computers

MDPI

*Article*

# Enhancing MPC-Based MCA Through Deep Learning for Adaptive Tuning

**Sari Al-serri [1,*], Mohammad Reza Chalak Qazani [2,3], Shady Mohamed [1], Saeid Nahavandi [4] and Houshyar Asadi [1,*]**

[1] Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Geelong, VIC 3216, Australia; shady.mohamed@deakin.edu.au
[2] College of Science and Engineering, James Cook University, Townsville, QLD 4811, Australia; mohammadreza.chalakqazani@jcu.edu.au or m.r.chalakqazani@gmail.com
[3] Faculty of Computing and Information Technology (FCIT), Sohar University, Sohar 311, Oman
[4] School of Engineering, Swinburne University of Technology, Hawthorn, VIC 3122, Australia; saeid.nahavandi@ieee.org
* Correspondence: salserri@deakin.edu.au (S.A.-s.); houshyar.asadi@deakin.edu.au (H.A.)

## Abstract

High-fidelity motion cueing in driving simulators is essential for delivering a realistic and immersive user experience. However, the trade-off between motion accuracy and computational efficiency often hinders achieving this. Fixed-horizon Model Predictive Control (MPC)-based Motion Cueing Algorithm (MCA) frameworks frequently struggle to adapt to rapid dynamic changes in vehicle behaviour, resulting in suboptimal simulator responses. Their reliance on worst-case horizon tuning can result in inefficient platform usage and increased computational load, limiting computational efficiency and practical deployment. This study presents an adaptive MPC-based MCA designed to enhance the fidelity of motion platforms used in vehicle dynamic simulations. The proposed method dynamically adjusts the MPC prediction horizon to improve overall simulation performance while minimising motion sensation error. Within the simulation environment, the prediction horizon is adaptively updated at each simulated control step according to recent tracking-performance metrics, enabling responsiveness to varying vehicle dynamic models and driving scenarios. The system was developed and implemented using Python and MATLAB environments, with Long Short-Term Memory (LSTM) networks employed to enhance the adaptability and precision of prediction horizon adjustments. Due to safety constraints, the proposed framework was evaluated exclusively within a simulation environment and compared against both classical MPC-based MCA and RL MPC-based MCA. Experimental results demonstrate that the proposed adaptive framework improves workspace utilisation and substantially reduces computational load compared with the classical and RL-based MPC-based MCA approaches, while maintaining competitive motion cueing tracking performance. The adaptive system effectively enhances linear displacement (LD), ensuring better alignment of motion cues with platform constraints. While minor trade-offs were observed in root mean square error (RMSE) and correlation coefficients (CCs) for sensed angular velocity (SAV) and sensed specific force (SSF), the framework improves workspace utilisation and computational efficiency while maintaining competitive motion cueing performance. Furthermore, the adaptive LSTM-MPC framework substantially reduces computational load, achieving approximately 44.26 times faster execution compared with the classical MPC-based MCA and approximately 30.03 times faster execution compared with the RL MPC-based MCA. These findings highlight the potential of integrating deep learning (DL) with MPC to optimise the trade-off between motion cueing performance, platform utilisation, and computational efficiency in driving simulators.



Academic Editor: Paolo Bellavista

Received: 11 May 2026
Revised: 17 June 2026
Accepted: 17 June 2026
Published: 18 June 2026

**Keywords:** driving simulator; motion cueing algorithm; model predictive control; adaptive tuning; deep learning; LSTM networks; computational complexity; vehicle dynamics simulation

## 1. Introduction

The use of simulators has become indispensable across multiple industries, including public transportation, logistics, mining, and education, as they provide realistic motion sensations in safe, controlled environments. These systems heavily depend on the Motion Cueing Algorithm (MCA) to recreate the complex dynamics of vehicle motion while adhering to the physical constraints of the simulator platform. The effectiveness of MCA directly impacts the fidelity of the simulated experience, necessitating advanced control techniques to optimise performance while maintaining operational feasibility [1]. Classical MCA remains one of the most widely used methods due to its simplicity and computational efficiency introduced by Conrad and Schmidt [2]. It employs high-pass and low-pass filters to separate translational and rotational motion cues; however, its fixed parameters and inability to adapt to varying motion conditions often result in suboptimal motion fidelity and increased susceptibility to motion sickness. The adaptive MCA was introduced by [3] to address these limitations. Challenges remained in terms of computational load, overshooting during motion generation, and violation of platform constraints. To address these issues, Telban and Cardollo [4] introduced the Optimal MCA, aiming to minimize perceived motion error while explicitly incorporating platform limitations and human perception models into the control strategy. The Optimal MCA further improves motion cueing by incorporating vestibular and proprioceptive system models to minimise perceived motion errors. Among the various MCA frameworks, Model Predictive Control (MPC) has gained significant attention due to its predictive capabilities and ability to handle system constraints explicitly, which was first introduced by [5]. Unlike classical or optimal approaches, MPC solves an optimisation problem at each time step by predicting system behaviour over a finite horizon, determining an optimal control sequence that balances accuracy of movement, computational performance and management of platform constraints [6]. This receding horizon strategy enables MPC to effectively handle multivariable control problems while enforcing constraints, making it a preferred approach for motion cueing applications. These advancements underscore the continuous evolution of MCA methodologies, positioning MPC as a promising solution for achieving high-fidelity motion cueing in simulation-based environments. Despite its advantages, traditional MPC implementations face critical limitations, particularly concerning computational complexity and model dependency. The real-time feasibility of MPC is often constrained by the high computational burden associated with solving optimisation problems at each iteration. Moreover, its reliance on a fixed mathematical model renders it sensitive to modeling errors and system uncertainties, which can degrade control accuracy [7]. These challenges have driven researchers to explore data-driven techniques, particularly deep learning (DL), to enhance MPC adaptability and efficiency.

Authors [8] proposed an adaptive neural network model-based predictive control approach, which highlights the effectiveness of combining ML with MPC. The authors [9] explored integrating learning-based approaches with MPC to enhance safety and adaptability in control systems. Their work discusses the challenges of incorporating ML into MPC frameworks, particularly in ensuring stability and constraint satisfaction while improving real-time performance. They propose a structured methodology that balances data-driven learning with rigorous control-theoretic principles, highlighting its applications in robotics, autonomous systems, and industrial automation. Their findings demonstrate that learning-

based MPC can improve adaptability to dynamic environments while maintaining safe control execution. Scholars in [10] proposed a learning-based approach for approximating MPC laws using neural networks trained with both control input and gradient data. By leveraging structural information from explicit MPC and incorporating gradients during training, their approach improves the learning of MPC control policies. The study in [11] presented a hybrid MPC-based MCA that reduces computational cost by combining explicit and implicit control strategies. In [12], the study proposed an MPC-based MCA tailored for driving simulators, integrating a vehicle dynamics model and a human vestibular system model to enhance motion perception fidelity, and demonstrating superior tracking accuracy and reduced steady-state error compared to a classical PID controller. The scholars [13] proposed an auto-tuning procedure for optimising the prediction and control horizons of MPC-based MCA applied to a Robocoaster motion platform. Their method employs the mean variance mapping optimisation technique to minimise false motion cues by automatically adjusting the control parameters. The proposed approach effectively balances translational and rotational motion cues while improving motion fidelity and maintaining computational feasibility, making it applicable for advanced motion simulation tasks.

Recent studies have also highlighted the growing role of artificial intelligence, simulation-based optimisation, and human-centred vehicle simulation in improving the realism and efficiency of vehicle-related systems. Vehicle simulators have been integrated with virtual reality and physical motion mechanisms to improve immersion and reproduce vehicle motion sensations [14]. Learning-based control has also been explored for autonomous cruise control and path-following applications [15], while optimisation-based intelligent systems have been used to model driver behaviour under different road and traffic conditions [16]. In addition, deep learning methods have shown strong capability in modelling temporal motion patterns and interaction-dependent behaviour for trajectory prediction tasks [17]. These studies support the broader trend of integrating artificial intelligence with vehicle simulation and control systems. However, limited attention has been given to using deep learning for online adaptive tuning of MPC-based MCA parameters, particularly the prediction horizon, to balance motion fidelity and computational efficiency in driving simulators.

The integration of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks into MPC frameworks has proven to be a promising solution for adaptive control. Unlike conventional MPC methods that rely heavily on predefined system models, DL-MPC can refine predictions using data-driven models, enabling more responsive adjustments in control strategies. Scholars in [18] introduced an LSTM-based MPC framework tailored for multimode industrial process control. Unlike conventional multimodel or switching MPC approaches that require multiple predictive models and complex switching strategies, the proposed method leverages the sequential learning capability of LSTM networks to capture dynamic behaviour across different operating modes using a single predictive model. The integration of adaptive gradient descent optimisation further enhances its real-time applicability. Their approach demonstrates improved control accuracy, reduced overshoot, and efficient computation, offering a robust and scalable solution for complex multimode systems. Similarly, scholars in [19] presented a real-time adaptive predictive control framework integrating ML techniques to enhance nonlinear process control. Their approach employs RNNs to model system dynamics while incorporating Lyapunov-based MPC to support closed-loop stability and optimality. The study introduces an event-triggered online learning mechanism that updates the RNN model in real time, addressing model uncertainty and improving closed-loop performance. The proposed method was validated on a chemical process system, demonstrating its capability to enhance control smoothness and dynamic response in the presence of disturbances.

Further evidence of the effectiveness of recurrent learning models in MPC can be seen in recent RNN- and LSTM-based predictive control studies. Wong et al. [20] developed an RNN-based MPC framework for continuous pharmaceutical manufacturing, where recurrent models were used for system identification and prediction within the MPC structure. Meng et al. [21] proposed an RNN-LSTM-based MPC approach for a corn-to-sugar process, in which the trained recurrent model was integrated into the predictive control formulation to regulate the target process output under setpoint changes and disturbances. These studies show that recurrent learning models can strengthen MPC performance by capturing temporal dependencies, and improving prediction accuracy in dynamic control environments. In addition to improving the predictive model itself, adaptive MPC research has also explored the dynamic adjustment of the prediction horizon. Scholars in [22] introduced an adaptive horizon multistage nonlinear MPC strategy designed to reduce the high computational burden associated with traditional multistage nonlinear MPC approaches. By leveraging nonlinear programming sensitivities, the prediction horizon is updated at each time step to ensure that scenario trajectories reach their terminal regions efficiently. The proposed method maintains recursive feasibility and input-to-state practical stability while reducing computational delay without compromising control performance, making it suitable for real-time applications in uncertain nonlinear systems.

A crucial design parameter in MPC is the prediction horizon ($N_P$), which determines how far into the future the system predicts its response, while a longer prediction horizon can improve control accuracy through greater foresight, it also increases computational complexity, posing challenges for real-time applications. Conversely, a shorter prediction horizon reduces computational load but may result in suboptimal control performance due to limited prediction depth. To address this trade-off, DL-based adaptive horizon selection techniques have emerged as a promising solution. In particular, LSTM networks enable adaptive adjustment of $N_P$ by leveraging historical control data and performance metrics, improving responsiveness to changing system dynamics. This integration of DL into MPC frameworks enhances adaptability and performance in dynamic environments. In the context of motion cueing, traditional fixed-horizon MPC-based MCA often struggle to balance motion fidelity and computational efficiency, especially when system parameters evolve during simulation.

Despite advances in optimisation-based and learning-based tuning approaches, important challenges remain. Meta-heuristic methods typically produce fixed MPC parameters that cannot adapt to dynamic motion variations. Reinforcement Learning (RL)-based approaches improve adaptability by interacting with the simulation environment to identify effective controller configurations, such as prediction and control horizons. However, these approaches generally select fixed horizon settings and have primarily focused on improving motion cueing performance rather than reducing the computational burden associated with large prediction horizons. Consequently, a research gap remains in the development of a computationally efficient adaptive MPC-based MCA capable of continuously adjusting the prediction horizon according to dynamic motion variations while maintaining competitive motion cueing performance.

To overcome this limitation, this research proposes an LSTM-based adaptive prediction horizon strategy integrated within an MPC-based MCA framework. The proposed method enables adaptive adjustment of $N_P$ during controller execution, improving motion cueing fidelity and enhancing overall simulation performance. By leveraging the predictive capability of DL, the proposed approach aims to improve motion cueing performance, enhance workspace utilisation, and increase the overall effectiveness of driving simulator experiences. An overview of the proposed architecture is illustrated in Figure 1.

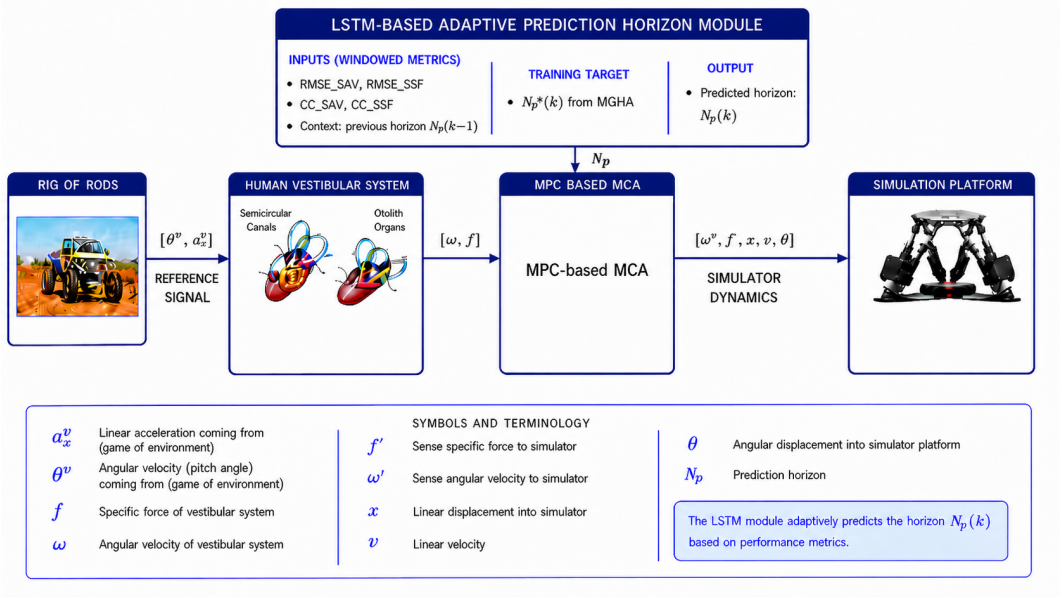


**Figure 1.** Proposed LSTM-based adaptive MPC-based MCA framework.

## 2. Proposed Method

### *2.1. MPC-Based MCA Formulation*

MPC is adopted as the core framework for the proposed MCA because of its ability to predict future system behaviour while explicitly handling platform and actuator constraints. In this study, the MPC-based MCA is formulated using a discrete-time state-space representation of the combined vestibular and motion platform dynamics. The system model is expressed as

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k), \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k), \end{aligned} \tag{1}$$

where $\mathbf{x}(k)$ denotes the state vector, $\mathbf{u}(k)$ is the control input vector, and $\mathbf{y}(k)$ is the output vector. The matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ define the discrete-time system dynamics. In this work, $\mathbf{D} = \mathbf{0}$, meaning that the current input does not instantaneously affect the output.

Human Vestibular Model

In motion cueing, tilt coordination is used to reproduce part of the longitudinal acceleration sensation through platform pitch motion. The perceived specific force along the longitudinal direction can be approximated as

$$S_F = f_x = a_x + g\theta, \tag{2}$$

where $a_x$ is the longitudinal acceleration, $\theta$ is the pitch angle, and $g$ is the gravitational acceleration.

The human vestibular system is represented using semicircular canal (SCC) and otolith (OTO) models. The SCC model is used to describe angular velocity perception, while the OTO model is used to describe specific force perception. In continuous time, these models are expressed as

$$\frac{\hat{\omega}}{\omega} = \frac{\tau_1 \tau_b s^2}{(1+\tau_b s)(1+\tau_1 s)}, \qquad \frac{\hat{f}}{f} = K_{\mathrm{OTO}} \frac{\tau_a s + 1}{(\tau_L s + 1)(\tau_s s + 1)}, \tag{3}$$

where $K_{\mathrm{OTO}} = 0.4$, $\tau_L = 5.3$ s, $\tau_s = 0.016$ s, $\tau_a = 13.2$ s, $\tau_1 = 5.3$ s, and $\tau_b = 30$ s [4]. These transfer functions are discretised using zero-order hold with a sampling time of $T_s = 0.01$ s, resulting in the discrete-time SCC and OTO state-space models $(\mathbf{A}_{\mathrm{SCC}}, \mathbf{B}_{\mathrm{SCC}}, \mathbf{C}_{\mathrm{SCC}}, \mathbf{D}_{\mathrm{SCC}})$ and $(\mathbf{A}_{\mathrm{OTO}}, \mathbf{B}_{\mathrm{OTO}}, \mathbf{C}_{\mathrm{OTO}}, \mathbf{D}_{\mathrm{OTO}})$, respectively.

By stacking the SCC and OTO states into the vestibular state vector $\mathbf{x}_{\mathrm{v}}(k)$, the combined vestibular model is written as

$$\begin{aligned} \mathbf{x}_{\mathrm{v}}(k+1) &= \begin{bmatrix} \mathbf{A}_{\mathrm{SCC}} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\mathrm{OTO}} \end{bmatrix} \mathbf{x}_{\mathrm{v}}(k) + \begin{bmatrix} \mathbf{B}_{\mathrm{SCC}} \\ \mathbf{B}_{\mathrm{OTO}} \end{bmatrix} \mathbf{u}(k), \\ \mathbf{y}_{\mathrm{v}}(k) &= \begin{bmatrix} \mathbf{C}_{\mathrm{SCC}} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{\mathrm{OTO}} \end{bmatrix} \mathbf{x}_{\mathrm{v}}(k) + \begin{bmatrix} \mathbf{D}_{\mathrm{SCC}} \\ \mathbf{D}_{\mathrm{OTO}} \end{bmatrix} \mathbf{u}(k). \end{aligned} \tag{4}$$

The motion platform dynamics are represented by a discrete-time model with state vector $\mathbf{x}_{\mathrm{c}}(k)$ and input $\mathbf{u}_{\mathrm{c}}(k)$:

$$\mathbf{x}_{\mathrm{c}}(k+1) = \mathbf{A}_{\mathrm{c}}\mathbf{x}_{\mathrm{c}}(k) + \mathbf{B}_{\mathrm{c}}\mathbf{u}_{\mathrm{c}}(k), \qquad \mathbf{A}_{\mathrm{c}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B}_{\mathrm{c}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}. \tag{5}$$

The complete state vector is then defined as

$$\mathbf{x}(k) = \begin{bmatrix} \mathbf{x}_{\mathrm{v}}^T(k) & \mathbf{x}_{\mathrm{c}}^T(k) \end{bmatrix}^T. \tag{6}$$

Accordingly, the assembled discrete-time model becomes

$$\begin{aligned} \mathbf{x}(k+1) &= \begin{bmatrix} \mathbf{A}_{\mathrm{v}} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\mathrm{c}} \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} \mathbf{B}_{\mathrm{v}} \\ \mathbf{B}_{\mathrm{c}} \end{bmatrix} \mathbf{u}(k), \\ \mathbf{y}(k) &= \begin{bmatrix} \mathbf{C}_{\mathrm{v}} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{\mathrm{c}} \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} \mathbf{D}_{\mathrm{v}} \\ \mathbf{D}_{\mathrm{c}} \end{bmatrix} \mathbf{u}(k), \end{aligned} \tag{7}$$

where $\mathbf{A}_{\mathrm{v}}$, $\mathbf{B}_{\mathrm{v}}$, $\mathbf{C}_{\mathrm{v}}$, and $\mathbf{D}_{\mathrm{v}}$ represent the combined vestibular block. These matrices are defined as

$$\begin{aligned} \mathbf{A}_{\mathrm{v}} &= \mathrm{blkdiag}(\mathbf{A}_{\mathrm{SCC}}, \mathbf{A}_{\mathrm{OTO}}), \\ \mathbf{B}_{\mathrm{v}} &= \begin{bmatrix} \mathbf{B}_{\mathrm{SCC}} \\ \mathbf{B}_{\mathrm{OTO}} \end{bmatrix}, \\ \mathbf{C}_{\mathrm{v}} &= \mathrm{blkdiag}(\mathbf{C}_{\mathrm{SCC}}, \mathbf{C}_{\mathrm{OTO}}), \\ \mathbf{D}_{\mathrm{v}} &= \begin{bmatrix} \mathbf{D}_{\mathrm{SCC}} \\ \mathbf{D}_{\mathrm{OTO}} \end{bmatrix}. \end{aligned} \tag{8}$$

The regulated output vector is defined as

$$\mathbf{y}(k) = \begin{bmatrix} f'(k) \\ \omega'(k) \\ x(k) \\ v(k) \\ \theta'(k) \end{bmatrix}, \qquad n = 5, \tag{9}$$

where $f'(k)$ and $\omega'(k)$ represent the perceived specific force and perceived angular velocity, respectively. In addition, $x(k)$ is the linear displacement (LD), $v(k)$ is the linear velocity, and $\theta'(k)$ is the platform angular displacement. The input vector is given by

$$\mathbf{u}(k) = \begin{bmatrix} a_x(k) \\ \theta(k) \end{bmatrix}, \qquad m = 2, \tag{10}$$

where $a_x(k)$ is the longitudinal acceleration input and $\theta(k)$ is the pitch-angle input supplied to the MPC-based MCA.

### *2.2. Adaptive Prediction Horizon Formulation*

In the classical MPC-based MCA, the prediction horizon is fixed throughout the simulation. In contrast, the proposed adaptive formulation updates the prediction horizon online according to the current motion condition. Let $N_p(k)$ denote the prediction horizon selected at sampling instant $k$. The control horizon is kept fixed as

$$N_c = N_c^{\text{fix}}, \qquad N_c^{\text{fix}} \leq N_p(k). \tag{11}$$

The use of a fixed control horizon limits the number of decision variables in the optimisation problem, while allowing the prediction horizon to vary according to the motion demand. The control input increments are defined as

$$\Delta\mathbf{u}(k) = \mathbf{u}(k) - \mathbf{u}(k-1). \tag{12}$$

The stacked future control increments and predicted outputs are given by

$$\Delta\mathbf{U}(k) \in \mathbb{R}^{(N_c^{\text{fix}} m)\times 1}, \qquad \mathbf{Y}(k) \in \mathbb{R}^{(N_p(k) n)\times 1}. \tag{13}$$

At each sampling instant $k$, the condensed prediction model is expressed as

$$\mathbf{Y}(k) = \mathbf{f}\big(N_p(k), \mathbf{x}(k)\big) + \mathbf{G}\big(N_p(k), N_c^{\text{fix}}\big)\Delta\mathbf{U}(k), \tag{14}$$

where $\mathbf{f}\big(N_p(k), \mathbf{x}(k)\big)$ is the free-response vector and $\mathbf{G}\big(N_p(k), N_c^{\text{fix}}\big)$ is the lifted dynamic matrix. The free-response vector is written as

$$\mathbf{f}\big(N_p(k), \mathbf{x}(k)\big) = \mathbf{F}\big(N_p(k)\big)\mathbf{x}(k), \tag{15}$$

where

$$\mathbf{F}\big(N_p(k)\big) = \begin{bmatrix} \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^{N_p(k)} \end{bmatrix}. \tag{16}$$

The lifted dynamic matrix is defined as

$$\mathbf{G}\big(N_p(k), N_c^{\text{fix}}\big) = \begin{bmatrix} \mathbf{CB} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{CAB} & \mathbf{CB} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^{N_p(k)-1}\mathbf{B} & \mathbf{CA}^{N_p(k)-2}\mathbf{B} & \cdots & \mathbf{CA}^{N_p(k)-N_c^{\text{fix}}}\mathbf{B} \end{bmatrix}. \tag{17}$$

Therefore, although the physical plant model remains linear time-invariant, the lifted prediction model becomes horizon-dependent through $N_p(k)$. In this sense, the proposed controller can be described as an adaptive-horizon, time-varying MPC formulation, where the optimisation problem is updated at each sampling instant according to the selected prediction horizon. However, the underlying state-space model, system matrices, and physical dynamics remain unchanged.

Let $\mathbf{R}_{\text{ref}}(k) \in \mathbb{R}^{(N_p(k)n)\times 1}$ denote the stacked reference vector over the current prediction horizon. The adaptive MPC cost function solved at sampling instant $k$ is written as

$$\begin{aligned} J_k(\Delta\mathbf{U}) = &\left(\mathbf{R}_{\text{ref}}(k) - \mathbf{Y}(k)\right)^T \mathbf{Q}\left(N_p(k)\right)\left(\mathbf{R}_{\text{ref}}(k) - \mathbf{Y}(k)\right) \\ &+ \mathbf{U}(k)^T \mathbf{S}\mathbf{U}(k) + \Delta\mathbf{U}(k)^T \mathbf{R}\Delta\mathbf{U}(k), \end{aligned} \tag{18}$$

where $\mathbf{Q}\left(N_p(k)\right) \succeq 0$, $\mathbf{S} \succeq 0$, and $\mathbf{R} \succeq 0$ are weighting matrices for output tracking, input magnitude, and input increments, respectively. The matrix $\mathbf{U}(k)$ denotes the stacked absolute input sequence, while $\Delta\mathbf{U}(k)$ denotes the stacked input increments.

The output weighting matrix is constructed from a fixed per-step weighting matrix $Q_{\text{step}}$ as

$$\mathbf{Q}\left(N_p(k)\right) = \text{kron}\left(I_{N_p(k)}, Q_{\text{step}}\right). \tag{19}$$

Thus, when $N_p(k)$ changes, the size of $\mathbf{Q}\left(N_p(k)\right)$, $\mathbf{R}_{\text{ref}}(k)$, and $\mathbf{Y}(k)$ changes accordingly. However, the per-step weighting structure $Q_{\text{step}}$ remains fixed in the adaptive-horizon stage.

The output constraints are also expressed over the currently selected prediction horizon:

$$\mathbf{Y}_{\min}\left(N_p(k)\right) \leq \mathbf{Y}(k) \leq \mathbf{Y}_{\max}\left(N_p(k)\right). \tag{20}$$

Additional actuator, state, and workspace constraints are retained within the MPC optimisation. Therefore, the proposed adaptive-horizon formulation changes the prediction length used by the controller while preserving the same physical model and constraint structure.

Substituting the condensed prediction model in Equation (14) into the cost function in Equation (18) yields the quadratic programming problem solved at each sampling instant:

$$\min_{\Delta\mathbf{U}(k)} \frac{1}{2}\Delta\mathbf{U}(k)^T \mathbf{H}(k)\Delta\mathbf{U}(k) + \mathbf{g}(k)^T \Delta\mathbf{U}(k), \tag{21}$$

subject to

$$\mathbf{M}(k)\Delta\mathbf{U}(k) \leq \mathbf{b}(k). \tag{22}$$

Here, $\mathbf{H}(k)$, $\mathbf{g}(k)$, $\mathbf{M}(k)$, and $\mathbf{b}(k)$ are horizon-dependent matrices. Their dependency on $k$ arises from the online selection of $N_p(k)$, which changes the lifted prediction model and the stacked constraints. Consequently, the controller operates as an adaptive-horizon MPC while the underlying system model remains unchanged.

### *2.3. Practical Stability and Feasibility Considerations*

The proposed adaptive MPC-based MCA adjusts only the prediction horizon while maintaining a fixed linear time-invariant (LTI) plant model, a fixed control horizon, and an unchanged MPC constraint structure. Consequently, although the lifted prediction matrices and optimisation problem vary with the selected prediction horizon $N_p(k)$, the underlying system dynamics remain unchanged throughout the simulation.

A rigorous Lyapunov-based proof of stability is beyond the scope of this study. Nevertheless, several design features support the practical feasibility and boundedness of the proposed adaptive-horizon formulation.

First, the prediction horizon is constrained within predefined bounds. Such bounded horizon selection is consistent with previous horizon optimisation studies for MPC-based MCAs, which demonstrated that effective prediction horizons can be selected from finite admissible ranges while maintaining motion fidelity and computational efficiency [23,24]. This prevents excessively short horizons that may degrade prediction quality and excessively long horizons that may unnecessarily increase computational complexity.

Second, output constraints, actuator limits, and platform workspace constraints are enforced at every optimisation step regardless of the selected prediction horizon. Therefore, horizon adaptation modifies only the prediction length while preserving the physical operating limits of the motion platform.

Third, the adaptive mechanism modifies only the prediction horizon and does not alter the plant model, controller architecture, or constraint definitions. Consequently, the controller transitions between horizon-dependent prediction models without introducing abrupt structural changes to the optimisation problem, thereby reducing the likelihood of undesirable horizon-switching chattering.

Furthermore, no optimisation infeasibility, divergence, or unstable behaviour was observed across any investigated simulation scenario. All simulations satisfied the imposed actuator, output, and workspace constraints throughout operation. Together, the bounded horizon adaptation, persistent constraint enforcement, and successful closed-loop operation provide empirical evidence supporting the practical feasibility and boundedness of the proposed adaptive LSTM-MPC-based MCA.

Furthermore, the adaptive horizon mechanism does not modify the actuator limits, output constraints, or platform workspace constraints imposed by the MPC formulation. The adaptation process affects only the prediction horizon used by the controller, while all physical safety constraints remain unchanged and are enforced at every optimisation step. Consequently, the proposed framework cannot command platform motions beyond the predefined operating limits solely as a result of horizon adaptation. This provides an additional level of safety for future implementation on physical motion platforms. Although a formal recursive-feasibility proof is beyond the scope of this study, the adaptive mechanism modifies only the prediction horizon while preserving the plant model, control horizon, cost function structure, and all imposed constraints. Since the horizon is selected from a finite admissible set and all optimisation constraints remain enforced at every step, the controller operates within a bounded family of MPC formulations sharing the same physical dynamics and safety limits. Consequently, horizon adaptation changes the prediction depth rather than the underlying control problem structure, providing additional justification for practical feasibility and bounded closed-loop operation.

### *2.4. Windowed Tracking Metrics*

The adaptive horizon is selected using short-window performance metrics computed from the sensed angular velocity (SAV) and sensed specific force (SSF) channels. Let $T_W$ denote the window duration, and let

$$W = \frac{T_W}{T_s} \tag{23}$$

be the corresponding number of samples. The active sliding window at sampling instant $k$ is defined as

$$\mathcal{W}_k = \{j \mid k - W + 1 \leq j \leq k\}. \tag{24}$$

For each channel $\text{ch} \in \{\text{SAV}, \text{SSF}\}$, with reference signal $r_{\text{ch}}(j)$ and output signal $y_{\text{ch}}(j)$, the windowed mean values are

$$\bar{r}_{\text{ch},k} = \frac{1}{W} \sum_{j \in \mathcal{W}_k} r_{\text{ch}}(j), \qquad \bar{y}_{\text{ch},k} = \frac{1}{W} \sum_{j \in \mathcal{W}_k} y_{\text{ch}}(j). \tag{25}$$

The windowed root mean square error is calculated as

$$\text{RMSE}_{\text{ch}}(k) = \sqrt{\frac{1}{W} \sum_{j \in \mathcal{W}_k} \left(r_{\text{ch}}(j) - y_{\text{ch}}(j)\right)^2}. \tag{26}$$

The corresponding windowed correlation coefficient is

$$\mathrm{CC}_{\mathrm{ch}}(k) = \frac{\sum_{j\in\mathcal{W}_k}(r_{\mathrm{ch}}(j)-\bar{r}_{\mathrm{ch},k})(y_{\mathrm{ch}}(j)-\bar{y}_{\mathrm{ch},k})}{\sqrt{\sum_{j\in\mathcal{W}_k}(r_{\mathrm{ch}}(j)-\bar{r}_{\mathrm{ch},k})^2}\sqrt{\sum_{j\in\mathcal{W}_k}(y_{\mathrm{ch}}(j)-\bar{y}_{\mathrm{ch},k})^2}}. \tag{27}$$

In this study, $T_s = 0.01$ s and $T_W = 1$ s, resulting in a window length of $W = 100$ samples. Therefore, at every sampling instant, the most recent one-second interval is used to compute the four adaptive features:

$$\{\mathrm{RMSE}_{\mathrm{SAV}}(k), \mathrm{RMSE}_{\mathrm{SSF}}(k), \mathrm{CC}_{\mathrm{SAV}}(k), \mathrm{CC}_{\mathrm{SSF}}(k)\}. \tag{28}$$

These metrics provide a compact description of the recent tracking performance and are used to guide the prediction horizon adaptation.

### *2.5. Metric-Guided Horizon Adaptation*

Before training the LSTM policy, a Metric-Guided Horizon Adaptation (MGHA) mechanism is used to generate interpretable horizon labels. The tracking errors for the SAV and SSF channels are defined as

$$e_{\mathrm{SAV}}(j) = r_{\mathrm{SAV}}(j) - y_{\mathrm{SAV}}(j), \qquad e_{\mathrm{SSF}}(j) = r_{\mathrm{SSF}}(j) - y_{\mathrm{SSF}}(j), \tag{29}$$

for $j \in \mathcal{W}_k$. The corresponding RMSE values are

$$\mathrm{RMSE}_{\mathrm{SAV}}(k) = \sqrt{\frac{1}{W}\sum_{j\in\mathcal{W}_k} e_{\mathrm{SAV}}^2(j)}, \qquad \mathrm{RMSE}_{\mathrm{SSF}}(k) = \sqrt{\frac{1}{W}\sum_{j\in\mathcal{W}_k} e_{\mathrm{SSF}}^2(j)}. \tag{30}$$

In this study, the MGHA parameters were selected as $s_1 = 0.05$, $s_2 = 0.05$, $\lambda = 0.5$, and $\mu = 0.3$. The parameters $s_1$ and $s_2$ define the scaling of the SAV and SSF RMSE terms within the exponential mapping, while $\lambda$ determines the relative weighting between SAV and SSF tracking performance. The parameter $\mu$ controls the contribution of the correlation-based term to the overall difficulty score used for horizon adaptation. The selected values were determined empirically to provide a balanced contribution from the SAV and SSF performance metrics and to avoid saturation of the exponential mapping. Since MGHA is used only as a label-generation mechanism, a detailed sensitivity analysis was considered beyond the scope of the present study and is identified as future work.

The mean correlation coefficient is defined as

$$\overline{\mathrm{CC}}(k) = \frac{1}{2}(\mathrm{CC}_{\mathrm{SAV}}(k) + \mathrm{CC}_{\mathrm{SSF}}(k)). \tag{31}$$

A scalar difficulty score $E(k) \in [0,1]$ is then computed as

$$\begin{aligned} E(k) = {} & \lambda\left(1 - e^{-\mathrm{RMSE}_{\mathrm{SAV}}(k)/s_1}\right)/2 \\ & + (1-\lambda)\left(1 - e^{-\mathrm{RMSE}_{\mathrm{SSF}}(k)/s_2}\right)/2 + \mu\frac{1}{2}(1 - \overline{\mathrm{CC}}(k)), \end{aligned} \tag{32}$$

where $s_1$ and $s_2$ are positive scaling constants, and $\lambda$ and $\mu$ are design weights. A larger value of $E(k)$ indicates more difficult motion conditions, such as larger tracking errors or lower correlation with the reference signals.

The adaptive prediction horizon is obtained as

$$N_p^\star(k) = \mathrm{round}[N_{\min} + E(k)(N_{\max} - N_{\min})], \tag{33}$$

subject to

$$N_{\min} \leq N_p^{\star}(k) \leq N_{\max}. \tag{34}$$

This mechanism assigns shorter horizons during low-demand motion segments and longer horizons during more demanding segments. These generated $N_p^{\star}(k)$ values are used as target horizon values for training the LSTM horizon policy.

### *2.6. Computational Considerations*

The proposed formulation keeps the control horizon fixed at $N_c = N_c^{\text{fix}}$, while the prediction horizon $N_p(k)$ is adapted online. This design is important because the number of decision variables in the condensed optimisation problem is determined by $N_c^{\text{fix}} m$, whereas the prediction horizon affects the size of the predicted output vector, lifted matrices, and stacked constraints.

For a prediction horizon $N_p(k)$ and fixed control horizon $N_c^{\text{fix}}$, the condensed QP contains

$$N_c^{\text{fix}} m \tag{35}$$

decision variables. The number of output-related constraints increases with $N_p(k)$. Therefore, reducing the prediction horizon during low-demand motion segments can reduce the computational burden, while increasing it during high-demand segments can preserve prediction quality and motion cue fidelity.

To reduce online computational overhead, the state-independent lifted matrices are precomputed and stored for all admissible prediction horizon values. Specifically, the following matrices are cached offline:

$$\left\{ \mathbf{F}(N_p), \mathbf{G}(N_p, N_c^{\text{fix}}) \right\}, \qquad N_p \in [N_{\min}, N_{\max}]. \tag{36}$$

At runtime, the controller selects the cached matrices corresponding to the current value of $N_p(k)$, computes

$$\mathbf{f}\big(N_p(k), \mathbf{x}(k)\big) = \mathbf{F}\big(N_p(k)\big)\mathbf{x}(k), \tag{37}$$

and assembles

$$\mathbf{Q}\big(N_p(k)\big) = \text{kron}\Big(I_{N_p(k)}, Q_{\text{step}}\Big). \tag{38}$$

As a result, the additional computational cost associated with switching the prediction horizon is small compared with solving the QP itself. This makes the adaptive-horizon formulation suitable for repeated controller execution with online horizon adaptation.

### *2.7. LSTM-Based Horizon Policy Learning*

Although MGHA provides an interpretable mapping from tracking metrics to prediction horizon values, its rule-based structure may not fully capture the temporal relationship between recent motion behaviour and suitable horizon selection. Therefore, an LSTM network is trained to learn this mapping from sequential data.

At each sampling instant $k$, the LSTM receives a short sequence of feature vectors and predicts the current prediction horizon:

$$N_p(k) = f_{\text{LSTM}}(\mathbf{x}_{k-L+1:k}), \tag{39}$$

where

$$\mathbf{x}_{k-L+1:k} = [\mathbf{x}(k-L+1), \ldots, \mathbf{x}(k)] \tag{40}$$

is a sequence of $L$ feature vectors. Each feature vector contains the windowed RMSE and CC metrics for SAV and SSF, together with the previous prediction horizon value

$N_p(k-1)$. The scalar output of the LSTM is inverse-scaled, clipped to the admissible range $[N_{\min}, N_{\max}]$, and rounded before being used by the MPC.

It should be noted that MGHA is used solely as a label-generation mechanism during the offline training stage and is not used during controller deployment; while MGHA computes prediction horizon values from a predefined analytical mapping based on instantaneous tracking metrics, the proposed LSTM learns an approximation of the underlying horizon-selection behaviour from sequential data. By exploiting temporal dependencies across multiple time steps, the LSTM can capture nonlinear relationships between recent tracking performance and suitable horizon selection that are not explicitly represented within the MGHA formulation. Consequently, the trained LSTM serves as a compact adaptive policy that replaces the online evaluation of handcrafted adaptation rules while providing improved generalisation across different motion conditions and driving scenarios. In addition to learning temporal dependencies, the use of an LSTM offers several practical advantages over directly applying the MGHA rule. Since the LSTM processes a sequence of recent performance metrics rather than relying solely on instantaneous measurements, it can produce smoother prediction horizon transitions and reduce unnecessary horizon fluctuations caused by short-term variations in the tracking signals. This behaviour has the potential to reduce horizon-switching chattering and improve controller consistency during rapidly changing driving conditions. Furthermore, once trained, the LSTM provides a compact adaptive policy that can be deployed without explicitly evaluating handcrafted adaptation rules or manually retuning MGHA parameters. The learned framework is also more scalable, as additional performance indicators, driving scenarios, or system variables can be incorporated through retraining without requiring reformulation of the underlying adaptation rule. Therefore, although the LSTM is trained using MGHA-generated labels, its purpose is not merely to replicate the MGHA mapping but to provide a smoother, scalable, and computationally efficient adaptive policy that can exploit temporal information unavailable to the instantaneous MGHA formulation.

For completeness, the LSTM cell equations are given as

$$\mathbf{f}_k = \sigma\Big(\mathbf{W}_f \mathbf{x}_k + \mathbf{U}_f \mathbf{h}_{k-1} + \mathbf{b}_f\Big), \tag{41}$$

$$\mathbf{i}_k = \sigma(\mathbf{W}_i \mathbf{x}_k + \mathbf{U}_i \mathbf{h}_{k-1} + \mathbf{b}_i), \tag{42}$$

$$\tilde{\mathbf{c}}_k = \tanh(\mathbf{W}_c \mathbf{x}_k + \mathbf{U}_c \mathbf{h}_{k-1} + \mathbf{b}_c), \tag{43}$$

$$\mathbf{c}_k = \mathbf{f}_k \circ \mathbf{c}_{k-1} + \mathbf{i}_k \circ \tilde{\mathbf{c}}_k, \tag{44}$$

$$\mathbf{o}_k = \sigma(\mathbf{W}_o \mathbf{x}_k + \mathbf{U}_o \mathbf{h}_{k-1} + \mathbf{b}_o), \tag{45}$$

$$\mathbf{h}_k = \mathbf{o}_k \circ \tanh(\mathbf{c}_k), \tag{46}$$

where $\sigma(\cdot)$ is the logistic sigmoid function, $\tanh(\cdot)$ is the hyperbolic tangent function, $\circ$ denotes the Hadamard product, $\mathbf{c}_k$ is the cell state, and $\mathbf{h}_k$ is the hidden state.

### *2.8. LSTM Training and Online Horizon Prediction*

The LSTM horizon policy is trained using data collected from a bank of driving scenarios $\mathcal{S}$. During each simulation, the windowed tracking metrics and the corresponding prediction horizon values are recorded. The input feature vector at time step $k$ is defined as

$$\mathbf{x}_k = \big[\mathrm{RMSE}_{\mathrm{SAV}}(k), \mathrm{RMSE}_{\mathrm{SSF}}(k), \mathrm{CC}_{\mathrm{SAV}}(k), \mathrm{CC}_{\mathrm{SSF}}(k), N_p(k-1), \ldots\big], \tag{47}$$

and the corresponding target output is

$$y_k = N_p(k). \tag{48}$$

The collected dataset is divided into training, validation, and test sets. Feature scaling and target normalisation are fitted using the training set only and then applied to the validation and test sets. This helps avoid information leakage between the training and evaluation stages.

The LSTM is trained as a supervised regression model by minimising the difference between the predicted horizon and the target horizon:

$$\mathcal{L}_{\text{sup}} = \frac{1}{N} \sum_{k=1}^{N} \left( \hat{N}_p(k) - N_p(k) \right)^2, \tag{49}$$

where $N$ is the total number of training samples, $\hat{N}_p(k)$ is the prediction horizon estimated by the LSTM, and $N_p(k)$ is the target prediction horizon.

During runtime, the trained LSTM predicts the prediction horizon at each sampling instant. At the beginning of the simulation, $N_p(0)$ is set to a nominal value. Then, at every sampling instant, the recent sequence of windowed metrics is passed to the LSTM to estimate $\hat{N}_p(k)$.

The hyperparameters used for training the LSTM horizon policy are summarised in Table 1.

**Table 1.** LSTM model hyperparameters used in the study.

| Hyperparameter | Value |
|---|---|
| Optimiser | Adam |
| Maximum Epochs | 300 |
| Batch Size | 32 |
| LSTM Units | 64, 32 |
| Dropout Rate | 30% |
| Activation Functions | ReLU |
| Sequence Length | 10 |
| Number of Features | 12 |

## 3. Experimental Setup and Results

### *3.1. Experimental Setup Details*

Motion references were generated in *Rig of Rods* (RoR) using a bank of scenarios including hard braking, slalom, sharp-turn, and bumpy-road manoeuvres. The condensed MPC QP was solved in MATLAB R2023b with $T_s = 0.01$ s and fixed control horizon $N_C$, while the learned horizon policy ran in Python 3.11 (TensorFlow/Keras) via the MATLAB–Python bridge. All simulations were performed on a workstation equipped with an Intel Core i7-10870H CPU running at 2.20 GHz, 32 GB RAM, and an NVIDIA GeForce RTX 3080 Laptop GPU with 8 GB memory under a 64-bit Windows operating system. At each step $k$, SAV and SSF were buffered in a sliding window $\mathcal{W}_k = \{k - W + 1, \ldots, k\}$ to compute $\text{RMSE}_{\text{SAV}}(k)$, $\text{RMSE}_{\text{SSF}}(k)$, $\text{CC}_{\text{SAV}}(k)$, and $\text{CC}_{\text{SSF}}(k)$, together with the previous horizon $N_P(k-1)$. The Python policy output $\widehat{N}_P(k)$ was clipped to $[N_{\min}, N_{\max}]$, rounded to the nearest integer, and used by the MPC to select the corresponding lifted matrices from a precomputed cache; the per-step output weight $Q_{\text{step}}$ remained fixed and we assembled $Q = \text{kron}\big(I_{N_P(k)}, Q_{\text{step}}\big)$ online. Bounds, actuator limits, and MAS-based state constraints were identical across methods. Primary endpoints were mean per-step execution time and workspace utilisation via LD; secondary endpoints were the windowed RMSE and CC for SAV and SSF. No human-subject trials were conducted; SAV/SSF CC and RMSE served as perception-channel surrogates, with SSQ and presence scores deferred to on-platform studies. The RL MPC-based MCA baseline follows the DQN-based horizon optimisation framework presented in [25]. In this framework, a Deep Q-Network (DQN) agent is trained

offline to select prediction and control horizon configurations. The state vector consists of vehicle and platform variables including longitudinal acceleration, angular velocity, sensed specific force, sensed angular velocity, LD, linear velocity, and pitch angle. The action space consists of discrete prediction horizon and control horizon selections, while the reward function is formulated using the RMSE and CC of SSF and SAV.

### *3.2. Training and Evaluation of the LSTM Horizon Policy*

The policy was developed in Python using TensorFlow/Keras, with pandas, NumPy, and scikit-learn for preprocessing. The RoR dataset comprised time-series examples constructed from 12 engineered features—base RMSE/CC for SAV/SSF, first differences, and short rolling means (window size 3, $\bar{x}^{(3)}(k) = \frac{1}{3}\left[x(k) + x(k-1) + x(k-2)\right]$) with sequences of 10 steps used to predict the normalised $N_P$. Feature vectors were robust-scaled; the target used min–max scaling. An 80/20 split, early stopping, and learning-rate reduction were applied.

On the test set, the best LSTM (64/32 units, 30% dropout, batch norm, ReLU dense) achieved

- RMSE: 0.2158;
- Mean Absolute Error (MAE): 0.1493;
- $R^2$: 0.6722.

Although the achieved $R^2$ value indicates moderate agreement between the LSTM predictions and the MGHA-generated target horizons, the MPC controller remains subject to the same actuator, output, and workspace constraints regardless of the selected prediction horizon. Consequently, prediction errors primarily affect the selected prediction depth and computational efficiency rather than directly compromising constraint satisfaction. In practice, horizon-prediction errors typically result in neighbouring horizon selections within the admissible range, limiting their impact on controller behaviour. This is consistent with the simulation results, where no infeasibility, instability, or constraint violations were observed despite imperfect horizon-prediction accuracy.

Predictions are clipped to $[100, 450]$ and rounded before use.

### *3.3. Adaptive MPC-Based MCA Performance*

A comparative analysis among the classical MPC-based MCA, RL MPC-based MCA, and adaptive LSTM–MPC-based MCA is summarised in Table 2. The primary evaluation results show that the RL MPC-based MCA achieves better SSF correlation and lower RMSE values than the LSTM-based method for some tracking metrics. However, the proposed LSTM-MPC-based MCA achieves the highest SAV correlation and substantially improves LD utilisation. Therefore, the proposed method should not be interpreted as outperforming the RL MPC-based MCA in every tracking metric. Rather, its principal advantage lies in its ability to adapt the prediction horizon during controller execution according to recent motion conditions, thereby improving platform utilisation and reducing computational load while maintaining competitive motion cueing performance.

Table 2 presents the primary evaluation results. To further assess the robustness of the proposed framework, two additional driving scenarios were also evaluated. The corresponding results are summarised in Tables 3 and 4. Although the absolute performance values vary across scenarios due to differences in manoeuvre characteristics and motion demands, several consistent trends can be observed. Across all three investigated scenarios, the adaptive LSTM-MPC-based MCA maintains competitive tracking performance while consistently demonstrating improved workspace utilisation and reduced computational burden relative to the classical MPC-based MCA and RL MPC-based MCA.

**Table 2.** Performance comparison of classical, RL, and LSTM-enhanced MPC-based MCA.

| Metric | Classical MPC-Based MCA | RL MPC-Based MCA | LSTM-MPC-Based MCA |
|---|---|---|---|
| CC of SSF | 0.6160 | 0.5990 | 0.5720 |
| CC of SAV | 0.0546 | 0.0610 | 0.0707 |
| RMSE of SSF | 0.6330 | 0.6440 | 0.6610 |
| RMSE of SAV | 0.1000 | 0.1120 | 0.1290 |
| Linear displacement (m) | 0.0620 | 0.0890 | 0.1460 |

**Table 3.** Performance comparison under an additional driving scenario.

| Metric | Classical MPC-Based MCA | RL MPC-Based MCA | LSTM-MPC-Based MCA |
|---|---|---|---|
| CC of SSF | 0.9794 | 0.9812 | 0.9708 |
| CC of SAV | 0.1226 | 0.0656 | 0.3196 |
| RMSE of SSF | 0.0695 | 0.0701 | 0.0860 |
| RMSE of SAV | 0.0533 | 0.0501 | 0.0458 |
| Linear displacement (m) | 0.0213 | 0.0170 | 0.0420 |

**Table 4.** Performance comparison under a third driving scenario.

| Metric | Classical MPC-Based MCA | RL MPC-Based MCA | LSTM-MPC-Based MCA |
|---|---|---|---|
| CC of SSF | 0.8234 | 0.8722 | 0.8812 |
| CC of SAV | 0.4020 | 0.5042 | 0.6006 |
| RMSE of SSF | 0.0316 | 0.0293 | 0.0320 |
| RMSE of SAV | 0.0157 | 0.0148 | 0.0138 |
| Linear displacement (m) | 0.0103 | 0.0127 | 0.0370 |

In the second driving scenario, the proposed framework achieved the highest SAV correlation coefficient and the lowest SAV RMSE while maintaining comparable SSF tracking performance. Similarly, in the third driving scenario, the adaptive LSTM-MPC-based MCA achieved the highest SSF correlation coefficient, the highest SAV correlation coefficient, and the lowest SAV RMSE among the three methods. These results provide additional evidence that the benefits of adaptive prediction horizon selection are not limited to a single driving manoeuvre or operating condition. Nevertheless, further validation using a broader range of representative driving scenarios remains an important direction for future work.

Figure 2 illustrates the alignment between the SSF and the reference signal. The adaptive LSTM-MPC-based MCA follows the general SSF motion profile and responds actively to dynamic variations. However, the RL-based method achieves better SSF correlation and lower SSF RMSE. Therefore, the SSF result indicates a trade-off between tracking accuracy and the improved adaptability, workspace utilisation, and computational efficiency achieved by the proposed LSTM-based method. In the present study, RMSE and CC were used as objective surrogate measures of motion cueing performance, consistent with common practice in the motion cueing literature. We acknowledge that no direct human-subject evaluation was conducted; therefore, exact perceptual acceptability boundaries cannot be established solely from the reported RMSE and CC values. To provide additional context, Figures 2 and 3 compare the SSF and SAV responses with the corresponding reference signals. Furthermore, the revised manuscript relates the results to established vestibular perception thresholds reported in the literature. Previous studies have reported angular velocity perception thresholds ranging from approximately 2.0–3.0 deg/s for roll, 2.0–3.6 deg/s for pitch, and 1.6–2.6 deg/s for yaw. Similarly, linear acceleration perception thresholds of approximately 0.17 $m/s^2$ for surge and sway and 0.28 $m/s^2$ for heave have been reported in vestibular modelling studies.

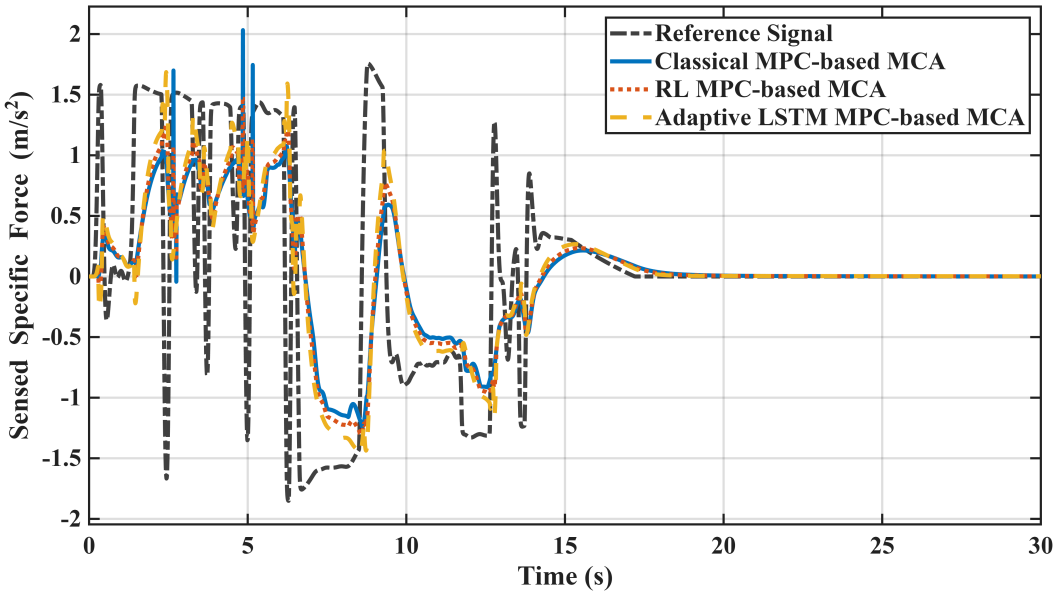


**Figure 2.** Comparison of sensed specific force and reference signal.

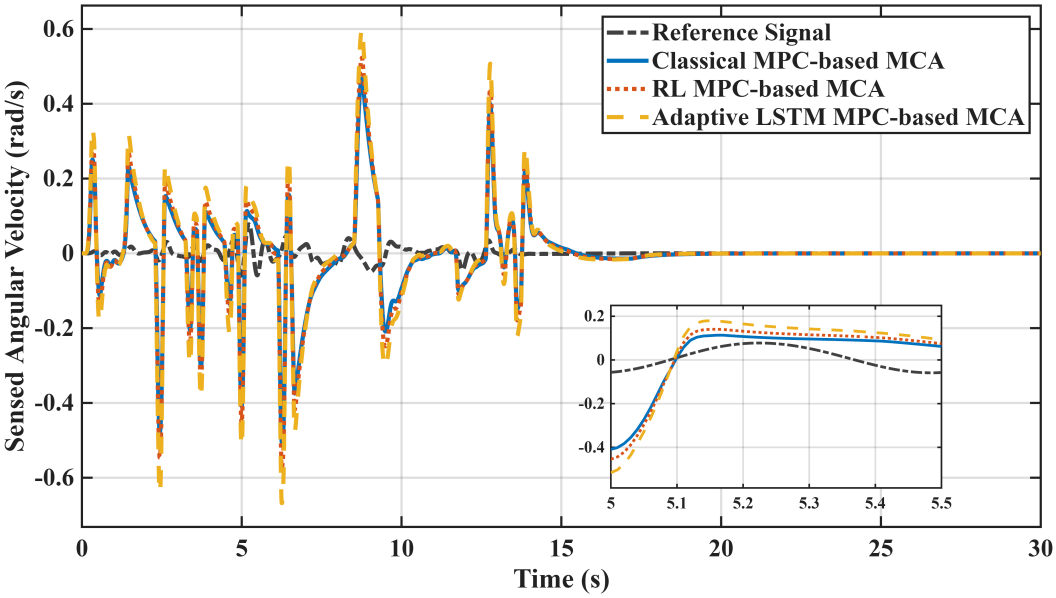


**Figure 3.** Comparison of sensed angular velocity and reference signal.

However, RMSE and CC do not directly correspond to vestibular perception thresholds. Consequently, it is not possible to rigorously determine whether the observed tracking differences are above or below human perceptual limits without dedicated human-subject experiments. The term "minor trade-off" was therefore intended to describe the relatively small reduction in objective-tracking performance compared with the substantial improvements in workspace utilisation and computational efficiency achieved by the proposed framework.

Figure 3 further evaluates the alignment of the SAV with the reference signal, demonstrating the enhanced capability of the adaptive LSTM-MPC-based MCA to track dynamic motion variations. Compared to the classical MPC-based MCA and RL MPC-based MCA, the adaptive approach shows better responsiveness to rapid changes while maintaining close agreement with the reference signal.

Figure 4 presents the SSF error over time for the classical MPC-based MCA, RL MPC-based MCA, and adaptive LSTM-MPC-based MCA. The adaptive method shows better responsiveness during dynamic changes while keeping the SSF error within a similar overall range.

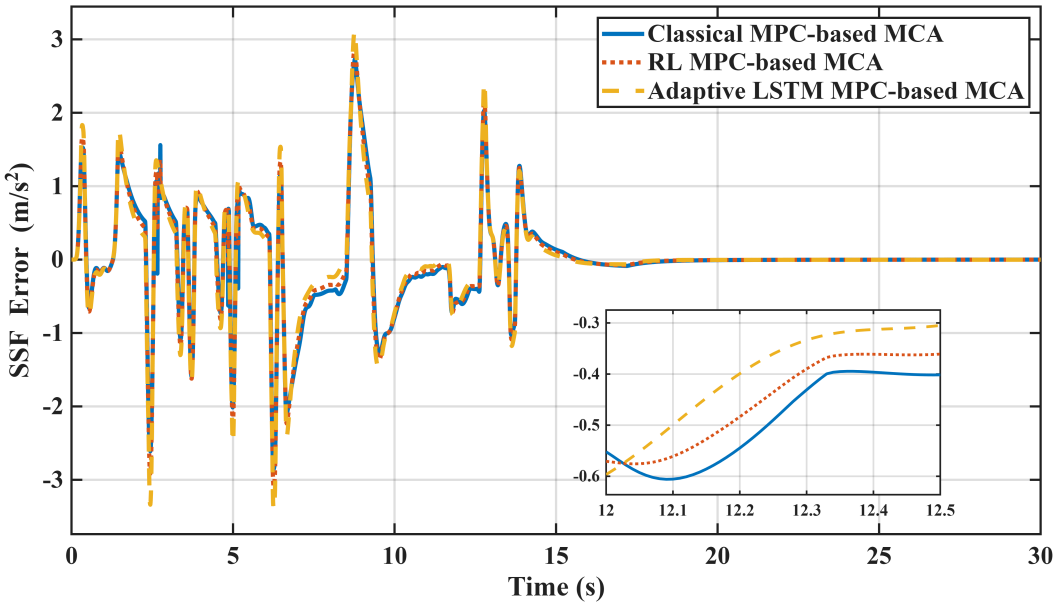


**Figure 4.** Comparison of specific force sensing error over time.

Figure 5 compares the SAV error for the classical MPC-based MCA, RL MPC-based MCA, and adaptive LSTM-MPC-based MCA. The adaptive approach exhibits slightly higher fluctuations due to its responsiveness to dynamic motion variations. Compared to the classical MPC-based MCA and RL MPC-based MCA, the adaptive method remains within a similar error range while responding more actively to rapid changes in the motion profile.

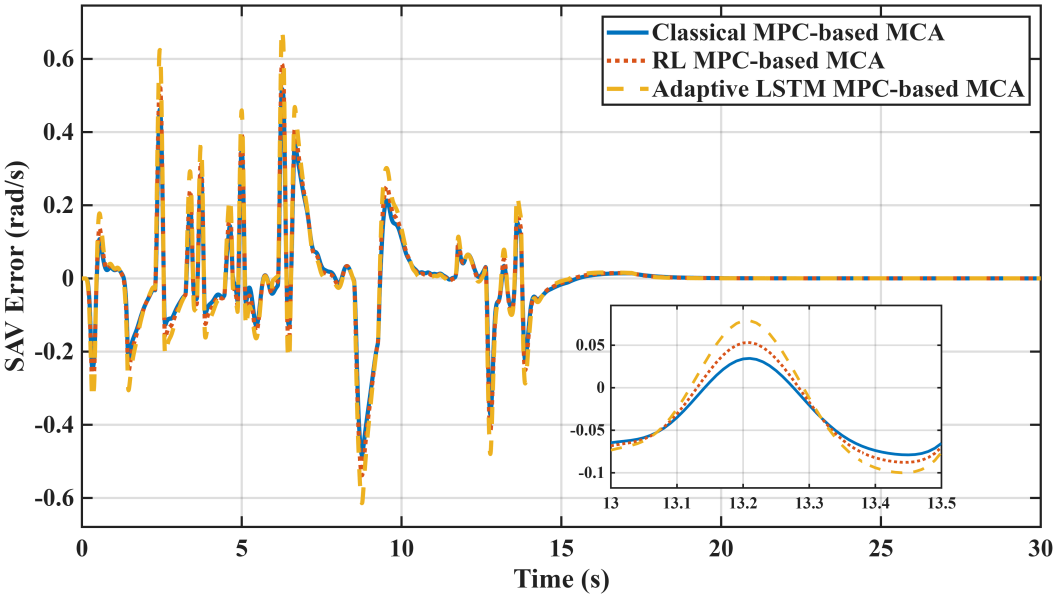


**Figure 5.** Comparison of sensed angular velocity error over time.

Figure 6 presents the LD response for the classical MPC-based MCA, RL MPC-based MCA, and adaptive LSTM-MPC-based MCA. The adaptive MPC demonstrates better workspace utilisation, particularly during peak displacement scenarios. Compared to the classical MPC-based MCA and RL MPC-based MCA, the adaptive approach achieves larger LD, indicating improved use of the available simulator workspace.

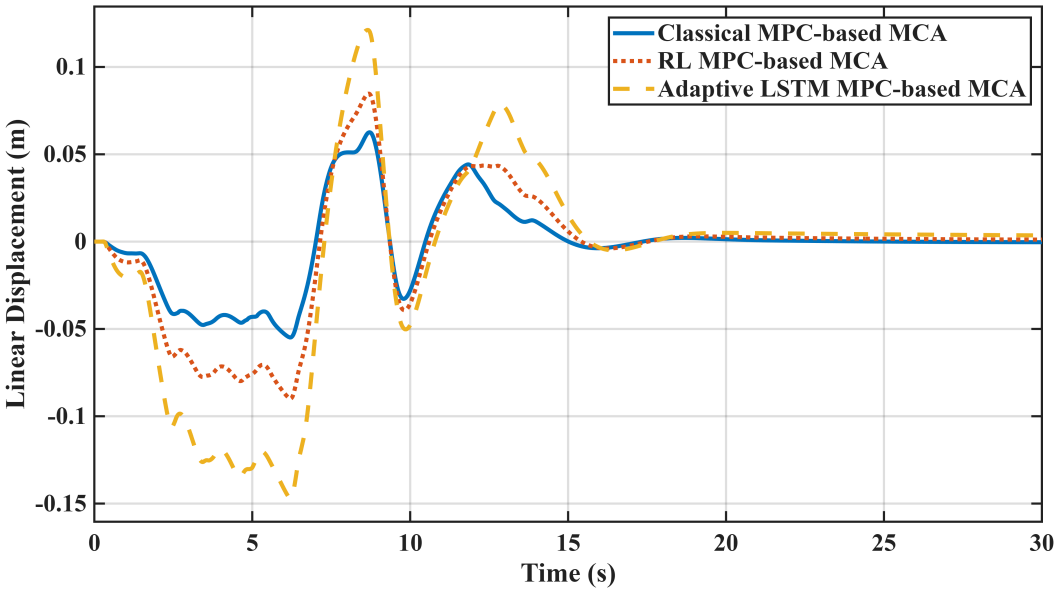


**Figure 6.** Comparison of linear displacement over time.

Figure 7 illustrates the pitch response for the classical MPC-based MCA, RL MPC-based MCA, and adaptive LSTM-MPC-based MCA. The adaptive model closely follows dynamic variations while maintaining a similar overall motion profile. Compared to the classical MPC-based MCA and RL MPC-based MCA, the adaptive approach shows a more responsive pitch behaviour while maintaining a similar overall motion trend.

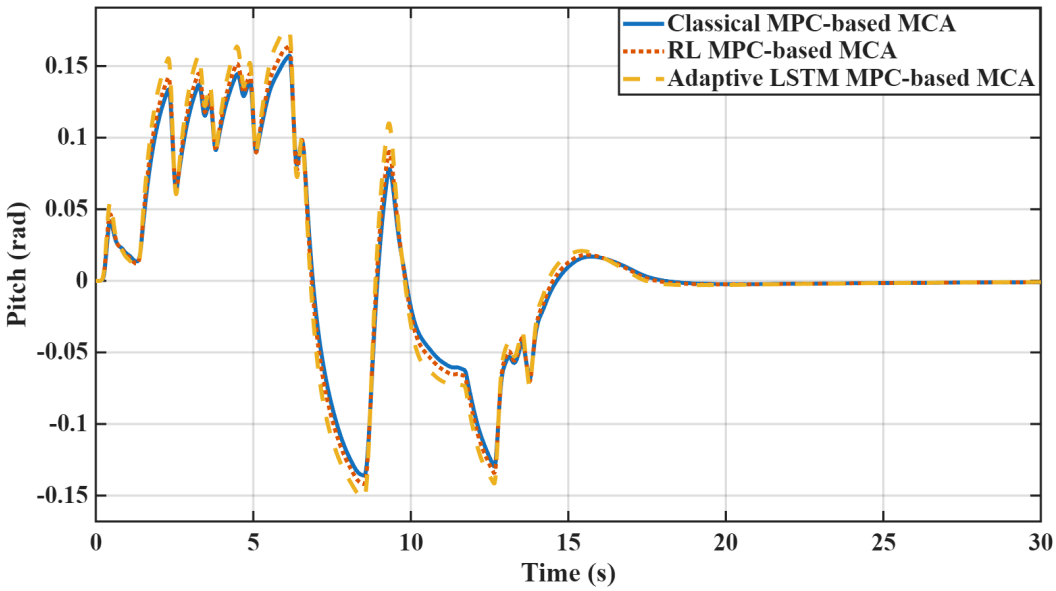


**Figure 7.** Comparison of pitch over time.

Figure 8 compares the per-iteration execution time measured within the MATLAB–Python simulation environment for the classical MPC-based MCA, RL MPC-based MCA, and adaptive LSTM-MPC-based MCA. The classical approach exhibits high variability, with spikes exceeding 7 s per iteration. Since all controller variants were implemented using the same `quadprog` solver and identical solver settings, these differences do not arise from solver configuration. Instead, they reflect the computational burden associated with the fixed prediction horizon, which produces larger lifted prediction matrices, larger Hessian matrices, and consequently larger quadratic programming problems at every optimisation step. The RL MPC-based MCA reduces the execution time compared with the classical approach

but still maintains a higher computational load than the adaptive method. In contrast, the adaptive LSTM-MPC-based MCA maintains a more stable execution time, averaging below 1 s per iteration, by predicting a suitable horizon and selecting the corresponding precomputed lifted matrices. Although the RL-based method achieves better performance in some tracking metrics, the proposed LSTM-based method offers a more favourable balance between tracking performance, workspace utilisation, and computational efficiency.

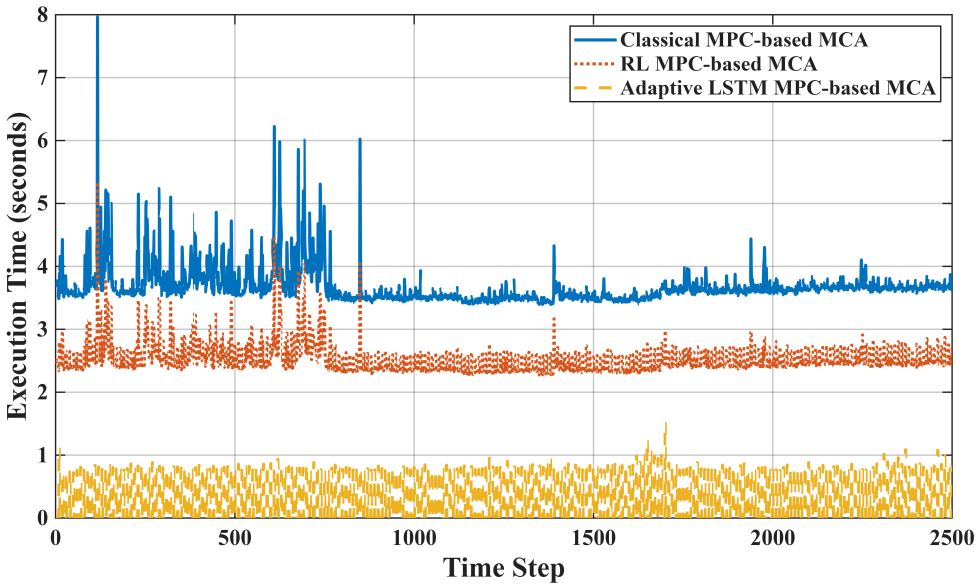


**Figure 8.** Execution time comparison: classical MPC-based MCA, RL MPC-based MCA, and adaptive LSTM-MPC-based MCA.

It should be noted that the reported execution times are provided solely to compare the relative computational burden of the controller formulations and should not be interpreted as evidence of validated real-time implementation. The adaptive prediction horizon mechanism operates online at each simulated control step within the offline MATLAB–Python simulation environment, where the prediction horizon is updated according to recent tracking-performance metrics. The reported timings include software-interaction overhead associated with the co-simulation framework and therefore do not represent real-time wall-clock execution at the controller sampling interval of $T_s = 0.01$ s. Although the measured execution times exceed the sampling interval, the results demonstrate that the proposed adaptive framework substantially reduces computational complexity relative to both the classical MPC-based MCA and the RL MPC-based MCA. Consequently, the contribution of this work is the demonstrated reduction in computational burden and the improved practical feasibility of future real-time implementation on suitable hardware platforms, rather than the validation of real-time operation itself.

The findings highlight the differences between the classical MPC-based MCA, RL MPC-based MCA, and adaptive LSTM-MPC-based MCA in terms of motion fidelity, adaptability, platform utilisation, and computational efficiency. While the classical and RL-based approaches demonstrate competitive tracking behaviour, the adaptive framework offers enhanced responsiveness to dynamic motion variations. The reduction in SSF correlation and the increase in RMSE observed in the adaptive model are counterbalanced by its improved SAV correlation, enhanced LD utilisation, and lower computational load. Although the adaptive approach introduces a marginal increase in error, it provides a more resource-efficient alternative compared to both the classical and RL-based methods. Additionally, LD is notably improved in the adaptive framework, allowing for better platform utilisation and smoother transitions in motion cueing. Furthermore, computational load analysis reveals

a clear advantage of the adaptive LSTM-MPC-based MCA over both the classical MPC-based MCA and RL MPC-based MCA. The classical method exhibits high execution time variability, with frequent computational spikes, indicating a heavier processing burden, while the RL-based method reduces this load but still remains computationally higher than the adaptive approach. In contrast, the adaptive framework maintains consistently lower execution times due to its dynamic prediction horizon adjustment, reducing computational overhead while preserving high motion cueing accuracy. This improvement increases the practical feasibility of future real-time implementation, where both computational efficiency and adaptability are essential.

## 4. Conclusions

MCA plays a critical role in replicating vehicle dynamics and providing realistic motion feedback within the inherent physical constraints of simulation platforms. In the context of driving simulation, MPC provides a principled framework for integrating DL techniques, enabling adaptive optimisation of control parameters to enhance overall system performance. Such advanced control methodologies are broadly applicable to autonomous vehicle development, driver training programmes, vehicle testing protocols, and studies of human motion perception, making them valuable in both industrial practice and academic research.

This work introduced an adaptive LSTM-MPC-based MCA framework that adaptively adjusts the prediction horizon using short-window tracking metrics. Compared with the classical fixed-horizon MPC-based MCA and RL MPC-based MCA, the proposed approach demonstrated improved adaptability, enhanced LD utilisation, and substantially reduced computational load while maintaining competitive motion cueing tracking performance. Although the RL MPC-based MCA achieved superior performance in certain SSF tracking metrics, the proposed framework provided a more favourable balance between tracking performance, workspace utilisation, and computational efficiency through adaptive prediction horizon selection. The improved LD utilisation achieved by the proposed framework allows the motion platform to make more effective use of its available workspace while remaining within platform constraints. In practical terms, this enables the simulator to reproduce a broader range of vehicle manoeuvres, including sustained acceleration, braking, cornering, and lane-change events, without premature workspace saturation. Consequently, improved workspace utilisation contributes to enhanced motion cue fidelity and increases the operational capability of the driving simulator.

The modest increase observed in some tracking errors is considered an acceptable trade-off given the improvements in workspace utilisation and computational efficiency. Importantly, the adaptive LSTM-MPC-based MCA achieved approximately 44.26 times faster execution than the classical MPC-based MCA and approximately 30.03 times faster execution than the RL MPC-based MCA. Furthermore, the adaptive framework maintained lower and more stable execution times throughout the simulation, demonstrating the effectiveness of online horizon adaptation in reducing optimisation complexity.

By coupling DL with MPC, the proposed framework acquires data-driven adaptability by learning from recent motion conditions to refine prediction horizon selection, resulting in a more intelligent and computationally efficient motion cueing strategy. Overall, the results demonstrate that adaptive prediction horizon tuning can improve workspace utilisation and substantially reduce computational burden while maintaining competitive motion cueing performance.

A limitation of the present study is that the proposed framework was evaluated exclusively in a simulation environment. Although simulation-based testing enables systematic assessment of motion cueing performance, workspace utilisation, and computational effi-

ciency under controlled and repeatable conditions, it cannot fully capture human perceptual responses or simulator-sickness effects. Furthermore, the vestibular model used in this study represents a generic human perception model and does not explicitly account for individual physiological differences. Consequently, the reported results should be interpreted as demonstrating the feasibility and performance of the proposed framework for a nominal human observer within a simulation environment.

Future work will focus on implementation using a physical motion platform, hardware-in-the-loop validation, and human-subject experiments to evaluate motion perception, simulator sickness, and user experience under real operating conditions. In addition, personalised vestibular models will be investigated to assess the impact of inter-subject variability on motion cueing performance and adaptive controller behaviour. Future research may also extend the proposed adaptive LSTM-MPC framework beyond motion cueing applications toward emerging intelligent transportation and energy system integration scenarios, including vehicle-to-home (V2H) and vehicle-to-grid (V2G) systems incorporating uncertainty-aware energy management strategies [26,27]. Furthermore, the framework could be extended to simulate hydrogen fuel-cell vehicles and their interaction with future hydrogen infrastructure, supporting the evaluation of sustainable transportation and net-zero energy systems [28].

**Author Contributions:** Conceptualization, S.A.-s. and H.A.; methodology, S.A.-s.; software, S.A.-s.; validation, S.A.-s., M.R.C.Q. and H.A.; formal analysis, S.A.-s.; investigation, S.A.-s.; writing—original draft preparation, S.A.-s.; writing—review and editing, M.R.C.Q., S.M., S.N. and H.A.; supervision, H.A., S.N. and S.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The data presented in this study are available from the corresponding author upon reasonable request.

**Acknowledgments:** The authors would like to acknowledge the support provided by the Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| MCA | Motion Cueing Algorithm |
| MPC | Model Predictive Control |
| DL | Deep Learning |
| LSTM | Long Short-Term Memory |
| MGHA | Metric-Guided Horizon Adaptation |
| SAV | Sensed Angular Velocity |
| SSF | Sensed Specific Force |
| LD | Linear Displacement |
| RMSE | Root Mean Square Error |
| CC | Correlation Coefficient |

## References

1. Koyuncu, A.B.; Erçelik, E.; Comulada-Simpson, E.; Venrooij, J.; Kaboli, M.; Knoll, A. A novel approach to neural network-based motion cueing algorithm for a driving simulator. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; pp. 2118–2125. [CrossRef]
2. Conrad, B.; Schmidt, S.F. *The Calculation of Motion Drive Signals for Piloted Flight Simulators*; Technical Report; NASA: Washington, DC, USA, 1969. Available online: https://ntrs.nasa.gov/citations/19700017803 (accessed on 24 July 2024).

3. Parrish, R.V.; Dieudonne, J.E.; Bowles, R.L.; Martin, D.J., Jr. Coordinated adaptive washout for motion simulators. *J. Aircr.* **1975**, *12*, 44–50. [CrossRef]
4. Houck, J.A.; Telban, R.J.; Cardullo, F.M. *Motion Cueing Algorithm Development: Human-Centered Linear and Nonlinear Approaches*; Technical Report; NASA: Washington, DC, USA, 2005. Available online: https://ntrs.nasa.gov/ (accessed on 24 July 2024).
5. Dagdelen, M.; Reymond, G.; Kemeny, A.; Bordier, M.; Maïzi, N. Model-based predictive motion cueing strategy for vehicle driving simulators. *Control Eng. Pract.* **2009**, *17*, 995–1003. [CrossRef]
6. Fang, Z.; Kemeny, A. Explicit MPC motion cueing algorithm for real-time driving simulator. In Proceedings of the 7th International Power Electronics and Motion Control Conference, Harbin, China, 2–5 June 2012; Volume 2, pp. 874–878. [CrossRef]
7. Gardezi, M.S.M.; Hasan, A. Machine Learning-Based Adaptive Prediction Horizon in Finite Control Set Model Predictive Control. *IEEE Access* **2018**, *6*, 32392–32400. [CrossRef]
8. Wang, S.W.; Yu, D.L.; Gomm, J.B.; Page, G.F.; Douglas, S.S. Adaptive Neural Network Model-Based Predictive Control for Air–Fuel Ratio of SI Engines. *Eng. Appl. Artif. Intell.* **2006**, *19*, 189–200. [CrossRef]
9. Hewing, L.; Wabersich, K.P.; Menner, M.; Zeilinger, M.N. Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annu. Rev. Control. Robot. Auton. Syst.* **2020**, *3*, 269–296. [CrossRef]
10. Winqvist, R.; Venkitaraman, A.; Wahlberg, B. Learning Models of Model Predictive Controllers Using Gradient Data. *IFAC-PapersOnLine* **2021**, *54*, 7–12. [CrossRef]
11. Chadha, A.; Jain, V.; Lazcano, A.M.R.; Shyrokau, B. Computationally-efficient motion cueing algorithm via model predictive control. In Proceedings of the 2023 IEEE International Conference on Mechatronics (ICM), Loughborough, UK, 15–17 March 2023; pp. 1–6. [CrossRef]
12. Hameed, A.; Abadi, A.S.S.; Ordys, A. Model predictive control based motion cueing algorithm for driving simulator. *J. Syst. Sci. Syst. Eng.* **2024**, *33*, 607–626. [CrossRef]
13. Pham, D.A.; Nguyen, D.T. Auto-tuning Prediciton and Control Horizon of Model Predictive Control Approach to Motion Cueing Algorithm Applied in Robocoaster Motion Platform. In *International Conference on Advanced Mechanical Engineering, Automation and Sustainable Development*; Springer: Cham, Switzerland, 2021; pp. 648–654. Available online: https://www.springerprofessional.de/en/auto-tuning-prediciton-and-control-horizon-of-model-predictive-c/20367666 (accessed on 27 September 2024).
14. Riera, J.V.; Casas, S.; Alonso, F.; Fern'andez, M. A VR-Enhanced Rollover Car Simulator and Edutainment Application for Increasing Seat Belt Use Awareness. *Computers* **2021**, *10*, 55. [CrossRef]
15. Andalibi, M.; Shourangizhaghighi, A.; Hajihosseini, M.; Madani, S.S.; Ziebert, C.; Boudjadar, J. Design and Simulation-Based Optimization of an Intelligent Autonomous Cruise Control System. *Computers* **2023**, *12*, 84. [CrossRef]
16. Bejarano, L.A.; Montenegro, C.E.; Espitia, H.E. Optimization of a Fuzzy System Used to Characterize the Factors That Affect Drivers on Urban Roads. *Computers* **2023**, *12*, 70. [CrossRef]
17. Ruan, K.; Di, X. InfoSTGCAN: An Information-Maximizing Spatial-Temporal Graph Convolutional Attention Network for Heterogeneous Human Trajectory Prediction. *Computers* **2024**, *13*, 151. [CrossRef]
18. Huang, K.; Wei, K.; Li, F.; Yang, C.; Gui, W. LSTM-MPC: A deep learning based predictive control method for multimode process control. *IEEE Trans. Ind. Electron.* **2023**, *70*, 11544–11554. [CrossRef]
19. Wu, Z.; Rincon, D.; Christofides, P.D. Real-time adaptive machine-learning-based predictive control of nonlinear processes. *Ind. Eng. Chem. Res.* **2020**, *59*, 2275–2290. [CrossRef]
20. Wong, W.C.; Chee, E.; Li, J.; Wang, X. Recurrent Neural Network-Based Model Predictive Control for Continuous Pharmaceutical Manufacturing. *Mathematics* **2018**, *6*, 242. [CrossRef]
21. Meng, J.; Li, C.; Tao, J.; Li, Y.; Tong, Y.; Wang, Y.; Zhang, L.; Dong, Y.; Du, J. RNN-LSTM-Based Model Predictive Control for a Corn-to-Sugar Process. *Processes* **2023**, *11*, 1080. [CrossRef]
22. Mdoe, Z.; Krishnamoorthy, D.; Jaschke, J. Adaptive horizon multistage nonlinear model predictive control. In Proceedings of the 2021 American Control Conference (ACC), New Orleans, LA, USA, 25–28 May 2021; pp. 2088–2093. [CrossRef]
23. Mohammadi, A.; Asadi, H.; Mohamed, S.; Nelson, K.; Nahavandi, S. Optimizing Model Predictive Control Horizons Using Genetic Algorithm for Motion Cueing Algorithm. *Expert Syst. Appl.* **2018**, *92*, 73–81. [CrossRef]
24. Qazani, M.R.C.; Jalali, S.M.J.; Asadi, H.; Nahavandi, S. Optimising Control and Prediction Horizons of a Model Predictive Control-Based Motion Cueing Algorithm Using Butterfly Optimization Algorithm. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
25. Al-serri, S.; Qazani, M.R.C.; Mohamed, S.; Arogbonlo, A.; Al-ashmori, M.; Lim, C.P.; Nahavandi, S.; Asadi, H. Optimising Horizons in Model Predictive Control for Motion Cueing Algorithms Using Reinforcement Learning. In Proceedings of the 2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Kuching, Malaysia, 6–10 October 2024; pp. 2793–2800. [CrossRef]
26. Giannelos, S.; Pudjianto, D.; Strbac, G. Smart Home Economic Operation Under Uncertainty: Comparing Monte Carlo and Stochastic Optimization Using Gaussian and KDE-Based Data. *Oper. Res. Perspect.* **2025**, *15*, 100348. [CrossRef]

27. Giannelos, S. Option Valuation of Smart Grid Technology Projects Under Endogenous and Exogenous Uncertainty. Ph.D. Dissertation, Imperial College London, London, UK, 2016.
28. Pudjianto, D.; Ameli, H.; Giannelos, S. Holistic Evaluation of the Roles and Values of Electrolysers in a Net-Zero Energy System. In Proceedings of the 14th International Conference on Renewable Power Generation (RPG 2025), Shanghai, China, 24–26 October 2025; pp. 432–440. [CrossRef]