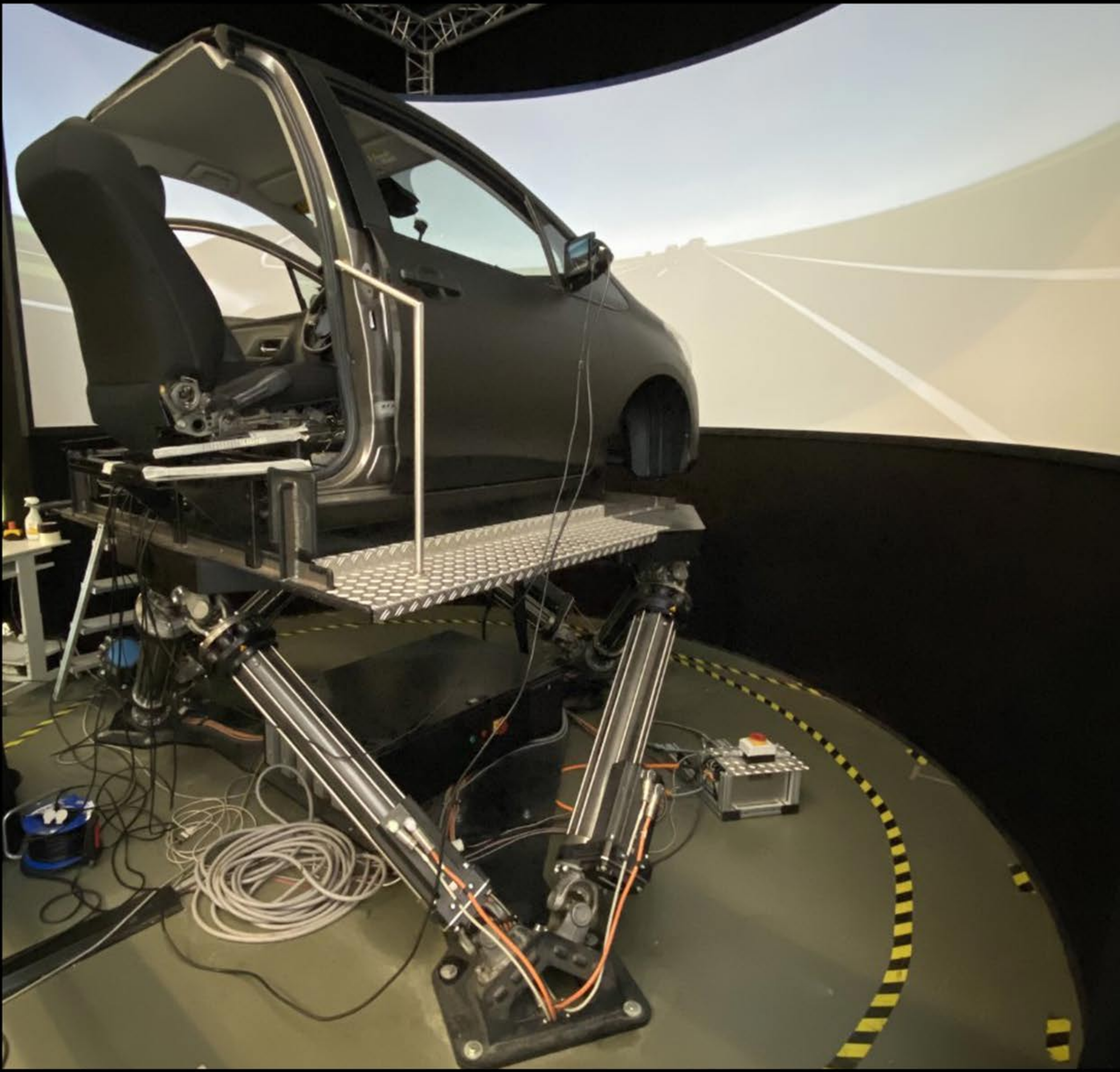
Hybrid MPC based Motion Cueing Algorithm for Driving Simulators

Akhil Chadha



Hybrid MPC based Motion Cueing Algorithm for Driving Simulators

by

Akhil Chadha

in partial fulfillment of the requirements for the degree of
Master of Science in Mechanical Engineering
at the Delft University of Technology

Student number:	5223121	
Graduation committee:	Dr. Barys Shyrokau,	Chair and Supervisor, CoR-IV, TU Delft
	Ir. Vishrut Jain,	Supervisor, CoR-IV, TU Delft
	Dr. Daan Pool,	External Committee Member, AE, TU Delft

Cognitive Robotics

TU Delft Delft University of Technology

Abstract

Driving simulators have been used in the automotive industry for many years now. They have been vastly employed for conducting tests in a safe, reproducible and controlled immersive virtual environment. The ability of the simulator to recreate the in-vehicle experience for the occupant is established through motion cueing algorithms. Such algorithms have consistently been developed with model predictive control (MPC) acting as the main control technique. Currently available MPC based methods either compute the optimal controller online or derive an explicit control law in an offline setting. These approaches limit the applicability of MPC for real-time applications due to online computational expense and offline memory storage issues.

This thesis report presents a solution to deal with issues of offline and online solving through a combined/hybrid approach. For this, explicit MPC is used to provide an initial guess as warm start for the implicit MPC based motion cueing algorithm. From the simulations, it was observed that the presented hybrid approach was able to reduce online computational load by shifting it offline using the explicit MPC. Further, braking constraints and adaptive washout weights were implemented in the hybrid motion cueing algorithm, to improve the specific force tracking performance and reduce any false cue occurrences. Finally, emulator studies were performed to realize driving simulator performance with the hybrid MPC approach. The thesis concludes by showing the improvements made with the developed algorithm and proposes recommendations for future work.

Acknowledgement

The last two years of my masters program here at TU Delft have been full of challenges which have definitely helped me in growing as a person. Now that this program is about to finish, I would like to take the opportunity to thank the people who have helped me along the way.

Firstly, I would like to thank Dr. Barys Shyrokau for allowing me to do my thesis in his research group and work on this exciting topic. Thank you for your knowledge, experience and advice throughout the course of my thesis project. Also, I would like to thank my daily supervisor Vishrut Jain for his constant support and guidance during the thesis.

Secondly, I would like to express my appreciation towards my family who helped me get to where I am today. Thank you for always believing in me and pushing me to excel. It would not have been possible without your support.

Lastly, I would like to thank my friends from TU Delft and from back home in India who I could always count on for moral support and a helping hand throughout this journey.

Akhil Chadha
Delft, August 2022

Contents

List of Figures

List of Tables

Abbreviations

MPC	Model predictive control
DOF	Degree of freedom
PWA	Piecewise affine
mpLP	Multi-parametric linear programming
mpQP	Multi-parametric quadratic programming
DIL	Driver in loop
MCA	Motion cueing algorithm
IP	Interior point
SQP	Sequential quadratic programming
RTI	Real time iteration
OCP	Optimal control problem
LNL	Linear nonlinear
NLL	Nonlinear linear
MPT	Multi parametric toolbox
RMSE	Root mean square error
eMPC	Explicit model predictive control
iMPC	Implicit model predictive control
DAVSi	Delft Advanced Vehicle Simulator

1 Introduction

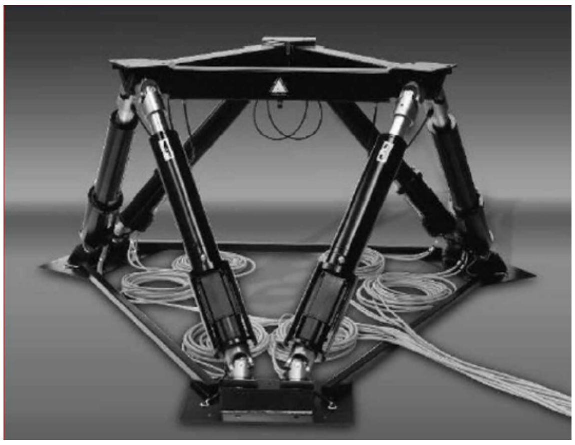
Before implementation of any new advancement in vehicular technology, it is usually tested upon using driving simulators. These simulators are used to create the same experience that one would feel on the road without any damages to the real vehicle. Thus, it is extremely important that the simulator movements take place with the aim of recreating the in-vehicle experience. To achieve this, a motion cueing algorithm is used, which acts as the control technique for the driving simulator's movements. It governs the process allowing the simulator to function properly so that a similar feeling of motion is experienced by the user. It also maximizes workspace utilization during motion cueing [1]. In addition, visual, audio, vestibular and haptic cues are used in the simulator set-up. These cues work together to create a virtual environment mimicking on-road experience.

In motion cueing, a vehicle model is used which takes in the driver input and sends the output to the created motion cueing algorithm. Then, the algorithm decides the desired motion which is performed by the driving simulator. Sensed specific force values $f_{spec,s}$ are obtained from the simulator which are calculated using simulator dynamics. These perceived accelerations comprise of two components; platform translational acceleration $a_{tran,p}$ and the gravitational tilt g_{tilt}. This ensures that all acceleration forces are considered, allowing us to study the human body's movement in space during the cueing process. Sensed values are then compared with the actual specific force $f_{spec,a}$ obtained from the vehicle model. The computed error is then fed back into the motion cueing algorithm block as feedback in order to improve results for the next time step. Some cueing algorithms however do not employ an error calculation step.

1.1. Driving Simulators

As stated previously, a driving simulator is used to recreate the in-vehicle experience. For this reason, the general layout of a driving simulator comprises of a visual projector, an audio system and the driving platform. The visual projector shows the corresponding road environment, audio system simulates real world on-road noises and the driving platform creates the desired motion according to the vehicle's position and orientation.

There are various types of driving simulators and the Gough-Stewart design is one of the most famous driving simulators to be used over the years. It is a six-legged parallel manipulator which has a high stiffness and load capacity, making it a favourable choice for a virtual environment to recreate the in-vehicle experience. An example of what the simulator looks like can be seen in Figure 1.1a. The platform as shown in the figure comprises of three main components; the fixed base attached to the ground, the platform or moving frame on which the vehicle cockpit is situated and the actuator legs [2]. This simulator is able to provide motion in 6 degrees of freedom (DOF) ie. three translational motions (surge x, sway y and heave z) and three rotational motions (roll ϕ, pitch θ and yaw ψ). Apart from the 6 DOF Gough-Stewart simulator, other simulators are also used, such as the 3 DOF motion platform commonly used for racing games or a more advanced driving simulator comprising of 8 DOF motion capabilities with a separate rail and tilting platform. Such a simulator has been developed at Renault called the ULTIMATE driving simulator [3]. An example of what the simulator looks like can be seen in Figure 1.1b. A 9 DOF simulator has also been developed comprising of a platform base moved by a tripod, upon which the hexapod is situated as studied and used in [4].


(a) 6 DOF Gough-Stewart motion platform [1]

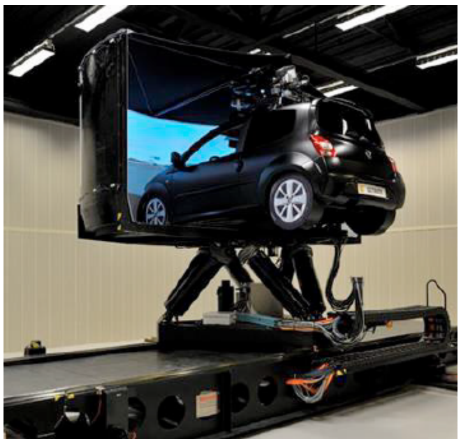

(b) 8 DOF Renault ULTIMATE motion platform [3]

Figure 1.1: Different kinds of driving simulators

The geometrical model of the Gough-Stewart platform can be seen in Figure 1.2. A crucial aspect of this geometrical model are the reference frames of the driving simulator. As shown in Figure 1.2, three separate frames can be defined as follows:

- Inertial frame of reference - This reference frame is fixed to the base of the driving simulator and does not move. It is denoted by ”I.F”.
- Platform frame of reference - This reference frame is fixed at the moving platform of the simulator. It moves with the platform and is denoted by ”P.F”.
- Driver frame of reference - This is the driver reference frame which acts at the driver eye point. It is denoted by ”D.F”.

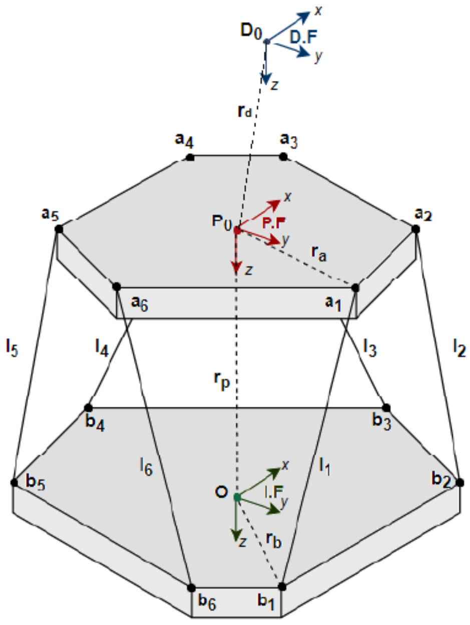


Figure 1.2: Geometrical model of the Gough-Stewart motion platform [5]

1.2. Problem Definition

Motion cueing algorithms are used to replicate the in-vehicle motion inside the simulator while ensuring that workspace boundaries are adhered to. To realize this, several motion cueing algorithms have been developed and implemented in different driving simulators. This includes conventional filter based algorithms, which use the property of high and low pass filters to feed input into the driving simulator. Such algorithms have shown promise but suffer from issues such as minimum workspace utilization and inability to take constraints into account. This leads to using advanced control techniques such as Model Predictive Control (MPC). MPC allows incorporation of explicit constraints, as part of the algorithm to ensure maximum workspace utilisation. By using this control technique, an improvement in the capability of the motion cueing algorithm can be observed. It is mainly of two types; implicit (online) or explicit (offline) with the former approach being most commonly used in this field.

It is important to understand that even though there have been advancements made in this domain, by using MPC, there is still a prominent restraining factor present in the form of computational efficiency. For the implicit controller, the online computation time tends to exceed the sampling rates needed to run the algorithm and maintain realistic simulator motion. Using the explicit controller, as an alternative, seems like a good solution as the control law is computed offline leading to a simple control input look-up table selection in real-time. However, this approach suffers from memory storage issues and has limitations in using large prediction horizons with fast sampling rates for high dimensional problems; which is why it cannot be used on its own. As a consequence there is need to shift the bulk of the online computational load offline to allow efficient online working of the MPC controller in real-time.

1.3. Research Objective

As mentioned earlier, advanced control techniques such as MPC have been extensively used for real-time implementation of motion cueing algorithms. But, the main issue observed is that these algorithms are computationally expensive. For this, a combined/hybrid approach will be developed which uses the initial explicit MPC solution as a guess/warm start for the final implicit MPC based motion cueing algorithm. This type of an algorithm has not yet been implemented in the domain of motion cueing algorithms and thus would act as a novel approach for this field.

Thus, the main goal for the thesis can be outlined as **to develop a hybrid MPC based motion cueing algorithm to mitigate online computation costs for real-time implementation in driving simulators**, with the following research objectives:

- Develop a motion cueing algorithm which combines explicit and implicit MPC based approaches to improve online computation time while maintaining performance for a simple cueing problem.
- Extend the developed motion cueing algorithm for higher degree of freedom problems to evaluate the capabilities of the hybrid approach
- Perform emulator studies, taking 6 DOF hexapod dynamics into account, to analyse the algorithm's simulation results and performance.

The work done by [6] in developing the explicit MPC based cueing algorithm will act as a benchmark for designing the explicit controller in this study. This algorithm will be altered according to our simulator specifications and requirements and be used as the initial guess for the hybrid cueing algorithm.

1.4. Thesis Layout

This thesis report is divided into 7 chapters. Chapter 1 provides a brief introduction into motion cueing algorithms for driving simulators and concludes with the thesis goal. Chapter 2 provides the necessary theory required to understand the working of motion cueing algorithms in detail, with information about conventional filter and MPC based algorithms. The main controller design used in this research is presented in chapter 3. In chapter 4, the developed 2 DOF motion cueing algorithm is explained along with extensions to take 4 DOF capabilities into account. Chapter 5 describes the test cases used in performing simulations with the developed motion cueing algorithms. Simplified actuator dynamics are used in these simulations and the results obtained are discussed. In chapter 6, emulator results with the Delft Advanced Vehicle Simulator (DAVSi) are shown. Finally, in chapter 7, conclusion and recommendations for future work are laid out.

Additional information which would be too extensive for the main report is presented in the Appendices as follows:

- Appendix A contains the scientific paper based on the work done in this research
- Appendix B has information about different human perception models used in literature along with an extensive comparative study done using these models.
- Appendix C contains additional simulation results for the designed algorithms.
- Appendix D has information on the toolboxes used to design and develop the motion cueing algorithms

2 Background

2.1. Human Perception

Perception deals with the human senses that generate signals from the environment through sight, hearing, touch, smell, taste etc. [7]. For driving simulators, there are four main relevant perception systems present in the human body. These can be categorised into audio, visual, somatosensory and vestibular systems; which are explained in detail as follows:

- Audio System - The human body has an audio perception system which is able to take in information with respect to audio signals in the nearby surrounding environment. It does this with the help of the human ear and is extremely useful for not only recognising objects around the human body but, to also get a sense of position in space [8].
- Visual System - The visual system is used to identify position and orientation in space. It does this by using the human eye as its main component. Using this it is able to confirm information that our audio system provides us by matching the signal with the particular object around us. For example; when we hear a person speak from behind, our visual perception system is able to identify the human speaking by looking at them [9].
- Somatosensory System - This system comprises of various sensory neurons that are responsive to different kinds of motion but mostly focus on the physical part. The central nervous system uses these neurons, such as peripheral receptors and neural pathways, to attain information regarding touch, pain, temperature, position and movement of different parts of the body. When dealing with driving simulators, this is also sometimes referred to as haptic feedback; vibrations on the steering wheel [10], [11].

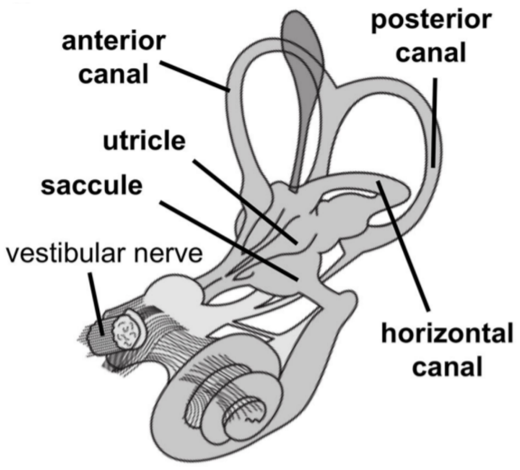


Figure 2.1: Vestibular system of the human body [12]

- Vestibular System - The vestibular system relates to recognising and sensing the human body's movement in space. This characteristic is used in a vast variety of central nervous system features such as body posture, spatial navigation and bodily self-consciousness. It is present inside the human ear and its components can be seen in Figure 2.1 [12].
  The peripheral vestibular organs in the inner ear consist of otoliths. These comprise of the utricle and saccule which are used to sense linear accelerations. It also consists of the semicircular canals which comprise of the anterior, posterior, and horizontal canal (for yaw movement whereas the other two are together responsible for pitch and roll movements) used to sense rotational velocities. The vestibular nerve projects signals from otoliths and semicircular canals to the central nervous system [12]. The process of sensing the required acceleration and angular velocity occurs when our head starts to move. Due to this movement, the semicircular canals which are filled with a viscous fluid called the endolymph exerts pressure on the cupula which is present at the end of each canal. This pressure stimuli is then converted into a nerve discharge sending information to the central nervous system [13]. The same procedure takes place for the otolith organs.

From the above listed perception systems, the vestibular system is of primary importance for motion cueing algorithms as it deals with motion in space. For including the effect of the vestibular system in the algorithm itself, two main components are taken into account. These are the semi-circular canals and the otolith organs as they sense the rotational velocities and linear accelerations. Apart from this, specific force formulation can also be used to represent the perceived accelerations. The different kinds of vestibular models present along with an in-depth analysis is presented in Appendix B.

2.2. Conventional Motion Cueing Algorithms

A conventional filter based algorithm is a type of motion cueing algorithm that uses the properties of high and low pass filters ie. to break down input frequencies into high and low frequency motions, to create the same experience inside the simulator as that would be felt on a real road [14]. This takes place using different kinds of filter based algorithms which are as follows:

- Classical Washout Filter Based Algorithm
- Optimal Washout Filter Based Algorithm
- Adaptive Washout Filter Based Algorithm

2.2.1. Classical Washout Filter Based Algorithm

A classical washout filter based algorithm was the first of its kind filter based motion cueing algorithm, that was used to send motion commands to the driving simulator. A common scheme of this algorithm can be seen in Figure 2.2.

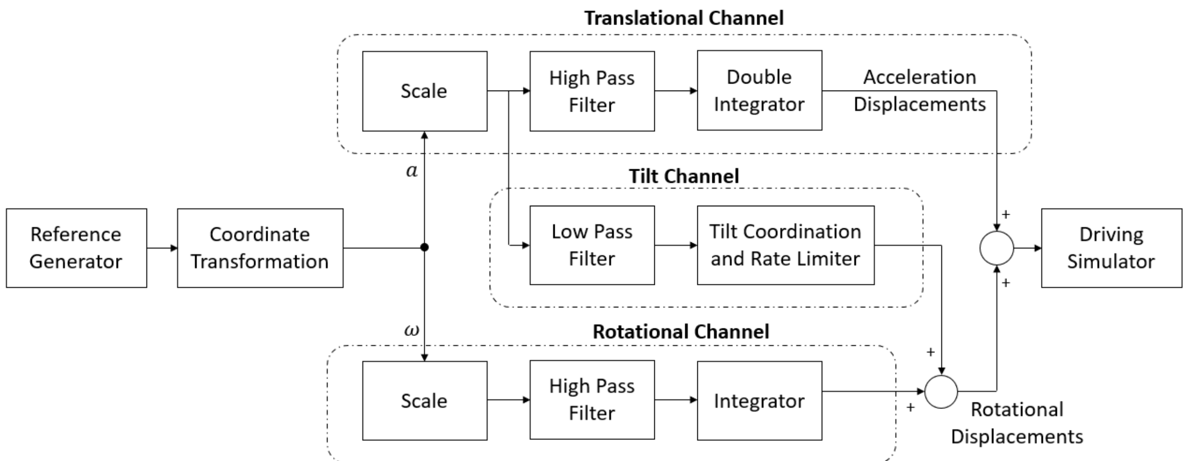


Figure 2.2: Scheme of a classical washout filter based algorithm

The algorithm first starts with the reference generator which is used to send in the inputs, translational acceleration and rotational velocities, to the motion cueing algorithm. Then, the algorithm is divided into three separate channels as shown in the figure. Before the respective input signal is sent to each of these channels, two actions take place. First, the frame of reference of the signal is changed to the inertial reference frame with the help of Euler angles. Second, the transformed signals are scaled appropriately. This scaling action takes place to ensure that the amplitude of the input signal is reduced, as humans are not able to distinguish between forces acting on them on the real road and the specific forces that are experienced inside the simulator. Thus, allowing the execution of more realistic motions with greater amplitudes without exceeding the physical limits of the simulator [15]. The scaling action that takes place for translational acceleration can be seen in Equation 2.1.

$$a_{scaled} = K_{factor} \cdot a_{input} \tag{2.1}$$

Here, a_{scaled} is the scaled translational acceleration, K_{factor} is the scaling factor and a_{input} is the input translational acceleration. A similar action occurs for rotational velocity as well. Once the scaling action is complete, the individual channels execute their functions which are described below:

- Translational Channel - This channel takes in the translational acceleration as the input in the appropriate reference frame after scaling has been done. The signal is passed through the high pass filter and consequently integrated twice to compute the translational displacement. There are two reasons why a high pass filter is present which are as follows:
  - The first reason is to allow the washout effect to take place. Washout here means that the simulator returns to its neutral position when the motion has finished. This allows the simulator to utilize the entire available workspace for its next desired movement. This is done by using a third order high pass filter as convergence takes place for a 3^{rd} order filter ie. displacement of the motion platform tends to zero for translational accelerations. The filter is a combination of 2^{nd} and 1^{st} order filters whose expression can be seen in the equation below [16]:

    $$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta \cdot \omega_n \cdot s + \omega_n^2} \cdot \frac{\omega_w}{s + \omega_w} \tag{2.2}$$

    Where ζ is the damping factor, ω_n is the cut-off frequency and ω_w is the washout frequency allowing the washout effect to take place.
  - The second reason why the high pass filter is used is because the low translational accelerations can drive the actuators for a prolonged period of time. This drives the driving simulator to its physical limits, not allowing anymore acceleration components to be reproduced. Thus, these low sustained accelerations are filtered out [5], [14], [17], [18].
- Tilt Channel - Although the low sustained accelerations drive the simulator to its physical limits, these signals are still important to us. In order to realize the effect of these signals, we exploit a key feature of the human body and especially the otolith organ (which act as a high pass filter) that they are unable to distinguish between the gravito-inertial and translational acceleration forces acting on the human body. So, a tilt channel is first used to extract the low sustained accelerations using a low pass filter with the same cutoffs as the high pass filter. Then, an illusion or false sense of motion is created by tilting the platform below the perception threshold using the rate limiter so that the driver is unable to realize the tilting effect taking place. This tilt effect in the lateral and longitudinal direction is used to reproduce the acceleration terms laterally and longitudinally with the help of the gravitational force, while ensuring that the orientation of the gravito-inertial forces align with the orientation of the real acceleration. The gravity vector can be seen in Equation 2.3 [19].

  $$g_{tilt} = R_x(\theta) \cdot R_y(\phi) \cdot \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{bmatrix} \tag{2.3}$$
- Rotational Channel - The rotational channel is similar to the translational channel but deals with inputted rotational velocities. Here, a high pass filter is again used as the low rotational velocities are not reproducible and irrelevant. Also, the low velocity component is taken care of by the tilt

channel as explained before. The high pass filter in this case is a 2^{nd} order transfer function as we are now dealing with velocities. The expression of the filter is as follows [16]:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.4}$$

Where, ζ is the damping factor, ω_n is the cut-off frequency. Once the low rotational velocities are filtered out, the resulting signal is added to the output received from the tilt channel. Finally, an integrator is used to compute the rotational displacements.

The classical washout based algorithm is an easy to execute motion cueing approach. It is computationally inexpensive as we are only dealing with simple filters. However, it does have some disadvantages while executing the algorithm. The first issue is that it is tuned for the worst case maneuver. This means that the algorithm does not utilize the maximum available workspace for simple or gentle maneuvers, thus leading to poor workspace utilization. Another issue is that it is a feedforward technique as seen in the scheme in Figure 2.2. This leads to suboptimal results as no feedback is provided to the algorithm at any point of time [14], [15], [19]. In order to tackle these issues, two different kinds of filter based algorithms have been developed which will now be explained.

2.2.2. Optimal Washout Filter Based Algorithm

This is a separate class of filter based algorithms whose scheme can be seen in Figure 2.3. In this, the feedforward nature of the classical washout algorithm is dealt with by providing feedback. The formulation also includes a vestibular system as seen in the scheme. In developing an optimal washout filter, the problem is to determine a model or set of transfer functions $W(s)$, that relates the simulator motion input u_{sim} to the driving motion u_{veh}, so that a cost function constraining the sensation error is minimized. Here, sensation error refers to the error between the sensed motion of the driver in the vehicle and the driver in the simulator. A sample cost function used in the algorithm can be seen below:

$$J = E\left\{\int_0^t \left(e^T Q e + x_d^T R_d x_d + u_s^T R u_s\right) dt\right\} \tag{2.5}$$

Where e is the sensation error. The sensed motion consists of the sensed rotational velocity and sensed translational accelerations. x_d are the system states that consist of the simulator's displacement as well as their rates. Sometimes, integrated states are also a part of the system states as it can result in better washout motion. Finally, u_s is the control action of the driving simulator [20]. The cost function defined above is then minimized by solving the Algebraic Riccati Equation giving optimal results.

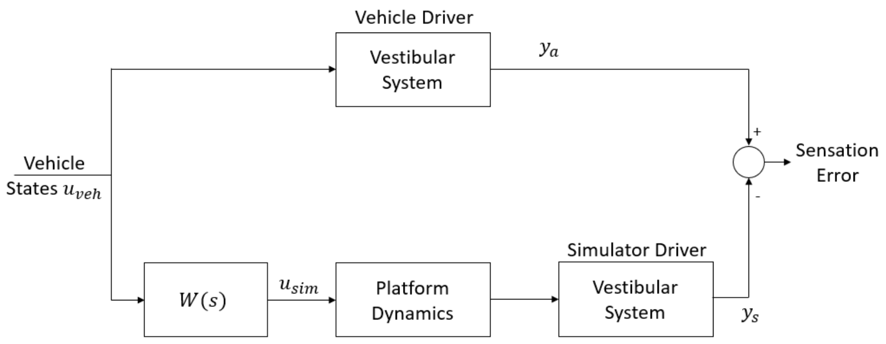


Figure 2.3: Scheme of a optimal washout filter based algorithm [20]

The optimal washout filter based algorithm is able to solve the feedforward issue of the classical approach and improve performance. But, it is also tuned for the worst case maneuver and thus, is not able to maximise workspace utilisation. Adaptive filters are used to tackle this problem which is explained in the upcoming section.

2.2.3. Adaptive Washout Filter Based Algorithm

An adaptive filter based approach tries to solve the issue faced by the classical and optimal techniques. The adaptive filter based algorithm works with adaptive filters which aim at improving maneuvers for gentle/simple cases and give more realistic cues. This is because, previously the filters were tuned for the worst case maneuver which resulted in poor workspace management. Now, adaptive filters can be incorporated into the system. An example of how this takes place can be seen in Figure 2.4.

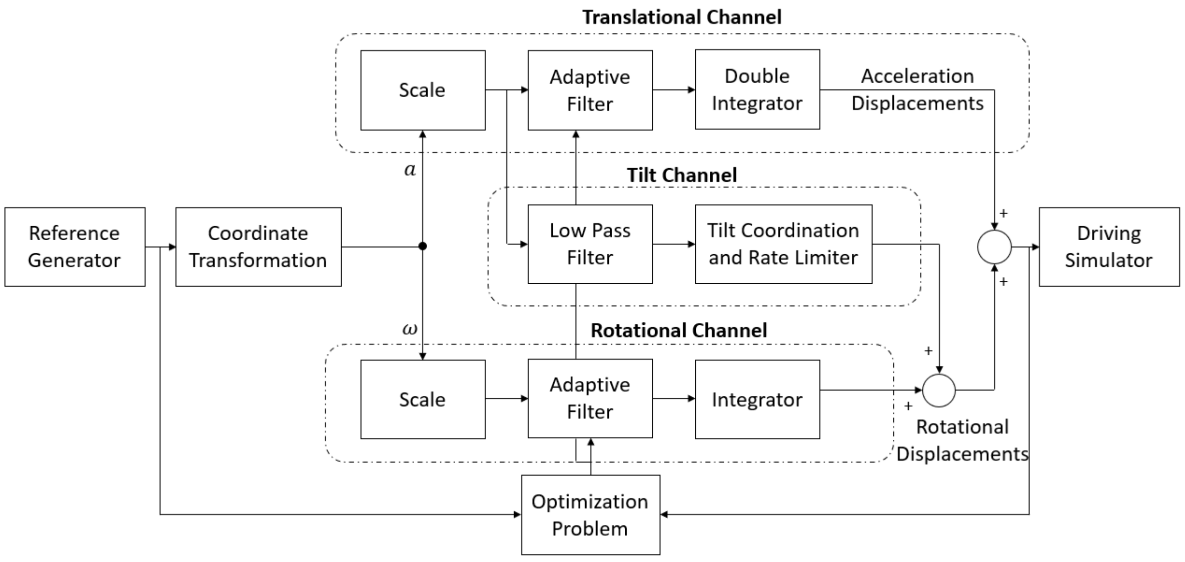


Figure 2.4: Scheme of an adaptive filter based algorithm

Adaptive filters are introduced in the translational and rotational channels as seen in the figure above. These filters are used to minimize a cost function which can be of the form as seen in Equation 2.6. The cost function in this case has terms which compute the difference between the sensed motion of the real vehicle and the driving simulator along with terms comprising of the simulator velocity $\dot{y}_s$ and displacement y_s. Individual weights are applied on each of the terms and a term for the adaptive filter is also added, which is maneuvered according to Equation 2.7. In this, K is the additional tuning parameter. Further, additional terms can also be added on the rotational displacement and velocity in the cost function as done in the reference derivation in [21]. In this implementation of the adaptive filter based algorithm, the tilt low-pass filter gain remains fixed and the high-pass filter gains are adapted to allow the cost function to be minimized.

$$J_y = \frac{1}{2}\left[W_1\,(a - a_s)^2 + W_2\,(\dot{\theta} - \dot{\theta}_s)^2 + W_3\dot{y}_s^2 + W_4 y_s^2 + \textstyle\sum_{i=1}^{2} W_{i+6}\Delta P_i^2\right] \tag{2.6}$$

$$\dot{P}_i = -K_i\frac{\partial J}{\partial P_i}, \quad i = 1, \ldots, 6 \tag{2.7}$$

This concludes the conventional filter based algorithms. Although they have a simple approach and are relatively easy to execute, all three types of filter based algorithms suffer from one glaring issue; they are unable to include explicit constraints as part of their problem formulation. To solve this, an alternate approach which is able to take these factors into account is needed. This is where model predictive control is used as part of the problem formulation in the motion cueing algorithm.

2.3. Model Predictive Control

2.3.1. What is model predictive control?

Model predictive control or MPC is an optimal control problem that uses a dynamic model to forecast system behavior and optimize the forecast to produce the best decision which is the control move at the current time [22]. The overall structure of the MPC scheme is categorised as a feedback control problem.

MPC is a control strategy that as defined before, is extensively used for solving optimization problems. This is because, it gives an optimal solution while handling system constraints and can be designed offline as well as in real-time. A drawback that MPC suffers from is having high computational load and need of an accurate system model. A general MPC scheme can be seen in the figure below:

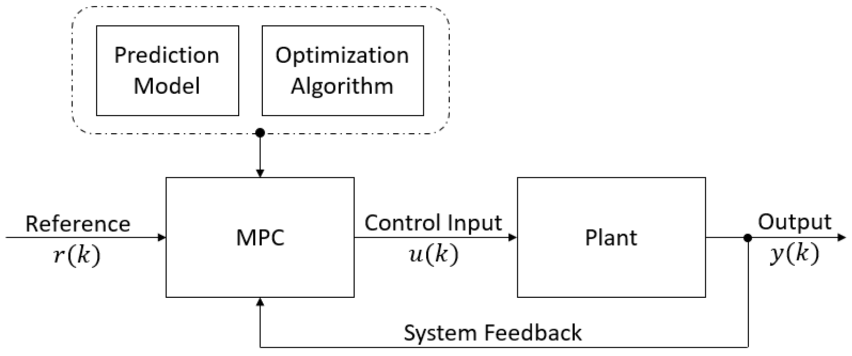


Figure 2.5: Model predictive control scheme

In this block diagram, it can be observed that the MPC block comprises of two main components; the prediction model and the optimization algorithm. The prediction model is used to predict the evolution of future states and the optimization algorithm uses this prediction to compute the best possible control input to achieve the desired results. A reference signal $r(k)$ is provided to the MPC block which performs the desired action according to the optimization algorithm and outputs the control input $u(k)$ for the plant. The output is then received and system feedback is sent back to the MPC block for the next iteration.

The MPC algorithm follows a receding horizon policy in which it first measures the current state $x(k)$ at the current time k. Then, it sets $x_0 = x(k)$ and performs the optimization problem. Upon solving the optimization problem, a sequence of control inputs is found from which the first control input is applied to the model to update the states. Once this is complete, the entire frame is shifted by one timestep and the procedure is repeated once again. This occurs until convergence is reached. The window for which the control input is calculated is known as the prediction horizon N_p. It means that we do not consider events in the future after the prediction horizon nor do we consider events that took place in the past. A layout of the MPC procedure taking place can be observed in Figure 2.6.

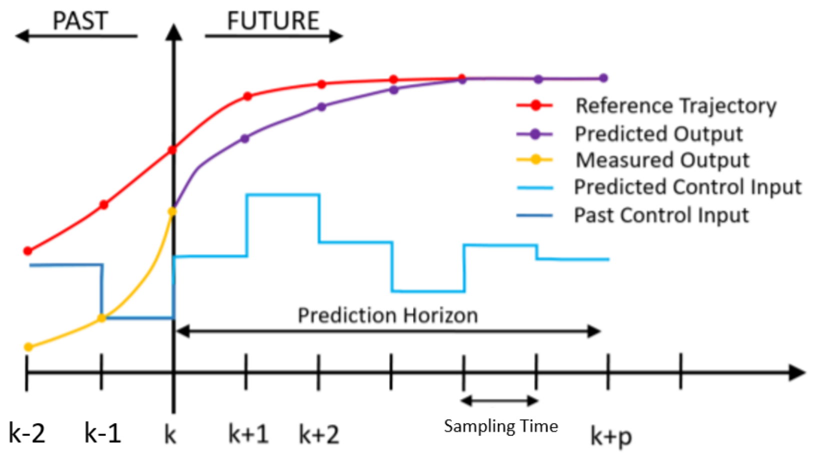


Figure 2.6: Model predictive control problem layout [23]

Based on the general scheme of the MPC algorithm, a sample MPC problem is now formulated to better understand the concept.

2.4. Modeling an MPC Problem

While creating an MPC controller, the first step is to have a model of the system for which the controller is being formulated. An example of what this looks like can be seen below:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{2.8}$$

Here, $x(k)$ contains the states of the model, $u(k)$ is the control input and $y(k)$ is the output. This state space model can also be represented as $x(k+1) = f(x(k), u(k))$. Once the model has been formulated, constraints are applied to the system. This is done in the form of lower and upper bounds and can be of the following types:

- State constraints - Constraints applied on individual states used in the model. For e.g. $-1 \leq x_1 \leq 1$. Here x_1 refers to the first state of the states $x(k)$.
- Input constraints - Constraints applied on the control input. For e.g. $u(k) \leq 5$
- State-input constraints - Constraints applied on some states and control inputs at the same time in relation to one another
- Output constraints - Constraints applied on the output of the system

After the state-space model and constraints associated with it have been formulated, the cost function of the optimization problem can be created. The cost function used in the MPC problem comprises of two main parts:

- Stage cost - Applies penalties on the state and the control
- Terminal cost - Applies penalties on the last step of the problem (at N_p) to allow asymptotic stability to be achieved.

Using the above defined MPC cost function components, the mathematical representation of a sample cost function set can be seen as follows:

$$\begin{aligned} & l(x(k), u(k)) = x(k)^T Q x(k) + u(k)^T R u(k) \\ & V_f(x(N_p)) = x(N_p)^T P x(N_p) \\ & J\left(x_0, u_{N_p}\right) = \sum_{k=0}^{N_p-1} l(x(k), u(k)) + V_f(x(N_p)) \\ & \text{s.t. } u \in \mathbb{U}, x \in \mathbb{X} \end{aligned} \tag{2.9}$$

Where, Q, R, P are positive definite weighting matrices, $l(x(k), u(k))$ is the stage cost, $V_f(x(N_p))$ is the terminal cost and $J(x_0, u_{N_p})$ is the cost function.

Finally, the horizon for which the control inputs would be computed is defined. This involves defining the following two components:

- Prediction horizon - It is the number of future control intervals that the designed controller must evaluate. Thus, it is equal to the length of the optimization window and the control input vector can be represented as follows:

$$u(k) = 0, 1, 2, \ldots, N_p - 1, N_p \tag{2.10}$$

  Further, it is used in defining the look ahead time which is the product of sampling time T_s and N_p.

- Control horizon - The control horizon N_c is chosen to be less than or equal to the prediction horizon and determines the number of parameters used in capturing the future control trajectory. By keeping $N_c \leq N_p$, the computational effort needed to compute the control trajectory prediction reduces. However, using an extremely small control horizon with respect to the prediction horizon can lead to sub-optimal results. Thus, there is a trade-off in computation time and performance based on which the N_c is selected. Its control vector representation is as follows:

$$u(k) = 0, 1, 2, ..., N_c - 1, N_c \tag{2.11}$$

where, $u(k)$ for $N_c + 1, ..., N_p$ are assumed to be constant.

The final MPC problem comprising of the cost function, state space model and the system constraints can be summarised as follows:

$$\begin{aligned} \min_{u_{N_p}} \quad & J(x_0, u) \\ \text{s.t.,} \quad & x(k+1) = f(x(k), u(k)) \\ & x_{low} \leq x(k) \leq x_{upper} \\ & u_{low} \leq u(k) \leq u_{upper} \\ & x(N) \in \mathbb{X}_f \end{aligned} \tag{2.12}$$

The concept of model predictive control and its formulation shown above has been referred from [22], [24].

2.4.1. Types of MPC

The MPC problem can be formulated in the following three ways:

- Implicit MPC - This is a type of MPC in which the optimal control sequence $U^*_{N_p}$ is obtained by solving the optimization problem online at each time step. The main aspect to be considered while using implicit MPC algorithms is to look at computational complexity of the problem which depends on the prediction model and the choice of the cost function as well [25].

- Explicit MPC - An explicit approach on the other hand uses parametric programming to reduce the given optimization problem to a series of piecewise affine (PWA) functions offline. This is done by first restating the initial optimization problem in the form of a multi-parametric linear or quadratic programming problem (mpLP or mpQP) [25]–[27]. An example can be seen below:

$$\begin{aligned} V^*(x) = \min_{u_{N_p}} J(x_0, u) \\ \text{s.t. the constraints} \end{aligned} \tag{2.13}$$

A value function is created as seen above. Here, $J(x_0, u)$ is the original cost function which can be of the type as seen in Equation 2.9. The final transformed mpQP/mpLP can be written as follows:

$$\begin{aligned} V^*(x) = \min_{U} \quad & \frac{1}{2} U^T H U + x^T F U \\ \text{s.t. } & LU \leq b + Ex \end{aligned} \tag{2.14}$$

Where L, b and E are suitable constant matrices created after conversion from the initial optimization problem to mpLP/mpQP. This is used to create a look-up table containing all the feasible parameters partitioned in the form of critical control regions. Then, while running the algorithm, the look up table can be used to compute the control action based on the position in the corresponding critical region. An example of what the control action looks like can be seen below [26], [28]:

$$U(x) = F_i x + G_i \text{ if } x \in \mathcal{CR}_i. \tag{2.15}$$

Where, $\mathcal{CR}_i$ are the control regions as visualized in Figure 2.7 for an example problem comprising of two states and a control input.

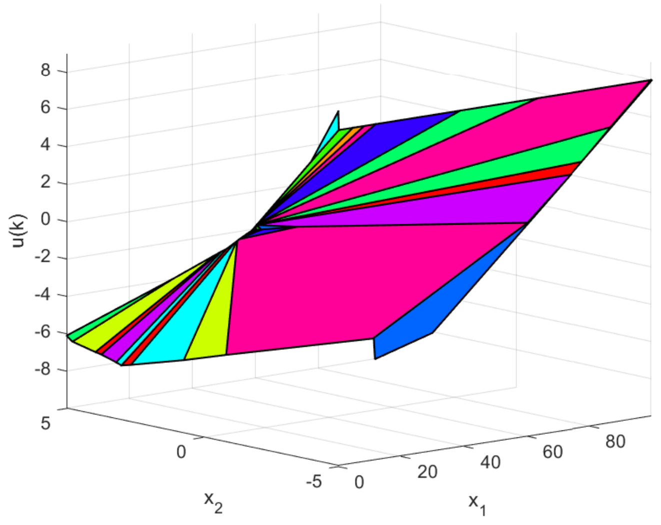


Figure 2.7: An example of critical regions for a two state problem with a control input

Explicit MPC is used to deal with the issue of computational load present in the implicit approach, but, this also results in memory storage issues when dealing with problems having huge computational complexity.

- Combined/Hybrid MPC - Apart from the implicit and explicit approaches present, a combination technique has also been developed and used in literature. It uses the functionalities of both implicit and explicit MPC approaches, with the explicit MPC algorithm acting as the initial/warm start for the online MPC block (by means of a look up table). This is effectively done to advance the starting point of the online optimization problem, which reduces online computation time and load. Such algorithms have been developed and used in the past starting from Zeilinger et al. [29], who specified the advantages of using a warm start technique (reduced number of iterations to solve the problem online) in a combined approach. The same technique was also used by Jost et al. [30] with modifications in the explicit initialisation. Since then it has been further used in other applications such as curve tilting [31] and lateral motion stabilisation [32].

2.4.2. Past Literature Work

Motion cueing algorithms have developed extensively over the past decade with MPC acting as the main control technique. Initially, this implementation was done for linear models only, using an implicit based MPC algorithm. Beghi et al.[19] described a linear MPC based approach which had the human perception model from Telban [33] in its formulation. The MPC problem was created by applying constraints on the driver eye point position and velocity with its implementation done in real-time. This work extended previous research done by the authors which focused on providing drivers with complete information about future values, thus making it an offline simulation based problem using a virtual driver.

Garrett et al. [34] also developed a linear implicit MPC based algorithm but used the perception model from Nahon and Reid [35] instead. In order to reduce the computational burden, the MPC problem was divided into four sub-problems; longitudinal/pitch, lateral/roll, vertical and yaw. Further, the QP solver was modified in order to reduce computation time as well. This was done by implementing the warm start capabilities of the active set strategy into the interior point method. This modification showed a slight advantage when compared with the classical quadprog solver as observed from the simulation studies present in the paper. For future work, the authors recommended to combine all DOFs as they have an effect on each other and can lead to better results. Also, Degdelen et al. [36] worked on the ULTIMATE driving simulator which is an 8 DOF model. They took the human perception model from Young [37] into account and also applied perception thresholds as part of its constraints. The actuator controller was used to deal with the non-linearities of the motion system and to formulate the problem as a linear optimization problem. This approach also did not account for all the DOFs during the experimentation; something which leads to sub-optimal results and was observed in previous work as well while dealing with linear models.

The MPC based approach using linear system models gives better results when compared with conventional filter based algorithms but suffers from problems as well. As these algorithms employ constraints in the driver reference frame, to keep the problem linear, sub-optimal results are obtained. This is because working in this reference frame creates difficulties in realizing the available workspace and thus an approximation is taken. Further, in some works as mentioned before, all DOFs are not taken into account which can also lead to poor results.

After seeing the issues that linear models deal with, nonlinear MPC based algorithms started being used which were able to take the nonlinear system dynamics into account. Bruschetta et al. [4] worked with a 9 DOF driving simulator. For this, the nonlinear MPC algorithm employed the vestibular model by Telban. Constraints were applied on the actuator lengths and the model was then compared with a linear MPC based approach. A set of constraints were also applied on making sure that no interference takes place between the actuator actions. For this, measurements were collected and then mapped to fit a nonlinear function imitating the closed surface. The results showed improvements from the linear model as it was now able to take non-linearities of the system into account. The entire study was performed using `ACADO` and the SQP-RTI algorithm [38]. A thesis by Katliar [39] analysed two different approaches; an offline approach in which all desired motion trajectories are pre-calculated and then fed in, and a real-time DIL implementation. Mohammadi et al. [40] aimed at reducing the computational burden of selecting the correct prediction and control horizons respectively. Thus, an optimization problem was formulated using genetic algorithm. For this, a profiling table was first created using the original MPC problem. This table stored the simulation time, control and prediction horizon values which satisfied the constraints. Also, while creating the profiling table, values which led to timeouts were rejected. Then, surface fitting was done to compute the weights of the horizon terms. Once this was complete and the function was fitted, optimal horizon values were found using the main optimization problem which were minimum in total cost. These were then sent to the online MPC problem for performing the desired simulator motion.

Khusro et al.[5] developed an approach in which the driver gives an input to the vehicle model (throttle/brake/gear) to receive the initial velocity and acceleration values. These were then transformed to the driver reference frame and sent to the vestibular system (by Telban [33]) to compute the actual perceived values. This along with the inverse kinematics result was used by the reference generator which assigned all other states in the model except the velocity and acceleration terms (as these values were taken from the vestibular system output) to zero to allow the washout effect to occur. Then, the MPC applied constraints on the tilt rate and the actuator positions and solved the motion cueing problem using `ACADO` with the SQP-RTI technique. The algorithm was compared with the linear MPC and washout filter algorithms and showed better results across all test cases. Lamprecht et al. [41] implemented a cueing algorithm which is slightly different from the algorithm developed by Khusro. Here, a separate OCP is solved by using a reference trajectory. The solution of this problem is then passed through the human perception model which gives the desired angular velocity and acceleration values to be used by the MPC block. Tilt rate constraints are applied by taking the difference between the desired values and the ones received from the human perception model. Finally, weights are adapted and an Inverse Optimal Control (IOC) technique is applied with the real time driver prediction block by comparing the predicted and sample experiment values, and subsequently solving the optimization problem. Subjective analysis was done using experiments on a simulator to compare the developed algorithm with the classical washout filter based algorithm. Results showed that 70% of the participants preferred the MPC based algorithm over the classical washout filter based algorithm.

Some nonlinear based work has also been done using the ULTIMATE driving simulator. Soyer et al. [42] used two different methods to allow the tracking action to take place in the motion cueing algorithm. This was done by splitting up the linear rail and the nonlinear actuator (hexapod) dynamics respectively; as the simulator used was 8 DOF. Then, one of these dynamics was given priority while the other compensated for the remaining action to occur. Consequently, this required two MPC blocks as well. The first approach was the linear-nonlinear (LNL) scheme in which the hexapod rotation compensated for rail displacement while in the nonlinear-linear (NLL) approach the opposite happened. In both cases, the algorithms performed better with lower computation time when the prediction horizon of the compensating controller was smaller than the other (this works as now the initial MPC block gets a larger room to work with, giving a very close approximation to be compensated by the second MPC block). Before doing this, when the prediction horizons were the same, it was observed that the NLL algorithm had a higher computation time than the LNL approach and thus, was not preferred amongst the two.

Apart from cueing algorithms based on implicit MPCs, work has also been done using the offline model of the controller. Fang et al. [6] implemented an explicit MPC based motion cueing algorithm which included braking constraints to reduce the platform velocity and tilt rate near the displacement limits of the simulator. This helped in improving stability of the algorithm and give better reference tracking. Using this algorithm, tests were performed in the ULTIMATE driving simulator, with perception threshold to limit the platform's rate of motion. Munir et al. [43] extended the work done by Fang by looking at the issue of complexity of the created explicit MPC problem. This refers to the number of control regions of the created explicit solution. The study also used the human vestibular model as part of the algorithm. Thus, the state space model comprised of two parts; the first was the system model which was the same as that of the previous paper. It included the linear position, linear velocity, tilt angle, tilt velocity and the perceived acceleration as the states. The second is the human vestibular model based on Telban's work. To deal with the issue of reaching real-time computational limits due to increasing number of states, a contractive set approach has been made in this paper. Contractive sets allow the problem domain to reduce over-time while ensuring that all the given constraints are satisfied. This allows the problem to deal with only the relevant workspace as time goes on, while starting with the maximum available workspace; thus reducing computational load. The results showed similar performance with reduced number of control regions. But, there is a limitation present on the control horizon when dealing with the full model. A value of 2 cannot be computed and is impractical. The prediction horizon chosen in the simulations was 1. Further, information on the effect on computation time and whether this approach can be applied to larger problems such as 4 or 6 DOF was not mentioned in the study.

To conclude, there has been work done using MPC algorithms in this domain ranging from linear to nonlinear models and from implicit to explicit MPC based algorithms across different kinds of simulators. As seen from literature, no particular method is perfect; implicit based approaches deal with computational time problems and explicit ones deal with memory storage issues. By taking the nonlinearity of the model into account, the performance does improve but the complexity also increases. Thus, a solution is needed which is able to take these issues into account and run the algorithm successfully in real-time.

2.5. Summary

In this chapter, different kinds of human perception systems present in the human body were first elucidated. Then, an overview on conventional filter based motion cueing algorithms was given. This included information on all three kinds of conventional algorithms starting from the classical approach to the adaptive washout filter based algorithm. Such algorithms are simple and easy to implement but suffer from issues related to sub-optimal workspace utilisation; due to the inability to include constraints as part of the problem formulation. In order to improve cueing performance, an advanced control technique called MPC is used. The basics of MPC and its variations were explained along with work done using this control technique. Finally, the major advantages and drawbacks of using MPC were listed. In the next chapter, the main controller design used in the proposed motion cueing algorithm will be explained in detail.

3 Main Controller Design

In this chapter, the main controller design used for the developed motion cueing algorithm will be presented. As mentioned in the previous sections, implicit and explicit MPC based approaches suffer from their respective individual issues related to computation time. Thus, a better alternative is needed for improving real-time performance. For this, a combined/hybrid MPC is used which was alluded to earlier in subsection 2.4.1. Such a controller advances the starting point of the online optimization problem by using the offline guess, which in-turn reduces online computation time. This type of a MPC has not yet been implemented in the domain of motion cueing algorithms and will be used to mitigate the issues that the implicit and explicit MPC based motion cueing algorithms face.

The scheme of the designed controller can be seen in Figure 3.1. First, initial state and reference values are sent to the explicit MPC block. Here, the reference values are the specific forces. Such forces are used to represent the perceived accelerations which are a sum of translational acceleration and gravitational tilt components, thus, taking all acceleration forces into account allowing us to study the human body's movement in space during motion cueing. These forces can be manually generated signals such as sine waves or be obtained from track simulations. The explicit MPC block, which is a pre-generated look-up table, performs a search to find the corresponding control region related to the state and reference values sent as input. Once the control region is selected, the associated control inputs which are the translational and rotational accelerations are taken (depending on the DOFs in the problem) and sent as an initial guess to the online MPC.

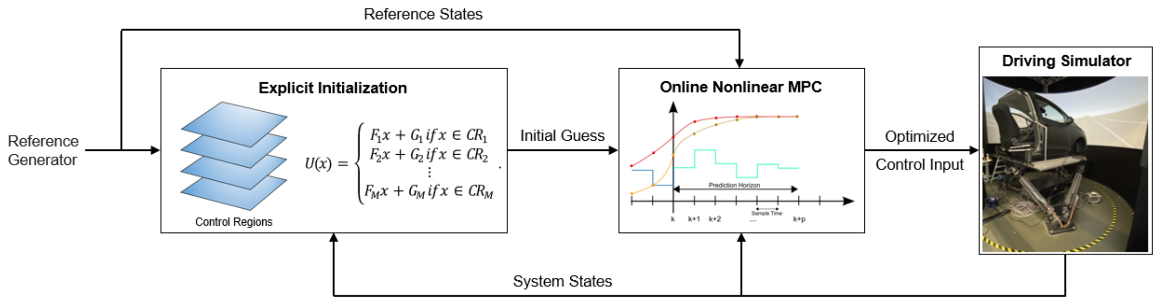


Figure 3.1: Scheme of the hybrid MPC based motion cueing algorithm

The implicit MPC block takes this guess as the initial starting point of the algorithm, along with the generated reference signal and state values to compute an optimized control input. This control input acts as the control command, sent to the driving simulator to perform the desired motion. The simulator dynamics used while applying the control inputs can be simplified actuator dynamics or advanced dynamics based on the driving simulator being used. Once the motion is complete, the states are updated and used as feedback for the next time step. This entire process is repeated until the simulation finishes.

For performing the simulation, the explicit control law is generated and stored using the open source Multi Parametric Toolbox (MPT) [44]. This is then used in the Simulink model of the hybrid MPC scheme as the initial guess. The model also contains the implicit controller which is generated using `ACADO`, along with the reference generator and other functionalities used for weight tuning and hybrid model selection. These are explained in detail in the subsequent sections. The Simulink model containing the entire hybrid scheme can be seen in Appendix D.

3.1. Types of Motion Cueing Models

The hybrid motion cueing algorithm can be used in different ways, depending on the initial control input guess and how it is applied. This application takes place inside the shift reference block, where the appropriate control input guesses are sent in to be used by the implicit controller. The guesses that are sent in are usually from the explicit MPC; which is used to compute and store the explicit control law. Apart from the control input guesses, reference state values are also sent in as part of the reference trajectory array. A general form of this array, used with explicit MPC control inputs u_{eMPC}, can be seen in Equation 3.1. This block generates the reference prediction for the entire prediction horizon of the implicit MPC controller which is updated at each time-step.

$$y_{in} = [repmat([yN, u_{eMPC}], N_p, 1)', yN'] \tag{3.1}$$

This shift reference equation which is sent in as input changes as per the model being used. As this research focuses on analysing differences in online computation time, the following four types of cueing models were used and studied:

- iMPC without any initial control input guess
- iMPC with control trajectory prediction - In this version, the control trajectory prediction received from the `ACADO` s-function is used as the initial guess for the next time step. The first control input from the trajectory prediction is selected and applied for the entire horizon in the shift reference block and Equation 3.1 is updated accordingly.
- Hybrid algorithm with first explicit MPC control input - In this model, the first control input from the explicit MPC look-up table is used and applied for the entire prediction horizon.
- Hybrid algorithm with all explicit MPC control inputs - The final hybrid model utilises all control inputs obtained from the explicit MPC controller and applies them for the entire horizon. As the sampling times vary between both controllers, explicit MPC inputs are applied in equal sections throughout the larger prediction horizon of the implicit MPC. For e.g. with a $N_{p,eMPC}$ of 5, the five control inputs are applied ten times each (1^{st} from 1-10, 2^{nd} from 11-20 and so on) for a $N_{p,iMPC}$ of 50.

The explicit controller was not considered separately while studying different kinds of cueing models. This is due to limitations in model complexity and using large N_p values with fast sampling rates; leading to poor performance. Thus, it is used in the hybrid scheme only as an initial guess, to kick-start the algorithm.

3.2. Summary

In this chapter, an overview of the hybrid controller used in developing the motion cueing algorithm has been explained. Further, information on different types of motion cueing models developed and analysed in this study was given. In the next chapter, the detailed description of the hybrid MPC algorithm containing both types of MPCs is elucidated. First, the 2 DOF algorithm is presented followed by the extended 4 DOF motion cueing algorithm.

4 Motion Cueing Algorithms

In this study, two kinds of motion cueing algorithms have been developed, for specific force tracking, which differ based on the number of DOFs taken into account. The first is a pitch-surge motion cueing algorithm followed by an extended version including sway-roll DOFs. The controller design explained in the previous chapter has been used for both algorithms and in this chapter, the individual components of each is explained in detail.

4.1. 2 DOF Hybrid MPC based Motion Cueing Algorithm

In this section, the explicit and implicit controllers used in the developed 2 DOF motion cueing algorithm are described. First, the design and features of the explicit controller are presented followed by the implicit MPC based algorithm.

4.1.1. Explicit MPC based Motion Cueing Algorithm

Explicit MPC involves computation of control regions offline to reduce the online simulation load by using a simple look-up table to select the appropriate control input.

The explicit MPC based motion cueing algorithm comprises of four states related to the simulator platform p, namely; platform displacement s_p, platform velocity v_p, tilt angle θ_p and tilt rate ω_p. These states allow the displacement and velocity effects in both translational and rotational motions to be taken into account for the 2 DOF problem. The state space equations used to formulate the model can be seen below:

$$\dot{x}(k) = \begin{cases} \dot{\omega}_p = a_{p,rot} \\ \dot{\theta_p} = \omega_p \\ \dot{v}_p = a_{p,tran} \\ \dot{s_p} = v_p \end{cases} \tag{4.1}$$

This results in a surge-pitch problem with translational and rotational accelerations $(a_{p,tran}, a_{p,rot})$ acting as the control inputs $u(k)$. The combined system can be represented in the following manner:

$$\dot{x}(k) = f(x(k), u(k)) \tag{4.2}$$

While designing this motion cueing algorithm, it is of major importance that the rotational motion is limited below the perception thresholds as explained before in subsection 2.2.1. Thus, constraints are applied on the tilt rate to ensure that the tilt coordination effects are not perceived by the driver in the simulator. From [45], it is known that most individuals have a comparatively high perception threshold with some outliers showing higher sensitivity towards the tilt rate; resulting in a lower threshold. Hence, to ensure that all individuals are not able to realize the tilting effect taking place, a lower value in the range of 2-4 deg/s is generally used [5], [33], [35], [41], [46], [47]. A summary of these perception threshold values can be seen in Appendix B. Based on this analysis, a value of 3 deg/s was chosen for the pitch tilt rate in the proposed motion cueing algorithm.

Further, a constraint is applied on the platform displacement so that the physical limits of the simulator are not exceeded. Along with this, constraints are also applied on the platform's velocity and translational control input to ensure that the speed of the platform can be controlled. The limit values for these constraints were chosen based on the DAVSi's capabilities and are listed in Equation 4.3.

$$\mathrm{X} = \begin{cases} -3deg/s \leq \omega_p \leq 3deg/s \\ -30deg \leq \theta_p \leq 30deg \\ -7.2m/s \leq v_p \leq 7.2m/s \\ -0.5m \leq s_p \leq 0.5m \\ -9.81m/s^2 \leq a_{p,tran} \leq 9.81m/s^2 \end{cases} \tag{4.3}$$

The goal of this formulated motion cueing algorithm is to track a reference specific force which is the output of the model. The formulation used to compute the specific force comprises of two components; translational acceleration and gravitational tilt. The gravitational tilt component takes the tilt angle multiplied by the gravity into account whereas the translational acceleration is the translational control input. This results in the following specific force formulation:

$$y(k) = f_{spec} = a_{p,tran} + g\sin\theta_p \tag{4.4}$$

Furthermore, the cost/objective function consists of weighted states, outputs and control inputs. The corresponding weights chosen for our simulations are presented below:

$$\begin{aligned} &W_{states} = diag([0,0,0,0]) \\ &W_{output} = 1 \\ &W_{controlinput} = diag([1e-3, 1e-3]) \end{aligned} \tag{4.5}$$

A value of 1 is assigned to the output as that is main goal of the algorithm. As the states are already constrained within the workspace limits, free movement is desired which is obtained by applying an extremely small/zero weight value. The control inputs are given same weights to ensure that the MPC performs motion, both translationally and rotationally, in equal quantities without giving one priority over the other. Their value is selected after performing simulations and looking at the overall contribution and effect on tracking performance. The current values of $1e-3$ allow smooth reference tracking (without chattering) to take place with equal contribution from both DOFs. Such weights ensure that the objective of the cost function is not conflicting during the simulation and they are applied using the `QuadFunction` in MPT. Also, these weights can be manipulated in the MPC formulation but give the same results as long as the ratio of weights remain the same.

Selection of sampling time is important for the motion cueing algorithm. If it is too small, it can limit the use of larger prediction horizons N_p due to high computation load. Using a small N_p in these situations can lead to poor performance due to limited future information. Also, applying a higher sampling time value can result in loss of information for applications requiring fast reaction/response. Thus, sampling time also depends on the frequency of control actions that the plant (simulator in this case) can work on. For the proposed motion cueing algorithm, a sampling time of 0.1s and a N_p of 5 was selected. This combination ensures that a look ahead time of 0.5s is maintained which allows sufficient future information to be considered for the cueing problem. Faster sampling rates with larger N_p cannot be taken due to the limitations of the explicit controller. More information about these limitations is presented and discussed in subsection 5.2.2.

Thus, the structure of the motion cueing algorithm is as follows:

$$\begin{aligned} \min_{u_{N_p}} \quad & J(x_0, u) \\ \text{s.t.,} \quad & x(k+1) = f(x(k), u(k)) \\ & x \in \mathrm{X} \\ & x(N) \in \mathbb{X}_f \end{aligned} \tag{4.6}$$

Move Blocking Constraint

A major drawback of using explicit MPC is the computation time needed in generating the control laws. As the problem complexity increases through additional states/inputs/constraints, the computation time rises exponentially. Due to such limitations, the complexity of the problem is often reduced by a large degree to allow controller execution to take place at the desired sampling rate and horizon values.

Instead of limiting the complexity of the problem, there is an alternate approach that can be taken to ensure that the desired problem works at reasonable sampling rates. This takes place by applying the move blocking functionality of MPT. Move blocking helps in constraining the predicted control moves over the horizon and acts as the control horizon for the MPC problem. A schematic view of how the control inputs can look like is shown in Figure 4.1.

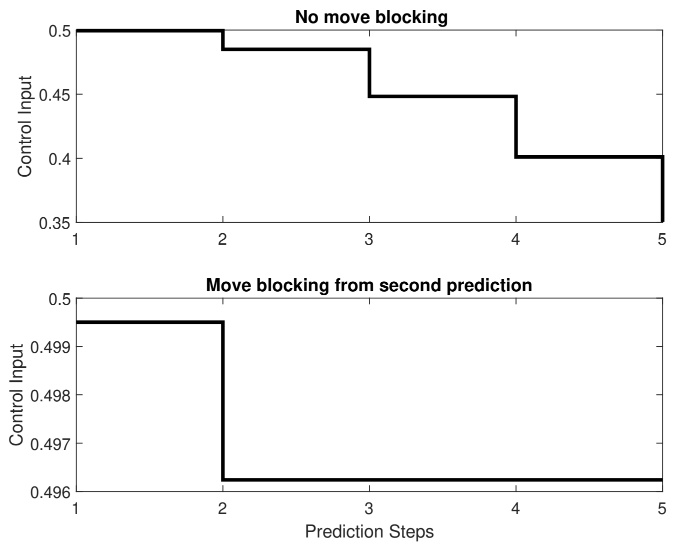


Figure 4.1: Effect of move blocking constraint

A reduction in computation time is observed when this constraint is applied while maintaining reasonable complexity and sampling rate for the problem at hand. The effect of using this constraint for the explicit MPC based motion cueing algorithm can be seen in subsection 5.2.2. For the proposed motion cueing algorithm, it was applied from the second prediction step.

Braking Constraints

Apart from the set of constraints taken into account for the explicit MPC based algorithm, additional braking constraints were also considered. These were first introduced in [6], which are used to slow down the platform velocity and tilt rate of the cueing algorithm when the platform displacement and tilt angles are about to reach their respective limits. Thus, a deceleration effect takes places without any washout taking place.

To implement this, two sets of constraints are used; one for platform velocity and the other for the tilt rate as follows:

$$s_{p,min} \leq c_v v_p T_{brk,p} + 0.5 c_u a_{p,tran} T_{brk,p}^2 \leq s_{p,max} \tag{4.7}$$

$$\theta_{p,min} \leq c_w \omega_p T_{brk,\theta} + 0.5 c_u a_{p,rot} T_{brk,\theta}^2 \leq \theta_{p,max} \tag{4.8}$$

where, $c_v = 1, c_w = 1, c_u = 0.45, T_{brk,\theta} = 0.5$, $T_{brk,p} = 2.5$ and s_p, θ_p thresholds are 0.5m and 30 deg (according to the DAVSi) respectively. The selection of these parameters is explained in section C.1 and the effect of using braking constraints is analysed in subsection 5.2.3. These constraints have not been used in the explicit algorithm, but have been studied for implementation in the online counterpart of the hybrid approach. Moreover, to validate the working of the explicit algorithm based on the model in [6], [43], a benchmark study was done as seen in section C.2.

4.1.2. Implicit MPC based Motion Cueing Algorithm

The implicit MPC based algorithm is designed to be slightly more complex compared to the explicit MPC based approach, through additional functionalities working at faster sampling rates and higher prediction horizon values. This is possible as at each time-step, the online solver only takes the current state and reference values into account, to compute the optimum control input. On the other hand, the offline version considers all feasible scenarios and computes the control input for each, after partitioning them in the form of control regions. Thus, using larger N_p and sampling time values, leads to increased computation time for the offline algorithm and in some situations an uncomputable solution. As the online computation load per time step is reduced, additional states, control inputs and constraints can be taken into account.

The states $\dot{x}(k)$ are updated by including platform accelerations (translational and rotational), which were previously acting as the control inputs. These are replaced by commanded accelerations, ($a_{cmd,tran}$ and $a_{cmd,rot}$), which are modelled incorporating a time delay in the system. This ensures that the acceleration ($a_{p,tran}$ and $a_{p,rot}$) values are achieved. The updated state space model is as follows:

$$\dot{x}(k) = \begin{cases} \dot{\omega}_p = a_{p,rot} \\ \dot{\theta}_p = \omega_p \\ \dot{v}_p = a_{p,tran} \\ \dot{s}_p = v_p \\ \dot{a}_{p,tran} = \dfrac{a_{cmd,tran} - a_{p,tran}}{T_s} \\ \dot{a}_{p,rot} = \dfrac{a_{cmd,rot} - a_{p,rot}}{T_s} \end{cases} \tag{4.9}$$

As far as the constraints are considered, braking constraints are added into the system which have already been alluded to before. These are used to improve performance by slowing down the system as it reaches its physical limits. The updated set of constraints acting on the system are as follows:

$$\mathrm{X}_{iMPC} = \begin{cases} -3deg/s \leq \omega_p \leq 3deg/s \\ -30deg \leq \theta_p \leq 30deg \\ -7.2m/s \leq v_p \leq 7.2m/s \\ -0.5m \leq s_p \leq 0.5m \\ -9.81m/s^2 \leq a_{p,tran} \leq 9.81m/s^2 \\ -0.5m \leq c_v v_p T_{brk,p} + 0.5 c_u a_{p,tran} T_{brk,p}^2 \leq 0.5m \\ -30deg \leq c_w \omega_p T_{brk,\theta} + 0.5 c_u a_{p,rot} T_{brk,\theta}^2 \leq 30deg \\ -5m/s^2 \leq a_{cmd,tran} \leq 5m/s^2 \end{cases} \tag{4.10}$$

Cost function weights remain the same as mentioned in Equation 4.5. This is because the goal is still to track the output specific force. The formulation of $y(k)$ also remains the same as seen in Equation 4.4.

As mentioned before, the implicit controller is able to work at a faster rate as compared to its explicit counterpart. Thus, a prediction horizon N_p of 50 was used with a sampling time T_s of $0.01s$ for the controller. These values were selected to maintain the same look ahead time of $0.5s$ as done for the explicit controller while working at a faster rate with more information and higher accuracy (due to increased horizon steps and faster reaction rate). The same look ahead time is desired for both controllers to allow 1:1 matching to take place, thus utilising all the available information from the explicit controller working on the same problem; but with a difference in the sampling rate and N_p. Also, a faster sampling rate of $0.001s$ was selected for the plant, to match the frequency of the DAVSi at $1kHz$.

The structure of the implicit motion cueing algorithm is as follows:

$$\begin{aligned} \min_{u_{N_p}} \quad & J(x_0, u) \\ \text{s.t.,} \quad & x(k+1) = f(x(k), u(k)) \\ & x \in \mathrm{X}_{iMPC} \\ & x(N) \in \mathbb{X}_f \end{aligned} \tag{4.11}$$

Washout

The last part of the implicit MPC based motion cueing algorithm involves adding washout weights into the model, which is another extension from the initial explicit MPC based algorithm. This is not included in the offline algorithm due to limitations of MPT.

Washout is added in the online algorithm to bring the platform back to its neutral position. This ensures that maximum workspace utilization takes place for the next simulation. To realize this, penalisation is provided on the states of the system, namely, the tilt rate ω_p and platform displacement s_p. The reason why these two states are selected is because the movement of the platform is restricted by limits on these states; due to limited workspace of the driving simulator. This limitation in translation/rotational movement leads to tracking error and false cues. False cues are undesirable as they reduce the overall feel of the cueing process taking place and can even lead to motion sickness. Thus, by applying weights on these two states, we can ensure that their respective limits are never met and that false cues are removed from the system. Implementation of these weights can take place in the following two ways:

- Constant Washout - Constant washout weights are used for the entire simulation. These weights need to be tuned for the specific signal being analysed. Thus, if the amplitude or frequency of the signal is altered, the ideal weights change and need to be tuned again. This makes using this approach a very time consuming task, requiring a better alternative in the form of adaptive weights.

- Adaptive Washout - Adaptive weights fix the above mentioned problem where tuning was needed every time a new signal was analysed. Here, an adaptive function, which is a 4^{th} order polynomial, is used to alter the weight values based on the thresholds and current values of the respective states; ω_p, s_p. This polynomial allows manual tuning of the shape of the adaptive function and the formulation for each of these two states can be seen in Equation 4.12. Figure 4.2 shows how the weight adapts itself based on the position of the platform. This nonlinear shape was chosen as it rapidly penalises the platform position when it is going towards its respective limit. From 0/neutral position to 0.25m, the rate of increase is low allowing free movement of the simulator to perform the desired motion. On the other hand, from 0.25m to the limit of 0.5m, a rapid increase is observed. This ensures that the priority of the objective function shifts towards bringing the platform back to its original position, while tracking the reference specific force. Thus, the chosen nonlinear function helps maintain freedom of movement for the simulator and changes priority to allow washout to occur when close to the limit.

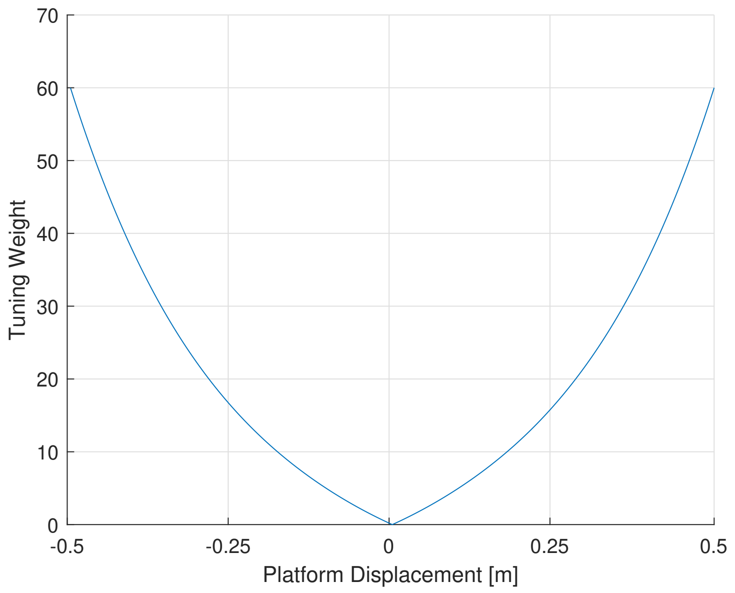


Figure 4.2: Adaptive weights for platform displacement s_p

$$
\begin{aligned}
W_{s_p} &= w_{s,1} + w_{s,2}\left(\frac{abs(s_{p,i})}{w_{s,5}}\right) + w_{s,3}\left(\frac{abs(s_{p,i})}{w_{s,5}}\right)^2 + w_{s,4}\left(\frac{abs(s_{p,i})}{w_{s,5}}\right)^4 \\
W_{\omega_p} &= w_{\omega,1} + w_{\omega,2}\left(\frac{abs(\omega_{p,i})}{w_{\omega,5}}\right) + w_{\omega,3}\left(\frac{abs(\omega_{p,i})}{w_{\omega,5}}\right)^2 + w_{\omega,4}\left(\frac{abs(\omega_{p,i})}{w_{\omega,5}}\right)^4
\end{aligned}
\tag{4.12}
$$

where, $w_{s,1} = 0.01, w_{s,2} = 20, w_{s,3} = 20, w_{s,4} = 20, w_{s,5} = 0.5, w_{\omega,1} = 0.0001, w_{\omega,2} = 0.7, w_{\omega,3} = 0.7, w_{\omega,4} = 0.7$ and $w_{\omega,5} = 3$.

The updated weight vector used by the implicit controller can be seen in Equation 4.13, with other weights remaining the same as before. The same formulation is used for both constant and adaptive weights, with W_{ω_p} and W_{s_p} acting as the washout weights.

$$
Weight = reshape(diag([1e-16W_{\omega_p} 1e-16W_{s_p} 1e-16 1e-16 1e-0 1e-3 1e-3]), 1, 9^2) \tag{4.13}
$$

4.2. 4 DOF Hybrid MPC based Motion Cueing Algorithm

The structure of the hybrid MPC based motion cueing algorithm remains the same as before; where control inputs from the explicit MPC are taken as an initial guess to compute the optimized control inputs online. In this section, the explicit MPC based motion cueing algorithm is first explained including additions made to consider lateral DOFs as part of the algorithm. Then, the online part of the hybrid scheme is elucidated concluding the design of the motion cueing algorithm.

4.2.1. Explicit MPC based Motion Cueing Algorithm

The motion cueing algorithm developed for 2 DOF problems is now extended to take sway-roll DOFs into account. By adding these additional DOFs into the motion cueing algorithm, some changes are made in the formulation of the MPC controller.

First, the state space model is updated to take the additional lateral DOFs into account. As done before, displacement and velocity states are used to represent the motion platform in both translational and rotational directions. Four additional states are added and the entire model can be seen below:

$$
\dot{x}(k) = \begin{cases}
\dot{\omega}_{p,long} = a_{p,long,rot} \\
\dot{\theta}_{p,long} = \omega_{p,long} \\
\dot{v}_{p,long} = a_{p,long,tran} \\
\dot{s}_{p,long} = v_{p,long} \\
\dot{\omega}_{p,lat} = a_{p,lat,rot} \\
\dot{\theta}_{p,lat} = \omega_p \\
\dot{v}_{p,lat} = a_{p,lat,tran} \\
\dot{s}_{p,lat} = v_{p,lat}
\end{cases}
\tag{4.14}
$$

Longitudinal and lateral subscripts refer to the pitch-surge and sway-roll DOFs respectively. This 4 DOF problem comprises of four control inputs including translational and rotational accelerations acting in both longitudinal and lateral directions:

$$
u(k) = \begin{cases}
a_{p,long,rot} \\
a_{p,long,tran} \\
a_{p,lat,rot} \\
a_{p,lat,tran}
\end{cases}
\tag{4.15}
$$

Thus, the combined system can be represented as follows:

$$
\dot{x}(k) = f(x(k), u(k)) \tag{4.16}
$$

Apart from the state-space model, additional constraints are applied to take the lateral dynamics into consideration. For this, perception thresholds are applied on both pitch and roll movements. For the proposed motion cueing algorithm, a value of 3 and 2.6 deg/s was chosen for pitch and roll tilt rates respectively. The selection of these values was done based on the same reasoning as mentioned before for the 2 DOF algorithm; by using the perception threshold analysis in Appendix B.

The remaining constraints use the same limits except for the platform displacement. As we are now dealing with additional degrees of freedom, the movement of the platform is limited by the limit circle, represented by $\sqrt{s_{p,long}^2 + s_{p,lat}^2} \leq 0.5^2$. However, the MPT toolbox does not support inclusion of nonlinear constraints, thus, the platform displacement limits are applied separately for the explicit controller with a value of $0.35m$; to ensure that the movement of the simulator takes place inside the limit circle. The control input received from the explicit MPC controller acts as an initial guess which is why a lower constrained value can be taken into account; with the implicit controller refining the final solution. The updated constraints used in the problem are as follows:

$$\mathrm{X} = \begin{cases} -3deg/s \leq \omega_{p,long} \leq 3deg/s \\ -2.6deg/s \leq \omega_{p,lat} \leq 2.6deg/s \\ -30deg \leq \theta_{p,long,lat} \leq 30deg \\ -7.2m/s \leq v_{p,long,lat} \leq 7.2m/s \\ -0.35m \leq s_{p,long,lat} \leq 0.35m \\ -9.81m/s^2 \leq a_{p,long,lat,tran} \leq 9.81m/s^2 \end{cases} \tag{4.17}$$

The goal of this motion cueing algorithm is still to track the output specific force. Rotations with respect to the x and y axis are used in deriving the gravitational tilt components as seen in Equation 4.18.

$$g_{tilt} = \begin{cases} g_{long} = g \sin\theta_{p,long} \\ g_{lat} = -g \cos\theta_{p,long} \sin\theta_{p,lat} \end{cases} \tag{4.18}$$

Taking the translational accelerations into account, the final output specific force can be written as follows:

$$y(k) = \begin{cases} f_{spec,long} = a_{p,long,tran} + g \sin\theta_{p,long} \\ f_{spec,lat} = a_{p,lat,tran} - g \cos\theta_{p,long} \sin\theta_{p,lat} \end{cases} \tag{4.19}$$

The cost/objective function used in the motion cueing algorithm consisted of weighted states, output and control inputs. Weights were selected in a similar manner as done for the 2 DOF algorithm and can be seen below:

$$\begin{aligned} &W_{states} = diag([0,0,0,0,0,0,0,0]) \\ &W_{output} = diag([1,1]) \\ &W_{controlinput} = diag([1e-3, 1e-3, 1e-3, 1e-3]) \end{aligned} \tag{4.20}$$

As the problem complexity has increased by adding 2 additional DOFs, a N_p of 2 is taken with a sampling time of $0.25s$. These values are selected to ensure that the same look ahead time of $0.5s$ is taken into account for the motion cueing problem. More information regarding the selection of this particular combination is presented in subsection 5.3.1.

4.2.2. Implicit MPC based Motion Cueing Algorithm

Due to additional DOFs, the implicit MPC algorithm is altered accordingly. For this, the state space model has additional states added in as done for the explicit MPC controller as follows:

$$\dot{x}(k) = \begin{cases} \dot{\omega}_{p,long} = a_{p,long,rot} \\ \dot{\theta}_{p,long} = \omega_{p,long} \\ \dot{v}_{p,long} = a_{p,long,tran} \\ \dot{s}_{p,long} = v_{p,long} \\ \dot{\omega}_{p,lat} = a_{p,lat,rot} \\ \dot{\theta}_{p,lat} = \omega_{p} \\ \dot{v}_{p,lat} = a_{p,lat,tran} \\ \dot{s}_{p,lat} = v_{p,lat} \\ \dot{a}_{p,long,tran} = \dfrac{a_{cmd,long,tran} - a_{p,long,tran}}{T_s} \\ \dot{a}_{p,long,rot} = \dfrac{a_{cmd,long,rot} - a_{p,long,rot}}{T_s} \\ \dot{a}_{p,lat,tran} = \dfrac{a_{cmd,lat,tran} - a_{p,lat,tran}}{T_s} \\ \dot{a}_{p,lat,rot} = \dfrac{a_{cmd,lat,rot} - a_{p,lat,rot}}{T_s} \end{cases} \tag{4.21}$$

As we can now take nonlinear constraints into account, the platform displacement is constrained by ensuring that the overall movement in longitudinal and lateral directions takes place inside the limit circle. Also, braking constraints are again applied to allow the platform velocity and tilt rate to be decreased as the simulator approaches the platform displacement and tilt angle limits. Further, to ensure that the tilt coordination effects are not perceived by the driver, tilt rate constraints are applied along with the other constraints, as done for the explicit algorithm.

Adaptive washout weights are also implemented to introduce washout effect using Equation 4.1.2. A prediction horizon N_p of 50 was selected with a sampling time T_s of $0.01s$ for the controller as done before for the 2 DOF algorithm. The entire formulation was again implemented using `ACADO` and the cueing algorithm was simulated in Simulink.

The structure of the 4 DOF implicit motion cueing algorithm can be seen below:

$$\begin{aligned} \min_{u_{N_p}} \quad & J\,(x_0, u) \\ \text{s.t.,} \quad & x(k+1) = f(x(k), u(k)) \\ & x \in \mathrm{X}_{iMPC,4DOF} \\ & x(N) \in \mathbb{X}_f \end{aligned} \tag{4.22}$$

4.3. Summary

In this chapter, the design of the explicit and implicit controllers used in the hybrid scheme was presented. The 2 DOF motion cueing algorithm was first elucidated which took the surge-pitch DOFs as part of the problem formulation. Washout weights and platform constraints used in the algorithm were explained. Then, the algorithm was extended for sway-roll DOFs, making it a 4 DOF cueing problem. Additional functionalities to take lateral dynamics into account, were presented for the design of the 4 DOF algorithm. In Chapter 5, the performance of the developed hybrid MPC based motion cueing algorithms is studied. The set of test cases and key performance indicators considered are listed, followed by results related to the 2 DOF and 4 DOF cueing algorithms. Further, the same set of four cueing models are used in evaluating the performance of the designed hybrid MPC controller as explained in Chapter 3.

5 Simplified Actuator Dynamics

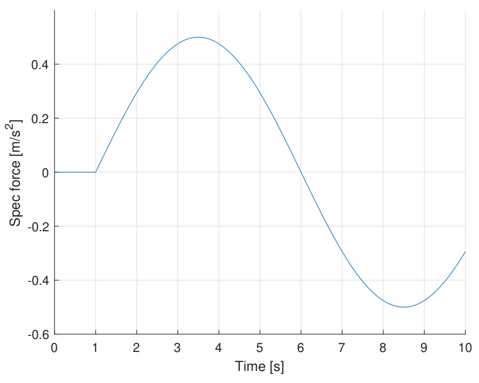
To see the performance of the designed motion cueing algorithms, simulations were performed using two kinds of simulator dynamics. In this chapter the simplified version is presented which is implemented in Simulink. This dynamics block is a representation of the state space model defined in the implicit MPC based motion cueing algorithm and can be seen in Figure D.2, for the 2 DOF motion cueing scenario.

5.1. Test Cases

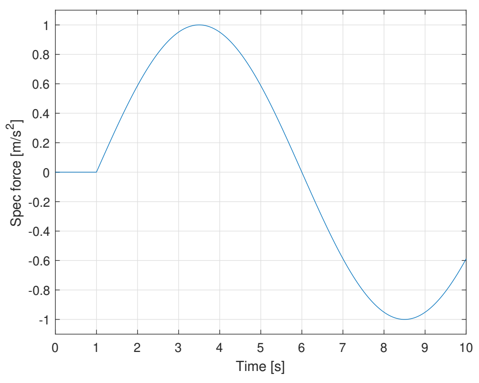
A set of reference signals were considered for evaluating the developed motion cueing algorithm. These signals allow a wide range of different scenarios to be taken into account that cover a reasonable range of operating conditions that the simulator might experience. Some of these signals have a high amplitude which are used to observe the performance of the simulator close to or over its potential. Further, frequencies are considered within the normal head movement range of 0.01 to 1 Hz [37], [48]. They are summarised in Table 5.1 and can be visualized in Figure 5.1.

Table 5.1: Reference signals used for simulations

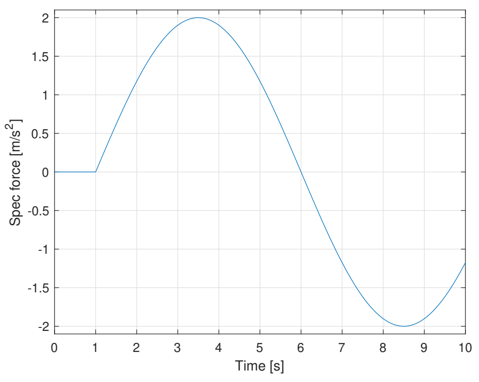
Type of reference signal	Amplitude $[\mathrm{m}/s^2]$	Frequency [Hz.]	Simulation time [s]	Scaling factor
Sine wave	0.5	0.1	10	1
	1	0.1	10	1
	2	0.1	10	1
	Combination of 4 sine waves: 1 0.8 0.1 0.6	Combination of 4 sine waves: 0.1 0.15 0.2 0.5	10	0.3
Step signal (from 1-4 seconds)	1	0.1	10	1
	2	0.1	10	1
Step (from 1-4 seconds) + sine wave (10-20 seconds)	0.5	0.1	20	1
	1	0.1	20	1



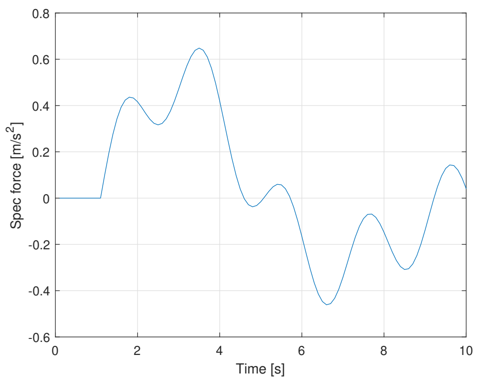
(a) Sine wave of amplitude 0.5



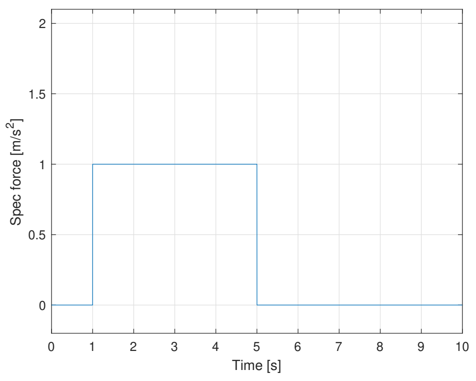
(b) Sine wave of amplitude 1



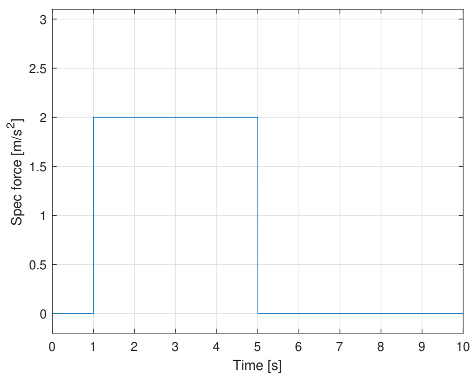
(c) Sine wave of amplitude 2



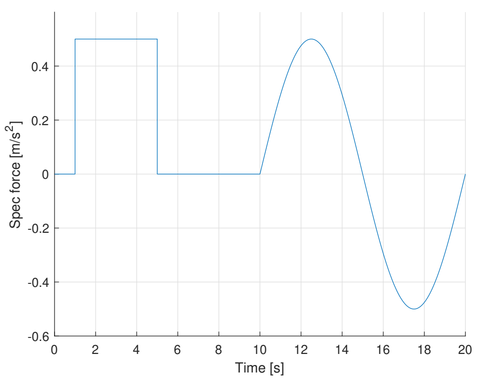
(d) Combined sine wave



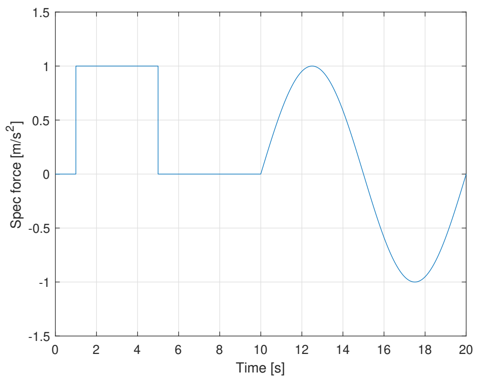
(e) Step signal of amplitude 1



(f) Step signal of amplitude 2



(g) Multiple event wave of amplitude 0.5



(h) Multiple event wave of amplitude 1

Figure 5.1: Reference signals used for simulations

The combined signal shown in Figure 5.1d is a combination of four sine waves as seen below:

$$sine_{combined} = k_{scale}(wave1 + wave2 + wave3 + wave4) \tag{5.1}$$

A scaling factor k_{scale} was chosen for this wave as the simulator is incapable of recreating the magnitudes of the original signal. By using a factor of 0.3, the overall amplitude and rate of change of the specific force is reduced. This results in a signal that can be tracked by the cueing algorithm throughout the simulation, while keeping the structure of the large amplitude wave. Also, for all these signals, the 1^{st} second was considered as the rest period to allow smooth transitions to occur from neutral position.

To perform a comprehensive comparison for each reference signal with different MPC based controllers, three performance indicators were used to compare the reference tracking and computation time performance as follows:

- **Mean Iterations** are used to check the number of iterations the ACADO solver takes to compute the control input throughout the simulation. This gives a good idea on how different MPC based controllers vary with each other. Lower number of mean iterations are preferred as they lead to reduced overall online computation time with lower load on the system.
- **Absolute Percentage Error** is used to see the percentage difference in mean iterations amongst the developed hybrid models when compared with the implicit MPC based cueing algorithm without any initial guess. It helps in recognising which models are performing best and whether a significant improvement is present on average.
- **Root Mean Square Error (RMSE)** was used to compute the differences between the values observed from the motion cueing algorithm and the reference signal being tracking. If this indicator is close to zero then the performance of the algorithm is very good. It is computed as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(x_{ref,i} - x_i\right)^2}{N}} \tag{5.2}$$

5.2. 2 DOF Simulations

5.2.1. Initial Explicit MPC Performance

Using the modelled controller in subsection 4.1.1, the explicit control law was generated and used to simulate different test scenarios as mentioned in the previous section. The results obtained for a simple sine wave of amplitude 2 can be seen in Figure 5.2. In this and the subsequent figures present in this report, three subplots are used to depict the performance of the motion cueing algorithm. The first subplot showcases the reference and actual specific forces along with its two components. The second plot shows the tilt angle and tilt rate during the cueing process. Lastly, the translational platform displacement and velocity is observed in the third subplot.

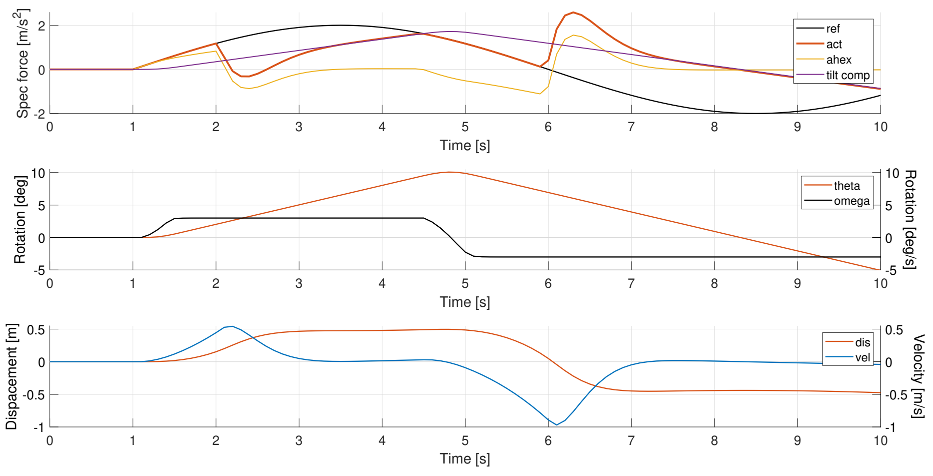


Figure 5.2: Initial explicit MPC motion cueing performance for sine wave of amplitude 2

From the results it can be seen that the algorithm is able to start tracking the reference sine wave relatively well before failing shortly afterwards at $2s$. This failure occurs at two regions where a peak and drop is observed respectively. It takes place when the algorithm reaches its tilt rate threshold or platform displacement limit. At this moment, the controller tries to continue tracking the signal but fails due to limited available workspace. Such a behaviour is expected as the amplitude is relatively high and the driving simulator cannot recreate those high magnitudes. Thus, at those points the algorithm fails, after which it eventually recovers and starts tracking again. This behaviour is seen for other signals with low amplitude as well. At these values, the width of the peak and drop is reduced but the same phenomenon is observed.

5.2.2. Effect of Move Blocking Constraint

Move blocking is a functionality of the MPT toolbox that constrains the predicted control moves over the horizon to reduce the number of decision variables in the optimization problem. Thus, it acts as a control horizon and is particularly useful for explicit MPC problems. This is because, with an increase in complexity of the problem, the computation time needed to compute the control laws increases exponentially. By reducing the number of decision variables, a significant decrease in overall computation time is observed.

The difference in computation time and RMSE tracking performance for the developed 2 DOF algorithm can be seen in Table 5.2. It can be observed that when move blocking is not applied, the computation time increases exponentially with a small increase in N_p. Even though the tracking performance improves due to an increase in look ahead time, the computation expense limits the use of high N_p values with the same sampling rate of $0.1s$. A value of 10 is not computable as the number of control regions to be computed continue to increase even after 2 days of computation time. On the other hand, by using the move blocking functionality (from the second prediction step), faster solutions are generated with a small compromise in RMSE performance. Further, the number of control regions also reduce drastically; requiring less memory space to store the control law. Larger N_p can also be considered for the 2 DOF scenario using this functionality (at this sampling rate), but to maintain the same look ahead time in both explicit and implicit MPCs, a N_p of 5 was used. Moreover, using faster sampling rates with larger N_p (while maintaining the same look ahead time) was not possible even with this functionality. This was due to the exponential increase in computation time resulting in uncomputable solutions.

Table 5.2: Influence of move blocking with different N_p values

Type	Prediction Horizon	Control Regions	Computation Time	RMSE (f_{spec})
Without move blocking	5	11131	~ 11 min	1.3358
	7	47100	~ 90 min	1.2912
	10	130000 >	2 days >	-
With move blocking	5	938	~ 34 s	1.5815
	7	1190	~ 90 s	1.4957
	10	1755	~ 3.1 min	1.4086
	12	2443	~ 6.7 min	1.3432
	15	3663	~ 18 min	1.2949
	17	4740	~ 26.5 min	1.2721
	20	6308	~ 61 min	1.2389

Finally, using this constraint, the explicit MPC controller was updated and the performance can be seen in Figure 5.3. Here, the move blocking effect takes place from the second prediction step, as mentioned before, which has a positive effect on the computation time with a small compromise in the RMSE values. The motion cueing algorithm performs with similar characteristics as seen in Figure 5.2.

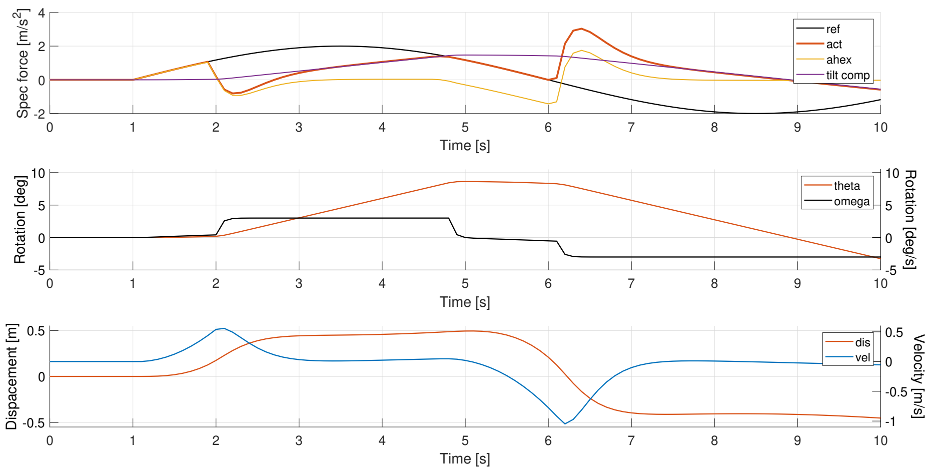


Figure 5.3: Explicit MPC motion cueing performance with move blocking constraint for sine wave of amplitude 2

5.2.3. Effect of Braking Constraints

As mentioned before in subsection 4.1.1, braking constraints are used to reduce the platform velocity and tilt rate as the simulator approaches its physical limits which has a positive effect on the performance of the motion cueing algorithm. This is because the driving simulator now has greater available workspace to perform the desired simulation with more maneuverability. Figure 5.4 shows the effect on the same simulation involving a sine wave of amplitude 2. It can now be observed that the algorithm is able to smoothen out the peaks and drops that were observed before. The effect of using these constraints on the platform state values can be seen in Figure 5.5. As the platform displacement reaches its limits, the velocity starts to reduce and approach zero allowing braking action to take place. Due to this, the limits are never reached resulting in improved motion capabilities to track the reference signal. On the other hand, when a similar action took place before applying these constraints, the platform displacement continued to stay at its limit even with a reduction in velocity. This was because the states were not constrained being dependent on each other which resulted in those peaks and drops as seen in Figure 5.2.

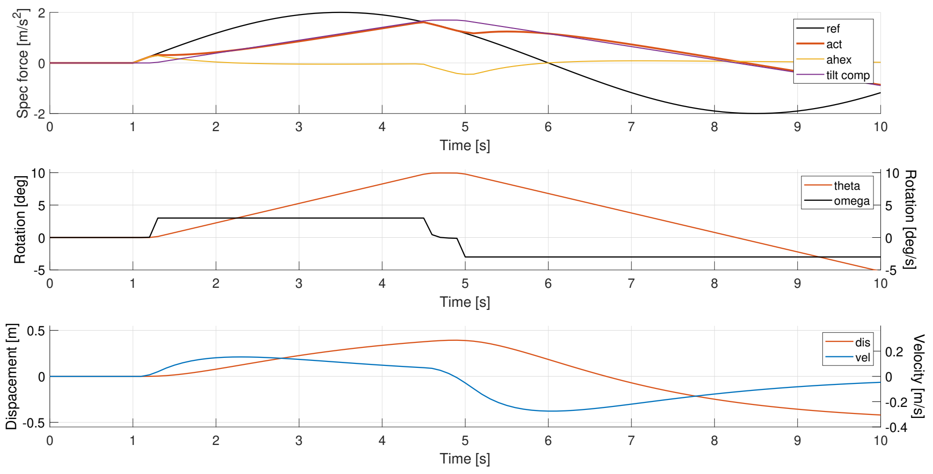


Figure 5.4: Explicit MPC motion cueing performance with braking constraint for sine wave of amplitude 2

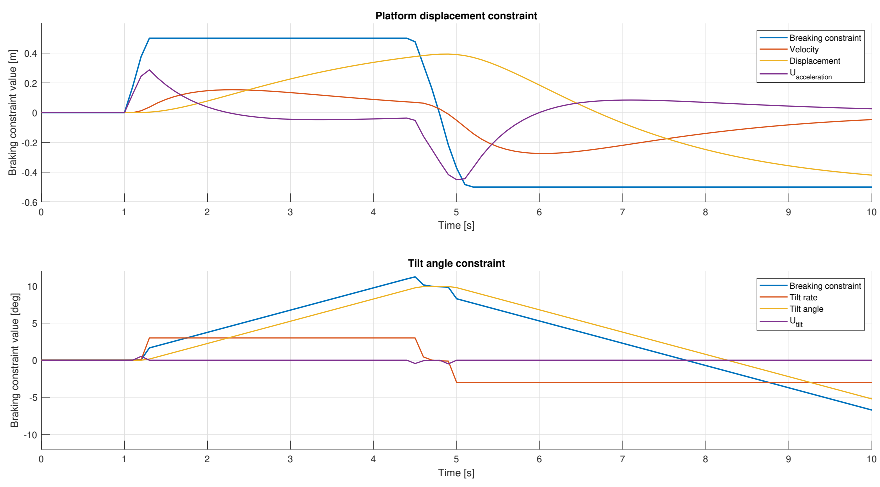


Figure 5.5: Braking constraint plot

Such a performance, with false cues is expected for a signal with a high rate of change in a short time span, having an amplitude of 2. On the other hand, at a lower amplitude of 0.5, improved tracking can be observed as seen in Figure 5.6. At this amplitude, the cueing algorithm is able to track the reference sine wave for the entire duration except at one position where a false cue occurs (at $5.5s$). This false cue makes the current performance of the algorithm undesirable which is why some additional weighting is needed. These adaptive parameters cannot be added in the explicit controller due to limitations of MPT. Further, the T_s and N_p cannot be increased to improve tracking, due to the exponential increase in control law computation time. Thus, the explicit solution is only used as an initial guess in this study.

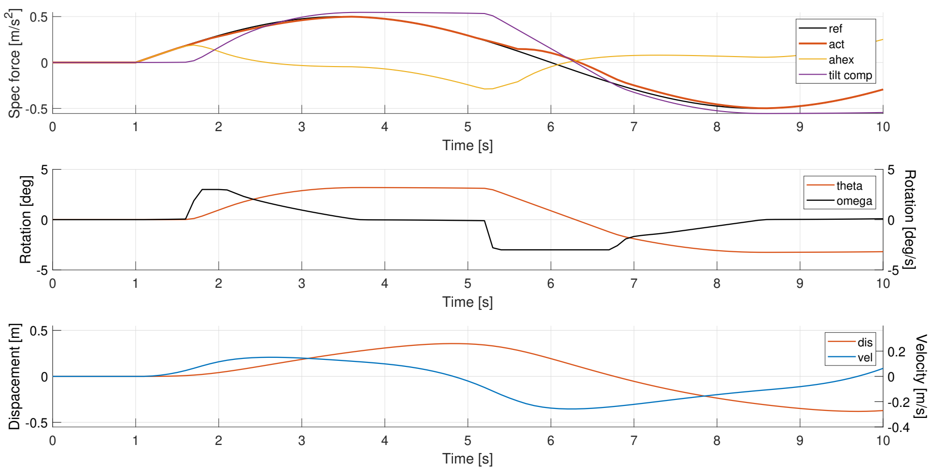


Figure 5.6: Explicit MPC motion cueing performance with braking constraint for sine of amplitude 0.5

Additional parametric studies with the explicit MPC are present in Appendix C.

5.2.4. Hybrid MPC Performance

The defined hybrid model in section 4.1 is tested across different conditions to see if there is any improvement in overall online computation time while maintaining similar performance. This involves tests with constant and adaptive weight functionalities as defined before for all the test cases present.

Constant Washout Weights

Constant washout weights are added on the tilt rate and platform displacement. The effect of using washout in the proposed model can be seen in Figure 5.7. It can be observed that any false cues which were present before have now gone away with very good tracking throughout the signal. This is because, using these weights, the algorithm is able to maneuver the simulator states to track the signal while keeping them within the desired limits and close to neutral position.

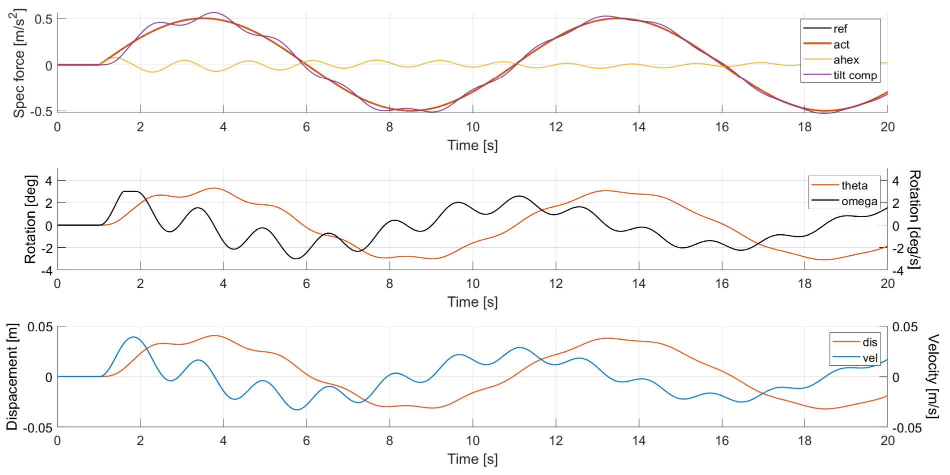


Figure 5.7: Motion cueing performance with braking constraints and washout weights for sine wave of amplitude 0.5

The washout weights used for this sine wave of amplitude 0.5 were 0.001 for tilt rate and 0.9 for the platform displacement respectively. In a similar manner, weights can be tuned for all the test cases taken into account separately. These are summarised in Table 5.3.

Table 5.3: Tuned washout weights for all test scenarios

Type of reference signal	Amplitude of reference signal	Best washout weight		Number of false cues	RMSE	
		Tilt rate	Platform displacement		Specific force	Platform displacement
Sine wave	0.5	0.001	0.9	0	0.0024	0.0242
	1	1.5	20	0	0.0473	0.1035
	2	0.01	30	2	1.1459	0.2464
	Combined	0.5	7	1	0.0742	0.0956
Step signal (from 1-4 seconds)	0.5	0.6	0.3	1	0.0436	0.0724
	1	0.3	1	2	0.2536	0.1625
	2	0.5	10	2	0.9871	0.2563
Step+Sine	0.5	0.8	0.6	2	0.0341	0.1016
	1	1.5	20	2	0.2154	0.1544

From the table, it can be inferred that at a low amplitude of 0.5, the cueing algorithm is able to perform good tracking as seen from the specific force and platform displacement RMSE values. As the amplitude increases to 1, specific force tracking slightly worsens but the RMSE still remains less than 0.05 for the sine wave and less than 0.3 for the step signal. This increase in value for the step signal is expected due to the sharp rise of the signal to be tracked; resulting in small false cues at the start and exit of the step signal. At even higher amplitudes of 2, the tracking starts to worsen resulting in false cues of greater magnitude. This behaviour, at such large values, is due to the limitations in the available workspace for the cueing algorithm. Further, when multiple event scenarios are simulated, constant weights work well for some parts of the simulation as it can only be tuned for a particular type of signal. Hence, a compromise is made in tracking performance resulting in more false cues as compared to the signals acting on their own.

Adaptive Washout Weights
Although constant washout weights work well and give good results, they suffer from issues when the signal type or amplitude is changed. This is because the performance of a constant washout weight is highly sensitive to the signal being used. The effect of using adaptive weights defined in Equation 4.1.2, can be seen in Table 5.4. It is observed that adaptive weights can improve performance by reducing the number of false cues generated across different scenarios, compared to using the same constant washout weight for all the test cases. In this table, 0.001 and 0.9 were the chosen constant weights for the tilt rate ω_p and platform displacement s_p respectively. Moreover, the number of false cues generated while using the adaptive weights show a similar to trend to the performance observed in Table 5.3. From this comparison it is evident, that the adaptive weights prove to be superior as compared to using the same washout weight for all the scenarios while saving tuning time and reducing false cues.

Table 5.4: Comparison between constant and adaptive washout weights in terms of false cues generated

Type of reference signal	Amplitude of reference signal	Constant washout weights			Adaptive washout weights		
		RMSE		Number of false cues	RMSE		Number of false cues
		Specific force	Platform displacement		Specific force	Platform displacement	
Sine wave	0.5	0.0024	0.0242	0	0.0030	0.0572	0
	1	0.1518	0.2313	2	0.1280	0.2018	0
	2	1.1797	0.2513	2	1.1432	0.2496	2
	Combined	0.0813	0.1162	2	0.0633	0.1280	1
Step signal (from 1-4 seconds)	0.5	0.0664	0.1029	2	0.0467	0.0915	1
	1	0.2648	0.1724	2	0.2943	0.1832	2
	2	0.9871	0.2565	2	0.9984	0.2961	2
Step+Sine	0.5	0.0470	0.0862	2	0.0331	0.0696	1
	1	0.2381	0.2130	4	0.2102	0.1997	2

An example simulation in Figure 5.8 shows multiple events taking place in the same simulation (step signal and a sine wave). It can be seen that the algorithm is able to track the reference signal while maintaining washout effect. It is able to adapt to the signal requirements without any compromise in specific force tracking. Similar performance can be observed when these signals act separately, whose results are present in Appendix C.

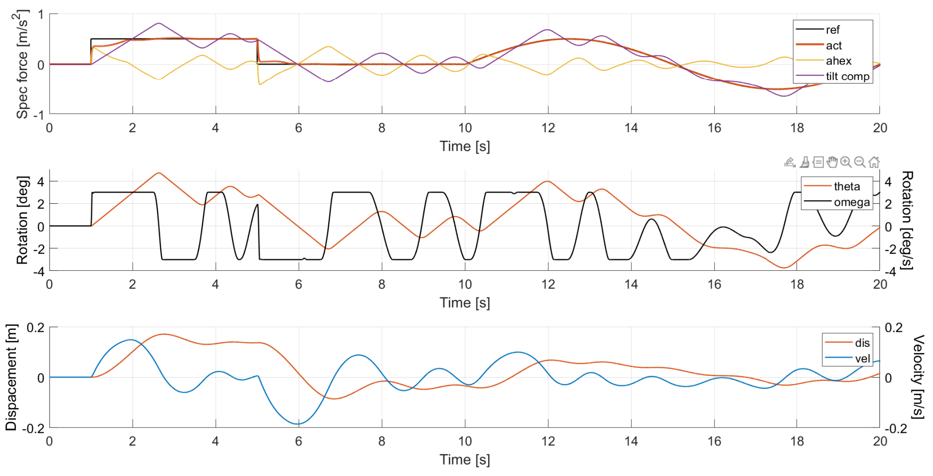


Figure 5.8: Motion cueing performance for multiple event signal with adaptive weights

Finally, using these adaptive weights, the model is updated and the same set of simulations are run for all the defined motion cueing models. The results in Table 5.5 show that the hybrid model with all explicit MPC control inputs, performs fastest with reduced mean iterations across all test cases analysed. The hybrid model with the first control input sent in gives second lowest mean iteration values with the implicit controller (without trajectory prediction) giving worst performance. The amount of mean iterations vary from one signal to another and there is no general trend that can be stated for the signals analysed.

From Table 5.6, it can be inferred that the average percentage improvement in computation time is around 35% and 20% for the hybrid controllers (with all and with first control inputs). On the other hand, the implicit controller (with control trajectory prediction) shows 15% improvement when compared with the implicit controller. Upon looking at this table in more detail, it can be seen that the amount of improvement in mean iterations goes down as the amplitude of the signal starts to increase. This can be due to the fact that at lower amplitude, the explicit guess is closer to the optimal control input whereas at larger amplitudes it is farther away. Also, the rate of drop for the hybrid controller with all inputs is less than the other two controllers because it is able utilise more information from the explicit MPC.

Table 5.5: Mean iteration values for all four motion cueing models with adaptive washout weights

Type of reference signal	**Amplitude of reference signal**	**Sampling time [s]**	**Mean iterations**			
			iMPC	**iMPC (with control trajectory prediction)**	**Hybrid (with first eMPC control input)**	**Hybrid (with all eMPC control inputs)**
Sine wave	0.5	iMPC = 0.01 eMPC = 0.1	1.0829	0.8771	0.7433	0.4625
	1		2.0290	1.7982	1.6214	1.4386
	2		1.4755	1.1628	1.1928	0.8771
	Combined		4.1778	3.6703	3.5754	3.2198
Step signal (from 1-4 seconds)	1		3.7862	3.2937	3.2657	2.4016
	2		1.9141	1.8092	1.7562	1.5904
Step+Sine	0.5		3.1289	2.0385	1.8006	1.4298
	1		2.7911	2.5977	2.3718	2.0225

Table 5.6: Absolute error values with iMPC for all hybrid models with adaptive washout weights

Type of reference signal	**Amplitude of reference signal**	**Sampling time [s]**	**Absolute Error % (with iMPC)**		
			iMPC (with control trajectory prediction)	**Hybrid (with first eMPC control input)**	**Hybrid (with all eMPC control inputs)**
Sine wave	0.5	iMPC = 0.01 eMPC = 0.1	19.00	31.36	57.29
	1		11.37	20.09	29.10
	2		21.19	19.16	40.55
	Combined		12.15	14.42	22.93
Step signal (from 1-4 seconds)	1		13.01	13.75	36.57
	2		5.48	8.25	16.91
Step+Sine	0.5		34.85	42.45	54.30
	1		6.93	15.02	27.54
Average Error (%)			15.50	20.56	35.65

Table 5.7 shows that the tracking performance is consistent across all motion cueing models in the respective test scenarios, with no model outperforming the other. A similar trend is observed with near perfect tracking at low amplitudes and worst performance at large values due to simulator workspace limitations. This means that the explicit MPC helps in reducing online computation load while maintaining tracking performance across all test cases. A similar study for the best constant washout weights is presented in Appendix C.

Table 5.7: Specific force RMSE values for all four motion cueing models with adaptive washout weights

Type of reference signal	**Amplitude of reference signal**	**Sampling time [s]**	**RMSE (f_{spec})**			
			iMPC	**iMPC (with control trajectory prediction)**	**Hybrid (with first eMPC control input)**	**Hybrid (with all eMPC control inputs)**
Sine wave	0.5	iMPC = 0.01 eMPC = 0.1	0.0030	0.0034	0.0044	0.0048
	1		0.1284	0.1381	0.1423	0.1409
	2		1.1432	1.1435	1.1439	1.1429
	Combined		0.0633	0.0631	0.0628	0.0561
Step signal (from 1-4 seconds)	1		0.2943	0.2963	0.2975	0.2911
	2		0.9884	0.9884	0.9878	0.9886
Step+Sine	0.5		0.0331	0.0322	0.0320	0.0345
	1		0.2102	0.2165	0.2220	0.2266

5.3. 4 DOF Simulations

Simulation studies were carried out to test the performance of the created 4 DOF motion cueing algorithm. The same set of test cases were taken into account as described before. First, the explicit MPC which acts as the initial guess was tested followed by the hybrid algorithm.

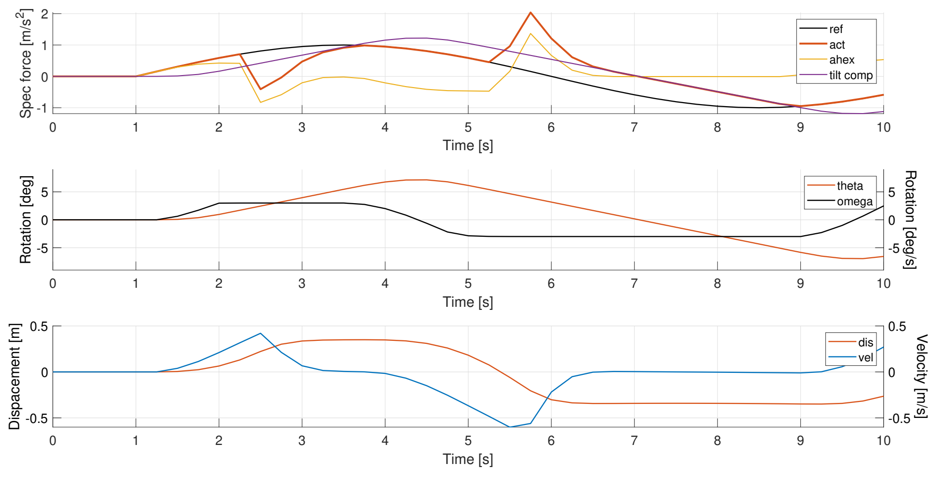
5.3.1. Explicit MPC Performance

The created explicit controller was tested by looking at the general tracking performance of the algorithm. Due to computation time limitations, a lower prediction horizon of 2 was chosen compared to the higher value of 5 taken for the 2 DOF case. This is because, with an increase in problem complexity, the computation time needed to generate the control laws increases exponentially which is the main drawback of using an explicit controller. A summary of difference in computation time is seen in Table 5.8. Here, it can be observed that with a N_p of 5 and N_c of 2, a solution was not computable as the number of computed control regions continued to increase at a fast rate even after 2 days of simulation time, with no scope of slowing down. Hence, such a prediction value is impractical and cannot be implemented for such a large scale cueing problem. To ensure that the look ahead time remains the same as the implicit controller, the sampling time was modified to 0.25 seconds with a N_p of 2. As done for the previous algorithm, the same look ahead time is preferred for both controllers, as it allows the initial guess to be provided for the same duration that the implicit controller is considering for computing the optimized control input.

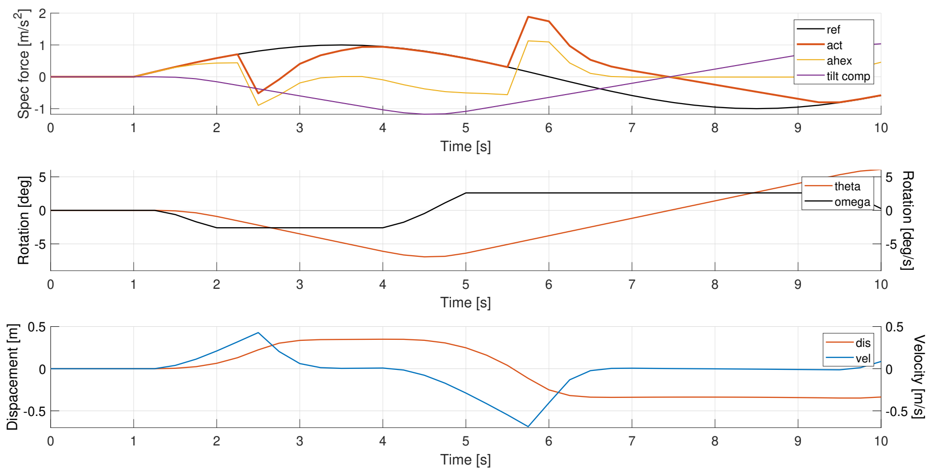
Table 5.8: Computation time difference with changing prediction horizon and sampling time

Prediction horizon	**Sampling time [s]**	**Number of control regions**	**Computation time**
2	0.25	20476	45 min
5 with move blocking from 2nd step	0.1	90000 >	2/3 days >

Simulations were run using the explicit controller and the tracking performance for a sine wave in both longitudinal and lateral direction can be seen in Figure 5.9. It can be observed that the controller is able to track the reference signal in a similar manner as expected when no washout weights or braking constraints are acting on the model. The same peaks and drops are observed when simulator states reach their saturation points. Apart from the tracking, the simulator is able to stay within its operating range while ensuring that the individual pitch and roll perception thresholds are not violated. But, due to limitations in improving overall performance as observed for the 2 DOF scenario, the explicit solution is only used as the initial guess for the 4 DOF study.



(a) Longitudinal motion



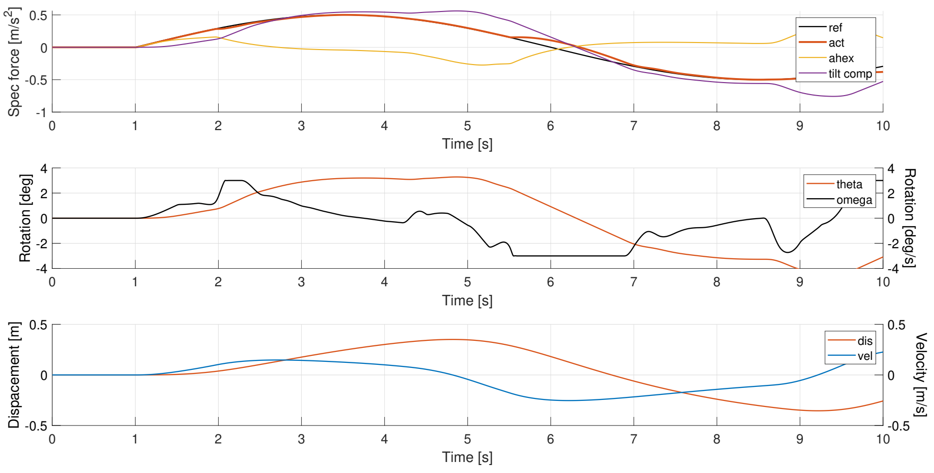
(b) Lateral motion

Figure 5.9: Sine wave of amplitude 1 and frequency 0.1

5.3.2. Hybrid MPC Performance

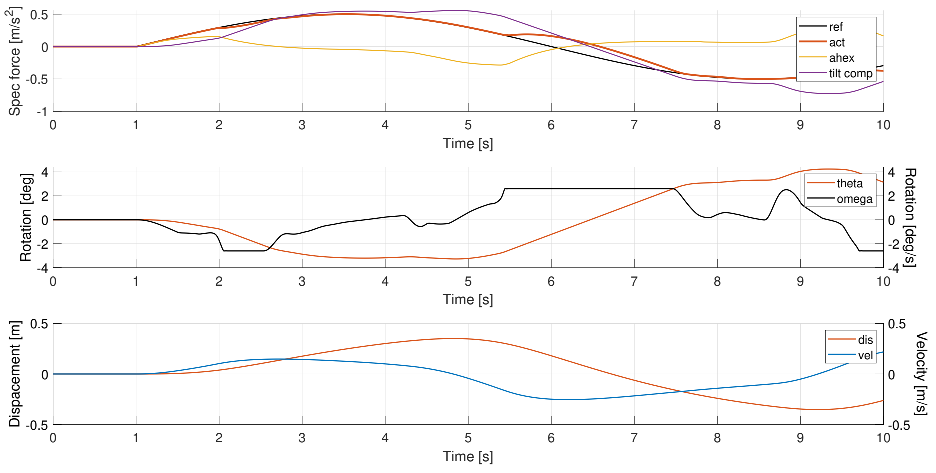
Once the explicit solution has been formulated and stored in the form of a look-up table, it is used as the initial guess for the implicit MPC. The algorithm derived and explained in subsection 4.2.2 is used to perform the online simulations. All three cueing models with an initial guess are analysed and compared with the implicit MPC based motion cueing algorithm using the same performance indicators.

The first set of simulations were performed without any adaptive washout weights acting to see how the general version of the algorithm behaves. This was done by analysing the tracking performance of a sine wave of amplitude 0.5; acting in both longitudinal and lateral directions. The algorithm is able to track the reference signal quite well due to the addition of braking constraints in the model, but suffers from a false cue starting at $5.5s$ as observed in Figure 5.10. This false cue can be attributed to the tilt rate limit being reached as seen in the subplots; resulting in a drop of tracking performance due to insufficient workspace available to perform the desired motion. Further, such an action takes place because there are no washout weights acting in the algorithm which help in bringing the simulator back to its neutral position.



(a) Longitudinal motion

Figure 5.10: Sine wave of amplitude 0.5 and frequency 0.1

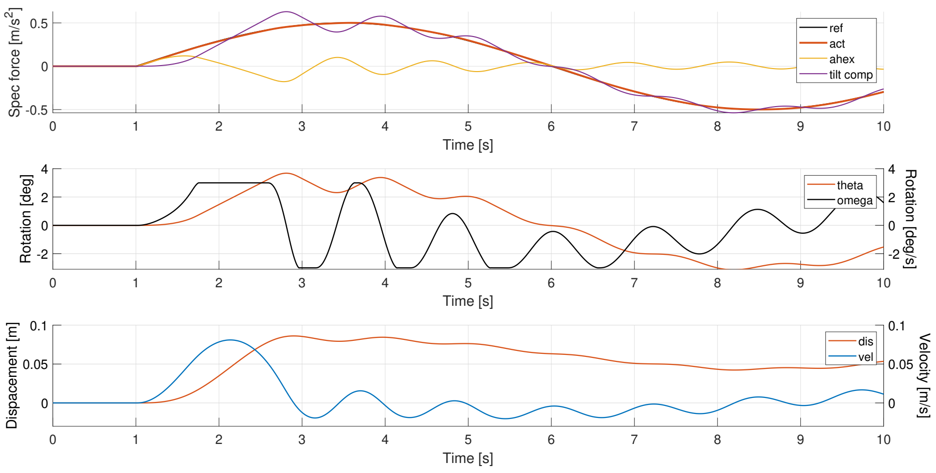


(b) Lateral motion

Figure 5.10: Sine wave of amplitude 0.5 and frequency 0.1 (contd.)

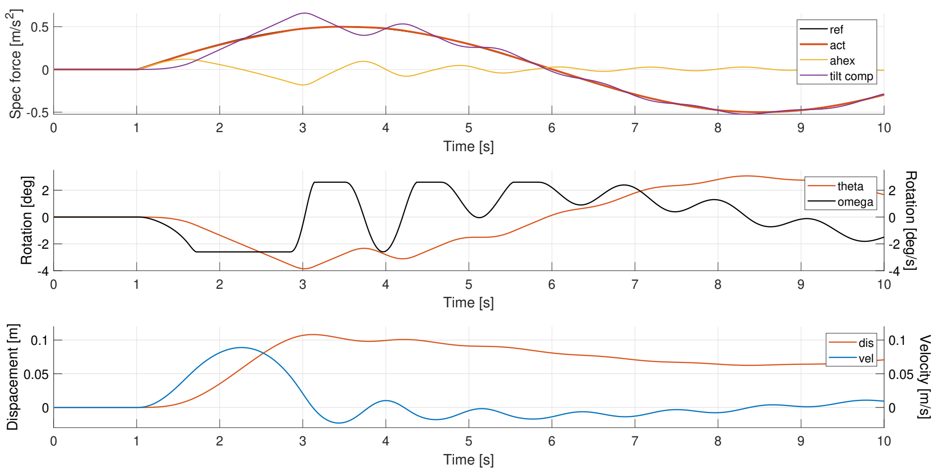
Adaptive Washout Weights

To improve tracking performance, adaptive washout weights are included in the algorithm. The formulation of these weights takes place in the same manner as explained in Equation 4.1.2. From the results in Figure 5.11 and Figure 5.12, showing the performance of a sine wave and a multiple event signal, it can be seen that using these weights as part of the motion cueing algorithm has a positive influence on the overall tracking performance. Even though in some parts of the simulation, the tilt rate reaches its saturation point, the adaptive weights allow it to go back to its neutral position while maintaining the simulator platform close to its neutral position as well. Thus, false cues which were observed before in those scenarios, where tilt rate limits were reached, are no longer present. Further, the algorithm is also able to adapt to varying signals within the same simulation as seen with the 2 DOF algorithm.



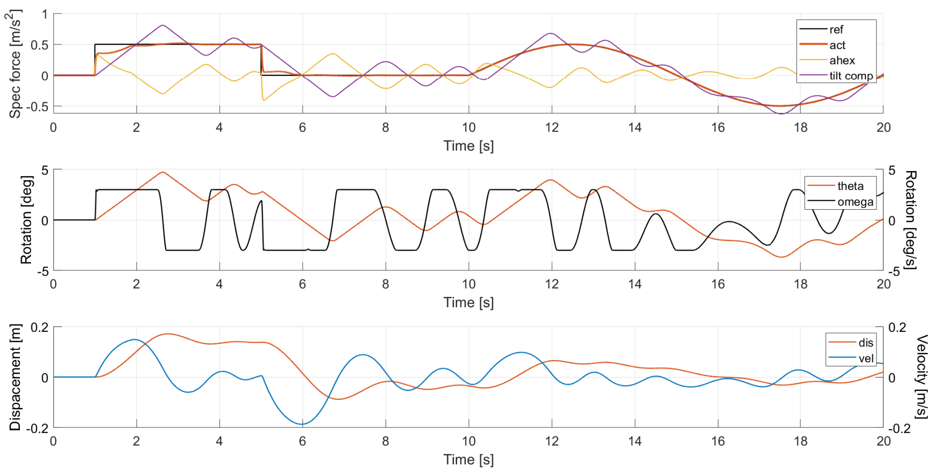
(a) Longitudinal motion

Figure 5.11: Sine wave of amplitude 0.5 and frequency 0.1



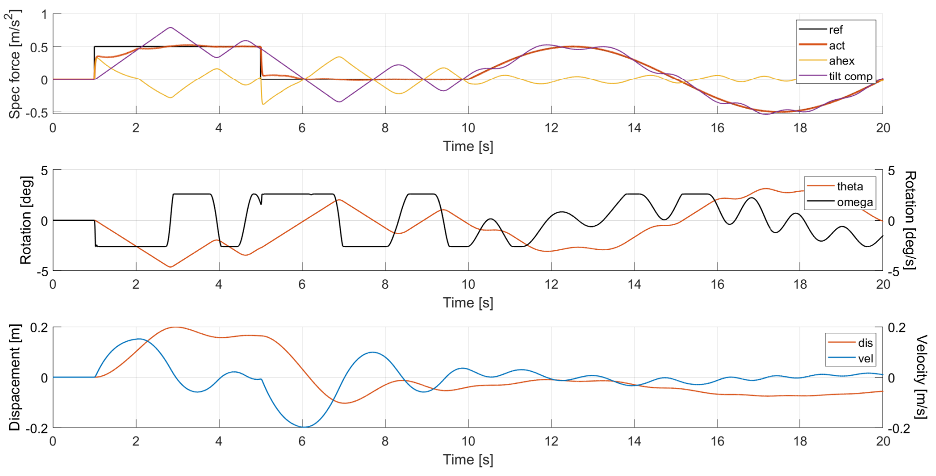
(b) Lateral motion

Figure 5.11: Sine wave of amplitude 0.5 and frequency 0.1 (contd.)



(a) Longitudinal motion

Figure 5.12: Multiple event wave



(b) Lateral motion

Figure 5.12: Multiple event wave (contd.)

As the updated algorithm gave good tracking performance, the online computation performance was also investigated, as done before for the 2 DOF simulations. Mean iterations observed in Table 5.9, for all models analysed, show that the hybrid model with all control inputs taken from the explicit controller gives fastest solutions. This controller reduces the online solving load by 30%, which is observed from the absolute error values shown in Table 5.10. The hybrid controller with first control input gives similar computation performance to the implicit controller with control trajectory prediction, with an improvement of around 10%. This could be because of only using the first control input resulting in limited information. This can also be caused due to a larger sampling rate difference amongst the explicit and implicit controllers in the 4 DOF hybrid algorithm due to explicit MPC limitations. Further, both of these cueing models show significant improvement at low amplitudes as seen for the sine wave and multiple event scenario. As the amplitude increases, the percentage improvement starts to drop. On the other hand, the hybrid controller with all control inputs shows improvement in all scenarios with a minor drop at high amplitude when compared to the other models. This can again be attributed to the quality and quantity of the control inputs utilised by the implicit controller. With more information, the implicit controller is able to provide the optimized control input at a faster and more uniform rate.

Table 5.9: Mean iteration values for all four motion cueing models with adaptive washout weights

Type of reference signal	**Amplitude of reference signal**	**Sampling time [s]**	**Mean iterations**			
			iMPC	**iMPC (with control trajectory prediction)**	**Hybrid (with first eMPC control input)**	**Hybrid (with all eMPC control inputs)**
Sine wave	0.5	iMPC = 0.01 eMPC = 0.25	2.4346	1.6494	1.6803	1.5075
	1		3.8611	4.0400	3.8082	2.4076
	Combined		8.6683	8.0699	7.8082	5.7612
Step signal (from 1-4 seconds)	1		7.0270	6.6623	6.9121	4.6024
	2		3.1578	3.0849	3.0180	2.4835
Step+Sine	0.5		5.3798	4.2339	4.0215	3.0920
	1		5.1864	5.1109	5.1634	3.8336

Table 5.10: Absolute error values with iMPC for all hybrid models with adaptive washout weights

Type of reference signal	Amplitude of reference signal	Sampling time [s]	Absolute Error % (with iMPC)		
			iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
Sine wave	0.5	iMPC = 0.01 eMPC = 0.25	32.25	30.98	38.08
	1		4.63	1.37	37.64
	Combined		6.90	9.92	33.53
Step signal (from 1-4 seconds)	1		5.19	1.63	34.50
	2		2.31	4.42	21.35
Step+Sine	0.5		21.30	25.25	42.52
	1		1.45	0.44	26.08
Average Error (%)			10.58	10.57	33.39

Lastly, the computational load decreases while making sure that the performance of the algorithm does not hinder, as seen in the RMSE values for all the models in Table 5.11 and Table 5.12. Similar trend is observed with very good tracking at low amplitude followed by a rise in RMSE as the amplitude of the signal increases. No particular cueing model outperforms the rest across all scenarios in both longitudinal and lateral directions. Further, the performance is similar in both directions. This is because the cost function allows equal contribution from both kinds of motions resulting in similar results.

Table 5.11: Specific force RMSE values (longitudinal motion) for all four motion cueing models with adaptive washout weights

Type of reference signal	Amplitude of reference signal	Sampling time [s]	RMSE (longitudinal)			
			iMPC	iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
Sine wave	0.5	iMPC = 0.01 eMPC = 0.25	0.0029	0.0029	0.0029	0.0029
	1		0.0702	0.0708	0.0709	0.0711
	Combined		0.0634	0.0613	0.0612	0.0621
Step signal (from 1-4 seconds)	1		0.2925	0.2925	0.2926	0.2932
	2		1.0115	1.0115	1.0123	1.0130
Step+Sine	0.5		0.0334	0.0332	0.0333	0.0327
	1		0.2554	0.2493	0.2486	0.2109

Table 5.12: Specific force RMSE values (lateral motion) for all four motion cueing models with adaptive washout weights

Type of reference signal	Amplitude of reference signal	Sampling time [s]	RMSE (lateral)			
			iMPC	iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
Sine wave	0.5	iMPC = 0.01 eMPC = 0.25	0.0034	0.0035	0.0035	0.0034
	1		0.2319	0.2320	0.2318	0.2320
	Combined		0.0690	0.0643	0.0643	0.0654
Step signal (from 1-4 seconds)	1		0.3243	0.3237	0.3237	0.3234
	2		0.9684	0.9684	0.9687	0.9696
Step+Sine	0.5		0.0380	0.0386	0.0385	0.0389
	1		0.3386	0.3386	0.3387	0.3380

5.4. Summary

In this chapter simulations were performed using simplified actuator dynamics for the 2 DOF and 4 DOF motion cueing algorithms respectively. A set of test scenarios were defined along with key performance indicators to analyse the results in detail. Simulations were first done with the explicit controller to analyse the simplified version of the implicit algorithm. Peaks and drops were observed in the simulations due to limitations in the available workspace. As additional functionalities, such as adaptive weights, used to improve the cueing performance cannot be included in the explicit algorithm, it was only used as an initial guess in this study. Apart from this, effect of move blocking was also analysed which helps in reducing control law computation time with a compromise in specific force tracking. From this analysis, the final N_p and sampling time values were selected for both 2 DOF and 4 DOF explicit controllers.

Next, the implicit controller used in the hybrid scheme was studied. To reduce peaks and drops obtained with the explicit controller, braking constraints were applied which helped in improving tracking performance for most of the simulation. Minor false cues were still obtained due to simulator limits of the tilt rate and platform displacement being reached. For this, washout weights were used in the algorithm to reduce false cues throughout the simulation. Here, constant washout weights were first studied followed by a comparison with adaptive weights. It was observed that by using the adaptive function, tuning time is saved with good tracking performance for the simulations performed; when compared with constant weights.

Once the individual controllers were analysed, the computation performance of the hybrid scheme was examined. This was done by using the four defined motion cueing models simulated for the same set of test scenarios. For both 2 DOF and 4 DOF cueing algorithms, the hybrid controller with all control inputs sent in as input gave best results with least mean iteration values when compared with the other three cueing models. The implicit controller gave highest mean iteration values across all test cases. Further, specific force tracking performance was also investigated for all models to check whether tracking is maintained while using the initial guess. The results showed that similar performance is obtained from the hybrid models for both 2 DOF and 4 DOF algorithms.

To conclude, it was observed that the developed hybrid scheme is able to reduce online computation costs while ensuring similar specific force tracking performance for both algorithms. Next, simulations using the emulator interface of the DAVSi will be shown along with virtual race track simulations with the 4 DOF motion cueing algorithm.

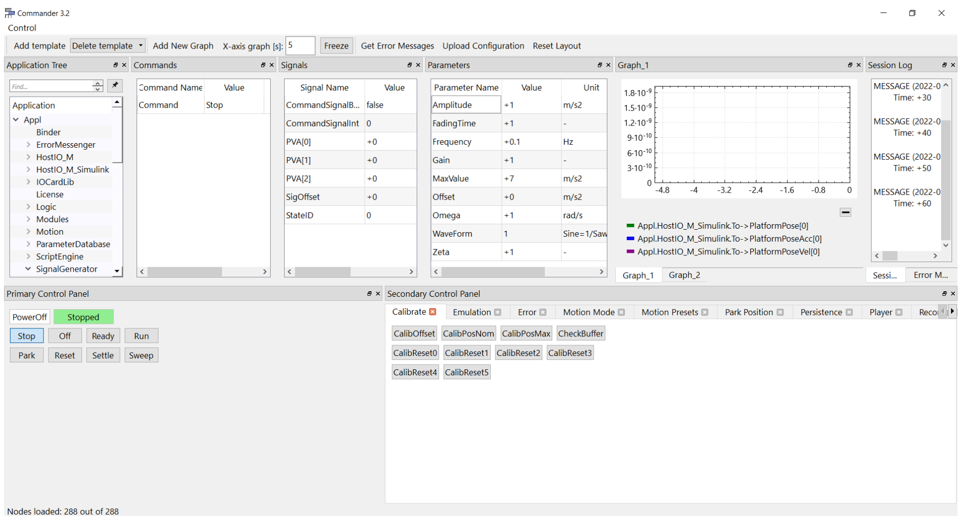
6 Advanced Simulator Dynamics

For the purpose of this study, an emulator environment was used to imitate the performance of the DAVSi, with the created motion cueing algorithm. The DAVSi is a 6 DOF driving simulator whose geometrical model has been explained in section 1.1. The system performance capabilities of the DAVSi are presented in the table below.

Table 6.1: System performance of the DAVSi

Motion	Displacement [m, deg]	Velocity [m/s, deg/s]	Acceleration [m/s^2, deg/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
Pitch θ	$\pm$24.3	$\pm$35.0	$\pm$260.0
Roll ϕ	-25.4 ... 28.4	$\pm$38.0	$\pm$260.0
Yaw φ	$\pm$25.0	$\pm$41.0	$\pm$510.0
Actuator	-1.297 ... 1.937	-	-

The emulator allows us to perform motion cueing tests without having to actually run the entire simulator hardware set-up. These offline simulations are run using the emulator's graphical user interface called Commander. The Commander allows the test scenarios to be built and run through the created Simulink file which contains the hexapod dynamics and the developed motion cueing algorithm. Using the GUI, analysis can be done on the platforms motion capabilities with respect to the reference signal being followed in the cueing algorithm. This includes monitoring signal and states, modifying parameters, viewing the status of the simulator and even generating plots of the signal responses. A general layout of the GUI can be seen in Figure 6.1. Therefore, the emulator setup helps in recognising how the DAVSi would actually perform in real life, if real-time hardware implementation of the algorithm took place. On the other hand, while performing simulations in real-time, the eMoveRT controller is used to control the actuation hardware and a connected host computer is used to operate the simulations taking place.



Command Name	Value
Command	Stop

Signal Name	Value
CommandSignalB...	false
CommandSignalInt	0
PVA[0]	+0
PVA[1]	+0
PVA[2]	+0
SigOffset	+0
StateID	0

Parameter Name	Value	Unit
Amplitude	+1	m/s2
FadingTime	+1	-
Frequency	+0.1	Hz
Gain	+1	-
MaxValue	+7	m/s2
Offset	+0	m/s2
Omega	+1	rad/s
WaveForm	1	Sine=1/Saw
Zeta	+1	-

🟩 Appl.HostIO_M_Simulink.To->PlatformPose[0]
🟦 Appl.HostIO_M_Simulink.To->PlatformPoseAcc[0]
🟪 Appl.HostIO_M_Simulink.To->PlatformPoseVel[0]

MESSAGE (2022-0
Time: +30
MESSAGE (2022-0
Time: +40
MESSAGE (2022-0
Time: +50
MESSAGE (2022-0
Time: +60

Figure 6.1: Commander graphical user interface

6.1. Simulation Setup

To analyze the performance of the DAVSi, a set of test scenarios were taken into account as done for offline simulations. The same simulations were run offline as well to allow proper comparison to take place and to see the tracking performance capabilities of the DAVSi. The test scenarios considered ranged from low to high amplitude signals, running for short to long duration. This gave a comprehensive outlook on the cueing algorithm's performance and can be seen as follows:

- Sine wave of amplitude $0.5m/s^2$ and frequency 0.1 Hz. with 10 seconds simulation time
- Step signal (from 1 to 4 seconds) of amplitude of amplitude $0.1m/s^2$ with 10 seconds simulation time
- Sine wave of amplitude $1m/s^2$ and frequency 0.01 Hz. with 60 seconds simulation time
- Combined sine wave comprising of 4 different signals as follows:
  - Wave 1: amplitude $0.5m/s^2$ and frequency 1 Hz.
  - Wave 2: amplitude $0.6m/s^2$ and frequency 0.1 Hz.
  - Wave 3: amplitude $0.1m/s^2$ and frequency 0.2 Hz.
  - Wave 4: amplitude $0.8m/s^2$ and frequency 0.15 Hz.

  The simulation was run for 20 seconds and a scaling factor of 0.3 was used to reduce the amplitude range and rate of change of the signal, resulting in the final reference signal as seen below:

$$sine_{combined} = k_{scale}(wave1 + wave2 + wave3 + wave4) \tag{6.1}$$

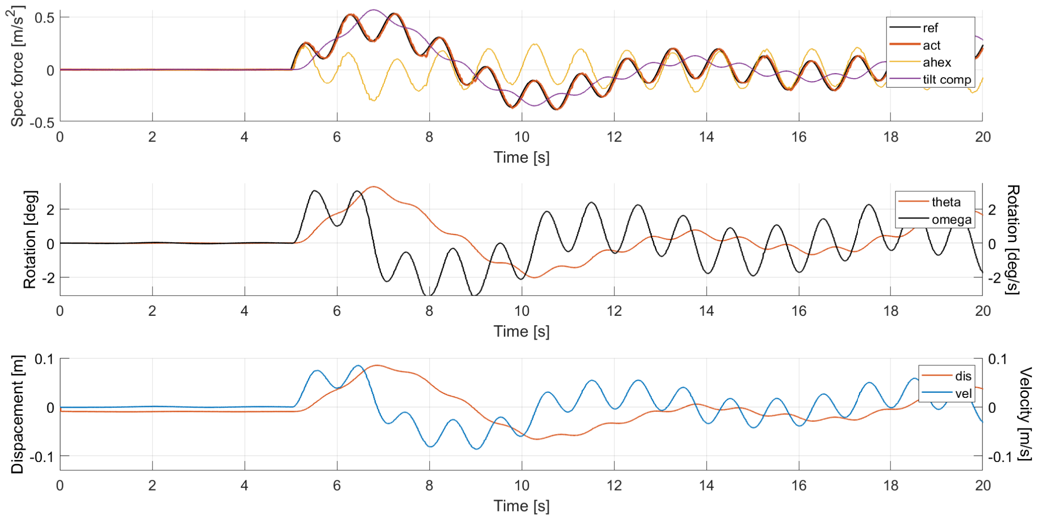
Simulations were performed using the 2 DOF motion cueing algorithm with hexapod dynamics. Additional slack variables were used for these simulations to improve robustness of the algorithm. These were applied to soften the tilt rate constraints which upon violation resulted in chattering and subsequent simulation failure in some test scenarios. Using these variables as part of the optimization problem, helps in avoiding such situations by allowing the constraint value to be slightly violated, while ensuring that the simulation does not fail. This implementation is done through additional constraints as seen in Equation 6.2. The remaining formulation and parameters used in the cueing algorithm were the same. All four developed models were studied under the same simulation setup, to see the effect on computation time through obtained mean iteration values.

$$\begin{aligned} \omega_{p,min} &\le \omega_p + s \\ \omega_p - s &\le \omega_{p,max} \\ 0 &\le s \end{aligned} \tag{6.2}$$

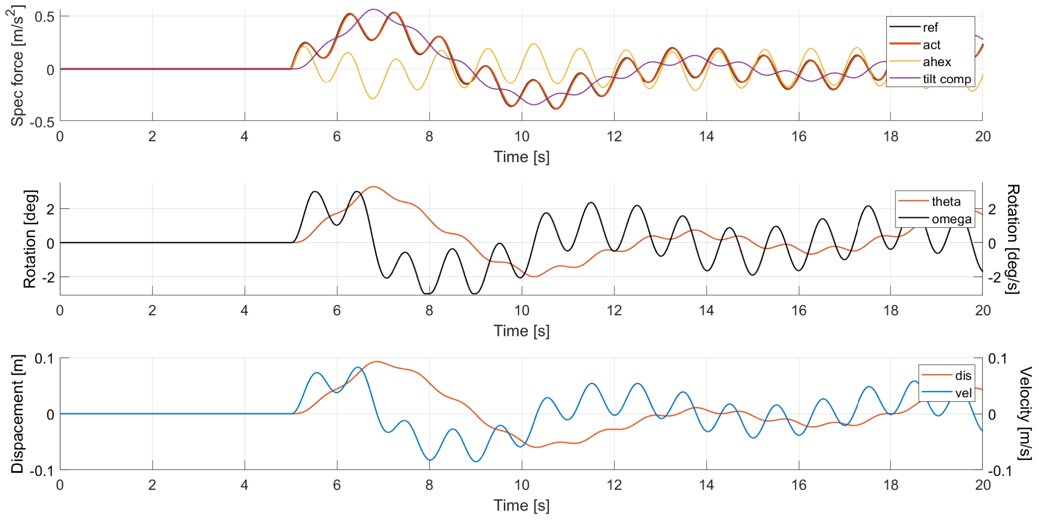
where, s is the slack variable

6.2. Simulation Performance

As mentioned in the previous section, four types of scenarios were considered. The first set of simulations were performed on signals with low amplitude, followed by multi-frequency and larger amplitude sine waves. In this section the combined sine wave is analysed with additional simulations presented in Appendix C.



(a) Emulator



(b) Offline simulation

Figure 6.2: Emulator performance comparison for combined sine wave

It can be inferred from Figure 6.2 that the emulator is able to track the desired reference signal with similar characteristics and performance when compared with offline simulation results. This occurs with sufficient contribution from both components of the specific force; linear platform acceleration and gravitational tilt. Neither of these components are erratic or extremely large in value, ensuring safe movement of the simulator with smooth tracking performance. No false cues are present in the simulation and washout effect takes place in a similar manner for both the emulator and offline simulations. Also, a larger rest period was used in this scenario to ensure that the emulator is in a stable condition before the desired motion begins. Thus, the emulator is able to work well with the designed motion cueing algorithm to track robust signals (with varying amplitude and frequency) and can be further used for extensive studies related to computation time.

While looking at the computation load required for different cueing models in Table 6.2, it can be seen that the mean iterations show a similar trend for both emulator and offline simulations; with the hybrid model with all control inputs giving lowest values. The other hybrid model with first explicit MPC control input also shows good improvement in mean iterations, whereas the implicit controller gives worst computation performance. For the step signal simulation, the mean iterations for both the hybrid controllers is close to or equal to 0 as at this low amplitude value, the initial guess is already very close to the optimized value; resulting in low online computation time.

The tracking for all the models is also analysed through RMSE values presented in Table 6.3. Offline simulations show superior RMSE tracking values for all the simulations when compared with the emulator results. This discrepancy is due to the difference in the simulator dynamics being used for both sets of simulations. However, looking at both sets of simulations independently, it can be seen that similar performance is obtained for all four cueing models across all test cases. Thus, the emulator which represents the movement and motion of the DAVSi shows similar trend in behaviour using this hybrid algorithm (in terms of reduced computation time while maintaining tracking performance) as observed in offline simulations.

Table 6.2: Mean iteration values for all four motion cueing models with the emulator and offline simulations

Type of simulation	Type of reference signal	Sampling time [s]	Mean iterations			
			iMPC	iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
Emulator	Sine wave of low amplitude	iMPC = 0.01 eMPC = 0.1	0.8611	1.3057	0.2319	0.1200
	Step signal		0.2647	0.5445	0.0160	0
	Sine wave of high amplitude		4.9128	4.8522	3.7820	2.5647
	Combined sine wave		0.6877	1.1679	0.6576	0.6092
Offline MCA simulation	Sine wave of low amplitude		1.0829	0.8772	0.7433	0.4625
	Step signal		1.3776	0.8881	0.0919	0
	Sine wave of high amplitude		1.5216	1.4521	1.3513	0.5232
	Combined sine wave		0.6377	0.5907	0.6012	0.2224

Table 6.3: Specific force RMSE values for all four motion cueing models with the emulator and offline simulations

Type of simulation	Type of reference signal	Sampling time [s]	RMSE (f_{spec})			
			iMPC	iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
Emulator	Sine wave of low amplitude	iMPC = 0.01 eMPC = 0.1	0.0112	0.0108	0.0104	0.0106
	Step signal		0.0148	0.0156	0.0162	0.0159
	Sine wave of high amplitude		0.0177	0.0089	0.0095	0.0077
	Combined sine wave		0.0212	0.0213	0.0218	0.0211
Offline MCA simulation	Sine wave of low amplitude		0.0030	0.0034	0.0044	0.0048
	Step signal		0.0051	0.0051	0.0048	0.0047
	Sine wave of high amplitude		0.0014	0.0013	0.0014	0.0013
	Combined sine wave		0.0098	0.0099	0.0099	0.0099

6.3. Virtual Track Simulations

The last study was done to see the performance of the developed motion cueing algorithm with a track simulation. This took place by feeding in track data as input to the cueing scheme. Data was obtained by simulating a scenario using the commercially available multibody software called IPG CarMaker. To analyse the effect in both longitudinal and lateral directions, the 4 DOF motion cueing algorithm was used in this study.

The Hockenheim race track was selected to perform these simulations in IPG CarMaker, as amongst the race tracks available it has the least elevation change of only $4.3m$. The race track can be visualized in Figure 6.3 which shows the plot of the center line of the track. Further, a high fidelity vehicle model of the Tesla Model S was used to perform the track simulation. The vehicle parameters of the model are present in Appendix C.

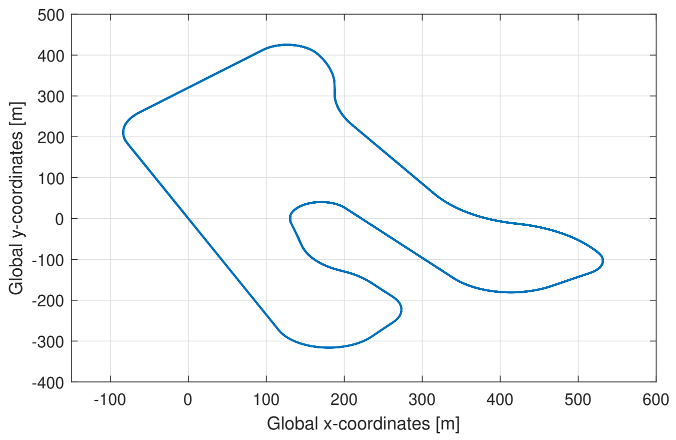


Figure 6.3: Hockenheim race track

The simulation was first run for the vehicle travelling at $120km/h$ for a single lap around the Hockenheim racetrack. User parameterized driver settings were used to perform the simulation with the following acceleration thresholds selected to imitate naturalistic driving style [49]:

$$\begin{aligned} a_{x,max} &= 3m/s^2 \\ a_{x,min} &= -4m/s^2 \\ a_{y,max} &= 4m/s^2 \end{aligned} \tag{6.3}$$

Once the simulation was complete in IPG CarMaker, longitudinal and lateral acceleration values were extracted. Before the extracted values could be used, they were passed through the vestibular system model by Telban [33]. This was done to ensure that only the perceived acceleration values are sent in; to be tracked by the motion cueing algorithm in the emulator interface of the DAVSi. The structure of this perception model can be seen below:

$$\frac{\hat{a}(s)}{a(s)} = \frac{k_{\text{oto}}\ (T_{\text{a}}s+1)}{(T_{\text{L}}s+1)\,(T_{\text{S}}s+1)} \tag{6.4}$$

Where, $k_{\text{oto}} = 0.4, T_{\text{a}} = 10, T_{\text{L}} = 5$ and $T_{\text{S}} = 0.016$. The selection of this model is based on an in-depth analysis which is presented in Appendix B.

Using these track reference values, 4 DOF simulations were run and the results obtained for longitudinal and lateral motion for a vehicle driving at 120km/h around the Hockenheim race track can be seen in Figure 6.4 and Figure 6.5.

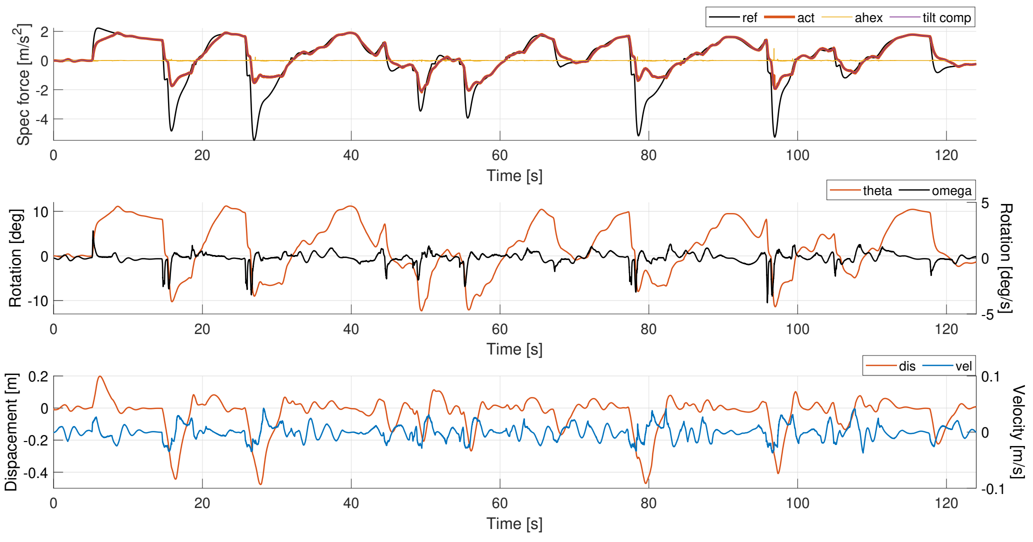


Figure 6.4: Longitudinal motion results for Hockenheim track simulation at 120km/h

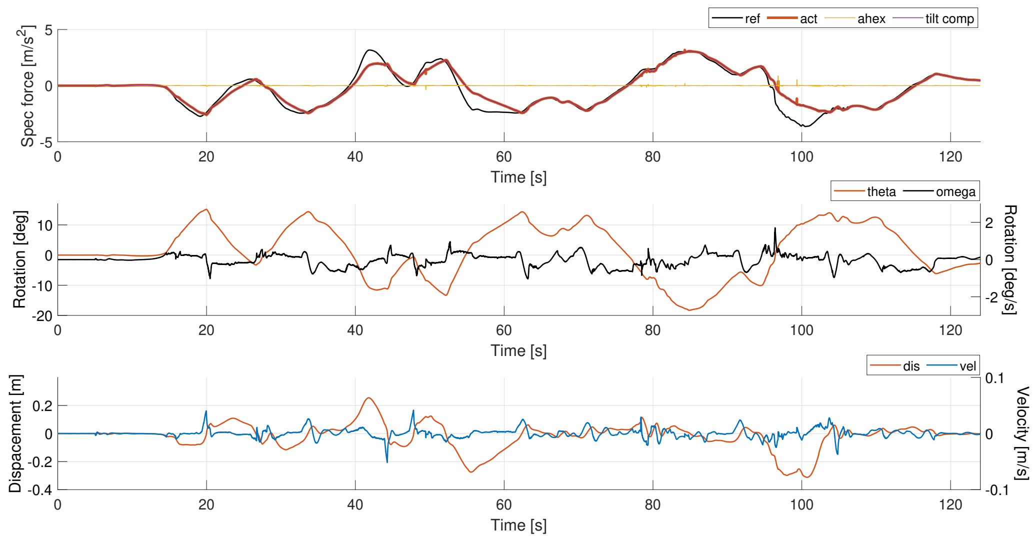


Figure 6.5: Lateral motion results for Hockenheim track simulation at 120km/h

From the figures, it can be inferred that the developed motion cueing algorithm gives good tracking performance throughout the simulation. This is seen in both longitudinal and lateral directions, with an RMSE value of 0.42, 0.21 respectively, for complete simulations around the Hockenheim race track. However, in some parts of the simulation, the cueing algorithm is unable to track the reference data perfectly. This occurs when the vehicle is travelling at a high speed and suddenly applies its brakes. An example can be seen at around $80s$ in Figure 6.4. At this time, the vehicle is driving through the third last corner before the main straight (which can be seen in Figure 6.3), requiring it to change direction and apply brakes at a fast rate; resulting in this immediate drop in longitudinal acceleration. In an attempt to achieve this motion inside the simulator, the tilt rate reaches its limit (and goes slightly out of bounds) while the platform displacement is also extended to its respective threshold, while staying inside the limit circle. Due to workspace limitations, the platform cannot be maneuvered anymore resulting in false cues.

Lastly, these results were quantified and a comparative study was done by simulating the same vehicle around the Hockenheim race track at slower speeds. This also included the mean iteration study with the four developed cueing models as done in the previous section. Table 6.4 shows the mean iterations observed for all four models. It can again be seen that the hybrid controllers reduce online computation time at all speed values when compared with the implicit controller. An improvement of 9% is seen with the hybrid controller using all explicit control inputs. The other hybrid controller shows an improvement of 5.9%, which is quite comparable to the implicit controller (with trajectory prediction) having a 5.1% difference in mean iterations. This behaviour is a replication of the trend observed with offline simulations presented in the previous chapter.

Table 6.4: Mean iteration values for all four motion cueing models with track simulations

Speed [km/h]	Mean iterations			
	iMPC	iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
80	0.5458	0.5302	0.5270	0.5243
100	0.6843	0.6511	0.6407	0.6133
120	0.6888	0.6363	0.6342	0.6025

A summary of the obtained RMSE values for all simulations performed is shown in Table 6.5. Here, it is confirmed that the RMSE tracking performance is maintained in both directions for all four cueing models. Also, a minor difference is observed in longitudinal and lateral directions at the same speed. This is due to the difference in signal values being tracked by the controller in both directions. As the speed starts to increase, the RMSE values worsen due to driving maneuvers being performed at a faster rate; resulting in larger false cues. The figures of the additional simulations done at lower speeds showing better performance is presented in Appendix C.

Table 6.5: Specific force RMSE values for all four motion cueing models with track simulations

Speed [km/h]	RMSE (longitudinal)			
	iMPC	iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
80	0.1894	0.1899	0.1891	0.1896
100	0.2565	0.2555	0.2690	0.2582
120	0.4279	0.4290	0.4105	0.4295

Speed [km/h]	RMSE (lateral)			
	iMPC	iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
80	0.1347	0.1344	0.1353	0.1349
100	0.1775	0.1760	0.1735	0.1794
120	0.2171	0.2111	0.2113	0.2158

6.4. Summary

This concludes the simulation studies performed with the developed motion cueing algorithms. It can be seen that with the emulator interface of the DAVSi, the cueing algorithm is able to give a similar trend in tracking and online computation time performance compared with the offline simulations. The emulator proves to be robust to different kinds of signals with varying amplitude and frequency. Also, track simulations using the extended 4 DOF cueing algorithm show that the cueing scheme works well with simulated track data of the Hockenheim race track at varying speeds. Here, the same trend of reduced mean iterations is observed with the designed hybrid controller. Also, tracking performance is maintained throughout the track simulation at all speeds analysed.

7 Conclusion and Recommendations

7.1. Conclusion

In the domain of motion cueing algorithms, several control techniques have been used ranging from conventional filter based algorithms to more advanced techniques using MPC. MPC based algorithms either compute an optimal controller online or derive an explicit control law in an offline setting. These approaches have limited applicability for real-time applications due to online computational expense and offline memory storage issues; leading to poor performance. Thus, the goal of this research was to implement a strategy to reduce the online computation load by shifting it offline using a hybrid MPC based algorithm.

First, a 2 DOF motion cueing algorithm was developed using the hybrid MPC based approach. For this, an explicit MPC based controller was created which comprised of displacement and velocity states in both surge-pitch DOFs, along with the acceleration components acting as the control inputs. The explicit controller was used to provide an initial guess acting as warm start for the implicit MPC based algorithm.The implicit controller was similar in its formulation but comprised of additional braking constraints and adaptive washout weights to improve specific force tracking. Based on this general hybrid formulation, four different kinds of cueing models were analysed by looking at performance indicators such as tracking performance and mean iterations. The explicit controller was not analysed separately due to limitations in model complexity and using large prediction horizon values with fast sampling rates; leading to poor performance. Thus, it was only used as an initial guess in this study. Out of all the hybrid controllers, the initial guess from the explicit MPC (with all control inputs) showed best results, with lowest mean iterations, when compared with the implicit controller.

Furthermore, this research was extended to 4 DOF motion cueing problems by adding in sway-roll DOFs. This increased the problem complexity for both explicit and implicit MPCs. The same set of test cases were analysed with the goal to track the reference specific force. From the results, it was observed that the hybrid controller with explicit MPC acting as the initial guess gave best results, by reducing the online computation load, as compared to just using the implicit controller on its own.

Also, emulator studies were performed to observe the performance of the DAVSi with the created motion cueing algorithm. For this, all the developed cueing models were studied with the emulator and tests were performed using different reference signals. The performance of the emulator (in terms of reference tracking and mean iterations) was compared with offline simulations, which showed uniform trend in results. Hybrid models again lowered the online computation load when compared to the implicit controller acting by itself. Lastly, virtual track simulations were performed by obtaining data from IPG CarMaker. The 4 DOF algorithm with a perception model was used in this study, which gave good tracking performance; for a virtual simulation of a car driving around the Hockenheim racetrack.

To conclude, an hybrid MPC based motion cueing algorithm has been developed which utilises explicit and implicit MPCs together. The explicit guess kick-starts the algorithm resulting in reduced number of iterations and computation time; to compute the optimized control input online. Tracking performance is maintained for a wide range of signals while working at a high sampling rate and prediction horizon. Moreover, emulator results with the DAVSi confirm the ability of the developed algorithm to reduce online computation time while maintaining tracking performance on a real-time system.

7.2. Recommendations

The developed hybrid MPC based motion cueing algorithm is able to work well for the designed scenarios and helps in reducing online computational expenses. The following recommendations can be made for future work:

First, additional testing should be done for the current 4 DOF motion cueing algorithm. Currently, virtual track simulations were performed using IPG CarMaker for the developed hybrid algorithm. These simulations were done using a vehicle simulated around a racetrack which showed good capabilities of the approach. In the future, this can be extended to take real road data representing natural driving conditions into account; with the emulator setup. These simulations can help in understanding the performance of the algorithm to a greater extent as seen with the race track simulations.

Second, real-time simulations should be performed, with human in loop, to see the actual working of the simulator with the developed motion cueing algorithm in real life. Even though the emulator is a direct representation of the DAVSi, performing a subjective analysis can prove to be beneficial to validate the findings of this study.

Third, investigation should be done in including a perception model as part of the motion cueing algorithm. By using an accurate mathematical model, such as the one developed by Telban, can lead to better approximation of the human vestibular model in the motion cueing algorithm. This can improve cueing performance due to generation of more realistic motion cues.

Lastly, this hybrid scheme can be extended by including yaw-heave DOFs, thus making it a 6 DOF motion cueing algorithm. In this version, additional actuator based constraints can be considered which improve overall functioning through efficient workspace management. Further, the test performance of the virtual simulator environment would be closer to real life simulations as all DOFs are taken in the cueing scheme. This will allow virtual simulations to be performed on tracks which have a higher rate of change of elevation than simulating the algorithm for flat road surfaces.

A Scientific Paper

This chapter contains the scientific paper that will be submitted to IEEE International Conference on Mechatronics 2023.

Hybrid MPC based Motion Cueing Algorithm for Driving Simulators

Abstract—**Driving simulators have been used in the industry for many years because of their ability to perform tests in a safe, reproducible and controlled immersive virtual environment. The improved performance of the simulator and its ability to recreate the same experience for the user is established through motion cueing algorithms (MCA). Such algorithms have constantly been developed with model predictive control (MPC) acting as the main control technique. Currently available MPC based methods either compute the optimal controller online or derive an explicit control law in an offline setting. These approaches limit the applicability of the MCA for real-time applications due to online computational expense and offline memory storage issues. This research presents a solution to deal with issues of offline and online solving through a hybrid approach. For this, explicit MPC is used to generate a look-up table to provide an initial guess as warm start for the implicit MPC based MCA. From the simulations, it is observed that the presented hybrid approach is able to reduce online computation load by shifting it offline using the explicit controller. Further, the algorithm is also tested with track simulation data in an emulator environment of a driving simulator, which gives good performance.**

Index Terms—**motion cueing algorithm, driving simulator, model predictive control**

I. Introduction

Driving simulators are used to perform tests on new advancements in vehicular technology. The virtual environment of the simulator is used to recreate the in-vehicle experience without any damages to the real vehicle. To do this, an MCA is used, which acts as the control tecnique for the driving simulator's movements. It governs the process allowing the simulator to function properly so that the same feeling of motion is experienced by the user and that maximum workspace utilization takes place [1].

In motion cueing, driver input is sent to the vehicle model which generates the reference signal to be tracked by the developed MCA. The MCA computes the desired motion to follow this reference signal, which is then performed by the simulator. The sensed specific force $f_{spec,s}$ (computed using simulator dynamics) comprises of two components, platform translational acceleration, $a_{tran,p}$ and the gravitational acceleration, which allows us to study the human body's movement in space during the cueing process. This is compared with the actual specific force value $f_{spec,a}$ obtained from the vehicle model. The computed error is then fed back into the MCA as feedback; to improve results for the next time step. Based on this, several kinds of MCAs have been implemented, which differ in terms of control techniques used.

Conventional filter based algorithms use the concept of high and low pass filters to reproduce the on-road experience inside the virtual environment [2]. They operate using three main channels. The first is the translational channel which takes in translational accelerations as input. It uses a high pass filter to filter out sustained low-frequency accelerations which can drive the simulator to its physical limits [2]–[5]. These filtered low-frequency accelerations are then recreated using tilt-coordination in the tilt channel [6]. Lastly, a rotational channel is present, which is similar to the translational channel but deals with inputted rotational velocities.

Different kinds of conventional algorithms have been developed such as the classical, optimal and adaptive washout based algorithms. The main issue such algorithms suffer from is that they are unable to take explicit constraints into account, leading to poor workspace utilization. Furthermore, some of these approaches like the classical washout based algorithm is a feedforward technique which results in poor performance. To overcome these problems, MPC based MCAs are commonly used. MPC is a control strategy that uses a dynamic model to forecast system behavior and optimize the future predictions to produce the best decision; control move at the current time [7]. This control strategy has been used in MCAs for over a decade with work done using two different types of controllers.

The first is the implicit/online controller which solves the optimization problem online at each time-step. Initially, linear MPC based MCAs were developed as seen in Beghi et al. [6] and Garrett et al. [8]. These improved performance when compared with the conventional algorithms, but gave sub-optimal results as the full capabilities (nonlinear dynamics) were not taken into account. Further, they employed constraints in the driver reference frame, to keep the problem linear, resulting in difficulties in realizing the available workspace. Thus, nonlinear based algorithms were developed as seen in Bruschetta et al. [9], who worked on a 9 DOF driving simulator. In this study, constraints were applied on the actuator lengths which showed improvements in performance when compared with the linear MPC based MCA. Khusro et al. [5] also developed a nonlinear MPC based MCA with actuator constraints. Perception thresholds were applied in this work to reduce false cues, along with adaptive weights which improved tracking performance. Lamprecht et al. [10] worked on a similar algorithm involving perception thresholds, but used a separate optimal control problem to predict future driver behaviour. Such MPC based MCAs give better and more accurate cueing performance compared to conventional and linear MPC based algorithms, but, they suffer from high online computation costs resulting in limitations for real-time implementation.

For this, other kinds of MPC based algorithms have been implemented. These use an explicit MPC, which pre-computes the solution and then uses it in the form of a look-up table online. This method reduces online computation time as seen in Fang et al [11]. In this study, a 2 DOF MCA was developed which was later extended by incorporating a vestibular model in [12]. Although this technique reduces online computation time, it suffers from memory storage issues along with restrictions in using large prediction horizons N_p with fast sampling rates. This is due to the exponential increase in control region computation time with an increase in complexity and scope of the problem, thus requiring a better alternative.

To overcome issues faced by implicit and explicit MPCs, a hybrid approach has been developed by Zeilinger [13]. An explicit controller was used to give an initial guess for the online optimization problem. The guess acts as a warm-start resulting in faster computation of the optimal control input. Since its inception, this technique has been used in applications such as curve tilting [14] and lateral motion stabilisation [15]. Jost et al. [16] also used this approach with modifications in the explicit initialisation.

In this work, a 4DOF hybrid MPC based MCA is designed to improve the computational speed of the algorithm while maintaining tracking performance. Hybrid MPC based approaches are yet to be implemented in the domain of cueing algorithms. This paper aims at establishing a hybrid approach for motion cueing, in doing so the computational performance of the proposed method is compared with the implicit MPC based MCA. The paper is structured as follows. In Section II, the controller design is explained which includes information about both MPCs present in the hybrid scheme. The test setup and simulations performed are presented in Section III. Conclusion and recommendations are listed in Section IV.

II. METHODOLOGY

A. Hybrid Scheme

Design of the hybrid MPC based MCA comprises of two main components, initialisation using explicit MPC and online computation using the implicit counterpart. A general scheme of the MCA can be seen in Fig. 1. The first step of this hybrid algorithm involves sending the initial state and reference values to the explicit MPC block. This block searches for the corresponding control region related to the state and reference values sent in. Once the control region is selected, the associated control inputs are sent to the online nonlinear solver, as initial guess. The reference values here refer to the specific force to be tracked by the MCA. With the information of the initial guess along with the current states and the reference signal, the MPC block computes the optimised control input. These are then sent to the simulator/plant dynamics block to apply the control input and update the states for the next time step. Once the state update is complete, the entire process is repeated until the simulation finishes.

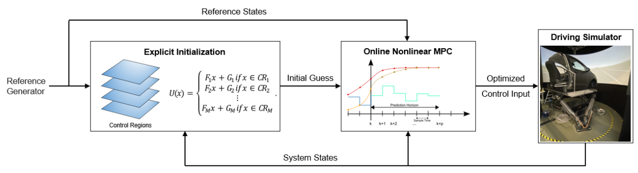


Fig. 1. Hybrid MPC scheme for the proposed motion cueing algorithm

B. Explicit MPC

As previously discussed, explicit MPC is used to compute a look-up table which is later used to provide an initial guess to the online solver. This comprises of state and reference values stored in the form of control regions. Each control region corresponds to a particular control input value, which is generated as follows: [17], [18].

$$U(x) = F_i x + G_i \text{ if } x \in \mathcal{CR}_i. \tag{1}$$

Where, $\mathcal{CR}_i$ are the control regions.

To generate the look-up table, an MCA is designed which takes the 4 DOF capabilities of the driving simulator into account, using the Multi Parametric Toolbox (MPT). This algorithm is a simplified version of the online counterpart as its goal is to provide an educated guess for warm start. Eight states are used in the model which take the platform displacement s_p, platform velocity v_p, tilt angle θ_p and tilt rate ω_p for pitch-surge and sway-roll DOFs into account. The state space equations are shown in (2).

$$\dot{x}(k) = \begin{cases} \dot{\omega}_{p,long} = a_{p,long,rot} \\ \dot{\theta}_{p,long} = \omega_{p,long} \\ \dot{v}_{p,long} = a_{p,long,tran} \\ \dot{s}_{p,long} = v_{p,long} \\ \dot{\omega}_{p,lat} = a_{p,lat,rot} \\ \dot{\theta}_{p,lat} = \omega_p \\ \dot{v}_{p,lat} = a_{p,lat,tran} \\ \dot{s}_{p,lat} = v_{p,lat} \end{cases} \tag{2}$$

Here, longitudinal and lateral subscripts refer to the pitch-surge and sway-roll DOFs respectively. Also, this problem contains four control inputs $u(k)$, comprising of translational and rotational accelerations acting in both longitudinal and lateral directions:

$$u(k) = \begin{cases} a_{p,long,rot} \\ a_{p,long,tran} \\ a_{p,lat,rot} \\ a_{p,lat,tran} \end{cases} \tag{3}$$

Thus, the combined system can be represented as follows:

$$\dot{x}(k) = f(x(k), u(k)) \tag{4}$$

Constraints are applied in the MCA to limit the motion capabilities of the driving simulator. Firstly, perception thresholds are applied on both pitch and roll movements to ensure that the driver does not realize the tilting action. Generally, a lower value in the range of 2-4 deg/s is used [5], [10], [19]–[22] and for the proposed MCA, 3 and 2.6 deg/s was chosen for pitch and roll tilt rates respectively.

Secondly, constraints are applied on the platform displacement to limit the translational displacement of the platform. As we are dealing with 4 DOFs, the movement of the platform is limited by the limit circle which is represented by $\sqrt{s_{p,long}^2 + s_{p,lat}^2} \leq 0.5^2$. However, as the explicit MPC is formulated using MPT, nonlinear constraints cannot be taken into account which is why the platform displacement limits are applied separately for the explicit controller with a value of $0.35m$. This ensures that the simulator movements are restricted within the limit circle. The control input obtained from the explicit MPC controller acts as an initial guess which is why a lower constrained value can be taken into account; with the implicit controller refining the solution. The constraints used in the problem are given below:

$$\chi_e = \begin{cases} -3deg/s \leq \omega_{p,long} \leq 3deg/s \\ -2.6deg/s \leq \omega_{p,lat} \leq 2.6deg/s \\ -30deg \leq \theta_{p,long,lat} \leq 30deg \\ -7.2m/s \leq v_{p,long,lat} \leq 7.2m/s \\ -0.35m \leq s_{p,long,lat} \leq 0.35m \\ -9.81m/s^2 \leq a_{p,long,lat,tran} \leq 9.81m/s^2 \end{cases} \tag{5}$$

The goal of this MCA is to track the output specific force. This specific force is defined by the two vector components, translational and gravitational tilt accelerations. Rotations with respect to the x and y axis are used in deriving the gravitational tilt components which are as follows:

$$g_{tilt} = \begin{cases} g_{long} = g \sin \theta_{p,long} \\ g_{lat} = -g \cos \theta_{p,long} \sin \theta_{p,lat} \end{cases} \tag{6}$$

Taking the translational accelerations into account, the final output specific force is given by:

$$y(k) = \begin{cases} f_{spec,long} = a_{p,long,tran} + g \sin \theta_{p,long} \\ f_{spec,lat} = a_{p,lat,tran} - g \cos \theta_{p,long} \sin \theta_{p,lat} \end{cases} \tag{7}$$

Furthermore, the cost/objective function consisted of weighted states, outputs and control inputs. The corresponding weights chosen for our simulations are presented in Table I. As the states are already constrained, a value of 0 is assigned to allow freedom of movement in the available workspace. Further, the output is given the highest weight as the goal of the algorithm is to track the specific force. Equal control input weights are used to ensure sufficient contribution from both translational and rotational motions.

TABLE I
MPC CONTROLLER WEIGHTS

Weight	States	Outputs	Control input
Value	0	1	1e-3

C. *Implicit MPC*

The second part of the hybrid approach is the online MPC based algorithm. This algorithm is able to take nonlinear constraints into account and is modelled using ACADO in MATLAB. The states $\dot{x}(k)$ of the cueing approach are also updated by adding the platform accelerations, previously used as the control inputs. These control inputs are replaced by commanded acceleration values, which are modelled incorporating a time delay in the system. The updated model is shown below:

$$\dot{x}(k) = \begin{cases} \dot{\omega}_{p,long} = a_{p,long,rot} \\ \dot{\theta}_{p,long} = \omega_{p,long} \\ \dot{v}_{p,long} = a_{p,long,tran} \\ \dot{s}_{p,long} = v_{p,long} \\ \dot{\omega}_{p,lat} = a_{p,lat,rot} \\ \dot{\theta}_{p,lat} = \omega_p \\ \dot{v}_{p,lat} = a_{p,lat,tran} \\ \dot{s}_{p,lat} = v_{p,lat} \\ \dot{a}_{p,long,tran} = \dfrac{a_{cmd,long,tran} - a_{p,long,tran}}{T_s} \\ \dot{a}_{p,long,rot} = \dfrac{a_{cmd,long,rot} - a_{p,long,rot}}{T_s} \\ \dot{a}_{p,lat,tran} = \dfrac{a_{cmd,lat,tran} - a_{p,lat,tran}}{T_s} \\ \dot{a}_{p,lat,rot} = \dfrac{a_{cmd,lat,rot} - a_{p,lat,rot}}{T_s} \end{cases} \tag{8}$$

The implicit controller allows us to take nonlinear constraints into account which is why the limit circle displacement constraint can be used directly. Apart from this, additional braking constraints are incorporated in the algorithm [11]. These constraints help in slowing down the platform velocity and tilt rate as the platform displacement and tilt angle reach their respective limits. Two sets of constraints are used for these; one for platform velocity and the other for the tilt rate as follows:

$$s_{p,min} \leq c_v v_p T_{brk,p} + 0.5 c_u a_{p,tran} T_{brk,p}^2 \leq s_{p,max} \tag{9}$$

$$\theta_{p,min} \leq c_w \omega_p T_{brk,\theta} + 0.5 c_u a_{p,rot} T_{brk,\theta}^2 \leq \theta_{p,max} \tag{10}$$

where, $c_v = 1, c_w = 1, c_u = 0.45, T_{brk,\theta} = 0.5$, $T_{brk,p} = 2.5$ and s_p, θ_p thresholds are 0.5m and 30 deg respectively.

The constraints used in the model are as follows:

$$\chi_i = \begin{cases} -3deg/s \leq \omega_{p,long} \leq 3deg/s \\ -2.6deg/s \leq \omega_{p,lat} \leq 2.6deg/s \\ -30deg \leq \theta_{p,long,lat} \leq 30deg \\ -7.2m/s \leq v_{p,long,lat} \leq 7.2m/s \\ -0.5m \leq s_{p,long,lat} \leq 0.5m \\ -9.81m/s^2 \leq a_{p,long,lat,tran} \leq 9.81m/s^2 \\ -0.5m \leq c_v v_{p,long} T_{brk,p} \\ \quad + 0.5 c_u a_{p,long,tran} T_{brk,p}^2 \leq 0.5m \\ -30deg \leq c_w \omega_{p,long} T_{brk,\theta} \\ \quad + 0.5 c_u a_{p,long,rot} T_{brk,\theta_{p,long}}^2 \leq 30deg \\ -0.5m \leq c_v v_{p,lat} T_{brk,p} \\ \quad + 0.5 c_u a_{p,lat,tran} T_{brk,p}^2 \leq 0.5m \\ -30deg \leq c_w \omega_{p,lat} T_{brk,\theta} \\ \quad + 0.5 c_u a_{p,lat,rot} T_{brk,\theta_{p,lat}}^2 \leq 30deg \\ -5m/s^2 \leq a_{cmd,long,lat,tran} \leq 5m/s^2 \\ \sqrt{s_{p,long}^2 + s_{p,lat}^2} \leq 0.5^2 \end{cases} \tag{11}$$

Finally, adaptive washout weights are added into the model to allow the driving simulator to reach its neutral position. Using adaptive weights allows different signals to be analysed without having to tune the weights for each signal being studied. This is done by applying weights on the platform displacement and tilt rate as they tend to reach their limits in every simulation. The formulation of the adaptive weight for these two states can be seen in (12) and (13). Fig. 2 shows how the weight adapts itself based on the position of the platform. A high weight is applied when the platform is close to its limit and a low weight when it is near the neutral position, thus allowing a washout effect to take place.

$$\begin{aligned} W_{s_p} =& w_{s,1} + w_{s,2}\left(\frac{abs(s_{p,i})}{w_{s,5}}\right) + w_{s,3}\left(\frac{abs(s_{p,i})}{w_{s,5}}\right)^2 \\ &+ w_{s,4}\left(\frac{abs(s_{p,i})}{w_{s,5}}\right)^4 \end{aligned} \tag{12}$$

$$\begin{aligned} W_{\omega_p} =& w_{\omega,1} + w_{\omega,2}\left(\frac{abs(\omega_{p,i})}{w_{\omega,5}}\right) + w_{\omega,3}\left(\frac{abs(\omega_{p,i})}{w_{\omega,5}}\right)^2 \\ &+ w_{\omega,4}\left(\frac{abs(\omega_{p,i})}{w_{\omega,5}}\right)^4 \end{aligned} \tag{13}$$

where, $w_{s,1} = 0.01, w_{s,2} = 20, w_{s,3} = 20, w_{s,4} = 20, w_{s,5} = 0.5, w_{\omega,1} = 0.0001, w_{\omega,2} = 0.7, w_{\omega,3} = 0.7, w_{\omega,4} = 0.7$, and $w_{\omega,5} = 3$.

The structure of the implicit MCA is as follows:

$$\begin{aligned} \min_{u_{N_p}} \quad & J(x_0, u) \\ \text{s.t.,} \quad & x(k+1) = f(x(k), u(k)) \\ & x \in \chi_i \\ & x(N) \in \mathbb{X}_f \end{aligned} \tag{14}$$

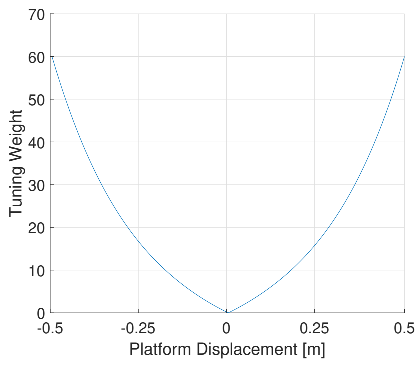


Fig. 2. Adaptive weights for platform displacement r_p

III. Simulation Results

A. Simulation Setup

A general set of test conditions are taken into account for simulating and observing the performance of the developed MCA. This includes specific force signals to be tracked by the cueing algorithm, which are in the form of sine waves and step signals along with multiple event waves (step signal + sine wave), for a range of amplitude ($0.5 - 2$ m/s^2) and frequency ($0.1 - 0.8$ Hz.) values.

While computing the explicit solution, a N_p of 2 is taken with a sampling time of 0.25 seconds. These values are selected to ensure that the look ahead time of 0.5s is used in the MCA. A higher N_p with faster sampling rates cannot be taken into account due to the exponentially high computation time from the explicit controller. The online version of the MCA however, is able to work at faster sampling rates and N_p values. Thus, N_p of 50 with a T_s of 0.01s is used to maintain the same look ahead time as the explicit controller.

Different MCA models are also considered to see which kind of approach performs best online. These are laid out as follows:

- Implicit MPC without any initial guess
- Implicit MPC with control trajectory prediction - In this model, the control trajectory prediction received from the ACADO s-function is used as the initial guess for the next time step. The first control input from the trajectory prediction is selected and applied for the entire horizon.
- Hybrid MCA with first explicit MPC control input - In this model, the first control input from the explicit MPC look-up table is used and applied for the entire prediction horizon.
- Hybrid MCA with all explicit MPC control inputs - This model utilises all control inputs obtained from the explicit MPC controller and applies them for the entire horizon. As the sampling times vary between both controllers, the explicit MPC inputs are applied in equal sections throughout the larger prediction horizon of the implicit MPC. For e.g. with a $N_{p,eMPC}$ of 5, the five control inputs are applied ten times each (1st from 1-10, 2nd from 11-20 and so on) for a $N_{p,iMPC}$ of 50.

Simplified actuator dynamics are used for offline simulations, whose results are showcased in the next section.

B. *Motion Cueing Performance*

Using the defined reference signals, simulations were run while looking at two main characteristics; specific force tracking and online computation time. In Fig. 3 and Fig. 4 the specific force tracking performance for a multiple event wave is present. This comprises of an initial step signal followed by a sine wave, both of amplitude 0.5 m/s^2. From the results, it can be seen that the cueing algorithm is able to track the reference signal while making sure that washout effect takes place in both longitudinal and lateral directions.

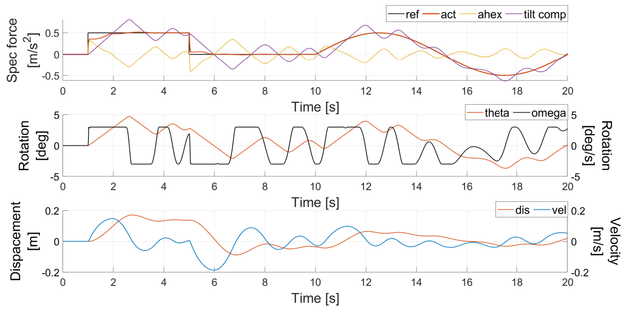


Fig. 3. Longitudinal motion for multiple event wave

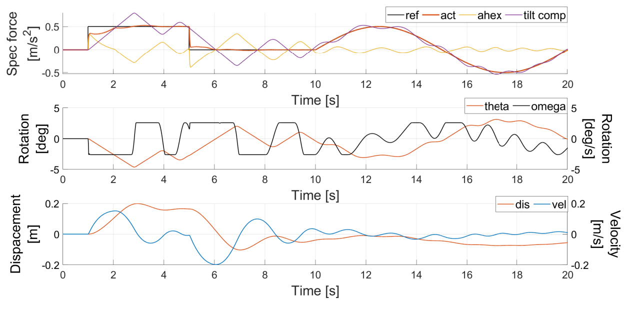


Fig. 4. Lateral motion for multiple event wave

Furthermore, the cueing algorithm is tested for their respective computational expense across different reference signal scenarios mentioned earlier. All the hybrid models are compared with the implicit MPC based cueing algorithm and the obtained results are presented in Fig. 5. The figure also shows the average tracking performance in both longitudinal and lateral directions for all scenarios. It can be seen that the developed hybrid models take less time to compute the optimized control input. The hybrid model with all explicit MPC control inputs performs best amongst all the models analysed by showing highest improvement in mean iterations from the implicit algorithm by 30%. This is done while keeping similar tracking performance in both longitudinal and lateral directions. Also, while performing the simulations the maximum iterations are set to 200.

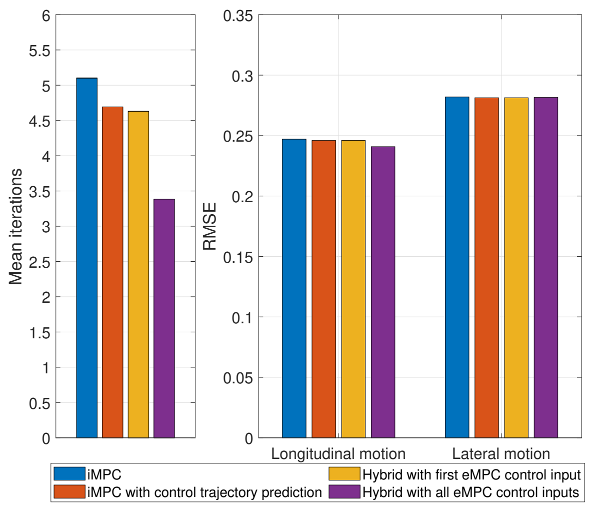


Fig. 5. Mean iterations and tracking performance for all scenarios

C. *Emulator Track Performance*

Simulations were also done with the emulator environment imitating the Delft Advanced Vehicle Simulator (DAVSi). The DAVSi is a 6 DOF driving simulator and using its emulator interface, tests can be performed without any damage limitations to the real system.

Full-track simulation tests were performed using this virtual environment. First, IPG CarMaker was used to simulate a vehicle driving at 120 km/h around the Hockenheim race track. Then, acceleration values were extracted, which were passed through the perception model by Telban [22] before using them as reference in the MCA. This was done to ensure that only the perceived acceleration values are sent to the MCA for performing the simulations with the emulator interface. From Fig. 6 and Fig. 7 it can be observed that the MCA is able to perform good reference signal tracking in both longitudinal and lateral directions throughout the simulation. An RMSE of 0.42, 0.21 is observed in both directions respectively. Further, similar trend in mean iterations is observed with the hybrid MCA improving online computation time performance. An improvement of 9% is seen with the hybrid model using all control inputs, whereas the other hybrid and implicit model shows an improvement of 5.9% and 5.1% respectively. Thus, the developed algorithm can be implemented and used with real track data in motion systems such as the DAVSi.

IV. Conclusion

In this study, a hybrid MPC based MCA has been developed which uses explicit and implicit MPCs together. The explicit MPC gives an initial guess which is used by the implicit MPC to kick-start the algorithm and compute the optimized control input at a faster rate. Amongst the developed cueing models, best computation time performance is observed from the model taking all explicit MPC control inputs as the initial guess. Moreover, to improve motion cueing, braking

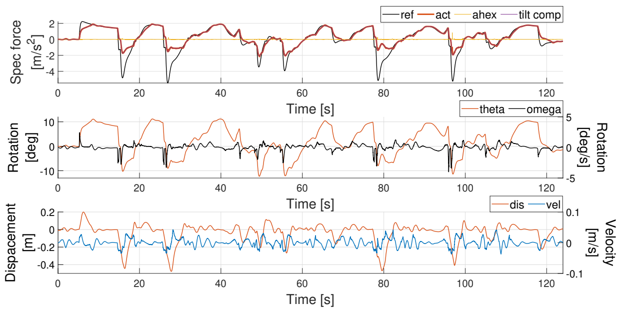


Fig. 6. Longitudinal motion results for Hockenheim track simulation

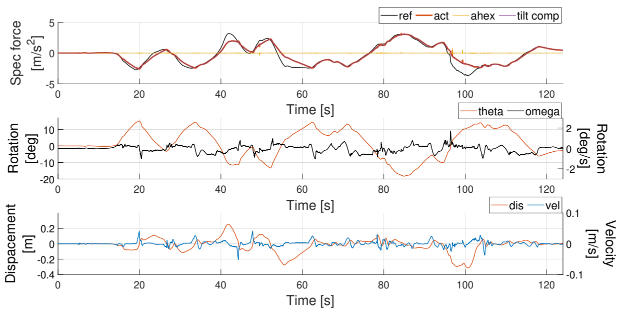


Fig. 7. Lateral motion results for Hockenheim track simulation

constraints are used to slow down the simulator when it is about to reach its physical displacement limits. Adaptive washout weights are also implemented which help in reducing any false cues that might occur; by bringing the simulator to its neutral position. Overall, the algorithm maintains tracking performance across all hybrid models while reducing online computation time. This is also seen in the track simulation done using the DAVSi which shows similar performance. In the future, human in loop experiments will be performed using the DAVSi to validate the findings of the paper. Also, extensions will be made to take 6 DOF motion capabilities into account as part of the cueing algorithm.

References

[1] S. Casas, R. Olanda, and N. Dey, "Motion cueing algorithms: A review," *International Journal of Virtual and Augmented Reality*, vol. 1, 2016.

[2] A. Stratulat, V. Roussarie, J. L. Vercher, and C. Bourdin, "Improving the realism in motion-based driving simulators by adapting tilt-translation technique to human perception," 2011.

[3] C. Seehof, U. Durak, and H. Duda, "Objective motion cueing test - experiences of a new user," 2014.

[4] M. A. Nahon and L. D. Reid, "Simulator motion-drive algorithms - a designer's perspective," *Journal of Guidance, Control, and Dynamics*, vol. 13, 1990.

[5] Y. R. Khusro, Y. Zheng, M. Grottoli, and B. Shyrokau, "Mpc-based motion-cueing algorithm for a 6-dof driving simulator with actuator constraints," *Vehicles*, vol. 2, 2020.

[6] A. Beghi, M. Bruschetta, and F. Maran, "A real time implementation of mpc based motion cueing strategy for driving simulators," 2012.

[7] B. J. Rawlings, Q. D. Mayne, and M. M. Diehl, *Model predictive control: Theory, Computation, and Design*, 2018, vol. 1.

[8] N. J. Garrett and M. C. Best, "Model predictive driving simulator motion cueing algorithm with actuator-based constraints," *Vehicle System Dynamics*, vol. 51, 2013.

[9] 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*, vol. 25, 2017.

[10] A. Lamprecht, D. Steffen, K. Nagel, J. Haecker, and K. Graichen, "Online model predictive motion cueing with real-time driver prediction," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[11] Z. Fang and A. Kemeny, "Explicit mpc motion cueing algorithm for real-time driving simulator," vol. 2, 2012.

[12] S. Munir, M. Hovd, Z. Fang, S. Olaru, and A. Kemeny, "Complexity reduction in motion cueing algorithm for the ultimate driving simulator," vol. 50, 2017.

[13] M. N. Zeilinger, C. N. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization," *IEEE Transactions on Automatic Control*, vol. 56, 2011.

[14] Y. Zheng, B. Shyrokau, T. Keviczky, M. A. Sakka, and M. Dhaens, "Curve tilting with nonlinear model predictive control for enhancing motion comfort," *IEEE Transactions on Control Systems Technology*, 2021.

[15] Y. Zheng and B. Shyrokau, "A real-time nonlinear mpc for extreme lateral stabilization of passenger vehicles," 2019.

[16] M. Jost and M. Mönnigmann, "Accelerating online mpc with partial explicit information and linear storage complexity in the number of constraints," 2013.

[17] I. Maurović, M. Baotić, and I. Petrović, "Explicit model predictive control for trajectory tracking with mobile robots," 2011.

[18] P. Bemporad, *Explicit Model Predictive Control*. London: Springer London, 2013, pp. 1–9. [Online]. Available: https://doi.org/10.1007/978-1-4471-5102-9_10-1

[19] J. Gutridge, "Three degree-of-freedom simulator motion cueing using classical washout filters and acceleration feedback," *Virginia Polytechnic Institute and State University*, 2004.

[20] L. D. Reid and M. A. Nahon, "Flight simulation motion-base drive algorithms: part 1. developing and testing equations," *UTIAS Report, No. 296*, 1985.

[21] A. B. Koyuncu, E. Ercelik, E. Comulada-Simpson, J. Venrooij, M. Kaboli, and A. Knoll, "A novel approach to neural network-based motion cueing algorithm for a driving simulator," 2020.

[22] R. J. Telban and F. Cardullo, "Motion cueing algorithm development: Human-centered linear and nonlinear approaches," *NASA TechReport*, 2005.

B Human Perception

B.1. Vestibular System Modeling

There are various kinds of vestibular system models implemented in literature. In this section, models which have been extensively used in past studies for motion cueing algorithms (as part of the problem formulation) are compared and deliberated upon. Further, a sensitivity analysis is also carried out to see how the model behaves to variations in parameter values.

As mentioned before, semicircular canals and otolith organs are the 2 main components used for taking vestibular system effects into account. For semicircular canals, work done by Telban [33], Reid [35], Young and Ormsby [37] is frequently used in the work related to motion cueing algorithms. These models use a general structure for the semicircular canals which can be seen in Equation B.1 (this model is a linear approximation to ensure that the computational time remains low and that the real-time performance is not affected due to nonlinearities).

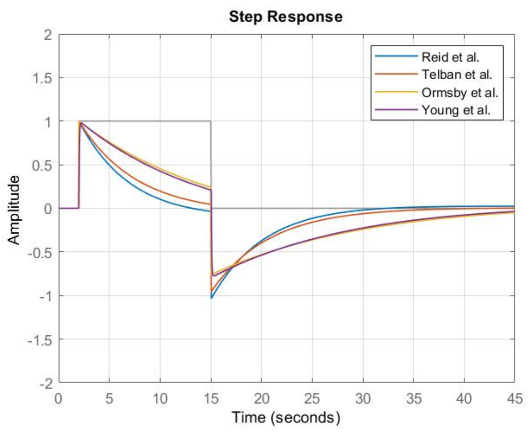
$$\frac{\hat{\omega}(s)}{\omega(s)} = \frac{T_{\mathrm{L}} T_a s^2}{(T_{\mathrm{L}} s + 1)\,(T_{\mathrm{S}} s + 1)\,(T_{\mathrm{a}} s + 1)} \tag{B.1}$$

where, $T_a, T_{\mathrm{L}}, T_{\mathrm{S}}$ are the relevant gain parameters which vary from one transfer model to another. A summary of the values taken for different transfer functions analysed can be seen in the table below:

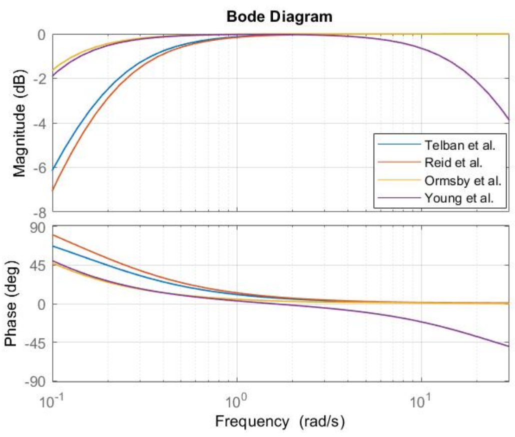
Table B.1: Transfer function parameters for semicircular canals

Parameter	Telban	Reid	Young	Ormsby
T_a(s)	80	30	30	30
T_{L}(s)	5.73	5.3	16	18
T_{S}(s)	0	0	0	0

Using these values, the corresponding step and frequency response plots can be generated to see the performance of the different transfer functions with varying parameter values. The step response plot shows how the transfer functions respond to a unit step input signal from 2 seconds to 15 seconds. The results can be seen in Figures B.1a and B.1b. Table B.2 shows the step response characteristics such as rise and settling time for all the transfer functions. Settling minimum and maximum values are also present in the table which specify the minimum and maximum value of the model once the response has risen.



(a) Step response



(b) Frequency response

Figure B.1: Step and frequency response for semicircular canals

Table B.2: Step response characteristics for semicircular canals

Parameter	Unit	Telban	Reid	Young	Ormsby
Rise Time	s	0.0022	0.0192	0.0393	0.0399
Settling Time	s	35.7432	32.7884	42.3197	42.2508
Settling Min.	amplitude	-0.9551	-1.0354	-0.7601	-0.7771
Settling Max.	amplitude	1	1	1	0.9781

From the results it can be observed that in the step response plot, the transfer functions from Ormsby and Young take a large time to converge to zero. Also, these two models do not exhibit a very large high pass behaviour as compared to the other two by Telban and Reid. In the frequency plots, it can be seen that, for the normal head movement range of 0.1 to 1 Hz., all the models show good behaviour as their magnitudes are constant and phase angle is close to zero. The model by Young is an exception as it suffers from a slight drop in phase angle in this range. This means that except this model, the others are able to act as a suitable angular velocity sensor for the normal head movement range and give a good approximation in predicting human perception using semicircular canals. Thus, upon looking at both the step and frequency responses, the models by Telban and Reid are preferred.

As done for the semicircular canals, a similar study is carried out for the otolith organs as well. For this, the work done by Telban [33], Ormsby and Young [48] is taken into account. The structure of the transfer function used by all these models can be seen below:

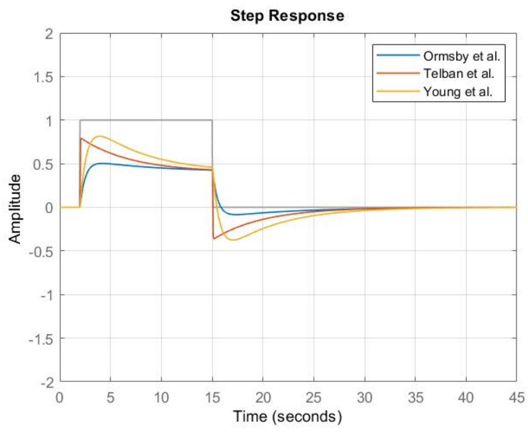
$$\frac{\hat{a}(s)}{a(s)} = \frac{k_{\text{oto}}\ (T_{\text{a}}s+1)}{(T_{\text{L}}s+1)\,(T_{\text{S}}s+1)} \tag{B.2}$$

The values for parameters mentioned in the equation above can be seen in Table B.3 (for the chosen transfer function models):

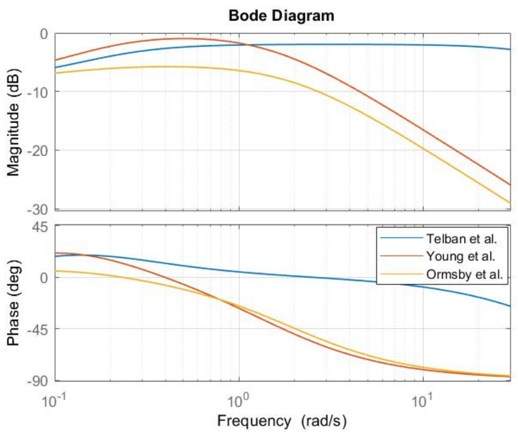
Table B.3: Transfer function parameters for otolith organs

Parameter	Telban	Young	Ormsby
K_{oto}	0.4	0.4	0.4
T_a(s)	10	13.2	10.1
T_{L}(s)	5	5.3	7.51
T_{S}(s)	0.016	0.66	0.51

Using these transfer functions, the step and frequency responses are again plotted as done for the semicircular canals. The results can be seen in Figures B.2a and B.2b with Table B.4 showing the step response characteristics.



(a) Step response



(b) Frequency response

Figure B.2: Step and frequency response for otolith organs

Table B.4: Step response characteristics for otolith organs

Parameter	Unit	Telban	Young	Ormsby
Rise Time	s	9.7673e-04	0.0466	0.1821
Settling Time	s	30.5390	33.6788	32.2377
Settling Min.	amplitude	-0.3629	-0.3749	-0.0838
Settling Max.	amplitude	0.7918	0.8151	0.5031

From the results it can be seen that the amongst the three models evaluated, the Ormsby model is unable to reach the high bandwidth that the other two models are able to. Further, between the Telban and Young models, Telban's model is able to give lower delay and has a higher bandwidth with an extremely small rise time. In the frequency response plot, the advantage of using the Telban model is more apparent as it is the only model which shows constant magnitude and zero phase angle value for normal head movement frequencies (0.1 to 1 Hz.). This means that the otolith model by Telban acts as a specific force transducer and a suitable sensor in the frequency range for predicting human perception. It can be used to recognize tilt orientation with respect to the local vertical better than other models analyzed. Hence, the Telban model is preferred amongst the investigated models.

The last type of analysis done is the sensitivity analysis. This test is performed on the Telban model (as it gives best performance amongst the models analysed) to see how the model reacts to variations in parameter values. For this, the sensitivity function is first computed using the expression below:

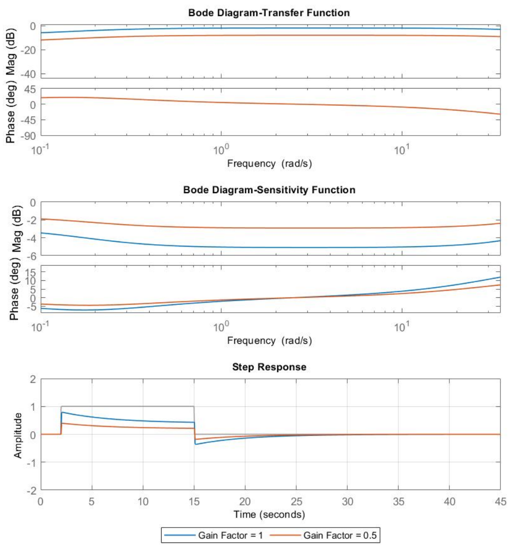
$$\text{Sensitivity Function } = \frac{1}{1+T(s)} \tag{B.3}$$

$$S_K^T = \frac{1}{1+T} = \frac{0.08s^2+5.016s+1}{0.08s^2+9.016s+1.4} \tag{B.4}$$

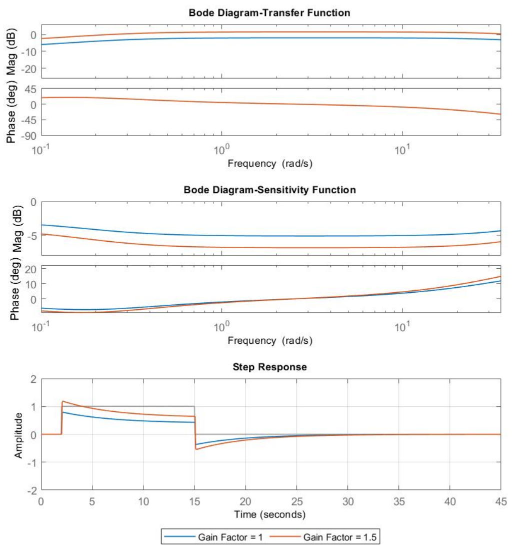
Here, T is the transfer function. Bode plots are then created for the sensitivity function and the maximum sensitivity value is computed using the expression shown below. The result obtained corresponds to the correct point which can be seen in Figure B.3.

$$\frac{d}{dw}\left|S_k^T(jw)\right| = 0 \tag{B.5}$$

Finally, the gain value parameter is varied to see what effect it has on the overall results. First, the gain is reduced by a factor of 0.5 as seen in Figure B.3a. It can be observed that although the sensitivity has indeed improved, reduction of the gain parameter results in low magnitude across the entire frequency range in the original transfer function. This effect of low magnitude in the low frequency range results in lower rise time and value in the step response as seen in the third subplot in Figure B.3a. This means that the transfer function is unable to follow slow moving step signals anymore. Thus, reducing the gain parameter is not advised. Upon increasing this parameter by the same factor as seen in Figure B.3b, the sensitivity increases which spoils the original transfer function's frequency and step responses. Hence, the sensitivity analysis shows that the system is sensitive to change in parameters and that the parameter values which have been taken for the Telban model show optimum results for this particular use case.



(a) Decreasing gain factor



(b) Increasing gain factor

Figure B.3: Sensitivity analysis

B.2. Perception Threshold Analysis

In order to set the perception threshold limits for both pitch and roll tilt motions, past literature was analysed to see the range of values taken into account while implementing such constraints. These are summarised in Table B.5. Based on the range of values mentioned along with the extensive study done in [45], threshold values in the range of 2-4 are generally considered. This ensures that all individuals are unable to perceive tilt below the threshold and that limited number of outliers are present. Based on this, values of 3, 2.6 deg/s were taken for pitch and roll perception thresholds respectively.

Table B.5: Perception threshold values from literature

Paper	Pitch threshold value [deg/s]	Roll threshold value [deg/s]
Gutridge et al. [46]	3.00	3.00
Reid et al. [35]	3.00	2.30
Khusro et al. [5]	3.60	3.00
Lamprecht et al. [41]	3.00	4.00
Telban et al. [33]	2.00	2.00
Koyuncu et al. [47]	2.04	2.04
Munir et al. [43]	6.00	-
Soyer et al. [42]	6.00	-
Nesti et al. [45]	-	6.30

C
Simulation Results

This appendix showcases additional simulation studies performed related to the explicit MPC controller and its functionalities. Selection of braking constraint parameters and the effect of changing these values on the cueing algorithm is elucidated. Further, influence of changing constraints and weights is shown along with an explicit MPC benchmark comparison. Lastly, additional hybrid MPC results are presented including a computation time study, emulator comparisons and virtual track simulations at varying speeds.

C.1. Selection of Braking Constraint Parameters

The selection of the braking constraint parameters takes place in the following manner:

To avoid the overshoot phenomenon when the platform is close to its limits the following condition is considered while tuning the weights:

$$\frac{c_v^2}{c_u} > 1 \tag{C.1}$$

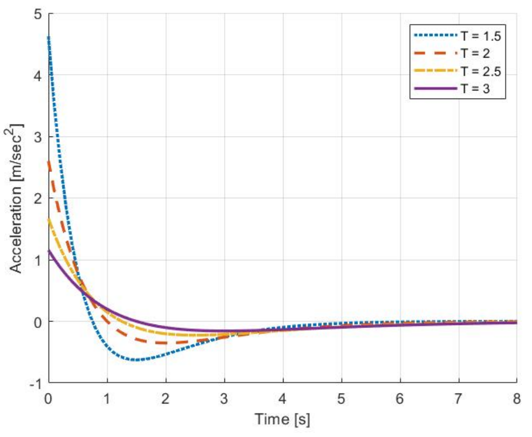
On the other hand, for selecting the $T_{brk,\theta}$ and $T_{brk,p}$ parameter values, an ideal simulator scenario was considered:

$$p(t) + c_v v(t)T + \frac{c_u u(t)T^2}{2} = x_{max} \tag{C.2}$$

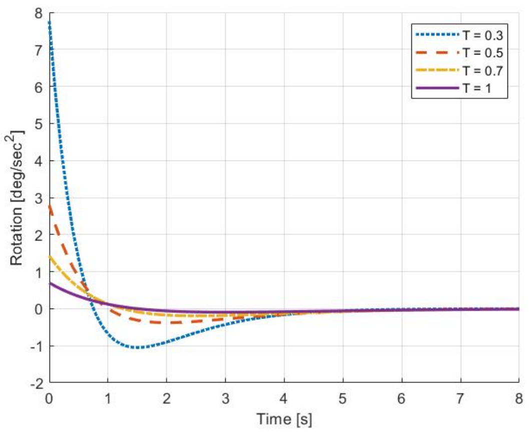
In state space form, this can be represented as a double integrator as follows:

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{u} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & \frac{-2}{c_u T^2} & \frac{-2c_v}{c_u T} \end{bmatrix} \cdot \begin{bmatrix} p \\ \dot{p} \\ u \end{bmatrix} \tag{C.3}$$

Using this representation, simulations were run for four sets of possible parameter values for both $T_{brk,\theta}$ and $T_{brk,p}$, the results of which can be seen in Figure C.1. It can be seen from the figures that different parameter values effect how the ideal simulator performs. Based on these simulation, 0.5 and 2.5 were the chosen parameter values for $T_{brk,\theta}$ and $T_{brk,p}$ respectively.



(a) Variation in parameter $T_{brk,p}$



(b) Variation in parameter $T_{brk,\theta}$

Figure C.1: Braking control law evolution for different parameter setting T

To further analyse the importance of these parameters and how their selection can vary the performance of the motion cueing algorithm, simulations were run with the developed algorithm; taking these four possible parameter values into account.

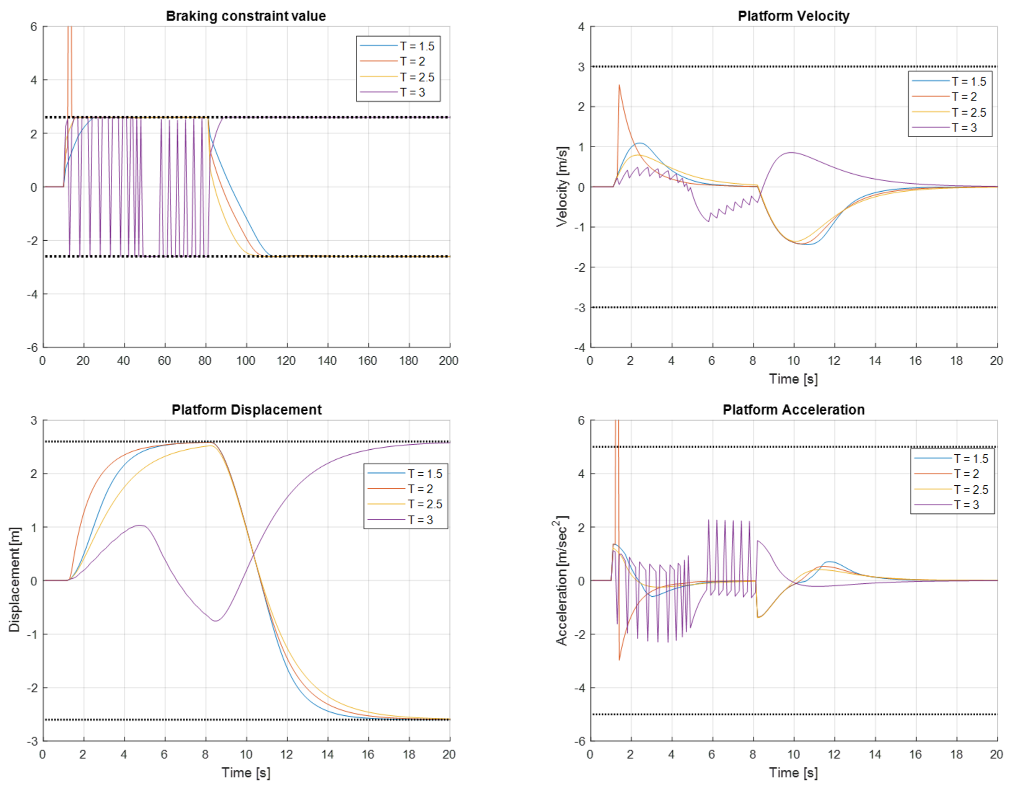


Figure C.2: Motion cueing performance with varying $T_{brk,p}$ parameter values

First, $T_{brk,p}$ was studied which consisted of using $1.5, 2, 2.5$ and 3 as the parameter values respectively. The results obtained with using values in the motion cueing algorithm can be in Figure C.2. It can be seen that the high parameter value of 3 is unstable from the braking constraint subplot. Along with this, the value of 2 fails to stay within the desired limits of the platform. Amongst 1.5 and 2.5, the parameter value of 2.5 provides better braking stability with lower platform velocity while allowing maximum workspace utilization to occur. Thus, the chosen value of 2.5 is correct. Second, $T_{brk,q}$ was studied which consisted of using $0.3, 0.5, 0.7$ and 1 as the parameter values respectively. The results obtained can be seen in Figure C.3. Using a high value of 1 or a low value of 0.3 gives unstable braking values and hence they are not recommended. Amongst 0.5 and 0.7, 0.5 is able to provide better stability and braking performance throughout the simulation. It stays within the limits and performs braking action when the tilt angle reaches/is about to reach its limits. Hence, the chosen value of 0.5 is best amongst the parameter values analysed.

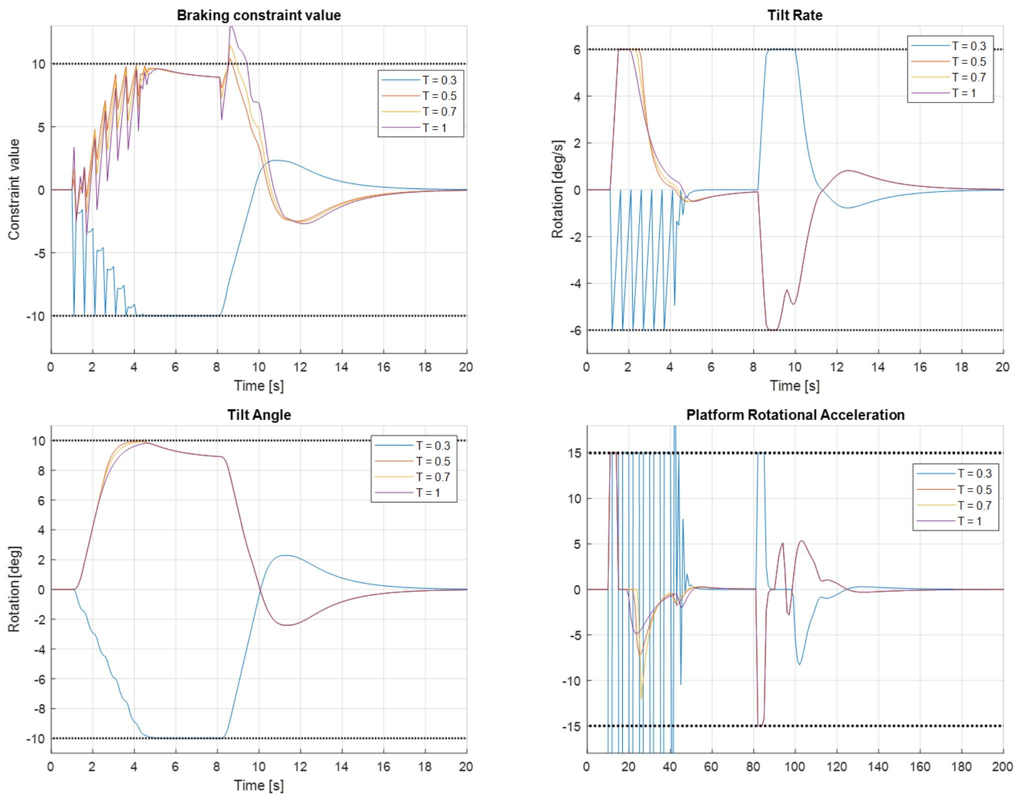


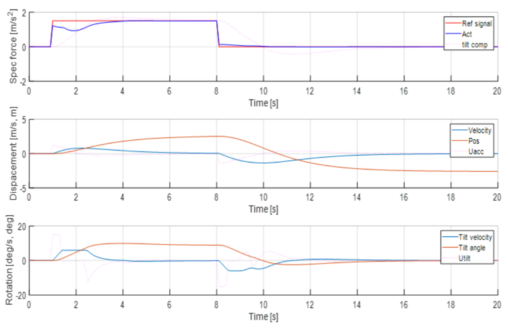
Figure C.3: Motion cueing performance with varying $T_{brk,q}$ parameter values

C.2. Explicit MPC Benchmark

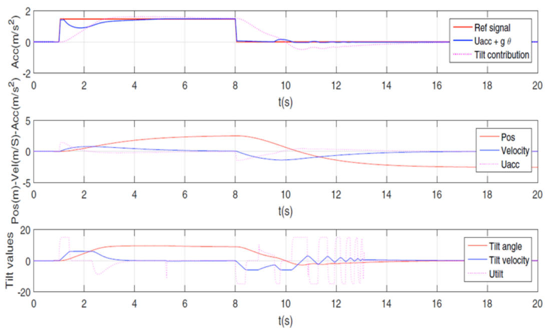
To ensure that the developed explicit MPC controller was correct in its formulation, a benchmark study was performed. This study used the same model formulation as explained in subsection 4.1.1 and is based on the work done by [6], [43]. The only difference from the developed explicit algorithm was in the constraint values used for the motion cueing states along with braking constraints added in. The parameter values chosen for the braking constraints were the same as described in Figure 4.1.1. Also, smaller control input weights of $1e-1$ were used. These changes are summarised below:

$$X = \begin{cases} -6deg/sec \leq \omega_p \leq 6deg/sec \\ -10deg \leq \theta_p \leq 10deg \\ -3m/sec \leq v_p \leq 3m/sec \\ -2.6m \leq r_p \leq 2.6m \\ -5m/sec^2 \leq a_{p,tran} \leq 5m/sec^2 \\ -15deg/sec^2 \leq a_{p,rot} \leq 15deg/sec^2 \\ -2.6m \leq c_v v_p T_{brk,p} + 0.5 c_u a_{p,tran} T_{brk,p}^2 \leq 2.6m \\ -10deg \leq c_w \omega_p T_{brk,\theta} + 0.5 c_u a_{p,rot} T_{brk,\theta}^2 \leq 10deg \end{cases} \tag{C.4}$$

This model uses more relaxed constraint values for platform displacement and tilt rate respectively. Apart from this, the remaining formulation was the same with the goal to track the output specific force. The same reference signals were analysed and compared with the actual results from the work done by the authors as seen in Figure C.4 and Figure C.5.

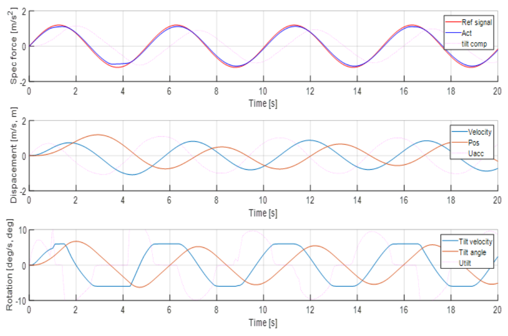


(a) Developed motion cueing algorithm

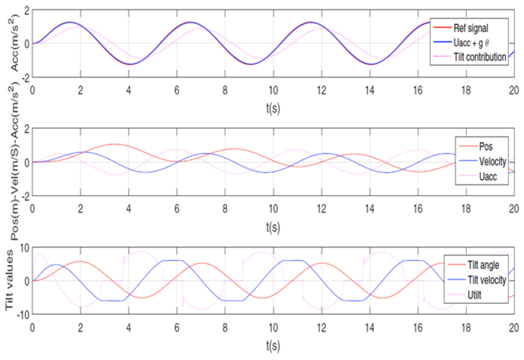


(b) Munir et al.[43]

Figure C.4: Step signal tracking performance comparison between created motion cueing algorithm and work done by [6], [43]



(a) Developed motion cueing algorithm



(b) Munir et al.[43]

Figure C.5: Sine wave tracking performance comparison between created motion cueing algorithm and work done by [6], [43]

The results show that the created motion cueing algorithm performs in the same manner as seen in [6], [43]. Thus, the developed motion cueing algorithm is created correctly for these set of constraints and can be further worked on for the purpose of this thesis with modifications.

C.3. Explicit MPC Parametric Studies

To analyse the created explicit controller in more detail, some parametric studies were performed to see the influence of certain parameters.

C.3.1. Influence of Constraints

This study was done to see the effect of adding/removing constraints from the system and what changes are observed in computation time for the explicit MPC. Constraints were also made more stringent to account for changes based on limits of the individual states. From the results obtained in Table C.1, it can be observed that with an increase in the number of constraints, the computation time increases rapidly with an exponential increase in the number of control regions. On the other hand, when the constraints are removed, it goes down. Upon making the limits more strict, computational load again increases.

Table C.1: Influence of constraints on computational load

Number of Constraints	Constraints	Control Regions	Computation Time [min]
5 (states + translational platform acceleration)	$-3.0\ deg/s \leq \omega_p \leq 3.0\ deg/s$ $-30\ deg \leq \theta_p \leq 30\ deg$ $-7.2\ m/s \leq v_p \leq 7.2\ m/s$ $-0.5\ m \leq r_p \leq 0.5\ m$ $-9.81\ m/s^2 \leq a_{p,tran} \leq 9.81\ m/s^2$	11131	~ 11
6 (states + both control inputs)	$-3.0\ deg/s \leq \omega_p \leq 3.0\ deg/s$ $-30\ deg \leq \theta_p \leq 30\ deg$ $-7.2\ m/s \leq v_p \leq 7.2\ m/s$ $-0.5\ m \leq r_p \leq 0.5\ m$ $-9.81\ m/s^2 \leq a_{p,tran} \leq 9.81\ m/s^2$ $-10\ deg/s^2 \leq a_{p,rot} \leq 10\ deg/s^2$	34887	~ 40
4 (all states except velocity + translational platform acceleration)	$-3.0\ deg/s \leq \omega_p \leq 3.0\ deg/s$ $-30\ deg \leq \theta_p \leq 30\ deg$ $-0.5\ m \leq r_p \leq 0.5\ m$ $-9.81\ m/s^2 \leq a_{p,tran} \leq 9.81\ m/s^2$	11120	~ 11
3 (all states except velocity)	$-3.0\ deg/s \leq \omega_p \leq 3.0\ deg/s$ $-30\ deg \leq \theta_p \leq 30\ deg$ $-0.5\ m \leq r_p \leq 0.5\ m$	9239	~ 9.1
3 (all states except velocity), stringent constraints	$-2.5\ deg/s \leq \omega_p \leq 2.5\ deg/s$ $-30\ deg \leq \theta_p \leq 30\ deg$ $-0.4\ m \leq r_p \leq 0.4\ m$	9182	~ 9.9

C.3.2. Influence of Braking Constraints on RMSE

Braking constraints not only decrease the platform velocity and tilt rate as the simulator approaches its physical displacement limits, but, in doing so it also reduces peaks and drops that might occur due to workspace limitations. This has a positive effect on the overall tracking performance which can be seen in the table below:

Table C.2: Influence of braking constraints on RMSE

Type of reference signal	Amplitude of reference signal	RMSE	
		Without braking constraints	With braking constraints
Sine wave	0.5	0.4957	0.0345
	1	0.8150	0.2933
	2	1.5828	1.1747
	Combined	0.6265	0.0844
Step signal (from 1-4 seconds)	0.5	0.4409	0.0480
	1	0.7126	0.3101
	2	1.3368	1.0070

C.3.3. Influence of Weights

This study was performed to see the effect of adding additional weights on the computation time of the explicit controller. Small weights of $1e-1$ were added on the states which were previously set to 0. From the results obtained in Table C.3, it can be see that with each additional weight added on the states, the overall computation time goes down with a decrease in the number of control regions.

Table C.3: Influence of weights on computational load

Weights	Control Regions	Computation Time [min]
$a_{spec} = 1$ States = [0 0 0 0] Control inputs = [1e-3 1e-3]	11131	~ 11
$a_{spec} = 1$ States = [1e-1 0 0 0] Control inputs = [1e-3 1e-3]	9269	~ 10.2
$a_{spec} = 1$ States= [1e-1 1e-1 0 0] Control inputs = [1e-3 1e-3]	8921	~ 9
$a_{spec} = 1$ States = [1e-1 1e-1 1e-1 0] Control inputs = [1e-3 1e-3]	8655	~ 8.8
$a_{spec} = 1$ States= [1e-1 1e-1 1e-1 1e-1] Control inputs = [1e-3 1e-3]	8524	~ 8.7

C.4. Hybrid MPC Performance

C.4.1. 2 DOF Motion Cueing Algorithm

Constant Washout Weights

Using the best washout weights defined in Table 5.3 in Chapter 5, the three hybrid models can be evaluated and compared against the original implicit MPC model. To check if there is any improvement in overall computation time, mean iterations over the entire simulation duration are computed and compared. For the simulations, $N_{p,eMPC} = 5, T_{s,eMPC} = 0.1s$ and $N_{p,iMPC} = 50$, and $T_{s,iMPC} = 0.01s$ is used. The results obtained for the defined signals and hybrid models can be seen in the table below:

Table C.4: Mean iteration values for all four motion cueing models with best constant washout weights

Type of reference signal	Amplitude of reference signal	Sampling time [s]	Mean iterations			
			iMPC	iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
Sine wave	0.5	iMPC = 0.01 eMPC = 0.1	0.2987	0.2767	0.2108	0.1998
	1		1.2697	1.1768	1.1249	1.4016
	2		1.5744	1.4515	1.3477	1.4326
	Combined		3.1089	2.6304	2.4915	2.5165
Step signal (from 1-4 seconds)	1		2.0829	1.9311	1.8811	1.7393
	2		1.3357	1.3427	1.2368	1.2547
Step+Sine	0.5		1.5352	1.4853	1.4288	1.1089
	1		1.6417	1.4683	1.4113	1.6652

It can be observed that the hybrid model with first explicit MPC control input gives least mean iteration values compared to the other three models evaluated. However, from the error values, the hybrid model with all explicit MPC control inputs provides higher average percentage difference from the implicit algorithm. This is due to larger improvements in particular test scenarios rather than showing improvements at a uniform rate across all scenarios. Further, the RMSE values present in Table C.6 show that the tracking remains consistent throughout with no particular model showing much better performance in the test cases analysed.

Table C.5: Absolute error values with iMPC for all hybrid models with best constant washout weights

Type of reference signal	Amplitude of reference signal	Sampling time [s]	Absolute Error % (with iMPC)		
			iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
Sine wave	0.5	iMPC = 0.01 eMPC = 0.1	7.36	29.42	33.11
	1		7.31	11.40	10.38
	2		7.80	14.40	9.00
	Combined		15.39	19.86	19.05
Step signal (from 1-4 seconds)	1		7.28	9.69	16.49
	2		0.52	7.40	6.06
Step+Sine	0.5		3.25	6.93	27.76
	1		10.56	14.03	1.43
Average Error (%)			7.44	14.14	15.41

Table C.6: Specific force RMSE values for all four motion cueing with best constant washout weights

Type of reference signal	Amplitude of reference signal	Sampling time [s]	RMSE (f_{spec})			
			iMPC	iMPC (with control trajectory prediction)	Hybrid (with first eMPC control input)	Hybrid (with all eMPC control inputs)
Sine wave	0.5	iMPC = 0.01 eMPC = 0.1	0.0024	0.0025	0.0028	0.0030
	1		0.0473	0.0477	0.0485	0.0481
	2		1.1459	1.1462	1.1479	1.1469
	Combined		0.0742	0.0742	0.0742	0.0747
Step signal (from 1-4 seconds)	1		0.2536	0.2540	0.2533	0.2553
	2		0.9871	0.9871	0.9871	0.9871
Step+Sine	0.5		0.0341	0.0335	0.0336	0.0337
	1		0.2154	0.2157	0.2168	0.2173

Adaptive Weights

Additional simulations with adaptive weights for the 2 DOF motion cueing algorithm can be seen in Figure C.6 and Figure C.7. In these simulations, sine and step waves are simulated seperately and it can be observed that good tracking performance is maintained throughout the simulation.

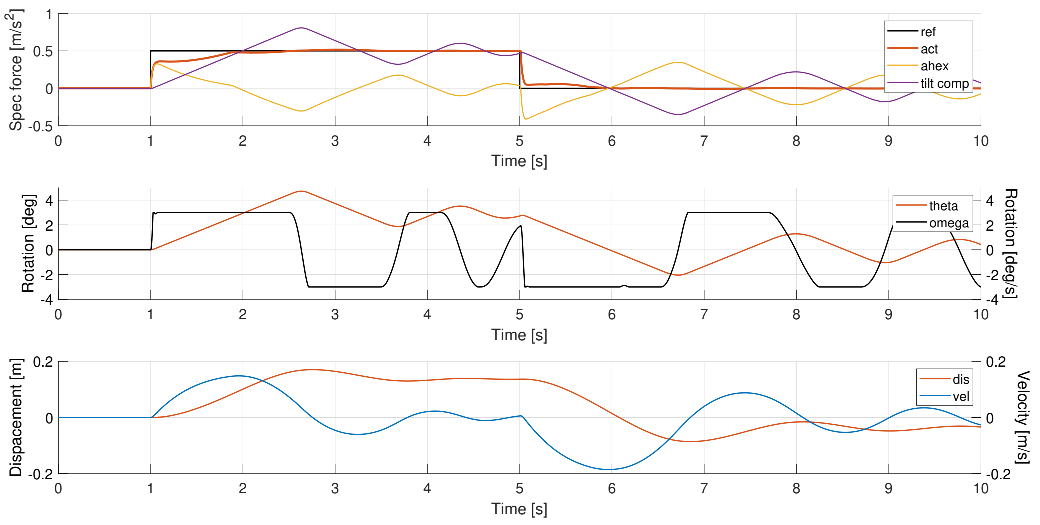


Figure C.6: Motion cueing performance for step signal with adaptive weights

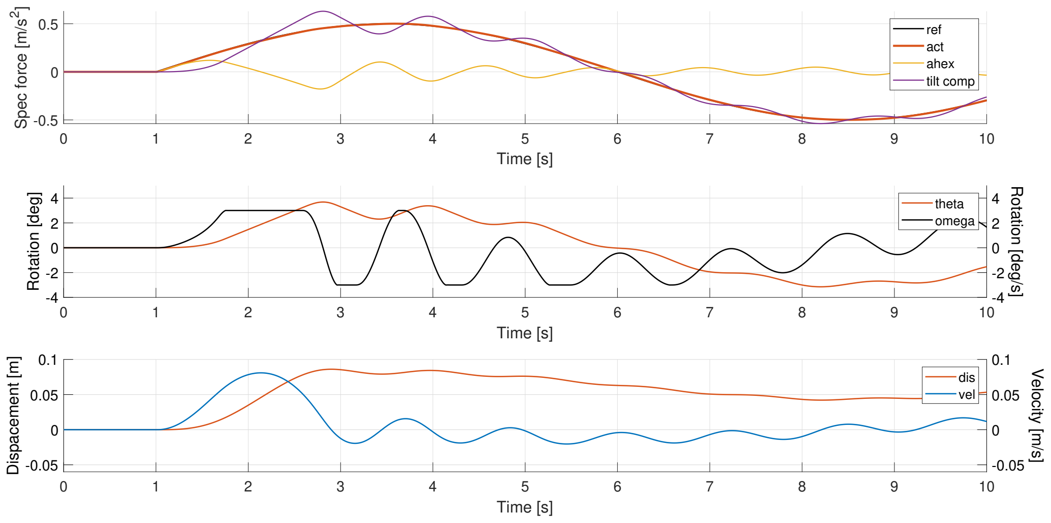
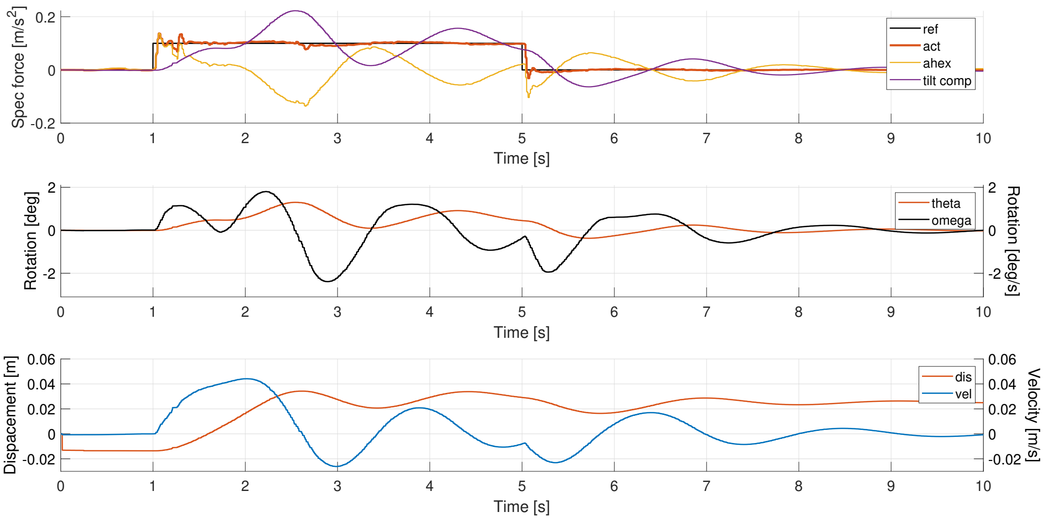


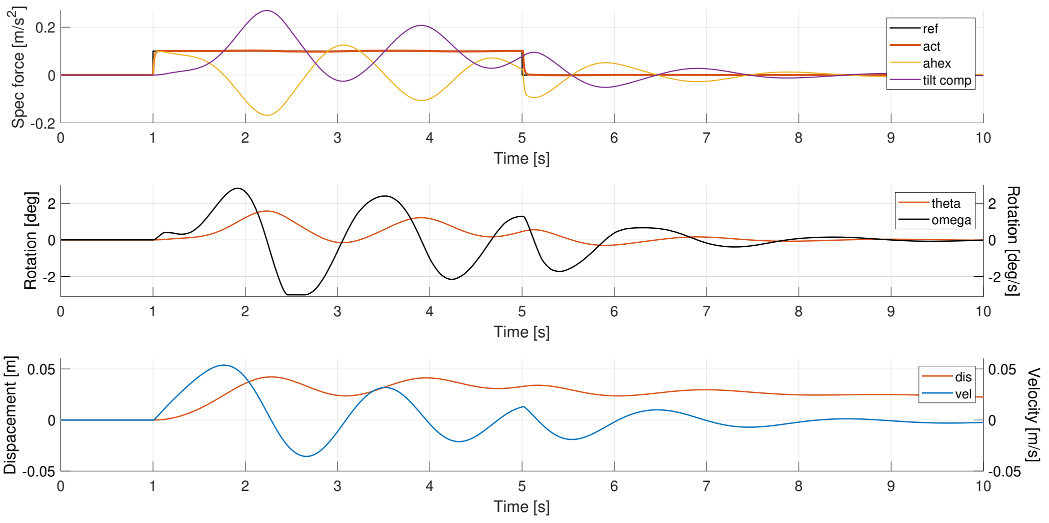
Figure C.7: Motion cueing performance for sine wave with adaptive weights

C.4.2. Emulator Simulations

In this section, step signal and sine wave comparisons are shown between the emulator and offline simulations.

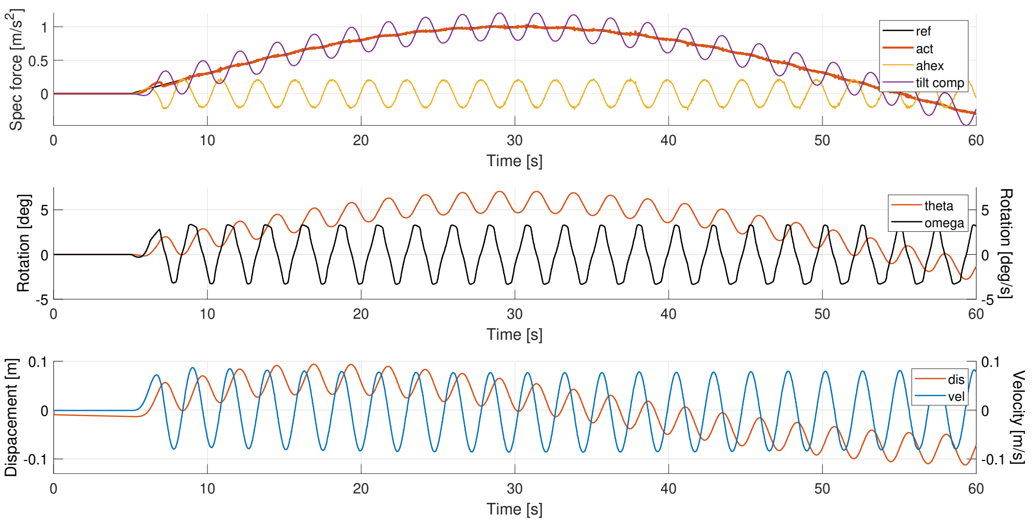


(a) Emulator

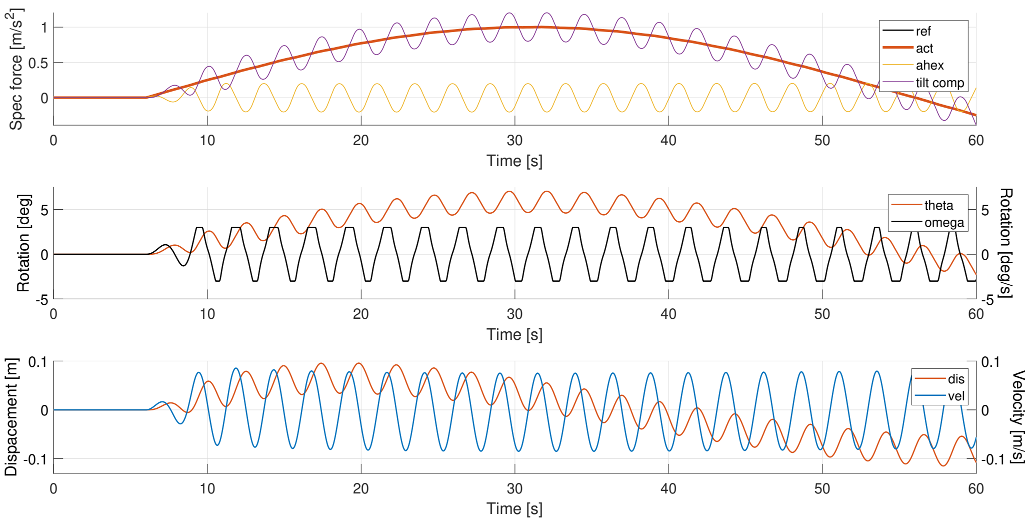


(b) Offline simulation

Figure C.8: Emulator performance comparison for step signal of amplitude 0.1



(a) Emulator



(b) Offline simulation

Figure C.9: Emulator performance comparison for sine wave of amplitude 1

Vehicle parameters of the Tesla Model S used in IPG CarMaker

Table C.7: Vehicle parameters

Parameter	Value
Mass of the vehicle (m)	2108 Kg
Distance of CoG from the rear axle (l_r)	1.500 m
Distance of CoG from the front axle (l_f)	1.470 m
Vehicle moment of inertia about z-axis $(I_z z)$	1585.3 kgm^2
Height of CoG	0.545 m
Coefficient of friction	1
Track Width (B)	1.70m

Virtual track simulation at 80 km/h

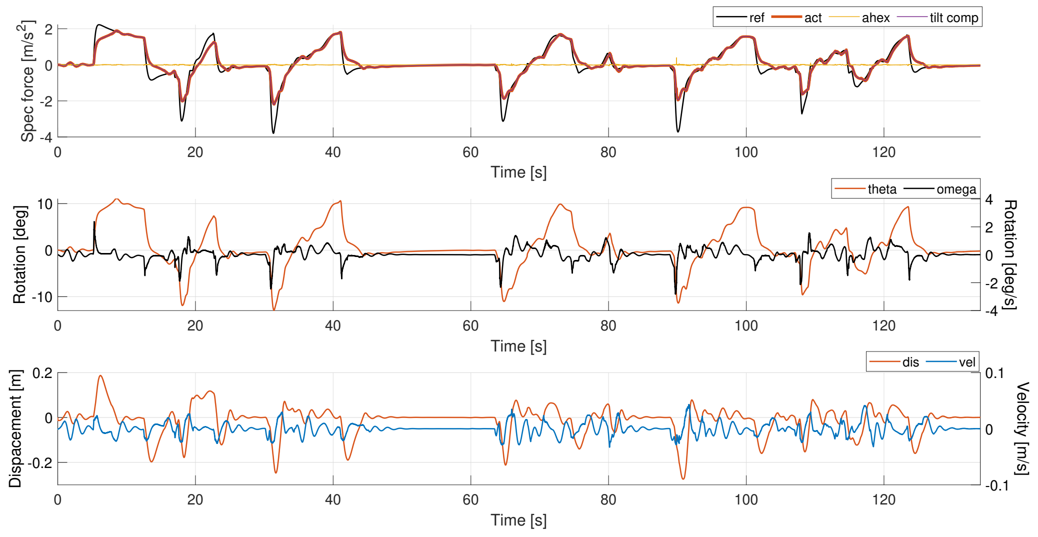


Figure C.10: Longitudinal motion results for Hockenheim track simulation at 80km/h

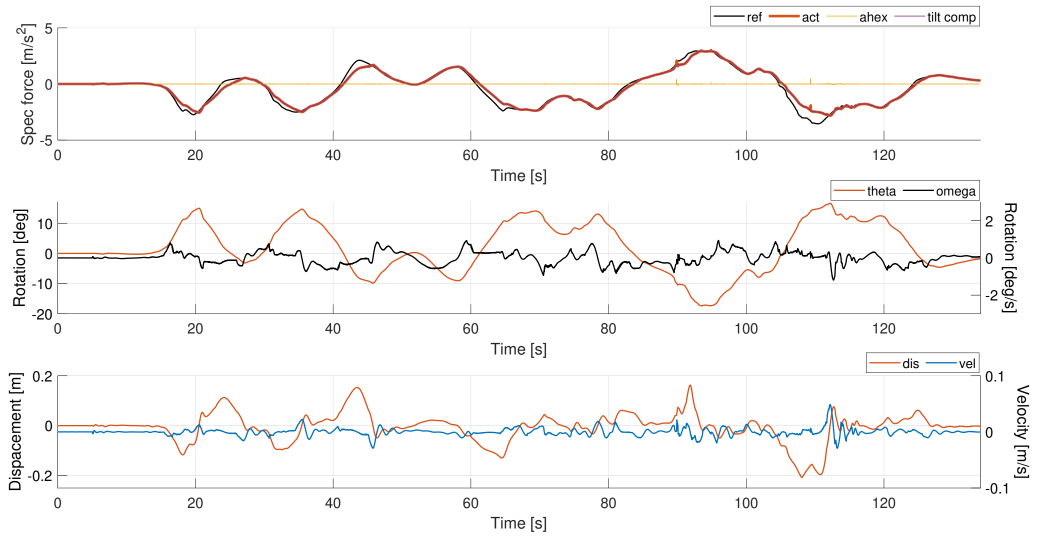


Figure C.11: Lateral motion results for Hockenheim track simulation at 80km/h

Virtual track simulation at 100 km/h

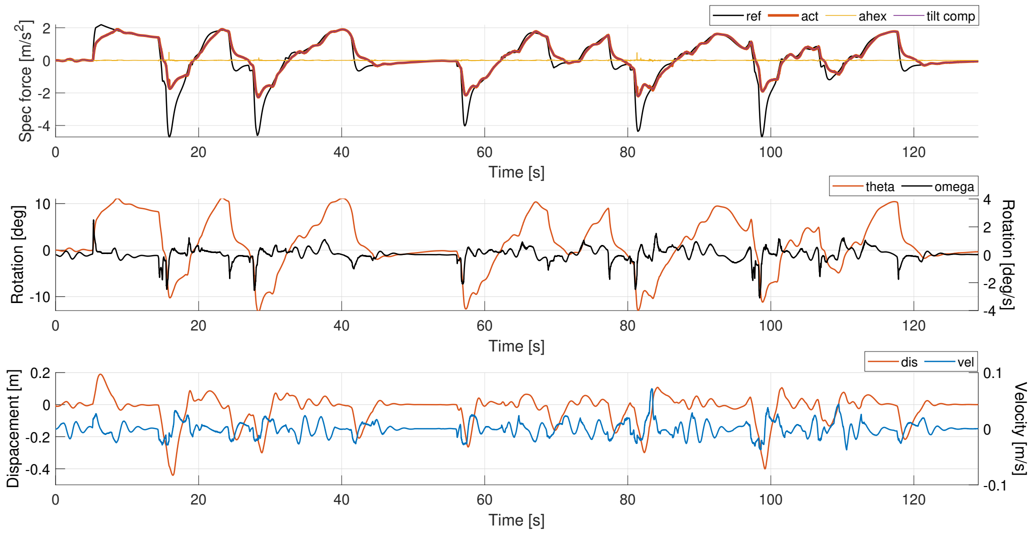


Figure C.12: Longitudinal motion results for Hockenheim track simulation at 100km/h

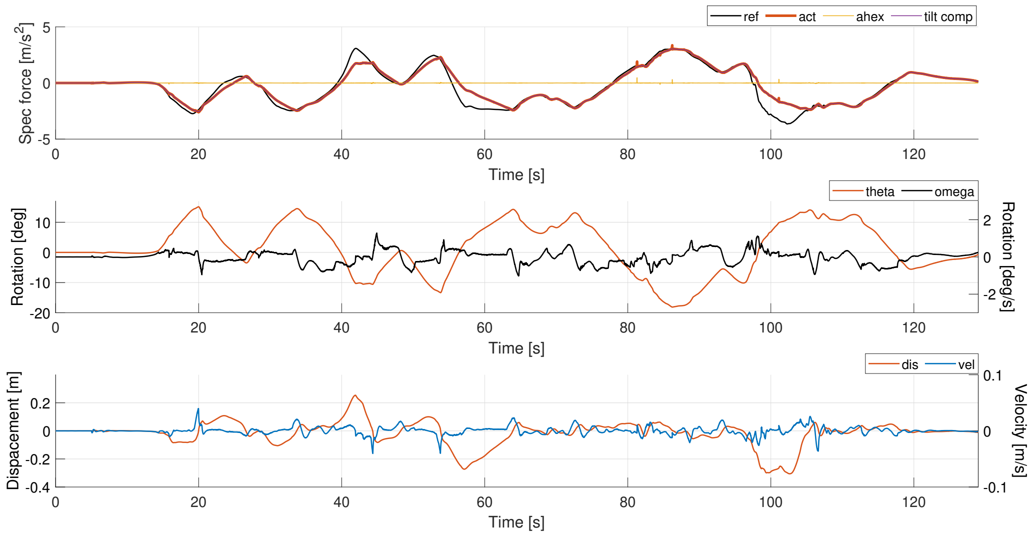


Figure C.13: Lateral motion results for Hockenheim track simulation at 100km/h

D

Toolbox Information

D.1. ACADO Toolkit

The ACADO toolkit was used to solve the nonlinear MPC problem. The toolkit allows easy formulation of the required MPC problem with different constraints and bounds. It has a user friendly MATLAB interface which is used to develop, generate and export a highly efficient self-contained C-code for the respective MPC problem. The toolkit also comes with various solvers which provide solutions with low computational expense and high accuracy [50].

For the developed implicit MPC motion cueing algorithm, the toolkit was used to generate the s-function which was later used in simulating the desired test cases. While formulating the problem, there were certain parameters that could be manipulated within the ACADO toolkit which can help in improving computation time and performance of the MPC controller. These settings are summarised in the table below:

Table D.1: ACADO solver settings

Parameter	ACADO Solver Setting
Hessian approximation	Gauss Newton
Discretization type	Multiple shooting
Integration type	Implicit Rung-Kutta integrator of order 2
Number of integration steps	3N
Levenberg-Marquadt regularization	1e-4
QP solver	qpOASES
Sparse QP solution	Full condensing
Maximum number of iterations	200
Hotstart QP	Yes

The toolkit uses the Real-Time Iteration (RTI) technique to solve the nonlinear MPC problem. This approach uses the Sequential Quadratic Programming (SQP) scheme to compute the solution of the optimization problem at each iteration. The reader is directed to [51] to know more about the basics of RTI and SQP.

Selection of the discretization method took place by comparing single and multiple shooting methods. Single shooting is a relatively simple approach for transcribing the optimal control problem. It works well for simple problems but fails when additional complexity is introduced into the system which can be in the form of nonlinearities. On the the hand, multiple shooting works by breaking up the trajectory into segments and solves each of them separately [52]. It is able to deal with nonlinearities and complicated problems which is why it is the chosen discretization method in this study.

Apart from this, the Gauss Newton method was chosen to approximate the Hessian as it only requires calculation of the 1^{st} order derivatives of the nonlinear output function. This helps in reducing the overall calculations needed in comparison with the complete Hessian. Also, the approximation from this method always gives a positive-semidefinite Hessian matrix which can be made positive-definite by adding a small Levenberg-Marquardt regularization term [39].

The QP solver was chosen based on the preferred implementation method from two types of algorithms, namely; interior point method and active set method. The interior point method generally converges in small number of iterations by using a weighted barrier function to replace the inequality constraints. On the other hand, the active set approach uses the warm start capabilities and current active set constraints to provide a solution to the QP. This solution can sometimes require more iterations but the overall computation time is usually less. Based on this functionality of the active set approach, a QP solver using this was selected in the ACADO toolkit. This solver is known as the qpOASES QP solver which assumes the Hessian matrix to be dense and performs a parametric active set method. Further, comparison with other commercial QP solvers has shown that it can outperform them in terms of reliability and computation time [39], [53].

Selection of parameters such as number of iterations, integration steps and hotstart was done by looking at the computation time and solver accuracy results. With high number of iterations, better accuracy can be obtained but it leads to high computation load as well. Thus, based on the requirements of the developed motion cueing algorithm, maximum iterations were limited to 200 with hotstart and 3N integration steps (3 integration steps per iteration for the entire horizon). The integration type was chosen as implicit Rung-Kutta integrator of order 2 due to its ability to provide better stability and numerical accuracy for a given computational effort [51].

The simulink model used to run the motion cueing algorithm can be seen in Figure D.1.

D.2. Multi-Parametric Toolbox

The Multi-Parametric Toolbox was used in computing the explicit control law for the motion cueing algorithm. The toolbox is open source and has a user friendly MATLAB interface [44].

The toolbox allowed easy formulation of the desired motion cueing problem. This was done by first defining the states followed by the constraints and weights. Lastly, MPC settings such as prediction horizon and move blocking was defined after which an MPC controller was generated using the following command:

```
ctrl = MPCController(model, N_prediction)        (D.1)
```

Here, `model` refers to the LTI system containing the state space equations and their corresponding constraints and weights.

Now that the explicit controller was defined with all the required parameters, the controller was converted into an explicit controller and used to generate the explicit control laws by the following command:

```
expctrl = ctrl.toExplicit()        (D.2)
```

These control regions containing the control laws were stored in the form of a mex file using the export functionality of the Multi-Parametric Toolbox. This mex file was then used in the hybrid MPC motion cueing algorithm to provide the initial guess.

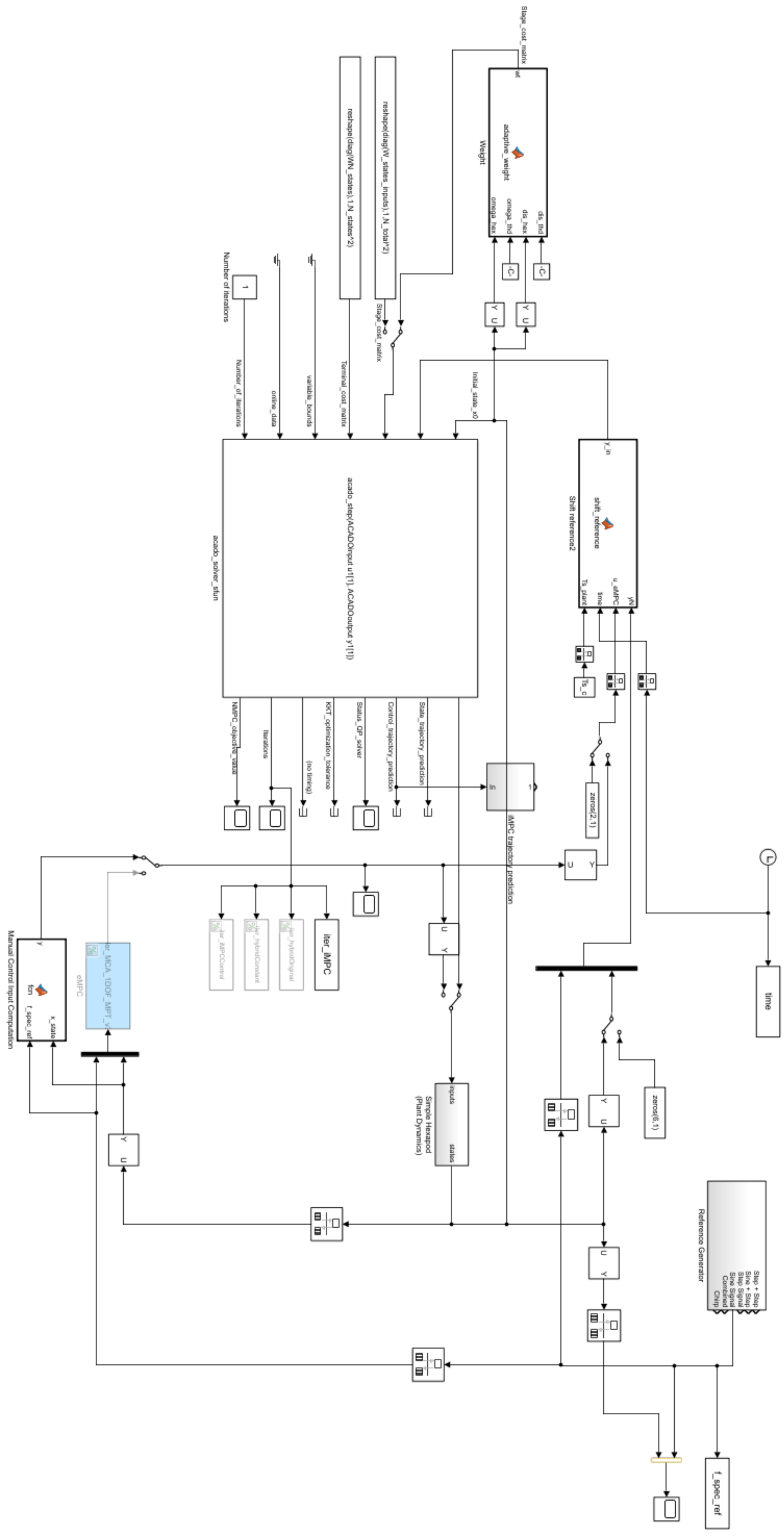


Figure D.1: Hybrid motion cueing algorithm simulink model

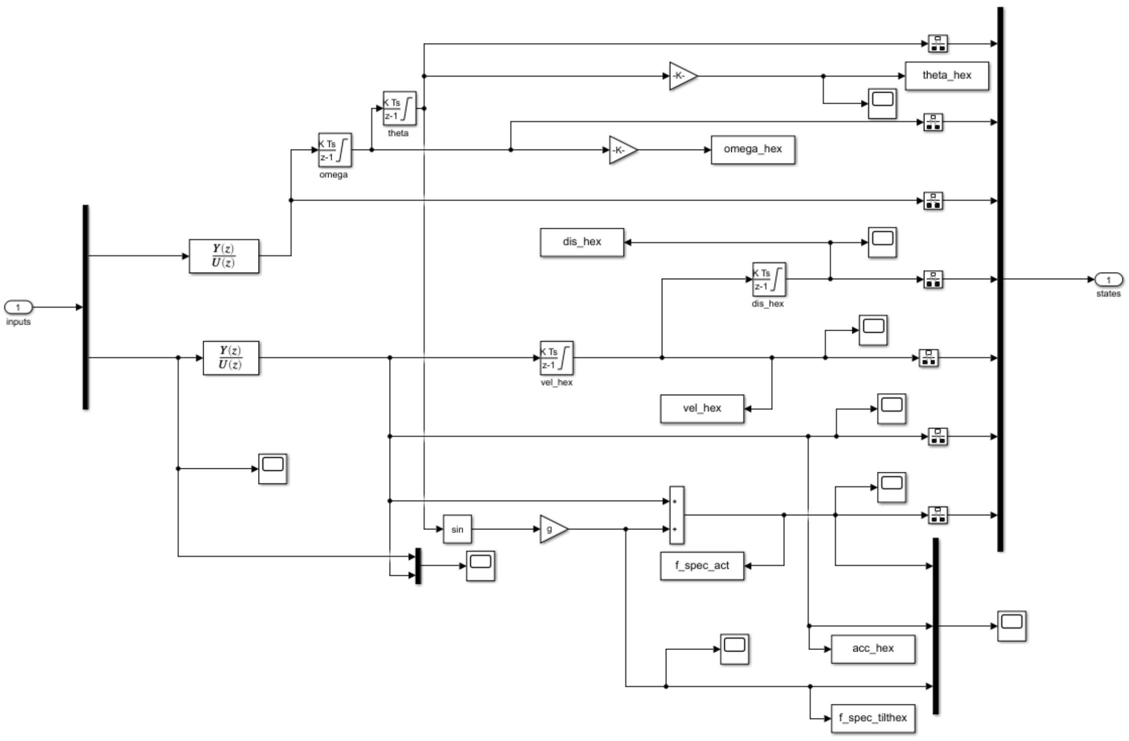


Figure D.2: Simplified actuator dynamics for 2 DOF motion cueing algorithm

References

[1] S. Casas, R. Olanda, and N. Dey, "Motion cueing algorithms: A review," *International Journal of Virtual and Augmented Reality*, vol. 1, 1 2016, ISSN: 2473-537X. DOI: `10.4018/ijvar.2017010107`.

[2] M. Furqan, M. Suhaib, and N. Ahmad, "Studies on stewart platform manipulator: A review," *Journal of Mechanical Science and Technology*, vol. 31, 9 2017, ISSN: 1738494X. DOI: `10.1007/s12206-017-0846-1`.

[3] G. Reymond and A. Kemeny, "Motion cueing in the renault driving simulator," *Vehicle System Dynamics*, vol. 34, 4 2000, ISSN: 00423114. DOI: `10.1076/vesd.34.4.249.2059`.

[4] 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*, vol. 25, 2 2017, ISSN: 10636536. DOI: `10.1109/TCST.2016.2560120`.

[5] Y. R. Khusro, Y. Zheng, M. Grottoli, and B. Shyrokau, "Mpc-based motion-cueing algorithm for a 6-dof driving simulator with actuator constraints," *Vehicles*, vol. 2, 4 2020. DOI: `10.3390/vehicles2040036`.

[6] Z. Fang and A. Kemeny, "Explicit mpc motion cueing algorithm for real-time driving simulator," vol. 2, 2012. DOI: `10.1109/IPEMC.2012.6258965`.

[7] M. O. Ward, G. Grinstein, and D. Keim, *- Human Perception and Information Processing*. 2020. DOI: `10.1201/b18379-7`.

[8] N. Schmitz, C. Spranger, and K. Berns, "3d audio perception system for humanoid robots," 2009. DOI: `10.1109/ACHI.2009.24`.

[9] M. L. Wolbarsht, "Visual perception. tom n. cornsweet," *The Quarterly Review of Biology*, vol. 46, 1 1971, ISSN: 0033-5770. DOI: `10.1086/406812`.

[10] M. Bruschetta, K. N. de Winkel, E. Mion, P. Pretto, A. Beghi, and H. H. Bülthoff, "Assessing the contribution of active somatosensory stimulation to self-acceleration perception in dynamic driving simulators," *PLoS ONE*, vol. 16, 11 November 2021, ISSN: 19326203. DOI: `10.1371/journal.pone.0259015`.

[11] G. Robles-De-La-Torre, "The importance of the sense of touch in virtual and real environments," *IEEE Multimedia*, vol. 13, 3 2006, ISSN: 1070986X. DOI: `10.1109/MMUL.2006.69`.

[12] C. Pfeiffer, A. Serino, and O. Blanke, "The vestibular system: A spatial reference for bodily self-consciousness," *Frontiers in Integrative Neuroscience*, vol. 8, APR 2014, ISSN: 16625145. DOI: `10.3389/fnint.2014.00031`.

[13] A. Kemeny and F. Panerai, "Evaluating perception in driving simulation experiments," *Trends in Cognitive Sciences*, vol. 7, 1 2003, ISSN: 13646613. DOI: `10.1016/S1364-6613(02)00011-6`.

[14] A. Stratulat, V. Roussarie, J. L. Vercher, and C. Bourdin, "Improving the realism in motion-based driving simulators by adapting tilt-translation technique to human perception," 2011. DOI: `10.1109/VR.2011.5759435`.

[15] C. Weiss, "Control of a dynamic driving simulator: Time-variant motion cueing algorithms and prepositioning," *Institut für Verkehrsführung und Fahrzeugsteuerung*, 2006.

[16] N. Nise, M. Perez, A. Perez, *et al.*, *Control Systems Engineering 7th Edition*. 2014, vol. 7th.

[17] C. Seehof, U. Durak, and H. Duda, "Objective motion cueing test - experiences of a new user," 2014. DOI: `10.2514/6.2014-2205`.

[18] M. A. Nahon and L. D. Reid, "Simulator motion-drive algorithms - a designer's perspective," *Journal of Guidance, Control, and Dynamics*, vol. 13, 2 1990, ISSN: 07315090. DOI: `10.2514/3.20557`.

[19] A. Beghi, M. Bruschetta, and F. Maran, “A real time implementation of mpc based motion cueing strategy for driving simulators,” 2012. DOI: 10.1109/CDC.2012.6426119.

[20] M. Aminzadeh, A. Mahmoodi, and M. Sabzehparvar, “Optimal motion-cueing algorithm using motion system kinematics,” *European Journal of Control*, vol. 18, 4 2012, ISSN: 09473580. DOI: 10.3166/EJC.18.363-375.

[21] N. J. I. Garrett and M. C. Best, “Driving simulator motion cueing algorithms–a survey of the state of the art,” *Proceedings of the 10th International Symposium on Advanced Vehicle Control (AVEC)*, 2010.

[22] B. J. Rawlings, Q. D. Mayne, and M. M. Diehl, *Model predictive control: Theory, Computation, and Design*. 2018, vol. 1.

[23] M. Derbeli, A. Charaabi, O. Barambones, and C. Napole, “High-performance tracking for proton exchange membrane fuel cell system pemfc using model predictive control,” *Mathematics*, vol. 9, 11 2021, ISSN: 22277390. DOI: 10.3390/math9111158.

[24] L. Wang, “Model predictive control: Design and implementation using matlab (t-3),” 2009. DOI: 10.1109/acc.2009.5159781.

[25] J. Drgona and L. Helsen, “Different problem classes and solution techniques for model predictive building control,” 2018.

[26] I. Maurović, M. Baotić, and I. Petrović, “Explicit model predictive control for trajectory tracking with mobile robots,” 2011. DOI: 10.1109/AIM.2011.6027140.

[27] A. Alessio and A. Bemporad, “A survey on explicit model predictive control,” vol. 384, 2009. DOI: 10.1007/978-3-642-01094-1_29.

[28] P. Bemporad, “Explicit model predictive control,” in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds. London: Springer London, 2013, pp. 1–9, ISBN: 978-1-4471-5102-9. DOI: 10.1007/978-1-4471-5102-9_10-1. [Online]. Available: https://doi.org/10.1007/978-1-4471-5102-9_10-1.

[29] M. N. Zeilinger, C. N. Jones, and M. Morari, “Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization,” *IEEE Transactions on Automatic Control*, vol. 56, 7 2011, ISSN: 00189286. DOI: 10.1109/TAC.2011.2108450.

[30] M. Jost and M. Mönnigmann, “Accelerating online mpc with partial explicit information and linear storage complexity in the number of constraints,” 2013. DOI: 10.23919/ecc.2013.6669259.

[31] Y. Zheng, B. Shyrokau, T. Keviczky, M. A. Sakka, and M. Dhaens, “Curve tilting with nonlinear model predictive control for enhancing motion comfort,” *IEEE Transactions on Control Systems Technology*, 2021, ISSN: 15580865. DOI: 10.1109/TCST.2021.3113037.

[32] Y. Zheng and B. Shyrokau, “A real-time nonlinear mpc for extreme lateral stabilization of passenger vehicles,” 2019. DOI: 10.1109/ICMECH.2019.8722930.

[33] R. J. Telban and F. Cardullo, “Motion cueing algorithm development: Human-centered linear and nonlinear approaches,” *NASA TechReport*, May 2005.

[34] N. J. Garrett and M. C. Best, “Model predictive driving simulator motion cueing algorithm with actuator-based constraints,” *Vehicle System Dynamics*, vol. 51, 8 2013, ISSN: 00423114. DOI: 10.1080/00423114.2013.783219.

[35] L. D. Reid and M. A. Nahon, “Flight simulation motion-base drive algorithms: Part 1. developing and testing equations,” *UTIAS Report, No. 296*, 1985.

[36] M. Dagdelen, G. Reymond, A. Kemeny, M. Bordier, and N. Maïzi, “Model-based predictive motion cueing strategy for vehicle driving simulators,” *Control Engineering Practice*, vol. 17, 9 2009, ISSN: 09670661. DOI: 10.1016/j.conengprac.2009.03.002.

[37] H. Asadi, S. Mohamed, C. P. Lim, S. Nahavandi, and E. Nalivaiko, “Semicircular canal modeling in human perception,” *Reviews in the Neurosciences*, vol. 28, 5 2017, ISSN: 03341763. DOI: 10.1515/revneuro-2016-0058.

[38] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl, “Auto-generated algorithms for nonlinear model predictive control on long and on short horizons,” 2013. DOI: 10.1109/CDC.2013.6760692.

[39] M. Katliar, “Optimal control of motion simulators,” *University of Freiburg - Faculty of Engineering*, 2020.

[40] A. Mohammadi, H. Asadi, S. Mohamed, K. Nelson, and S. Nahavandi, “Optimizing model predictive control horizons using genetic algorithm for motion cueing algorithm,” *Expert Systems with Applications*, vol. 92, 2018, ISSN: 09574174. DOI: 10.1016/j.eswa.2017.09.004.

[41] A. Lamprecht, D. Steffen, K. Nagel, J. Haecker, and K. Graichen, “Online model predictive motion cueing with real-time driver prediction,” *IEEE Transactions on Intelligent Transportation Systems*, 2021, ISSN: 1524-9050. DOI: 10.1109/tits.2021.3114003.

[42] M. Soyer, S. Olaru, and Z. Fang, “Motion cueing control design based on a nonlinear mpc algorithm,” *IFAC-PapersOnLine*, vol. 54, 6 2021, ISSN: 24058963. DOI: 10.1016/j.ifacol.2021.08.567.

[43] S. Munir, M. Hovd, Z. Fang, S. Olaru, and A. Kemeny, “Complexity reduction in motion cueing algorithm for the ultimate driving simulator,” vol. 50, 2017. DOI: 10.1016/j.ifacol.2017.08.2256.

[44] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, “Multi-Parametric Toolbox 3.0,” in *Proc. of the European Control Conference*, http://control.ee.ethz.ch/~mpt, Zürich, Switzerland, Jul. 2013, pp. 502–510.

[45] A. Nesti, C. Masone, M. Barnett-Cowan, P. R. Giordano, H. H. Bülthoff, and P. Pretto, “Roll rate thresholds and perceived realism in driving simulation,” 2012.

[46] J. Gutridge, “Three degree-of-freedom simulator motion cueing using classical washout filters and acceleration feedback,” *Virginia Polytechnic Institute and State University*, 2004.

[47] A. B. Koyuncu, E. Ercelik, E. Comulada-Simpson, J. Venrooij, M. Kaboli, and A. Knoll, “A novel approach to neural network-based motion cueing algorithm for a driving simulator,” 2020. DOI: 10.1109/IV47402.2020.9304825.

[48] H. Asadi, S. Mohamed, C. P. Lim, and S. Nahavandi, “A review on otolith models in human perception,” *Behavioural Brain Research*, vol. 309, 2016, ISSN: 18727549. DOI: 10.1016/j.bbr.2016.03.043.

[49] P. Bosetti, M. D. Lio, and A. Saroldi, “On the human control of vehicles: An experimental study of acceleration,” *European Transport Research Review*, vol. 6, 2 2014, ISSN: 18668887. DOI: 10.1007/s12544-013-0120-2.

[50] B. Houska, H. Ferreau, and M. Diehl, “ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization,” *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[51] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, “From linear to nonlinear mpc: Bridging the gap via the real-time iteration,” *International Journal of Control*, vol. 93, 1 2020, ISSN: 13665820. DOI: 10.1080/00207179.2016.1222553.

[52] M. Kelly, “Transcription methods for trajectory optimization: A beginners tutorial,” Jul. 2017.

[53] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, “Qpoases: A parametric active-set algorithm for quadratic programming,” *Mathematical Programming Computation*, vol. 6, 4 2014, ISSN: 18672957. DOI: 10.1007/s12532-014-0071-1.