient model predictive control-based motion cueing algorithm for the driving simulator. *Simulation, 92(11)*, pages 1025–1033, 2016.

[74] Z. Fang, D. Wautier, and A. Kemeny. Development and application of a fast mpc based motion cueing algorithm. *In proceedings of Driving Simulation Conference (DSC), Europe*, pages 109–116, 2019.

[75] A. Mohammadi. Enhancing human motion perception in model predictive motion cueing algorithm. *PhD. Thesis, Deakin University, Australia*, 2018.

[76] J. Venrooij, D. Cleij, M. Katliar, P. Pretto, H. Bülthoff, D. Steffen, F. Hoffmeyer, and H. Schoener. Comparison between filter- and optimization-based motion cueing in the daimler driving simulator. *In proceedings of Driving Simulation Conference and Exhbition, Paris, France*, 2016.

[77] J. Frasch, S. Sager, and M. Diehl. Parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computation, 7(3)*, page 289–329, 2015.

[78] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl. qpoases: A parametric active set algorithm for quadratic programming. *Mathematical Programming Computation, 6(4)*, page 327–363, 2014.

[79] A. Domahidi, A. Zgraggen, M. Zeilinger, M. Morari, and C. Jones. Efficient interior point methods for multistage problems arising in receding horizon control. *In proceedings of IEEE Conference on Decision and Control (CDC), Maui, HI, USA*, page 668–674, 2012.

[80] M. Diehl, H. Bock, and J. Schloder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization, 43(5)*, page 1714–1736, 2005.

[81] S. Gros, M. Zanon, Q. Rien, A. Bemporad, and M. Diehl. From linear to nonlinear mpc: bridging the gap via the real-time iteration. *International Journal of Control*, pages 1–19, 2016.

[82] B. Houska, H. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, pages 298–312, 2011.

[83] M. Kiehl. Parallel multiple shooting for the solution of initial value problems. *Parallel Computing. 20 (3)*, page 275–295, 1994.

[84] M. Vukov, A. Domahidi, H. Ferreau, M. Morari, and M. Diehl. Auto-generated algorithms for nonlinear model predictive control on long and on short horizons. *In proceedings of* 52^{nd} *IEEE Conference on Decision and Control*, pages 5113–5118, 2013.

[85] R. Bartlett, A. Wachter, and L. Biegler. Active set vs. interior point strategies for model predictive control. *In proceedings of the 2000 American Control Conference*, page 327–363, 2000.

[86] S. Adhau, S. Patil, D. Ingole, and D. Sonawane. Implementation and analysis of non-linear model predictive controller on embedded systems for real-time applications. 18^{th} *European Control Conference, Naples, Italy*, pages 3359–3364, 2019.

GLOSSARY

LIST OF ACRONYMS

AWF	Adaptive Washout Filter.
CoG	Centre of Gravity.
CWF	Classical Washout Filter.
DF	Driver frame of reference.
DOF	Degree of Freedom.
FVT	Final Value Theorem.
IF	Inertial frame of reference.
L-MPC	Linear Model Predictive Control.
MCA	Motion Cueing Algorithm.
MPC	Model Predictive Control.
NL-MPC	Non-linear Model Predictive Control.
OCP	Optimal Control Problem.
OWF	Optimal Washout Filter.
PWA	Piece-wise affine.
RTI	Real-time iteration.
SQP	Sequential Quadratic Programming.
VF	Vehicle frame of reference.