Machine Learning-Driven Model Predictive Control of Modular Multilevel Converters

Singh, S.; Stipanovic, D. M.; Lekic, A.

DOI
10.1109/ACCESS.2025.3614746

Publication date
2025

Document Version
Final published version

Published in
IEEE Access

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 September 2025, accepted 16 September 2025,
date of publication 26 September 2025, date of current version 2 October 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3614746

RESEARCH ARTICLE

Machine Learning-Driven Model Predictive Control of Modular Multilevel Converters

SUNNY SINGH[1], DUŠAN M. STIPANOVIĆ[2], (Fellow, IEEE), AND ALEKSANDRA LEKIĆ[1], (Senior Member, IEEE)

[1]Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD Delft, The Netherlands
[2]Coordinated Science Laboratory, University of Illinois at Urbana–Champaign, Urbana, IL 61801, USA

Corresponding author: Sunny Singh (s.singh-6@tudelft.nl)

This work was supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) Veni Project Smart and Flexible Control for a Power Electronics-based Electrical Grid (SAFE-GRID) under Project 20248.

ABSTRACT The Modular Multilevel Converter (MMC) has garnered significant interest recently due to its superior harmonic performance and improved efficiency in high-voltage direct current electrical grids. Model Predictive Control (MPC) is widely adopted for the MMC applications, as it provides a straightforward control design, facilitates the inclusion of multiple control objectives through a flexible cost function formulation, and offers excellent control performance. An emerging and promising solution involves integrating MPC with machine learning (ML)-based models, in which neural networks learn MPC behavior and predict the results as the traditional MPC does. In this paper, a multi-layered neural network is designed to approximate the control behavior of MPC correctly, enabling a substantial reduction in computational effort during real-time operation and replacing the complex optimization routines of MPC with lightweight neural network regression models that are both efficient and decoupled from the algorithmic complexity of traditional MPC. The performance of controllers is evaluated under both small and large disturbances in active power and reactive power.

INDEX TERMS Machine learning, modular multilevel converter, model predictive control, feedforward neural network.

I. INTRODUCTION

Modular Multilevel Converters are up-and-coming solutions among various voltage source converter technologies due to their excellent scalability, modularity, and suitability for high-power and high-voltage applications, particularly in high-voltage-direct-current transmission systems [1], [2]. MMC has garnered significant interest recently due to its modular structure and for providing several advantages, including superior harmonic performance and improved efficiency, including enhanced power quality, scalability, low harmonic distortion, high efficiency, fast dynamics response, and improved fault tolerance [1], [3], [4], [5], [6], [7].

Despite the many advantages that MMCs have, a significant limitation in controlling MMCs stems from their computational complexity. This challenge escalates rapidly with increasing numbers of submodules (SMs) or output voltage levels. The control algorithm's computational burden is growing significantly as more SMs are added, creating practical implementation constraints for high-voltage applications with numerous SMs. However, their complex internal configuration and numerous switching states pose significant challenges for precise and efficient control, requiring entering the new advanced control era, such as MPC and ML-based MPC approaches. Numerous model-based control techniques have been studied for MMC systems to control the MMCs and to highlight their importance, including Linear Quadratic Regulation (LQR) [8], adaptive control [9], and MPC [10]. Among various control strategies, MPC has proven particularly effective for MMCs, due to its predictive capability and flexibility in handling multivariable dynamics. MPC has gained widespread adoption; it can evaluate the system state and provides a robust framework for managing these complexities, offering optimal control solutions in real time while explicitly handling constraints [10], [11], [12].

However, a significant limitation in MPC applications arises from the extensive number of switching states, significantly increasing the computational load and complicating

The associate editor coordinating the review of this manuscript and approving it for publication was Riccardo Mandrioli.

real-time execution [13], [14], [15]. Despite its benefits, a major challenge of MPC lies in the intensive computational effort required for the online optimization [16]. The complexity escalates particularly in explicit MPC, where many input sequences must be evaluated at each sampling instant to determine the optimal control action. This computational demand becomes even more pronounced when the controller is tasked with regulating multiple constraint variables simultaneously, thereby complicating real-time implementation. The computational effort associated with MMC control has been reduced from the primary cost function by decoupling the SM capacitor-voltage regulation in [17]. In the work [18] further alleviates the burden by introducing a novel MPC framework that segments the cost function into three distinct components, each tailored for specific control objectives, effectively minimizing the number of evaluated states. A fast MPC approach is presented in [19], incorporating an SM voltage balancing method to enhance speed and responsiveness. Again, to address the issues, the Laguerre functions have been integrated into the formulation of discrete-time MMC models [10], [11], [20]. Using Laguerre functions significantly reduces dependency on the control horizon, thereby reducing the computational effort required for prediction and making it easily applicable. Despite these advancements, the fundamental nature of MPC remains unchanged, namely, the need to evaluate switching possibilities at every time step. Consequently, the applicability of MPC remains constrained to systems operating at relatively low switching frequencies.

To address these challenges, ML techniques, particularly neural networks, have been explored as a promising alternative. A basic neural network architecture can be trained offline by making use of the data from conventional MPC controlling the MMC. Once trained, the neural network can replicate the control behavior of MPC while significantly reducing the real-time computational burden. Moreover, the data-driven nature of neural networks offers increased robustness against model inaccuracies, making them suitable for practical implementations of predictive control in complex power conversion systems. A supervised learning framework is introduced in [21] and [22] to replicate the behavior of MPC with significantly lower computational complexity. The advancement in artificial intelligence has significantly improved the control capabilities of various kinds of power converters by enabling intelligent and efficient control strategies [23]. ML algorithms have demonstrated the potential to optimize converter real-time performance by learning from the historical data and dynamically adjusting the control parameters. By using supervised learning methods, it is possible to develop ML-based MPC controllers that mimic the decision-making process of MPC. These models can predict optimal control actions with minimal delay, making them efficient alternatives to traditional optimization-based approaches. Replacing conventional MPC with neural network model predictive control (NNMPC) offers a promising approach to mitigate the computational complexity associated with traditional MPC in MMC applications [24]. A developed ML model, such as a neural network, can be trained offline using data that is generated from the traditional MPC operating under various load conditions, current references, and model parameters. Once trained, the NNs can replicate the control behavior of the MPC while significantly reducing real-time computational requirements. This approach is thus suitable for real-time control applications, especially in systems like MMCs, where the high computational complexity of traditional MPC often poses a challenge [12], [24], [25].

This paper presents an ML-based MPC approach as an efficient alternative to traditional MPC for MMCs. Data were collected from the Laguerre-based MPC formulation to approximate the controller behavior and used to train a neural network. This trained model replicates the control policy, enabling fast predictions with reduced computational overhead. Simulation results depict that the ML-based controller closely matches the performance of traditional MPC and discrete linear quadratic regulator (DLQR) approaches in the unconstrained case, effectively regulating the MMC.

Based on the above discussion in the present article, the key contributions of the present work are highlighted as follows

1) We introduce a variable substitution to reformulate the state-weighting matrix in the cost function. Replacing the identity matrix with a tailored non-identity matrix enables unequal prioritization of state variables according to their significance for the control objectives.
2) The present study applies the proposed NNMPC approach to the MMC. It includes simulation results obtained with the traditional MPC and the benchmark DLQR controller to enable a clear comparative performance assessment.
3) The proposed controller is tested under different constraint scenarios, including rate and amplitude limits and combined constraints of both rate and amplitude, and their effect on the output current. Again, the sensitivity analysis using Laguerre parameters is systematically investigated

The present paper is organized as follows: Section II presents the discrete-time model of the MMC and MPC formulation. Section III details the development of the neural network-based MPC framework. Section IV provides the simulation results to validate the proposed control strategy. Section V comments on this paper's overall results and offers future potential directions.

II. MATHEMATICAL FORMULATION OF THE MMC AND MPC

A. STATE-SPACE MODEL OF MMC

The MMC presents a three-phase structure comprising three legs, each corresponding to a separate phase, $j \in \{a, b, c\}$. Every leg includes a lower and an upper arm, and each arm is composed of N_{SM} half-bridge SMs, as depicted in Fig. 1, where the voltages and currents in the upper (denoted as U)

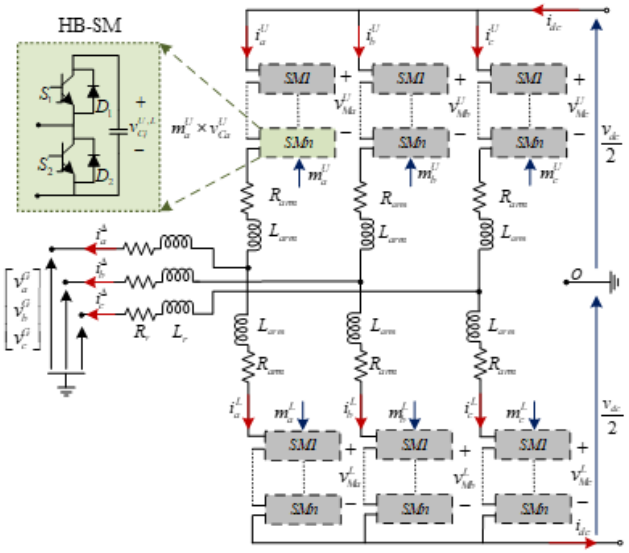

FIGURE 1. MMC topology.

and lower (denoted as L) arms can be described by:

$$v_{Mj}^{U,L} = \quad m_j^{U,L} v_{Cj}^{U,L}, \quad i_{Mj}^{U,L} = m_j^{U,L} i_j^{U,L} i_j^{U,L}. \tag{1}$$

Here, $m_j^{U,L}$ are the insertion indices of the upper and lower SMs, respectively, and $v_{Cj}^{U,L}$, $v_{Mj}^{U,L}$ represent sum of SM capacitor voltages, and equivalent voltages on SMs, respectively. The development of the converter model is based on the approach described in [26]. By adopting the Σ–Δ notation, it is possible to express the variables associated with the upper and lower arms of the converter as follows:

$$i_j^{\Delta} = i_j^U - i_j^L, \quad i_j^{\Sigma} = \frac{i_j^U + i_j^L}{2}, \tag{2a}$$

$$v_{Cj}^{\Delta} = \frac{v_{Cj}^U - v_{Cj}^L}{2}, \quad v_{Cj}^{\Sigma} = \frac{v_{Cj}^U + v_{Cj}^L}{2}, \tag{2b}$$

$$m_j^{\Delta} = m_j^U - m_j^L, \quad m_j^{\Sigma} = m_j^U + m_j^L \tag{2c}$$

$$v_{Mj}^{\Delta} = \frac{-v_{Mj}^U + v_{Mj}^L}{2} = \frac{-m_j^{\Delta} v_{Cj}^{\Sigma} + m_j^{\Sigma} v_{Cj}^{\Sigma}}{2} \tag{2d}$$

$$v_{Mj}^{\Sigma} = \frac{v_{Mj}^U + v_{Mj}^L}{2} = \frac{m_j^{\Sigma} v_{Cj}^{\Sigma} + m_j^{\Delta} v_{Cj}^{\Delta}}{2}. \tag{2e}$$

The dqz-frame variables [11], [26] are created using Park transformation:

$$P_{\omega_0}(t) = \frac{2}{3} \begin{bmatrix} \cos(\omega_0 t) & \cos\left(\omega_0 t - \frac{2\pi}{3}\right) & \cos\left(\omega_0 t - \frac{4\pi}{3}\right) \\ \sin(\omega_0 t) & \sin\left(\omega_0 t - \frac{2\pi}{3}\right) & \sin\left(\omega_0 t - \frac{4\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}.$$

Next, the differential equations are written by considering inductor currents and capacitor voltages as state variables:

$$\frac{d}{dt}\left(\vec{i}_{dq}^{\Delta}\right) = \frac{\vec{v}_{Mdq}^{\Delta} - \left(-\omega L_{eq}^{ac} J_2 + R_{eq}^{ac} I_2\right) \vec{i}_{dq}^{\Delta} - \vec{v}_{dq}^{G}}{L_{eq}^{ac}} \tag{3a}$$

$$\frac{d}{dt}\left(\vec{i}_{dq}^{\Sigma}\right) = -\frac{\vec{v}_{Mdq}^{\Sigma} + (R_{arm} I_2 - 2\omega L_{arm} J_2)\, \vec{i}_{dq}^{\Sigma}}{L_{arm}}, \tag{3b}$$

$$\frac{d}{dt}\left(i_z^{\Sigma}\right) = \frac{v_{dc}}{2L_{arm}} - \frac{v_{Mz}^{\Sigma} + R_{arm} i_z^{\Sigma}}{L_{arm}}, \tag{3c}$$

where ω is the angular frequency, and $L_{eq}^{ac} = L_r + \frac{L_{arm}}{2}$, $R_{eq}^{ac} = R_r + \frac{R_{arm}}{2}$. I_2 is the identity matrix of dimension 2×2, while $J_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$.

The operational dynamics of the MMC can be fully characterized by the following discrete-time differential equation: $\dot{\vec{x}}_m = A_m \vec{x}_m + B_m \vec{u}$, where $\vec{x}_m = [i_d^{\Sigma}, i_q^{\Sigma}, i_z^{\Sigma}, i_d^{\Delta}, i_q^{\Delta}]^T$ is the state that corresponds to the output current and $\vec{u} = [v_{Md}^{\Sigma}, v_{Mq}^{\Sigma}, v_{Mz}^{\Sigma} - \frac{v_{dc}}{2}, v_{Md}^{\Delta} - v_d^{\Delta}, v_{Mq}^{\Delta} - v_q^{\Delta}]^T$ represents difference between the corresponding voltages in the dq-frame [10], [11], [27]. Matrices A_m and B_m are given by:

$$A_m = \begin{bmatrix} \frac{-R_{arm}}{L_{arm}} & 2\omega & 0 & 0 & 0 \\ -2\omega & \frac{-R_{arm}}{L_{arm}} & 0 & 0 & 0 \\ 0 & 0 & \frac{-R_{arm}}{L_{arm}} & 0 & 0 \\ 0 & 0 & 0 & \frac{-R_{eq}^{ac}}{L_{eq}^{ac}} & -\omega \\ 0 & 0 & 0 & \omega & \frac{-R_{eq}^{ac}}{L_{eq}^{ac}} \end{bmatrix},$$

$$B_m = \text{diag}\left\{-\frac{1}{L_{arm}}, -\frac{1}{L_{arm}}, -\frac{1}{L_{arm}}, \frac{1}{L_{eq}^{ac}}, \frac{1}{L_{eq}^{ac}}\right\}. \tag{4}$$

The zero-order hold (ZOH) method [11] is used as standard in controller design and is applied here to discretize the MMC continuous state-space model [27]:

$$\vec{x}_m(k+1) = A_p(T_s)\vec{x}_m(k) + B_p(T_s)\vec{u}(k) \tag{5a}$$

$$\vec{y}(k+1) = C_p(T_s)\vec{x}_m(k), \tag{5b}$$

Where $C_p(T_s)$ is an identity matrix, and $A_p(T_s)$ and $B_p(T_s)$ are given as the exact solution of the differential equation as:

$$A_p(T_s) = e^{A_m T_s}, \qquad B_p(T_s) = A_m^{-1}\left(e^{A_m T_s} - I\right) B_m.$$

The resulting model provides AC and DC representations of the MMC.

B. MPC FORMULATION

The discrete-time MMC model can be characterized with $\vec{u}$ denoting the control signals and $\vec{x}_m$ representing the output signals. An integral action is incorporated into the output formulation to ensure zero steady-state error, producing the enhanced discrete-time MMC model. The standard MMC

model, is given by the following equation: (6)

$$\underbrace{\begin{bmatrix}\Delta\vec{x}_m(k+1)\\ \vec{y}(k+1)\end{bmatrix}}_{\vec{x}(k+1)} = \underbrace{\begin{bmatrix}A_p(T_s) & \mathbf{0}^T\\ C_p(T_s)A_p(T_s) & 1\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}\Delta\vec{x}_m(k)\\ \vec{y}(k)\end{bmatrix}}_{\vec{x}(k)} + \underbrace{\begin{bmatrix}B_p(T_s)\\ C_p(T_s)B_p(T_s)\end{bmatrix}}_{B}\Delta\vec{u}(k), \tag{6a}$$

$$\vec{y}(k) = \underbrace{[\mathbf{0}_m \quad I]}_{C}\vec{x}(k). \tag{6b}$$

Here, $\mathbf{0}$ is a zero matrix, $\Delta\vec{u}(k) = \vec{u}(k)-\vec{u}(k-1)$ and $\Delta\vec{x}_m(k+1) = \vec{x}_m(k+1) - \vec{x}_m(k)$ is the forward difference where $\vec{u}(k)$ and $\vec{u}(k-1)$ indicates the input to the system, for instance k and $k-1$, respectively.

In conventional MPC schemes, using the input increment $\Delta\vec{u}$ in the cost function, particularly under high sampling frequencies and stringent closed-loop performance requirements, necessitates many parameters, specifically a longer control horizon N_c. This can lead to numerical ill-conditioning and a significant increase in computational effort. To address this, a Laguerre function-based representation [28] is introduced that approximates the input increment sequence $\Delta u = [\Delta\vec{u}(k_i)\ \Delta\vec{u}(k_i+1)\ \cdots\ \Delta\vec{u}(k_i+N_c-1)]^T$. Instead of directly optimizing each increment, this method reformulates the problem using a pulse basis operator $\Delta\delta(i)$, allowing $\Delta\vec{u}(k_i+j)$ to be expressed as:

$$\Delta\vec{u}(k_i+j) = [\delta(i)\ \delta(i-1)\ \cdots\ \delta(i-N_c+1)], \tag{7}$$

where $\delta(i)$ denote the pulse operator is expressed as:

$$\delta(i) = \begin{cases}1, & \text{if } i = 0,\\ 0, & \text{if } i \neq 0.\end{cases} \tag{8}$$

This transformation significantly reduces the number of optimization parameters, decreasing the computational load during real-time implementation. Consequently, the MPC becomes more efficient and suitable for fast dynamic systems like MMCs [28].

At the k^{th} time step, the control parameter changes from $\Delta\vec{u}(k)$ to $\vec{\eta}$ as

$$\Delta\vec{u}(k+k_i|k) = L(k|k_i)^T\vec{\eta}, \tag{9a}$$

$$L(k_i+1|k) = A_l L(k_i|k), \tag{9b}$$

where

$$A_l = \begin{bmatrix}a & 0 & 0 & 0\\ \beta & a & 0 & 0\\ -\beta & \beta & a & 0\\ \vdots & \vdots & \vdots & \vdots\end{bmatrix},$$

$$L(0) = \sqrt{\beta}\begin{bmatrix}1 & -a & a^2 & \cdots & (-1)^{N-1}a^{N-1}\end{bmatrix}^T,$$

$$\bar{\eta} = \begin{bmatrix}c_1 & c_2 & \ldots & c_N\end{bmatrix}^T.$$

The parameter a represents Laguerre's network pole, where $0 < a < 1$ ensures network stability. In this work, we select $a = 0.237$ and define $\beta = \sqrt{1-a^2}$. The number of terms N used to approximate the system response is set to $N = 4$.

The optimal value of the vector, $\vec{\eta}$, is obtained by minimizing the cost function while also satisfying the constraints of equality and inequality. So the cost function in this case is derived based on the DLQR as the base, which is expressed as:

$$\min_{\vec{\eta}} J = \sum_{m=1}^{N_p}\vec{x}(k+m|k)^T Q\vec{x}(k+m|k) + \vec{\eta}^T R\vec{\eta}, \tag{10a}$$

$$\text{subject to } M\vec{\eta} \leq b, \tag{10b}$$

$$\vec{x}(k+m|k) = (\vec{r}(k) - \vec{y}(k+m|k))^T. \tag{10c}$$

The weighting matrices satisfy $Q \succeq 0$ (positive semidefinite) and $R \succ 0$ (positive definite). Also, $\mathbf{M}$ and $\mathbf{b}$ are column vectors representing constraint information related to amplitude, rate, or a combination of both, and $r(k)$ denotes the reference signal. While a common practice sets $Q = C^\top C$ (yielding an identity matrix in standard cases) where C is the output matrix, our specific control problem requires a modified Q matrix to be positive definite and different from the identity matrix. To achieve this transformation in a structured way, we modify the output matrix C in (6). Let us define $\widetilde{C} = EC$, where E is a nonsingular matrix. This change requires adjusting the output vector as $\tilde{\vec{y}}(k) = \widetilde{C}\vec{x}(k) = E\vec{y}(k)$, and the state vectoras:

$$\vec{x}(k+1) = \left[(\Delta\vec{x}_m(k+1))^T\ (E^{-1}\tilde{\vec{y}}(k+1))^T\right]^T. \tag{11}$$

By applying the following transformations, the corresponding augmented discrete-time MMC model in (6) can be reformulated as:

$$\vec{x}(k+1) = \begin{bmatrix}\Delta\vec{x}_m(k+1)\\ E^{-1}\tilde{\vec{y}}(k+1)\end{bmatrix} = A\vec{x}(k) + B\Delta\vec{u}(k)$$
$$= \begin{bmatrix}A_p & 0\\ C_pA_p & I\end{bmatrix}\begin{bmatrix}\Delta\vec{x}_m(k)\\ E^{-1}\tilde{\vec{y}}(k)\end{bmatrix} + \begin{bmatrix}B_p\\ C_pB_p\end{bmatrix}\Delta\vec{u}(k).$$

Therefore, these modifications are also valid for the augmented plant model.

The cost function in MPC is generally designed to minimize the deviation between the output signal and the desired setpoint (reference) signal. A standard form of such a cost function can be expressed mathematically, which may serve as a basis for developing the specific form of the cost function used in this work. To explore the structure and influence of the weighting matrix Q, we consider two distinct cases formulated and discussed in the subsequent sections.

The general cost function is of the form that minimizes the error between the setpoint (reference) signal and the output signal. It is expressed in the form:

$$J = \min_{\vec{\eta}}\sum_{m=1}^{N_p}\left[(\vec{r}(k_i) - \tilde{\vec{y}})^T \times(\vec{r}(k_i) - \tilde{\vec{y}}) + \vec{\eta}^T R\vec{\eta}\right], \tag{12}$$

or

$$J = \min_{\vec{\eta}} (R - Y)^T (R - Y) + \vec{\eta}^T R \vec{\eta}. \tag{13}$$

Our objective is to reformulate the cost function to resemble the structure of the standard DLQR cost function. To achieve this, we analyze the specific choices of the weighting matrix Q under which the cost function in (12) becomes equivalent to the DLQR form. We can align both formulations by selecting Q as a symmetric positive semi-definite matrix that emphasizes the tracking error similarly to the DLQR framework. This alignment ensures that our approach inherits the desirable stability and performance characteristics of DLQR control design. To investigate this equivalence, we consider two distinct cases for the setpoint signal, which will be discussed in detail as **Case I** and **Case II** in the following sections.

1) CASE I

Suppose $\vec{r}(k) = 0$, then from equation (12), we can assume that $\vec{e}(k+m|k) = 0 - \vec{\tilde{y}}(k+m|k)$. Then the cost function:

$$\begin{aligned} J &= \sum_{m=1}^{N_p} \left(0 - \vec{\tilde{y}}(k+m|k)\right)^T \left(0 - \vec{\tilde{y}}(k+m|k)\right) \\ &\quad + \vec{\eta}^T R \vec{\eta} \\ &= \sum_{m=1}^{N_p} x(k_i+m|k_i)^T C^T E^T E C x(k_i+m|k_i) + \vec{\eta}^T R \vec{\eta}. \end{aligned} \tag{14}$$

By comparing the cost functions in equations (10a) and (14), we find that they are identical when $r(k) = 0$ if and only if

$$Q = C^T E^T E C. \tag{15}$$

2) CASE II

In the case when $\vec{r}(k) \neq 0$. Then, based on equation (12), we want to reformulate the cost function in a DLQR-like structure, recall the output matrix as

$\widetilde{C} = E \begin{bmatrix} \mathbf{0} & I \end{bmatrix}$.

We define a vector $\vec{x}_r(k_i) = [\mathbf{0}^T \ (E^{-1}\vec{r}(k_i))^T]^T$, where $\vec{x}_r(k_i)$ and the augmented state variable $\vec{x}(k)$ have the same dimensions, with the number of zero entries equal to the dimension of $\Delta\vec{x}_m(k)$. This structure ensures that the reference can be expressed as

$$\vec{r}(k_i) = \widetilde{C}\vec{x}_r(k_i). \tag{16}$$

Here, it is crucial to note that during the processes of prediction and optimization, the setpoint signal $\vec{r}(k_i)$ remains constant throughout the optimization horizon. Again, in this case, the state vector is categorized as

$$\vec{x}(k+1) = \left[(\Delta\vec{x}_m(k))^T \ (E^{-1}(\vec{r}(k_i) - \vec{\tilde{y}}(k_i+m|k_i)))^T\right]^T \tag{17}$$

We choose a vector $\vec{\tilde{y}}$ and pre-assume the DLQR cost function and matrix Q:

$$J = \sum_{m=1}^{N_p} \vec{x}(k+m|k)^T Q \vec{x}(k+m|k) + \vec{\eta}^T R \vec{\eta},$$

by substituting the value of the Q and $\vec{x}$, one can obtain the following

$$\begin{aligned} J &= \sum_{m=1}^{N_p} \left[(\Delta\vec{x}_m(k_i+m|k_i))^T \ (E^{-1}\left(\vec{r}(k_i) - \vec{\tilde{y}}(k_i+m|k_i)\right))^T\right] \\ &\quad \times \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & E^T E \end{bmatrix} \begin{bmatrix} \Delta\vec{x}_m(k_i+m|k_i) \\ E^{-1}\left(\vec{r}(k_i) - \vec{\tilde{y}}(k_i+m|k_i)\right) \end{bmatrix} + \vec{\eta}^T R \vec{\eta} \\ &= \sum_{m=1}^{N_p} \Bigg[\left(\vec{r}(k_i) - \vec{\tilde{y}}(k_i+m|k_i)\right)^T \\ &\quad \times (E^{-1})^T E^T E E^{-1} \left(\vec{r}(k_i) - \vec{\tilde{y}}(k_i+m|k_i)\right) + \vec{\eta}^T R \vec{\eta} \Bigg]. \end{aligned}$$

After simplifying, we get

$$\begin{aligned} J = \min_{\eta} \sum_{m=1}^{N_p} \Bigg[&\left(\vec{r}(k_i) - \vec{\tilde{y}}(k_i+m|k_i)\right)^T \\ &\times \left(\vec{r}(k_i) - \vec{\tilde{y}}(k_i+m|k_i)\right) + \vec{\eta}^T R \vec{\eta} \Bigg]. \end{aligned} \tag{18}$$

This demonstrates that equations (12) and (18) are identical, confirming that the cost function aligns with the DLQR formulation used thus far.

Again, from equation (12) it follows that:

$$\begin{aligned} J &= \min_{\vec{\eta}} \sum_{m=1}^{N_p} \left[(r(k_i) - \vec{\tilde{y}})^T (r(k_i) - \vec{\tilde{y}}) + \vec{\eta}^T R \vec{\eta}\right], \\ J &= \min_{\vec{\eta}} \sum_{m=1}^{N_p} \left[((r(k_i) - \vec{\tilde{y}})^T (E^{-1})^T E^T E E^{-1} \right. \\ &\quad \left. \times ((r(k_i) - \vec{\tilde{y}}) + \vec{\eta}^T R \vec{\eta}\right] \\ &= \min_{\vec{\eta}} \sum_{m=1}^{N_p} \left[(E^{-1}(r(k_i) - \vec{\tilde{y}}))^T E^T E \right. \\ &\quad \left. \times (E^{-1}(r(k_i) - \vec{\tilde{y}})) + \vec{\eta}^T R \vec{\eta}\right] \end{aligned}$$

$$\begin{aligned} J &= \sum_{m=1}^{N_p} \left[(\Delta\vec{x}_m(k))^T \ (E^{-1}(\vec{r}(k_i) - \vec{\tilde{y}}(k_i+m|k_i)))^T\right] \\ &\quad \times \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & E^T E \end{bmatrix} \begin{bmatrix} \Delta\vec{x}_m(k_i+m|k_i) \\ E^{-1}\left(\vec{\tilde{y}}(k_i+m|k_i) - \vec{r}(k_i)\right) \end{bmatrix} \\ &\quad + \vec{\eta}^T R \vec{\eta} \\ &= \sum_{m=1}^{N_p} \vec{x}(k_i+m|k_i)^T (CE)^T (CE) \vec{x}(k_i+m|k_i) + \vec{\eta}^T R \vec{\eta}. \end{aligned} \tag{19}$$

which is equivalent to DLQR cost function. Therefore, we can confidently state that the cost function remains consistent with that of the DLQR, validating our approach even for a nonzero setpoint. Based on this discussion, we ensure that the DLQR is the base for our approach, and the optimization problem is defined in equation (10), where $Q = C^T E^T EC$, M and b are column vectors representing the constraint information related to both rate and amplitude limitations. For this problem, we set

$$R = 10^{-5} \times I, \qquad E = \begin{bmatrix} 2 & -1 \\ 3 & -2 \end{bmatrix}.$$

III. NEURAL NETWORKS BASED MPC CONSTRUCTION

Neural networks are powerful tools for modeling complex systems because they can process information in parallel and perform distributed computations. They are instrumental when accurate mathematical models are unavailable or only partial knowledge of the system states is accessible. With sufficient training data, neural networks can learn the system behavior and provide accurate approximations of system behaviors by removing the traditional computations.

Due to system identification by neural networks, their nonlinear characteristics, and their ability to learn directly from data without requiring prior system knowledge, neural networks are well-suited for use in MPC frameworks. They can capture the dynamics of complex, multivariable systems, making them highly valuable in advanced control strategies. There are several types of neural networks, such as the multilayer perceptron (MLP), radial basis function (RBF) network, and various recurrent neural networks (RNNs). Despite their differences, all these networks and chosen architectures share common components like layers, neurons (nodes), and weighted connections. In this work, we employ a particular multilayer neural network, which is widely used in various applications involving prediction, system identification, and control.

Since our application involves predicting multiple future outputs, we use the neural network iteratively. The output from each prediction step is fed back into the network as input for the next step. This approach allows us to generate a sequence of future values that can be integrated into the MPC strategy.

In our present work, a feedforward neural network with hidden layers, each using the hyperbolic tangent activation function, and a linear output layer was trained using the Levenberg-Marquardt algorithm (LM) (`trainlm`) in MATLAB to model the nonlinear system under study. The loss function is critical for training, quantifies prediction errors, and guides weight optimization.

To evaluate prediction accuracy, we employed the mean squared error (MSE) as the loss function, defined as follows:

$$\text{MSE} = \frac{1}{2N} \sum_{t=1}^{N} \left[y(t) - \hat{y}\right]^T \left[y(t) - \hat{y}\right],$$

where N is the number of data points, y_i is the true output, and $\hat{y}_i$ is the predicted output. MSE is suitable for regression due to its differentiability and sensitivity to errors, making it ideal for the continuous outputs of this problem.

To minimize the MSE, `trainlm` employs backpropagation to compute the gradients of the MSE with respect to the network weights, forming the Jacobian matrix. The LM algorithm then optimizes the weights. A detailed analysis of the LM algorithm is provided in Appendix A.

The complete setup for the NNMPC working process, data collection, neural network training, and using the trained model for control performance, is clearly described in Algorithm 1. Again, in order to help visualize each stage, a flowchart is also provided in Fig. 2, showing the step-by-step procedure from initial data preparation to real-time implementation on the MMC system.

Algorithm 1 Neural Network-Based Output Prediction for MPC

Input: System matrices A_p, B_p, C_p; Initial state $\vec{x}_0$; Control input sequence $\{\vec{u}_k\}_{k=1}^{N}$ from MPC
Output: Trained neural network $\mathcal{N}$ for output prediction

Initialize state: $\vec{x}_1 \leftarrow \vec{x}_0$;
for $k = 1$ **to** $N - 1$ **do**
  Compute output: $\vec{y}_k = C_p \vec{x}_k$;
  Update state: $\vec{x}_{k+1} = A_p \vec{x}_k + B_p \vec{u}_k$;
  Store input-output pair: $X_k = [\vec{u}_k, \vec{x}_k]$, $Y_k = \vec{y}_{k+1}$;
Assemble training data: $X = \{X_k\}_{k=1}^{N-1}$, $Y = \{Y_k\}_{k=1}^{N-1}$;
Define and train NN: $\mathcal{N} = \texttt{train}(X, Y)$ with activation functions `tansig` and `purelin`, training algorithm `trainlm`, epochs = 3000, regularization = 0.05;
Split data: 70% training, 15% validation, 15% testing;
Evaluate performance: compute MSE; compare predicted vs actual outputs;
foreach *new input-state pair* $[\vec{u}_k, \vec{x}_k]$ *during MPC execution* **do**
  Predict output: $\hat{\vec{y}}_{k+1} = \mathcal{N}([\vec{u}_k, \vec{x}_k])$;

The optimal network architecture was determined by targeting a minimum MSE of 10^{-5}, at which point training was terminated. The configuration that first attained this target MSE was selected among the candidate models. To provide a more comprehensive assessment of predictive performance, we also report a mean absolute error (MAE) of 0.09, which is less sensitive to large deviations, and a mean absolute percentage error (MAPE) of 0.01876%, which quantifies relative prediction accuracy. Upon completion of training, the model's generalization capability was evaluated on an independent validation dataset. If validation performance proved unsatisfactory—indicating potential underfitting or overfitting—the model was re-trained either by re-initializing

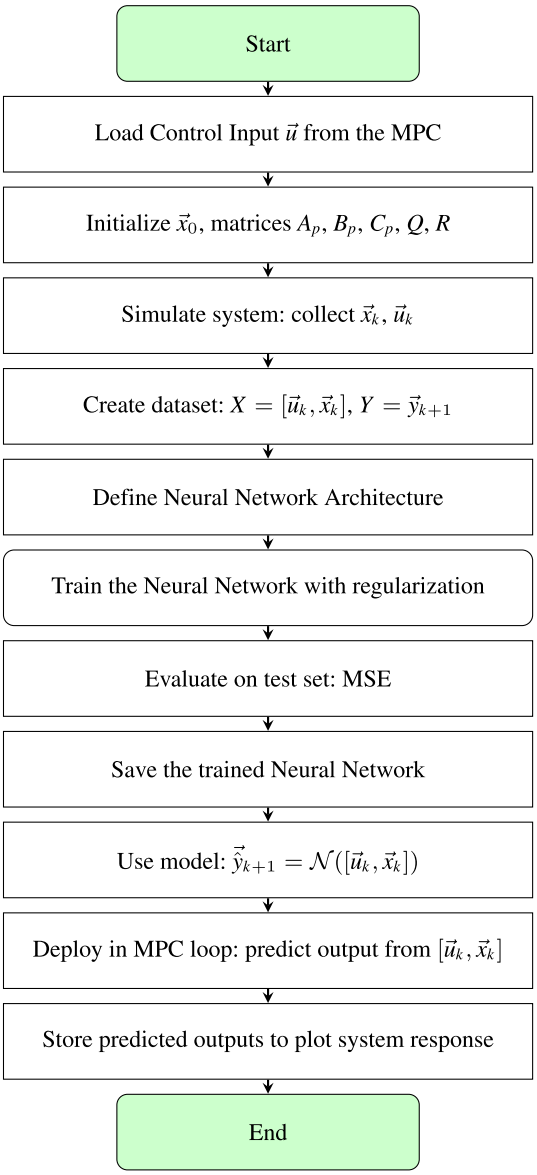

FIGURE 2. Flowchart of NNMPC algorithm.

weights and biases or by modifying the network structure to achieve improved accuracy.

IV. PERFORMANCE EVALUATION AND SIMULATION RESULTS OF MPC AND NNMPC

A. PARAMETERS FOR MMC

To evaluate the performance of the MMC model illustrated in Fig. 1, the parameters are given in Table 1.

The active and reactive power setpoints were intentionally varied to evaluate the dynamic performance of the MMC when controlled using both conventional MPC and NNMPC. These variations were clearly observed through the output currents in the *dq* reference frame. The study examines two main scenarios: small and large disturbances. For each scenario, simulations of two events were performed at different time intervals. Both active and reactive power variables were evaluated, each undergoing changes at specific time instances, as detailed in Table 2.

TABLE 1. System Parameters of MMC Used in Simulations.

Name	Symbol	Value
Arm inductance	L_{arm}	0.15 [p.u.]
Arm resistance	R_{arm}	0.0015 [p.u.]
AC filter inductance	L_f	0.12 [p.u.]
AC filter resistance	R_f	0.003 [p.u.]
Sample time	T_s	2 [ms]

TABLE 2. Simulation scenarios for NNMPC under small and large disturbances.

Changes	Variable	1st Change	2nd Change
Small	Active power	0.5 p.u. (10)	1.0 p.u. (30)
	Reactive Power	0.2 p.u. (20)	1.0 p.u. (30)
Large	Active power	-1.0 p.u. (40)	1.0 p.u. (80)
	Reactive Power	-1.0 p.u. (50)	0.5 p.u. (80)

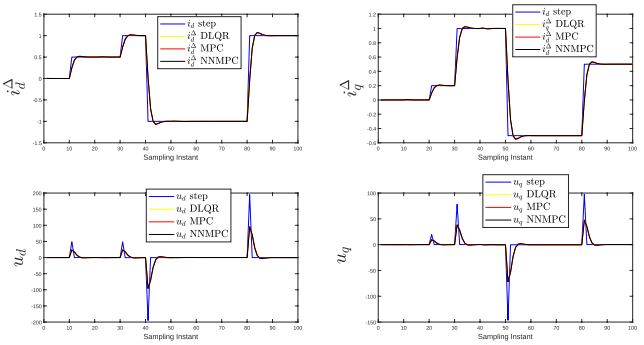

FIGURE 3. Case-study with unconstrained scenario showing $\Delta u_{d,q}$ dynamics.

B. SIMULATION RESULTS FOR THE UNCONSTRAINED CASE

First, we consider the unconstrained case, where no constraints are imposed on state variables. At the initial state, the active as well as reactive power setpoints are set to zero, as illustrated in Figs. 3–4. The traditional MPC, NNMPC, and DLQR responses closely align, tracking the same trajectory. When the setpoint changes, comparisons at various sampling instants reveal nearly identical responses between the controllers. Under small disturbances, MPC, NNMPC, and DLQR demonstrate comparable performance, exhibiting minimal overshoot, while for larger disturbances, the overshoot is larger. The MMC plant's output and control input results for the unconstrained case are depicted in Figs. 3–4. These results demonstrate the system's sensitivity to disturbance magnitude when state constraints are absent, underscoring its dynamic response under such conditions. Without constraints, both the MPC and NNMPC approaches yield performance comparable to the DLQR, as the DLQR is commonly used as a baseline for evaluating these controllers. This similarity arises because, in the unconstrained case,

the predicted plant output can also be obtained using the DLQR method. However, this equivalence holds only in the unconstrained scenario. The DLQR approach becomes inapplicable where introduced, as it cannot explicitly handle constraints on states or inputs. Therefore, in the constrained case, we focus on the predicted plant outputs obtained using MPC and NNMPC, which are inherently capable of handling such constraints. This capability is one of the key reasons why MPC and NNMPC are widely adopted in practical applications.

Analyzing system output responses under constrained scenarios is essential, as it provides valuable insights into controller performance and stability under practical operating conditions. Most of the MMC plant challenges are based on constraints.

C. SIMULATION RESULTS FOR CONSTRAINT SCENARIOS

One of the distinguishing features of MPC compared to DLQR is its inherent ability to handle constraints within the online optimization process. In this section, we investigate how three distinct constraint scenarios influence the plant's output by comparing the responses obtained from both traditional MPC and the proposed NNMPC approach:

1) Rate constraint;
2) Amplitude constraint;
3) Both rate and amplitude constraints.

Constraints on the system output may be considered in the current analysis, although this can be included and visualized if required; in some cases, they may lead to system instability.

1) RATE CONSTRAINT

To observe the effect of rate constraints, we have imposed three different rate constraint scenarios, which are outlined in the following cases:

(a) $|\Delta u_d| = |\Delta u_q| = 30$;
(b) $|\Delta u_d| = |\Delta u_q| = 40$;
(c) $|\Delta u_d| = |\Delta u_q| = 60$.
(d) unconstrained case.

The results obtained from both the MPC and NNMPC approaches exhibit similar behavior. Moreover, the significant impact of the rate constraints on the output signals i_d^{Δ} and i_q^{Δ} is observed. From the Figs. 5-7 it is clear that the rate constraints do not violate the corresponding constraints limit as we impose on it.

2) AMPLITUDE CONSTRAINT

To analyze the effect of amplitude constraints on the system's output dynamics, we consider the following limits:

(a) $|u_d| = |u_q| = 30$;
(b) $|u_d| = |u_q| = 40$;
(c) $|u_d| = |u_q| = 60$;
(d) unconstrained case.

The influence of these amplitude bounds on the system's output is illustrated in Figs. 8-10. The conventional MPC and the proposed NNMPC, both exhibit the similar performances

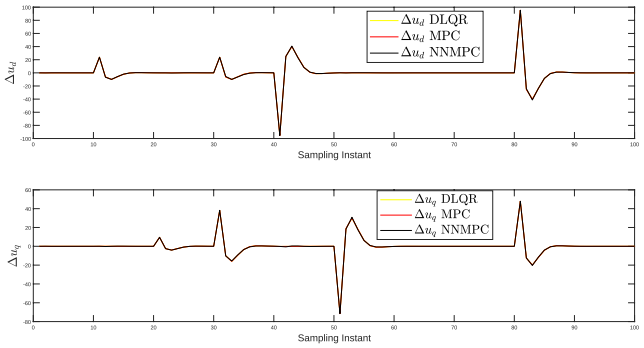

FIGURE 4. Case-study with unconstrained scenario showing $\Delta u_{d,q}$ dynamics.

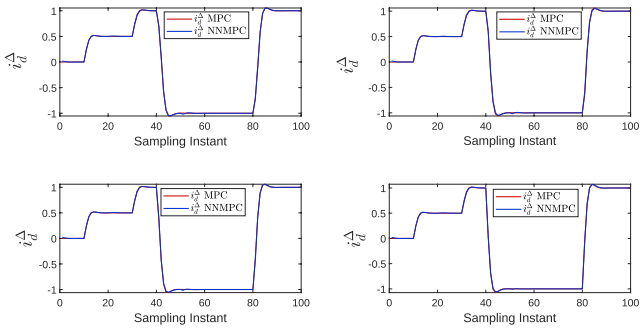

FIGURE 5. Dynamics of i_d^{Δ} current with constrained voltage scenarios: upper left - $|\Delta u_d| = 30$; upper right - $|\Delta u_d| = 40$; lower left - $|\Delta u_d| = 60$; lower right - $|\Delta u_d| = 400$ (unconstrained case).

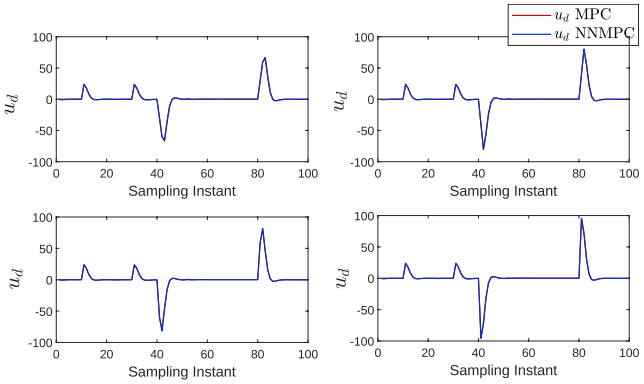

FIGURE 6. Dynamics of u_d voltage with constrained scenarios: upper left - $|\Delta u_d| = 30$; upper right - $|\Delta u_d| = 40$; lower left - $|\Delta u_d| = 60$; lower right - $|\Delta u_d| = 400$ (unconstrained case).

across all cases, demonstrating that the neural network has effectively learned the underlying control behavior of the MPC and accurately replicates its response. The control inputs remain within the specified amplitude limits from minor disturbances, with no violations observed. However, under larger disturbances, the control effort approaches the imposed bounds. As shown in Fig. 9, even when the inputs reach their limits, the constraint handling capability of both control strategies ensures compliance. Interestingly, Fig. 10 reveals that while the control inputs $u_{d,q}$ stay

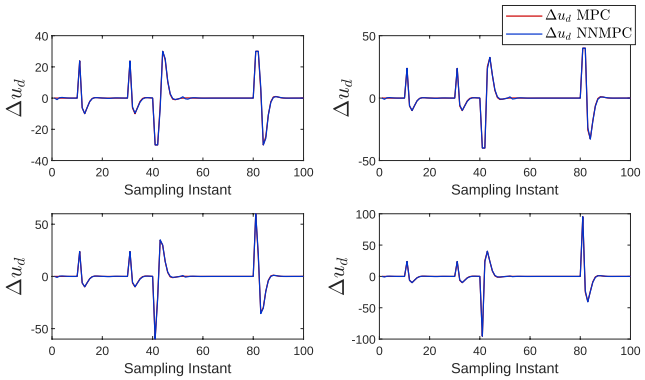

FIGURE 7. Dynamics of Δu_d voltage with constraints: upper left - $|\Delta u_d| = 30$; upper right - $|\Delta u_d| = 40$; lower left - $|\Delta u_d| = 60$; lower right - $|\Delta u_d| = 400$ (unconstrained case).

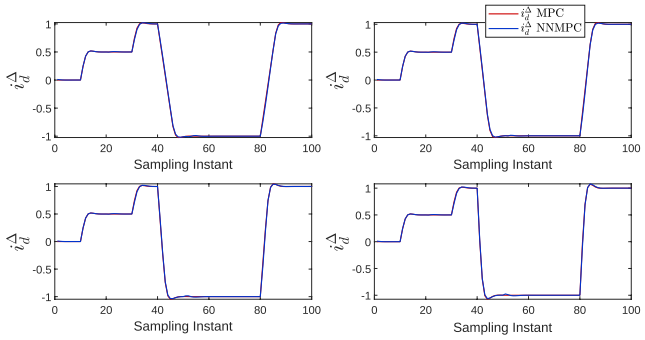

FIGURE 8. Dynamics of i_d^{Δ} current with amplitude constraints: upper left - $|u_d| = 30$; upper right - $|u_d| = 40$; lower left - $|u_d| = 60$; lower right - $|u_d| = 400$ (unconstrained case).

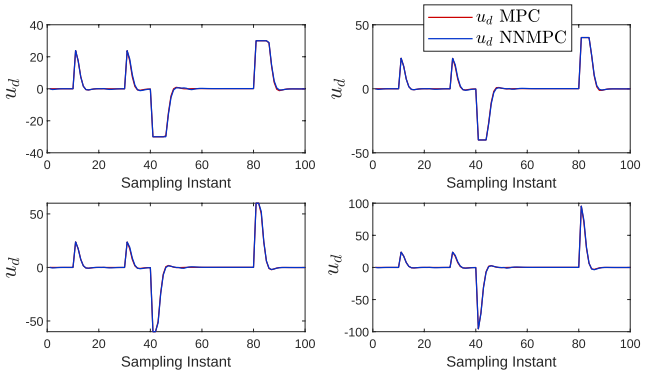

FIGURE 9. Dynamics of u_d current with amplitude constraints: upper left - $|u_d| = 30$; upper right - $|u_d| = 40$; lower left - $|u_d| = 60$; lower right - $|u_d| = 400$ (unconstrained case).

within their respective limits, the rate of change $\Delta u_{d,q}$ may temporarily exceed expected dynamics under significant disturbances. This suggests that while the absolute control values are constrained, the dynamics of the control signal are more aggressive in response to sudden changes in system conditions.

The dynamic behavior of the system under each of these amplitude constraints is illustrated in Figs. 8-10. These figures highlight how different amplitude limits influence

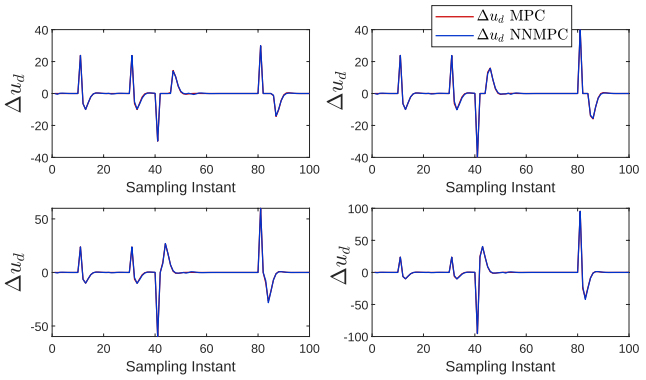

FIGURE 10. Dynamics of Δu_d voltage with amplitude constraints: upper left - $|u_d| = 30$; upper right - $|u_d| = 40$; lower left - $|u_d| = 60$; lower right - $|u_d| = 400$ (unconstrained case).

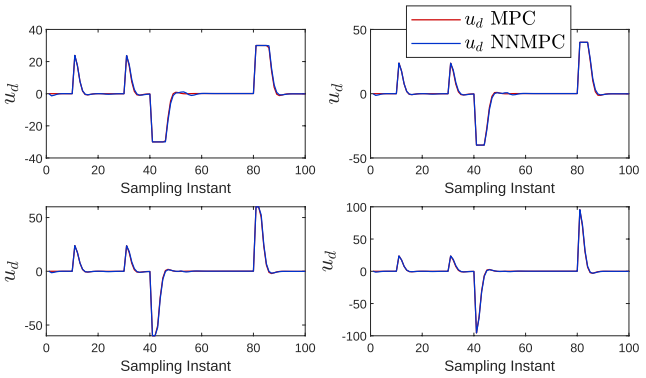

FIGURE 11. Dynamics of u_d with rate $|\Delta u_d|$ and amplitude $|u_d|$ constraints: upper left - case (a); upper right - case (b); lower left - case (c); lower right - case (d).

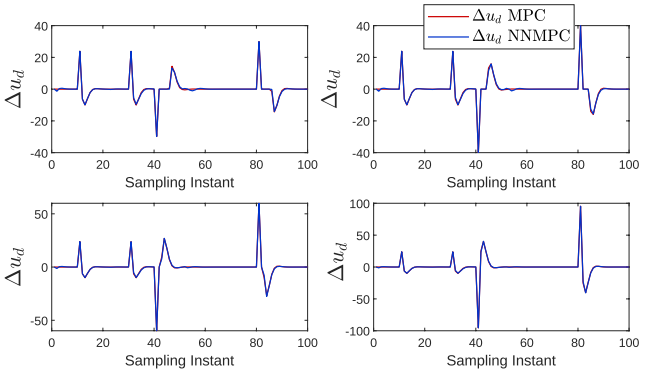

FIGURE 12. Dynamics of Δu_d with rate $|\Delta u_d|$ and amplitude $|u_d|$ constraints: upper left - case (a); upper right - case (b); lower left - case (c); lower right - case (d).

the overall system response. The constraints $|u_d| = |u_q| = 30$ significantly impact the control of the output response. Specifically, they lead to a noticeably higher undershoot when compared to the cases with higher constraints ($|\Delta u_d| = |\Delta u_q| = 60$) and the unconstrained scenario. This behavior highlights the trade-off between tight rate limitations and

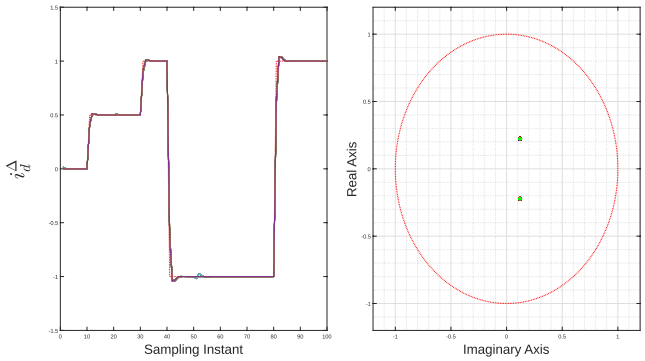

FIGURE 13. Effect of prediction horizon N_p with fixed Laguerre parameters (a, N).

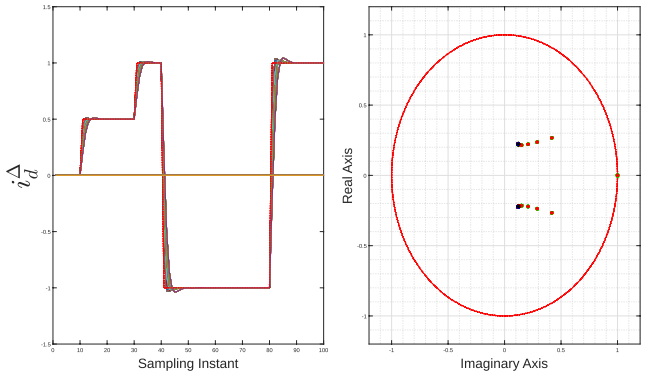

FIGURE 14. Effect of Laguerre pole a with fixed N_p and N.

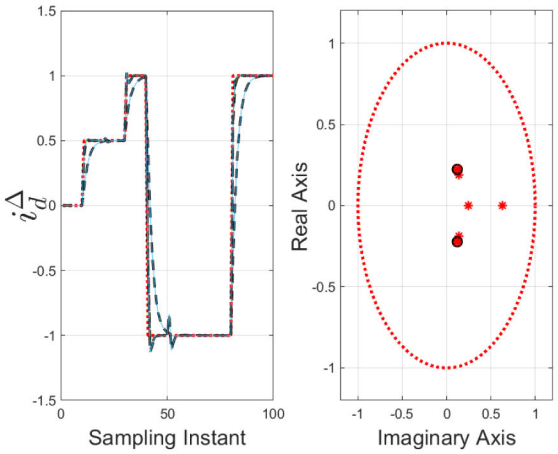

FIGURE 15. Effect of Laguerre terms N with fixed N_p and a.

the controller's ability to promptly correct deviations in the system responsively.

3) BOTH RATE AND AMPLITUDE CONSTRAINTS

To investigate the combined effect of rate and amplitude constraints on the output response, we impose the following limits To understand the effect of rate and amplitude constraints together on the output response, the following limits on rate and amplitude constraints are imposed:

(a) $|\Delta u_d| = |\Delta u_q| = 30$, $|u_d| = |u_q| = 30$;
(b) $|\Delta u_d| = |\Delta u_q| = 40$, $|u_d| = |u_q| = 40$;
(c) $|\Delta u_d| = |\Delta u_q| = 60$, $|u_d| = |u_q| = 60$.
(d) unconstrained case.

The corresponding responses of the system under both small and large disturbances are illustrated in Figs. 11-12. The results demonstrate that after imposing both amplitude and rate constraints, the control signals $u_{d,q}$ and $\Delta u_{d,q}$ remain within their specified limits across all disturbance levels. Furthermore, the traditional MPC and the proposed NNMPC yield nearly identical responses, effectively satisfying the constraints even under larger disturbances. This observation confirms that the neural network model successfully captures the constraint-handling behavior of the conventional MPC approach.

D. SIMULATION RESULTS BASED ON THE SENSITIVITY ANALYSIS

The influence of Laguerre's parameter (a, N) and the predictive horizon (N_p) is depicted in Figs. 13-15. The predictive horizon N minimally impacts system behavior for fixed values of a and N at given sampling time, T_s. Specifically, the response of the neural networks-based system remains identical for $N_p = 4$ and $N_p = 40$, demonstrating that the MMC model is essentially independent of the length of the predictive horizon in this case. In this case, the output of the NNMPC system follows the setpoint more accurately, as clearly depicted in Fig. 13. We fix the prediction horizon and sampling time to observe the Laguerre parameters' effect on the neural network-based model's output. The Laguerre parameters a and N exhibit more significant effects on system performance, as increasing the parameter N causes the system poles to converge toward those of the underlying DLQR controller. As a result, the behavior of the NNMPC closely resembles that of the DLQR controller. Although the system output shows significant fluctuations from the reference signal at some instances, this behavior is depicted in Fig. 14. Again, increasing the pole parameter a drives the system toward instability. Specifically, when the pole is chosen as $a = 1$, the eigenvalues shift toward the boundary of the unit circle. These results suggest that while the predictive horizon can be kept relatively short without performance degradation, careful selection of Laguerre parameters (N and a) is crucial for maintaining system stability and achieving desired control performance.

V. CONCLUSION

In this study, we present a neural network-based MPC strategy for controlling the MMC plant model. Traditional control methods often struggle with MMCs because of their complex structure and large number of submodules, which make real-time control more difficult and increase computational load. To overcome these challenges, a multi-

layer neural network is used to learn the plant's dynamic behavior. After training, the network can accurately predict the MMC system's output from the input variables, removing the need for a detailed mathematical model.

The major contribution of this paper is the development of a cost function in which the weighting matrix Q is non-identity. This matrix is derived through a transformation of the output matrix, necessitating corresponding adjustments in the augmented model of the MMC. The resulting non-identity Q matrix has unequal weighting of the system states, enabling more flexible tuning and a more precise representation of control objectives.

Further to evaluate the effectiveness of the proposed framework, we analyzed the plant's output behavior across different control scenarios. Firstly, we deeply considered the investigation of the unconstrained case, where it was observed that the output response from our proposed NNMPC controller is aligned with similar output response behaviour with the DLQR, traditional MPC, and the proposed NNMPC all produce similar output responses. The results confirm that the neural network is well-trained and successfully mimics the behavior of a traditional MPC. The proposed simulations show that the NNMPC framework can control the system effectively, providing accurate predictions while significantly reducing the computational cost. The proposed method performs well across different operating conditions, including setpoint tracking. These results show that the NNMPC approach is robust, adaptable, and well-suited for real-time control of MMC systems.

APPENDIX A

The objective of the training procedure is to find the network weight coefficients that minimize the discrepancy between the actual outputs $y(t)$ and the outputs produced by the artificial neural network, denoted by $\hat{y}$. More formally, the cost function employed for training in this work is the MSE (where N represents the total number of samples)

$$L(.) = \frac{1}{2N}\sum_{t=1}^{N}\left[y(t) - \hat{y}\right]^T\left[y(t) - \hat{y}\right]. \tag{20}$$

The LM algorithm is developed specifically to address the nonlinear least-squares problems. When working with experimental or observed data, least-squares problems are commonly used for model fitting. The process achieves this by defining an objective function, which sums the squared differences between the values predicted by the model and the actual measurements. Minimizing this sum results in the optimal parameters for the selected model, thereby reducing the difference between the model predictions and the actual values. This method serves as a cornerstone in data analysis, statistics, and numerous scientific applications, offering a robust framework for interpreting and modeling observed patterns [29]. One of the important properties of the LM method is its adaptive nature. When the current parameter estimates are far from the optimum, the algorithm behaves like gradient descent, helping to ensure stable and cautious convergence. Conversely, when the parameters approach their optimal values, it behaves more like the Gauss-Newton method, allowing for faster convergence. Due to this flexibility in choosing the search direction at each iteration, the LM algorithm tends to be more robust than the Gauss-Newton method alone, which is considered one of its main advantages. In general, curve fitting involves adjusting a model function $\hat{y}(t, p)$, which depends on a vector p of n parameters and an independent variable t to best match a set of N observed data points (t_i, y_i). The common way is to minimize the MSE between the measured y_i and the estimated output $\hat{y}(t_i, p)$, as given earlier in equation (20). The MSE can be expressed as below:

$$L(p) = \left(y - \hat{y}(p)\right)^T W\left(y - \hat{y}(p)\right), \tag{21}$$

Again,

$$L(p) = y^T Wy - 2\, y^T W\hat{y}(p) + \hat{y}(p)^T W\hat{y}(p). \tag{22}$$

where, the matrix W is diagonal, and its elements given by

$$W(i, i) = \frac{1}{2N}. \tag{23}$$

In general, curve fitting involves adjusting a model function $\hat{y}(t, p)$, which depends on an independent variable t and a vector of n parameters p, best to match a set of N When the model function $\hat{y}(t, p)$ is nonlinear concerning the parameter vector p, minimizing the cost function defined in equation (20) must be performed using iterative methods. One widely used approach is the gradient descent algorithm, which updates the parameter estimates in the opposite direction of the gradient of the target function.

The gradient of this function regarding the parameters p can be obtained as follows:

$$\frac{\partial L}{\partial p} = 2\left(y - \hat{y}(p)\right)^T W\frac{\partial}{\partial p}\left(y - \hat{y}(p)\right). \tag{24}$$

Due to independency of y with p, this expression can be rewritten as

$$= -2\left(y - \hat{y}(p)\right)^T W\frac{\partial\hat{y}(p)}{\partial p}. \tag{25}$$

Now the Jacobian matrix J is defined as the partial derivatives that capture the local sensitivity of the estimated outputs to parameter modifications

$$J = \frac{\partial\hat{y}}{\partial p}, \tag{26}$$

The gradient becomes:

$$\frac{\partial L}{\partial p} = -2\left(y - \hat{y}\right)^T WJ. \tag{27}$$

For every successive step, it is important to compute vector h that adjusts the parameter vector p. This update may guide the parameters toward the steepest descent of the target function.

The update vector for the basic gradient descent method can be estimated as follows

$$h_{\text{gd}} = \alpha_1 J^T W\big(y - \hat{y}\big), \tag{28}$$

where α_1 is a positive scalar that controls the step size in the gradient descent direction. By applying the first-order Taylor series expansion of the model output around the current parameter estimate p, the following expression is rewritten as

$$\hat{y}(p+h) \approx \hat{y}(p) + \frac{\partial \hat{y}}{\partial p} h = \hat{y} + Jh. \tag{29}$$

For the next step, derivation using the expression from equation (21) and substituting the linearized model output, we have:

$$\begin{aligned} L(p+h) \approx\ & y^T W y + \hat{y}^T W \hat{y} \\ & - 2\, y^T W \hat{y} - 2(y - \hat{y})^T WJh + h^T J^T WJh. \end{aligned} \tag{30}$$

In order to find the parameter h which minimizes this approximate cost function, we set its gradient with respect to h equal to zero:

$$\frac{\partial L(p+h)}{\partial h} \approx -2(y - \hat{y})^T W_1 J + 2\, h^T J^T W_1 J = 0. \tag{31}$$

Solving this condition leads to the update equation used in the Gauss–Newton algorithm:

$$(J^T WJ)\, h_{\text{gn}} = J^T W(y - \hat{y}). \tag{32}$$

The LM algorithm is smart because it can act like gradient descent or the Gauss–Newton method when needed. In every step, it updates the parameters using this rule:

$$\big(J^T WJ + \lambda I\big)\, h_{\text{LM}} = J^T W\big(y - \hat{y}\big), \tag{33}$$

where λ and I stand for the damping factor and identity matrix, respectively. When λ is large, the update behaves similarly to gradient descent, producing cautious steps toward the minimum. As λ decreases over successive iterations, the algorithm increasingly resembles the Gauss-Newton approach, allowing faster convergence near the optimum. The perturbation vector h_{LM} which further is expressed as:

$$h_{\text{LM}} = \big(J^T WJ + \lambda I\big)^{-1} J^T W(y - \hat{y}). \tag{34}$$

The convergence of the LM algorithm is particularly ensured if the cost function L consistently decreases for every step, which is inspired by the Lyapunov theory, requires:

$$\Delta L = L(p+h) - L(p) < 0. \tag{35}$$

Ensuring this condition holds throughout optimization is preferred. In order to formulate the expression for ΔL, one subtracts the value of the function $L(p)$ from $L(p+h)$, obtaining the following expression.

$$\Delta L = -2(y - \hat{y})^T WJh + h^\top J^T WJh. \tag{36}$$

This derivation illustrates how the parameter update h contributes to reducing the cost function at each iteration. By substituting the expression for h from equation (34) into the variation of the objective function ΔL given by (36), one arrives at:

$$\begin{aligned} \Delta L =\ & -2(y - \hat{y})^T WJ(J^T WJ + \lambda I)^{-1} J^\top W(y - \hat{y}) \\ & + \big[(J^T WJ + \lambda I)^{-1} J^T W(y - \hat{y})\big]^T \\ & \big(J^T WJ + \lambda I - \lambda I\big)\big(J^T WJ + \lambda I\big)^{-1} J^T W(y - \hat{y}) \\ =\ & -(y - \hat{y})^T WJ(J^T WJ + \lambda I)^{-1} J^T W_1 (y - \hat{y}) \\ & - \lambda (y - \hat{y})^T WJ\big(J^T WJ + \lambda I\big)^{-T} \big(J^T WJ \\ & + \lambda I\big)^{-1} J^\top W(y - \hat{y}) \\ =\ & -(y - \hat{y})^T WJ(J^T WJ + \lambda I)^{-1} J^T W(y - \hat{y}) \\ & - \lambda \big\|(J^T WJ + \lambda I)^{-1} J^T W(y - \hat{y})\big\|^2. \end{aligned} \tag{37}$$

Here, the last two terms are negative semidefinite or at least non-positive definite, ensuring that $\Delta L \leq 0$ under typical conditions. The first term in equation (37) is non-positive because it is a quadratic form involving the matrix $(J^\top WJ + \lambda I)^{-1}$, which is symmetric and positive definite. Similarly, the second term is non-positive as it is scaled by the damping parameter $\lambda > 0$. In the derivation, the notation $^{-T}$ indicates the transpose of the inverse matrix. Since $(J^T WJ + \lambda I)^{-1}$ remains positive definite, the only scenario where the expression in equation (37) does not yield a negative value—specifically when $\Delta L = 0$ occurs when $J^T W(y - \hat{y}) = 0$. This result effectively demonstrates that the algorithm achieves local stability in the Lyapunov sense concerning the quadratic objective function defined earlier in (20). A detailed analysis quantifying the degree of local stability is not included here, as it would require a more in-depth exploration beyond the scope of this work. It is worth mentioning that, in practical terms, the algorithm consistently performed effectively and successfully reached convergence.

REFERENCES

[1] A. Lesnicar and R. Marquardt, "An innovative modular multilevel converter topology suitable for a wide power range," in *Proc. IEEE Bologna Power Tech. Conf.*, vol. 3, May 2003, pp. 272–277.

[2] M. Vatani, B. Bahrani, M. Saeedifard, and M. Hovd, "Indirect finite control set model predictive control of modular multilevel converters," *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1520–1529, May 2015.

[3] P. Hu, R. Teodorescu, S. Wang, S. Li, and J. M. Guerrero, "A currentless sorting and selection-based capacitor-voltage-balancing method for modular multilevel converters," *IEEE Trans. Power Electron.*, vol. 34, no. 2, pp. 1022–1025, Feb. 2019.

[4] M. Chethan and R. Kuppan, "Optimizing inertia estimation in virtual synchronous generators with MMC-based proportional integral and model predictive control strategies," *IEEE Access*, vol. 13, pp. 43424–43434, 2025.

[5] A. Antonopoulos, L. Ängquist, and H. Nee, "On dynamics and voltage control of the modular multilevel converter," in *Proc. 13th Eur. Conf. Power Electron. Appl.*, Sep. 2009, pp. 3353–3362.

[6] L. Zhang, Y. Zou, J. Yu, J. Qin, V. Vittal, G. G. Karady, D. Shi, and Z. Wang, "Modeling, control, and protection of modular multilevel converter-based multi-terminal HVDC systems: A review," *CSEE J. Power Energy Syst.*, vol. 3, no. 4, pp. 340–352, Dec. 2017.

[7] A. Aslam and M. Raza, "Design and implementation of active control method for minimizing circulating current in MMC-VSC system," *IEEE Access*, vol. 13, pp. 124471–124482, 2025.

[8] J. D. López, J. J. Espinosa, and J. R. Agudelo, "LQR control for speed and torque of internal combustion engines," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 2230–2235, Jan. 2011.
[9] E. Lavretsky and K. A. Wise, "Robust adaptive control," in *Advanced Textbooks in Control and Signal Processing*. Cham, Switzerland: Springer, 2013, pp. 317–353.
[10] A. Shetgaonkar, L. Liu, A. Lekić, M. Popov, and P. Palensky, "Model predictive control and protection of MMC-based MTDC power systems," *Int. J. Electr. Power Energy Syst.*, vol. 146, Mar. 2023, Art. no. 108710.
[11] A. Shetgaonkar, A. Lekić, J. L. Rueda Torres, and P. Palensky, "Microsecond enhanced indirect model predictive control for dynamic power management in MMC units," *Energies*, vol. 14, no. 11, p. 3318, Jun. 2021.
[12] D. Wang, Z. J. Shen, X. Yin, S. Tang, X. Liu, C. Zhang, J. Wang, J. Rodriguez, and M. Norambuena, "Model predictive control using artificial neural network for power converters," *IEEE Trans. Ind. Electron.*, vol. 69, no. 4, pp. 3689–3699, Apr. 2022.
[13] D. Liao-McPherson, M. Huang, S. Kim, M. Shimada, K. Butts, and I. Kolmanovsky, "Model predictive emissions control of a diesel engine airpath: Design and experimental evaluation," *Int. J. Robust Nonlinear Control*, vol. 30, no. 17, pp. 7446–7477, Nov. 2020.
[14] B. Masalmeh, R. Prasad, V. Nougain, and A. Lekić, "Neural networks in RSCAD: Enhancing MMC-based HVDC simulation with advanced machine learning," *IEEE Trans. Ind. Appl.*, vol. 61, no. 2, pp. 2515–2526, Mar. 2025.
[15] A. Norouzi, H. Heidarifar, H. Borhan, M. Shahbakhti, and C. R. Koch, "Integrating machine learning and model predictive control for automotive applications: A review and future directions," *Eng. Appl. Artif. Intell.*, vol. 120, Apr. 2023, Art. no. 105878.
[16] S. D. Tavakoli, S. Fekriasl, E. Prieto-Araujo, J. Beerten, and O. Gomis-Bellmunt, "Optimal H_∞ control design for MMC-based HVDC links," *IEEE Trans. Power Del.*, vol. 37, no. 2, pp. 786–797, Apr. 2021.
[17] B. Gutierrez and S.-S. Kwak, "Modular multilevel converters (MMCs) controlled by model predictive control with reduced calculation burden," *IEEE Trans. Power Electron.*, vol. 33, no. 11, pp. 9176–9187, Nov. 2018.
[18] J.-W. Moon, J.-S. Gwon, J.-W. Park, D.-W. Kang, and J.-M. Kim, "Model predictive control with a reduced number of considered states in a modular multilevel converter for HVDC system," *IEEE Trans. Power Del.*, vol. 30, no. 2, pp. 608–617, Apr. 2015.
[19] Z. Gong, P. Dai, X. Yuan, X. Wu, and G. Guo, "Design and experimental evaluation of fast model predictive control for modular multilevel converters," *IEEE Trans. Ind. Electron.*, vol. 63, no. 6, pp. 3845–3856, Jun. 2016.
[20] B. S. Riar, T. Geyer, and U. K. Madawala, "Model predictive direct current control of modular multilevel converters: Modeling, analysis, and experimental evaluation," *IEEE Trans. Power Electron.*, vol. 30, no. 1, pp. 431–439, Jan. 2015.
[21] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Syst. Lett.*, vol. 2, no. 3, pp. 543–548, Jul. 2018.
[22] S. Wang, T. Dragicevic, Y. Gao, and R. Teodorescu, "Neural network based model predictive controllers for modular multilevel converters," *IEEE Trans. Energy Convers.*, vol. 36, no. 2, pp. 1562–1571, Jun. 2021.
[23] S. Zhao, F. Blaabjerg, and H. Wang, "An overview of artificial intelligence applications for power electronics," *IEEE Trans. Power Electron.*, vol. 36, no. 4, pp. 4633–4658, Apr. 2021.
[24] S. Wang, T. Dragicevic, G. F. Gontijo, S. K. Chaudhary, and R. Teodorescu, "Machine learning emulation of model predictive control for modular multilevel converters," *IEEE Trans. Ind. Electron.*, vol. 68, no. 11, pp. 11628–11634, Nov. 2021.
[25] N. Yousefi, J. Ebrahimi, and A. Bakhshai, "Neural network controller based on direct and indirect model predictive for modular multilevel converters," in *Proc. 25th Eur. Conf. Power Electron. Appl.*, Sep. 2023, pp. 1–8.
[26] G. Bergna-Diaz, J. Freytes, X. Guillaud, S. D'Arco, and J. A. Suul, "Generalized voltage-based state-space modeling of modular multilevel converters with constant equilibrium in steady state," *IEEE J. Emerg. Sel. Topics Power Electron.*, vol. 6, no. 2, pp. 707–725, Jun. 2018.
[27] A. Zama, A. Benchaib, S. Bacha, D. Frey, and S. Silvant, "High dynamics control for MMC based on exact discrete-time model with experimental validation," *IEEE Trans. Power Del.*, vol. 33, no. 1, pp. 477–488, Feb. 2018.
[28] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB*, vol. 3. Cham, Switzerland: Springer, 2009.
[29] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, Jun. 1963.

SUNNY SINGH received the B.Sc. and M.Sc. degrees in applied mathematics from Deen Dayal Upadhyaya Gorakhpur University, Gorakhpur, India, in 2014 and 2016, respectively, and the Ph.D. degree in mathematics from Indian Institute of Technology (BHU), Varanasi, India, in 2024. He is currently a Postdoctoral Researcher with Delft University of Technology (TU Delft), The Netherlands. His research interests include nonlinear dynamics, control theory, neural networks, and machine learning.

DUŠAN M. STIPANOVIĆ (Fellow, IEEE) received the B.S. degree in electrical engineering from the University of Belgrade, Belgrade, Yugoslavia, in 1994, and the M.S.E.E. and Ph.D. degrees in electrical engineering from Santa Clara University, Santa Clara, CA, USA, in 1996 and 2000, respectively. He was an Adjunct Lecturer and a Research Associate with the Department of Electrical Engineering, Santa Clara University, from 1998 to 2001; and a Research Associate with the Prof. Claire Tomlin's Hybrid Systems Laboratory, Department of Aeronautics and Astronautics, Stanford University, from 2001 to 2004. In 2004, he joined the University of Illinois at Urbana–Champaign, where he is currently a Professor with the Controls Group, Coordinated Science Laboratory, Department of Industrial and Enterprise Systems Engineering. His research interests include decentralized control and estimation, stability theory, optimal control, and differential games, with applications in control of autonomous vehicles, machine learning, precision agriculture, circuits, and telerehabilitation.

ALEKSANDRA LEKIĆ (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the School of Electrical Engineering, University of Belgrade, Belgrade, Serbia, in 2012, 2013, and 2017, respectively. She is with the Faculty of Electrical Engineering, Mathematics and Computer Science, TU Delft. She was an Assistant Professor (January 2020–April 2025) and an Associate Professor (since May 2025). She leads the Control of the HVdc/AC power systems team, which conducts advanced research in the field of power electronics and power system control. She was a recipient of the prestigious NWO Veni 2022 Grant in the Netherlands. She is an Associate Editor of *International Journal of Electrical Power and Energy Systems* (Elsevier).