Digital Chemical Engineering 16 (2025) 100262

Contents lists available at ScienceDirect

Digital Chemical Engineering

journal homepage: www.elsevier.com/locate/dche

Original Article

Neural network implementation of model predictive control with stability guarantees

Arthur Khodaverdian [a], Dhruv Gohil [a], Panagiotis D. Christofides [a,b,*]

[a] Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095, USA
[b] Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095-1592, USA

ARTICLE INFO

Keywords:
Neural network
Model predictive control
Nonlinear processes
Approximate optimal control

ABSTRACT

This work explores the use of supervised learning on data generated by a model predictive controller (MPC) to train a neural network (NN). The goal is to create an approximate control policy that can replace the MPC, offering reduced computational complexity while maintaining stability guarantees. Through the use of Lyapunov-based stability constraints, an MPC can be designed to guarantee stability. Once designed, this MPC can be used to generate a dataset of various state-space points and their resulting immediate optimal control actions. With the MPC dataset representing an optimal control policy, an NN is trained to function as a direct substitute for the MPC. The resulting approximate control policy can then be applied in real-time to the process, with stability guarantees being enforced through post-inference validation. If, for a given set of sensor readings, the NN yields control actions that violate the Lyapunov stability constraints used in the MPC, the control action is discarded and replaced with stabilizing control from a fallback stabilizing controller. This control architecture is applied to a benchmark chemical reactor model. Using this model, a comprehensive study of the stability, performance, robustness, and computational burden of the approach is carried out.

1. Introduction

Of the various methods for process control, Proportional–Integral–Derivative (PID) control has stood out as the most common framework. Estimates suggest that around 90% of industrial control systems use some form of PID in their control loops (Åström and Hägglund, 2001). PID is a feedback control framework that uses separate constant weights to determine the influence of past (Integral), present (Proportional), and future (Derivative) errors of a given sensor reading relative to a desired set-point. Its construction is that of a Single-Input-Single-Output (SISO) design, which limits a given PID controller to only control a single output variable. The SISO design can be used in parallel to give a non-interconnected Multiple-Input-Multiple-Output (MIMO) design, but the lack of interconnectivity is a limit on the design's ability to handle complex dynamics. Despite these shortcomings, PID, even today, is a control method that continues to evolve in terms of tuning methods and applications to further extract improvements from the basic structure (Vilanova and Visioli, 2012). In its most basic form, that is, a controller where only a Proportional term is used, only two floating point operations (FLOPS) are needed to calculate the control action for a given sensor reading and desired set-point, making proportional control an extremely computationally efficient operation. Similarly, the integral control term can be approximated with a summation and the derivative control term can be approximated with a finite difference approximation, meaning that both only require three FLOPS for a total of 8 FLOPS per PID. This computational efficiency, paired with only needing to tune a maximum of 3 parameters per controller, demonstrates the reasons why PID is so commonly used: simplicity and efficiency.

Simplicity is convenient, but it often comes with a loss of capability. PID, as a simple linear feedback controller, does not account for nonlinear dynamics, multivariable interactions, and constraints. For this reason, PID control's effectiveness is often limited to linear systems (even though it is commonly used to control nonlinear processes). For nonlinear systems, especially those with highly nonlinear dynamics, model-based optimal control methods yield more favorable results. Notably, Model Predictive Control (MPC) is a form of optimal control that determines control actions through numerical optimization of a cost function, which may use an approximated state trajectory based on a model of the process dynamics. MPC is a flexible framework with the ability to optimize a cost function for an arbitrary horizon length while also accounting for input constraints. Unlike PID controllers, which are unsuitable for multivariable problems with significant interactions, MPC's explicit use of the system's dynamics enables it to consider

* Corresponding author at: Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095, USA.
E-mail address: pdc@seas.ucla.edu (P.D. Christofides).

https://doi.org/10.1016/j.dche.2025.100262
Received 12 July 2025; Received in revised form 3 August 2025; Accepted 8 August 2025
Available online 18 August 2025

all of the interactions between the inputs and how these interactions evolve with time. This enables MPC to function as a general-purpose framework for process control of nonlinear systems. By tuning the cost function and the constraints, the behavior of the resulting process control can be customized as desired (Qin and Badgwell, 2003).

The use of a nonlinear model of the process dynamics is good for controller performance and robustness, but comes with a major consequence: MPC's computation time scales poorly with the dimensionality of the problem. In particular, the linear form of MPC is solvable with tools that have polynomial-time complexity (Peng et al., 2024). More specifically, certain MPC problems can be described via a convex quadratic programming problem. An extension of Karmarkar's algorithm has been shown to solve such problems with a time complexity of $\mathcal{O}\left(L^2 n^4\right)$ where L is the bit-length of the input and n is the number of variables. Some alternative methods improve on this for the convex case, such as those shown in Kapoor and Vaidya (1986), but for non-convex optimization problems that may arise in nonlinear constrained cases, the complexity becomes NP-hard (Pardalos and Vavasis, 1991).

In general, for nonlinear systems, MPC can achieve better control quality by incorporating nonlinear costs and/or nonlinear constraints. Such problems no longer necessarily fall into the linear or quadratic programming framework and are instead a form of nonlinear programming (NLP) problem. Solving an NLP problem requires nonlinear iterative methods which do not have a neat Big O notation to describe their time complexity; however, some approximations can be made for specific algorithms. One such method is the Sequential Least Squares Programming (SLSQP) method. This method, among other changes, replaces its quadratic programming subproblem with a linear least squares problem (Kraft, 1988). Given that these subproblems typically have time complexities of $\mathcal{O}\left(n^2\right)$ or $\mathcal{O}\left(n^3\right)$ (Gill et al., 1979), it can be reasonably assumed that MPC has an overall time complexity with polynomial scaling in the problem size. Examples of this are easily seen by considering the construction of the Jacobian matrix, which has a time complexity of $\mathcal{O}\left(N^2\right)$ where N is the size of the state array. Not only do key components of the optimization algorithm scale poorly with high-dimensional inputs and constraints, but there is the added issue of the optimization process being an iterative approach. Thus, as the input and output dimensions scale in size, and as more constraints get enforced, MPC may become infeasible for real-time applications beyond a certain scale (Xi et al., 2013).

Current research on how to handle this computational complexity issue is split. Some studies focus on how the system can be reduced or simplified without significant performance loss (Tomasetto et al., 2025; Alora et al., 2023; Zarrouki et al., 2023). Other studies focus on applying novel frameworks or optimizations to improve performance without reducing the system's scale (Meng et al., 2024; Yaren and Kizir, 2025; Adabag et al., 2024). A more recent development is the increase in research into the application of NNs to process control. The bulk of this research explores the use of NNs to model the system dynamics as opposed to using complex first-principles models. The goal of this research is to accelerate the computation time and/or improve the real-world accuracy of the model (Gordon et al., 2024; Patel et al., 2025; Alsmeier et al., 2024). Not only is this approach generally faster, but in some cases, its accuracy can outperform first-principles models (Macmurray and Himmelblau, 1995).

Given the success of NNs in modeling nonlinear dynamics, a natural extension is to investigate broader applications of this approach. Consequently, a novel effort has begun to focus on the use of machine learning techniques to approximate the entire MPC as opposed to modeling a portion of the MPC problem (Lucia and Karg, 2018). Among this research, there are some cases in which the topic of guaranteeing closed-loop stability or performance is explored. Some cases explore the topic of guaranteed constraint satisfaction by projection of the solved control actions into a set which ensures constraints are met (Bonzanini et al., 2020). Others explore novel training methods, such as the FORWARD-SWITCH method, which provides both constraint satisfaction and performance guarantees in linear systems (Ahn et al., 2022). Such approaches highlight that this method of control is potentially viable, but further research is required to resolve the lack of strict constraint handling, particularly for nonlinear cases, and to determine how the approach compares to MPC when faced with the same high-dimensional nonlinear problems that MPC may not be able to solve efficiently for real-time implementation (Gonzalez et al., 2024).

Motivated by the above considerations, this work explores these concerns and proposes a supervised learning approach that ensures stability through external enforcement of Lyapunov-based stability constraints as opposed to mapping or further optimization. This external enforcement either directly modifies the solved control action to satisfy simpler control action constraints, such as bounds, or replaces the control entirely with that of a fallback controller that is designed to satisfy the constraints in question. Whereas some approaches aim to ensure stability through modified training (Wang et al., 2022), this approach is method-agnostic, so long as sufficient stabilization constraints are enforced. If the stability constraint is violated, stability is guaranteed through fallback control to a reference stabilizing controller. This safeguards the closed-loop system against cases where erroneous outputs are generated. Simultaneously, this simplifies the design process and enables generalized compatibility. Using a benchmark nonlinear chemical process example, we demonstrate how a simple NN architecture and training process can reduce the time complexity of optimal control while retaining close mapping to the MPC. The train-time performance and run-time performance of the resulting NN are examined to provide insight into the computational benefits and performance impacts of the method.

2. Preliminaries

2.1. Notation

The transpose of vector x is denoted by $x^\top$. The set of real numbers is denoted by $\mathbb{R}$. The set subtraction of set B from set A yields a set of elements that are in set A but not in set B and is denoted as $A \backslash B$. Functions are denoted as $f(\cdot)$.

2.2. Class of systems

In this work, we focus on nonlinear multiple-input multiple-output (MIMO) continuous-time systems described by a system of nonlinear first-order ordinary differential equations (ODEs) of the form:

$$\dot{x} = F(x,u) = f(x) + \sum_{i=1}^{m} g_i(x)\, u_i \tag{1}$$

where the state vector $x = \left[x_1, x_2, \ldots, x_n\right] \in \mathbb{R}^n$ describes the process state variables which are assumed to be measurable at every sampling time t_k (i.e., state feedback control problem is considered). The control input vector $u = \left[u_1, u_2, \ldots, u_m\right] \in \mathbb{R}^m$ describes the applied control inputs, where $0 < m \leq n$. Each control input is bounded, i.e., $u_{i,\min} \leq u_i \leq u_{i,\max}\ \forall\ i = 1, 2, \ldots, m$ where $u_{i,\min}$ and $u_{i,\max}$ represent the lower and upper bounds of each control action, respectively. This bounded region, denoted $U \subset \mathbb{R}^m$, is a subset of the set of real numbers. The functions $f(\cdot)$ and $g_i(\cdot)\ \forall\ i = 1, 2, \ldots, m$ are assumed to be sufficiently smooth vector functions. Without loss of generality, we consider the origin as a steady-state of the open-loop system (i.e., Eq. (1) with $u_i = 0$, $\forall\ i = 1, 2, \ldots, m$) by assuming that $f(0) = 0$ (or $F(0,0) = 0$). We further designate the initial time as zero ($t_0 = 0$). Finally, the set $S(\Delta)$ is defined as the assortment of piecewise constant functions characterized by a period of Δ.

2.3. Stabilizability assumption

The existence of an explicit feedback controller $u(x) = \Phi(x) \in U$ that can ensure exponential stability of the origin of Eq. (1) is assumed. This assumption is referred to as the stabilizability assumption. Specifically, this stabilizability assumption states that there exists a stabilizing controller $u(x(t)) = \Phi(x) \in U$ in the sense that there exists a continuously differentiable control Lyapunov function $V(x)$ such that the following inequalities hold for all $x \in D$, where D is an open neighborhood around the origin:

$$c_1 |x|^2 \leq V(x) \leq c_2 |x|^2 \tag{2a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \leq -c_3 |x|^2 \tag{2b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq c_4 |x| \tag{2c}$$

where c_1, c_2, c_3, and c_4 are positive constants. Along with the fact that u is bounded, these assumptions ensure the existence of positive constants M_F, L_x, and L'_x that ensure that for all $x, x' \in D$ and $u \in U$, the following inequalities are satisfied:

$$\left| F(x', u) - F(x, u) \right| \leq L_x |x - x'| \tag{3a}$$

$$|F(x, u)| \leq M_F \tag{3b}$$

$$\left| \frac{\partial V(x)}{\partial x} F(x, u) - \frac{\partial V(x')}{\partial x} F(x', u) \right| \leq L'_x |x - x'| \tag{3c}$$

Remark 1. With respect to the stabilizability assumption, we note that it can be viewed as the analogue of requiring that the (A, B) pair is stabilizable in the context of linear systems and it is further expressed in terms of the existence of a control Lyapunov function which is the minimum requirement for stabilization of nonlinear systems.

2.4. Lyapunov-based model predictive control

The design of a stabilizing MPC with an explicitly defined region of guaranteed closed-loop stability is ensured through the use of a Lyapunov-based MPC (LMPC) formulated as follows (Mhaskar et al., 2006):

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_k + N\Delta} L(\tilde{x}(t), u(t)) \, dt \tag{4a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u(t)) \tag{4b}$$

$$u \in U, \ \forall \ t \in [t_k, t_k + N\Delta) \tag{4c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{4d}$$

$$\dot{V}(\tilde{x}(t_k), u(t_k)) \leq \dot{V}(\tilde{x}(t_k), \Phi(\tilde{x}(t_k))) \tag{4e}$$

where Δ is the sampling period and it represents the time between two consecutive sensor readings for the state x. In this context, it is also equivalent to the duration of the optimal control input u. N denotes the total number of sampling periods that the LMPC will simulate the state trajectory and optimize the control inputs for. In other words, N is the length of the prediction and control horizons. Eq. (4a) represents a generalized cost function that is to be optimized, which can be tuned as necessary. For simplicity, a quadratic cost function is used in this paper. Eq. (4b) represents the model that is used to simulate the process dynamics. In this work, the dynamic model is a first-principles-based model where Eq. (4d) initializes the state measurements of the model using sensor readings at t_k, which denotes an arbitrary initial condition. The final constraint is the Lyapunov time derivative constraint, also referred to as the stability constraint. This constraint utilizes the properties of the stabilizability assumption from Section 2.3 to ensure that the MPC's control action at each sampling time guarantees convergence of the closed-loop state to a small region around the origin. Thus, this constraint ensures faster, or at least as fast, convergence from the MPC's solution relative to the reference exponentially stabilizing controller, $\Phi(x)$. This stabilizing controller is often designed along with the Lyapunov function, increasing the complexity of the design process, which is why another form of Eq. (4e) can be used in practice, namely:

$$\dot{V}(\tilde{x}(t_k), u(t_k)) \leq -\alpha V(\tilde{x}(t_k)) \tag{5}$$

where α is a positive real number. This form of the constraint yields the same guarantees by forcing the $\dot{V}$ term to be negative proportionally to the magnitude of V as opposed to forcing it to be more negative than a reference controller, thereby allowing for stabilizability without the need to explicitly solve for a reference controller.

Remark 2. A major benefit of the LMPC design is its generalized support for cost functions. Although a quadratic cost function is used in this work, there are alternatives such as economic cost functions that incorporate economics through some function of the predicted state values $\tilde{x}$ and/or the current guess for the control input u. Such functions can also vary in time to account for economics, as has been demonstrated in our recent previous work (Khodaverdian et al., 2025).

Remark 3. The stability constraints under continuous-time implementation of the controller, $\Phi(x)$, guarantee exponential stability of the closed-loop system. Practically, this is not possible, as there is a non-negligible computation and signal transmission time that prevents continuous control. Thus, the control actions are applied in a sample-and-hold fashion. This causes the stability guarantees to only apply outside of a small region around the origin, as will be shown in Section 4.2 (i.e., it is possible to only establish convergence of the closed-loop state to a small region around the origin).

Remark 4. The Lyapunov stability constraints are only applied at the current sampling time, t_k, in the LMPC optimization problem and not throughout the entire prediction horizon, as in the MPC only the first control action is applied to the closed-loop system (receding horizon implementation). This is not a requirement, but it is a convenient simplification of the constraints. In the receding horizon implementation, the control input trajectory that MPC generates over the entire prediction horizon is discarded, with the exception of the first control input, which is applied to the process.

3. Neural network construction

3.1. Data generation

In order to get good fitting performance from a neural network trained via supervised learning, it is necessary to have a dense, diverse dataset of the desired operating region. In practice, it can be difficult to determine what a feasible operating region is and whether or not a given state is worth considering. Thus, the problem of generating sufficiently dense, diverse, and quality data is a system-specific problem that needs to be carefully considered. As a data-driven method, this step is critical.

A factor to consider is the fact that data generation can be done offline. The system dynamics, typically a first-principles model, along with a desired MPC to replace, provide a platform for the arbitrary generation of data. For any given state, MPC can be used to generate optimal control actions with the necessary constraints discussed earlier. This enables the minimal case for data generation with the state and control action pair. This is referred to as the minimal case, as MPC at the bare minimum requires an initial state and will output the resulting control action for the current sampling time. Additional data can be collected as needed, but it is critical to avoid mixing datasets built with different controller designs unless the NN is designed in such a way that enables the use of multiple models.

Remark 5. Some research on the topic of machine learning applied to model predictive control, often referred to as ML-MPC, may sound similar to the framework that is proposed in this paper; however, the two not only function differently, but can also be combined. The research on ML-MPC centers around an approach in which MPC is modified such that Eq. (4b) (i.e., the prediction model used in MPC) is no longer a first-principles model and is instead a data-driven NN model (Wu et al., 2019a,b). This modification is beneficial for highly complex systems where first-principles models fail to sufficiently capture the process dynamics. In these cases, it is beneficial to gather real-world data from the process and use it to train an NN model (research shows that recurrent neural network models are a viable option (Alhajeri et al., 2024)) that is capable of predicting the change in the process state for a given control action. In contrast, the present work uses offline data generated by an MPC to approximate the optimal control action for a given state and then uses this data to construct an NN that acts as the controller. These two frameworks can work at the same time, where one NN exists within the MPC as its model to better model the system dynamics, and the other NN is trained off of the data produced by this MPC to then function as a replacement used in the feedback loop. One NN applies the control action to predict the future state, and the other NN, as shown in this work, takes the current state and calculates the optimal control actions that should be applied to the process.

3.2. Controller neural network model type and training

The proposed controller implementation framework does not provide any specific guidance or have any specific restrictions on the neural network model type and training aspect of the controller design process. Unlike data generation, model training can be of any form that is desired. How the model is trained and the extent to which this is done impact the performance of the model, but the current framework is network performance agnostic and will guarantee closed-loop stability regardless of the type of neural network used to approximate the data and serve as the feedback controller. A model that performs poorly will simply result in the backup controller performing the bulk of the control actions. Thus, it is important to train the model to perform well if the quality of the control action and the resulting closed-loop performance are the priorities.

Remark 6. The robustness of this approach with respect to sensor noise and model uncertainty is dependent on the existence of the fallback controller. The NN-controller alone has no performance or stability guarantees as presented and will only attempt to mimic the LMPC with variable success. Thus, unless a fallback controller exists that is capable of enforcing the desired closed-loop stability and robustness properties with respect to measurement noise and model uncertainty, the implementation of the NN-controller on its own provides no closed-loop stability and robustness guarantees.

4. Guaranteeing stability for neural network-based control

This section will cover the design of the NN-based approximate implementation of MPC with stability guarantees.

4.1. Practical implementation

As illustrated in Fig. 1, the NN implementation is designed as a direct substitute for any given MPC design. The only required inputs to the system are $x(t_k)$ (the sensor signals at a given reference time frame t_k). Unlike MPC, which benefits from a good initial guess for the optimal control, the NN implementation does not require this guess as an input, although the framework enables the chosen NN to accept this form of input if desired.

The novelty in this design occurs after the output of the NN is acquired. A validation block is used to represent the steps used to ensure that the given solution is one that satisfies the chosen constraints. Notably, the guarantee of stability can be enforced here through checking if the Lyapunov stability constraints are satisfied for a given NN output. If true, then the output is left unchanged, but if false, then the constraint is forcefully satisfied through fallback to a stabilizing controller of any desired form. General constraint satisfaction is guaranteed through similar enforcement; either the constraints are enforced through direct modification of the NN output, such as clipping at control bounds, or a fallback controller that guarantees constraint satisfaction is used in cases where constraints are violated.

Remark 7. Because the framework does not involve a specialized training process, modified forms of MPC are supported. One such design that can be beneficial to consider is a two-layer implementation of MPC, which is a cyber-secure design that generates an optimal trajectory instead of optimal control (Khodaverdian et al., 2025). A linear controller can then derive suboptimal control by tracking this trajectory, but can do so while encrypted, as homomorphic encryption enables addition and/or multiplication operations on encrypted data. In the context of the NN implementation, instead of using MPC to solve the optimal control, and then solving the predicted trajectory based on this control, the NN can be trained to either directly output an estimate trajectory or to continue calculating approximate control signals that will then be applied to a state estimator to build the trajectory.

Remark 8. The design of this framework is intentionally left unspecified, as the precise goal of MPC varies between different processes. Instead, the importance of this framework comes in the form of the validation block. This simplifies the design process significantly. Now, there is no longer any required complexity involved in trying to design a training loop that ensures the constraints are met. As a consequence, poor fitting will yield poorer results with respect to the goals of the optimal control, but this will not come at the expense of losing the ability to stabilize the system. Additionally, this complexity can be implemented anyway if it is deemed beneficial for performance, as the enforcements are external.

4.2. Closed-loop stability results

As discussed in Section 2.3, the continuous-time form of the system under $\Phi(x)$ is rendered exponentially stable through the assumption's statement. In practice, the control action needs to be applied in a sample-and-hold fashion, which violates the continuous-time implementation, and thus we must establish that the stability of the closed-loop system is still guaranteed under sample-and-hold implementation of $\Phi(x)$ for sufficiently small sampling time.

4.2.1. Stability via a reference controller

The first form of the stability guarantee constraint is the form shown in Eq. (4e), which requires an exponentially stabilizing reference controller.

Theorem 1. *For a nonlinear system described by Eq. (1), given the existence of an explicit feedback controller $\Phi(x)$ that renders the nonlinear system exponentially stable with respect to the origin if applied continuously for any initial state satisfying $V\left(x\left(t_0\right)\right) \leq \rho$, there exists a positive constant ϵ_w such that if the following conditions are satisfied, the sample-and-hold implementation of $\Phi(x)$ ensures the convergence of the closed-loop state to a small region around the origin denoted $\Omega_{\rho_{\min}}$ and determined as follows:*

$$L_x' M_F \Delta - \frac{c_3}{c_2}\rho_s \leq -\epsilon_w \tag{6a}$$

$$\rho_{\min} = \max\left\{V\left(x\left(t_k+\Delta\right)\right) | V\left(x\left(t_k\right)\right) \leq \rho_s\right\} \tag{6b}$$

$$\rho_s < V\left(x\left(t_k\right)\right) \tag{6c}$$

$$\rho_s < \rho_{\min} < \rho \tag{6d}$$

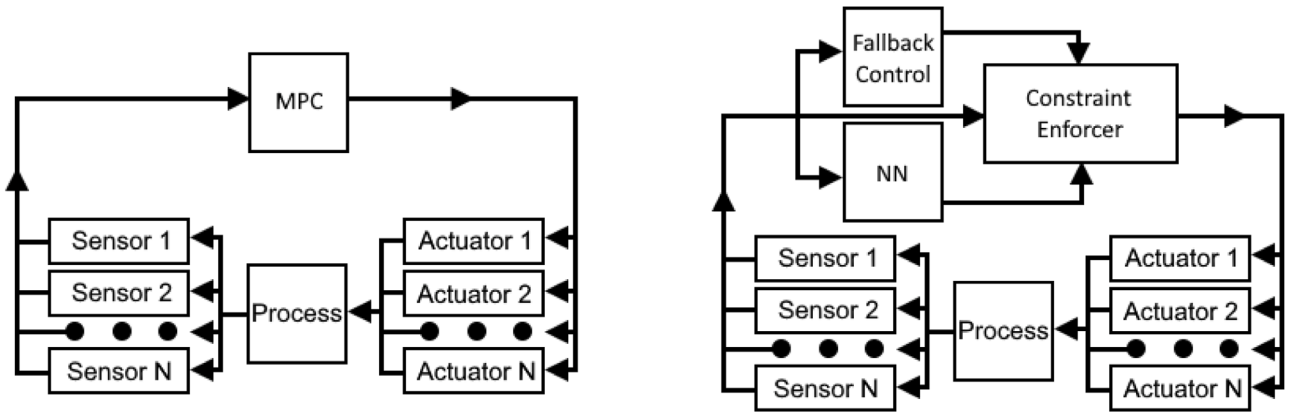

Fig. 1. Model Predictive Controller (Left) application block diagram for a general process vs Neural Network based controller with validation and fallback stabilizing controller (Right).

Proof. To start, we consider the reference controller itself. Due to the sample and hold implementation of the controller, we need to account for the fact that the state will continue to evolve while the control action remains fixed. Thus, we consider some time range representing a single sampling time interval denoted $t \in [t_k, t_k + \Delta)$. Here, t_k is the initial reference time frame, Δ is the sampling time, and thus the range of possible times between one sensor reading and the next includes all times besides $t_k + \Delta$, where the new sensor reading will be available. Thus, we can denote an arbitrary time frame of the time derivative of the Lyapunov function as

$$\dot{V}\left(x(t), \Phi\left(x\left(t_k\right)\right)\right) = \frac{\partial V(x(t))}{\partial x} F\left(x(t), \Phi\left(x\left(t_k\right)\right)\right) \tag{7}$$

After some algebraic manipulation, the expression becomes:

$$\begin{aligned}\dot{V}\left(x(t), \Phi\left(x\left(t_k\right)\right)\right) = &\frac{\partial V(x(t))}{\partial x} F\left(x(t), \Phi\left(x\left(t_k\right)\right)\right) \\ &- \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} F\left(x\left(t_k\right), \Phi\left(x\left(t_k\right)\right)\right) \\ &+ \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} F\left(x\left(t_k\right), \Phi\left(x\left(t_k\right)\right)\right)\end{aligned} \tag{8}$$

Via Section 2.3, the existence of the controller implies the existence of a continuously differentiable Lyapunov function. This further implies that the Lyapunov function and its derivatives are Lipschitz continuous, hence Eq. (3c). Additionally, the Lyapunov function must satisfy Eq. (2b). Thus, Eq. (8) simplifies to

$$\dot{V}\left(x(t), \Phi\left(x\left(t_k\right)\right)\right) \leq L_x' \left|x(t) - x\left(t_k\right)\right| - c_3 \left|x\left(t_k\right)\right|^2 \tag{9}$$

We can further simplify this using the integral triangle inequality as follows:

$$\begin{aligned}\left|x(t) - x\left(t_k\right)\right| &\leq \left|\int_{t_k}^{t} F\left(x(\tau), \Phi\left(x\left(t_k\right)\right)\right) \mathrm{d}\tau\right| \\ &\leq \int_{t_k}^{t} \left|F\left(x(\tau), \Phi\left(x\left(t_k\right)\right)\right)\right| \mathrm{d}\tau \\ &\leq \int_{t_k}^{t} M_F \mathrm{d}\tau \\ &\leq M_F\left(t - t_k\right) \\ &\leq M_F \Delta\end{aligned} \tag{10}$$

which can be substituted in Eq. (9) to yield:

$$\dot{V}\left(x(t), \Phi\left(x\left(t_k\right)\right)\right) \leq L_x' M_F \Delta - c_3 \left|x\left(t_k\right)\right|^2 \tag{11}$$

Note the existence of both a positive term and a negative term in the upper bound for the time derivative of the Lyapunov function. This implies that for a given non-zero sampling time, there exists a lower bound for $|x|$ denoted $\rho_s > 0$. We denote Ω_{ρ_s} as the level set of V where for any $V(x(t)) \leq \rho_s$, $\dot{V} \leq 0$ cannot be guaranteed. Thus, the sample and hold assumption results in incomplete convergence to the origin. As a result, the condition described by Eq. (6c) must be satisfied. This condition can be combined with Eq. (3b) to yield

$$\rho_s < V\left(x\left(t_k\right)\right) \leq c_2 \left|x\left(t_k\right)\right|^2 \tag{12}$$

$$\frac{\rho_s}{c_2} \leq \left|x\left(t_k\right)\right|^2 \tag{13}$$

$$-c_3 \left|x\left(t_k\right)\right|^2 \leq \frac{-c_3}{c_2} \rho_s \tag{14}$$

which results in

$$\dot{V}\left(x(t), \Phi\left(x\left(t_k\right)\right)\right) \leq L_x' M_F \Delta - \frac{c_3}{c_2} \rho_s \tag{15}$$

It can now be seen why Ω_{ρ_s} is necessary. Because of the sample and hold implementation, we get an upper bound on the rate of change to the Lyapunov function that can be positive for small values of $|x|$. Although the positive factor in the upper bound can be decreased, the only such method is to reduce Δ. It can be seen that as Δ approaches 0, we approach a continuous-time system, which will not have this issue. Thus, for any $\Delta > 0$, we have a small region around the origin (Ω_{ρ_s}) where we cannot guarantee the decrease of $V\left(x\left(t_k\right)\right)$. As a result, given a reference state within this region, Eq. (6b) describes a larger region where all such states will be contained within. Given sufficiently long run time, the final result will ultimately converge to $\Omega_{\rho_{min}}$ and remain in this region. States within $\Omega_{\rho_{\min}} \backslash \Omega_{\rho_s}$ will converge into Ω_{ρ_s}, but within Ω_{ρ_s} the sample and hold implementation prevents any guarantees of further convergence to the origin. In order to sufficiently stabilize the system, a sufficiently small Δ must be chosen, as a smaller Δ results in a smaller ρ_s before the upper bound of $\dot{V}$ becomes positive.

More broadly, we say that for sufficiently small Δ and for all $x\left(t_k\right) \in \Omega_\rho \backslash \Omega_{\rho_s}$ we can ensure

$$\dot{V}\left(x(t), \Phi\left(x\left(t_k\right)\right)\right) \leq -\epsilon_w \tag{16}$$

where ϵ_w is some positive constant, for $t \in [t_k, t_k + \Delta)$. As such, we can ensure that the control Lyapunov function will decay with time, ultimately causing the closed-loop state to converge to a small region around the origin. □

Theorem 2 below establishes closed-loop stability under the MPC of Eq. (4).

Theorem 2. *For the MPC described by Eq. (4) where Eq. (4b) describes a nonlinear system of the form described by Eq. (1), given the existence of an explicit feedback controller $\Phi(x)$ that renders the nonlinear system exponentially stable with respect to the origin if applied continuously for any initial condition satisfying $V\left(x\left(t_0\right)\right) \leq \rho$, there exists a positive constant*

ϵ_w such that if the following conditions are satisfied, the sample-and-hold implementation of $u(t)$ of the MPC ensures that the closed-loop state converges to a small region around the origin denoted $\Omega_{\rho_{\min}}$ characterized as follows:

$$L_x' M_F \Delta - \frac{c_3}{c_2}\rho_s \leq -\epsilon_w \tag{17a}$$

$$\rho_{\min} = \max\left\{V\left(x\left(t_k+\Delta\right)\middle| V\left(x\left(t_k\right)\right)\leq \rho_s\right)\right\} \tag{17b}$$

$$\rho_s < V\left(x\left(t_k\right)\right) \tag{17c}$$

$$\rho_s < \rho_{\min} < \rho \tag{17d}$$

Proof. We provide a sketch of the proof building on the proof of Theorem 1. Eq. (4e) constrains the control action solutions such that the time derivative of the Lyapunov function is more negative than that of the system if the exponentially stabilizing controller $\Phi(x(t))$ was used. Theorem 1 demonstrates how this reference controller ensures that the closed-loop system state converges to a small region around the origin when applied in a sample-and-hold manner. Thus, under the same constraints, the MPC solution will yield

$$\dot{V}(\tilde{x}(t_k),u) \leq \dot{V}(\tilde{x}(t_k),\Phi\left(\tilde{x}\left(t_k\right)\right)) \leq -\epsilon_w \tag{18}$$

Repeated application of this result yields convergence of the closed-loop system state to a small region around the origin. □

4.2.2. Stability via alternative Lyapunov stability constraint

Another convenient form is called the Alpha Form of the stabilizing constraint demonstrated in Eq. (5). Here, instead of relying on the existence of a stabilizing controller, we utilize the positive definite nature of the Lyapunov function to ensure a consistently negative time derivative of the function by using a constant $\alpha > 0$. Theorem 3 below establishes closed-loop stability under the MPC of Eq. (4) with the stability constraint of Eq. (4e).

Theorem 3. *Consider an MPC described by Eq. (4) where Eq. (4b) describes a nonlinear system of the form described by Eq. (1) and the constraint from Eq. (4e) is replaced with Eq. (5). Suppose there exists an explicit feedback controller $\Phi(x)$ that renders the nonlinear system exponentially stable with respect to the origin if applied continuously for any initial condition satisfying $V\left(x\left(t_0\right)\right)\leq\rho$. Then, there exists a positive constant ϵ_w and a positive constant α such that, if the following conditions are satisfied, then the sample-and-hold implementation of $u(t)$ ensures convergence of the closed-loop state to a small region around the origin denoted $\Omega_{\rho_{\min}}$ defined as follows:*

$$L_x' M_F \Delta - \alpha\rho_s \leq -\epsilon_w \tag{19a}$$

$$\rho_{\min} = \max\left\{V\left(x\left(t_k+\Delta\right)\middle| V\left(x\left(t_k\right)\right)\leq \rho_s\right)\right\} \tag{19b}$$

$$\rho_s < V\left(x\left(t_k\right)\right) \tag{19c}$$

$$\rho_s < \rho_{\min} < \rho \tag{19d}$$

Proof. As is done in Theorem 1, we can denote an arbitrary time frame of the time derivative of the Lyapunov function under MPC

$$\dot{V}\left(x(t),u\left(t_k\right)\right) = \frac{\partial V(x(t))}{\partial x} F\left(x(t),u\left(t_k\right)\right) \tag{20}$$

Through some algebraic manipulation of the expression.

$$\begin{aligned}\dot{V}\left(x(t),u\left(t_k\right)\right) = {} & \frac{\partial V(x(t))}{\partial x} F\left(x(t),u\left(t_k\right)\right) \\ & - \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} F\left(x\left(t_k\right),u\left(t_k\right)\right) \\ & + \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} F\left(x\left(t_k\right),u\left(t_k\right)\right)\end{aligned} \tag{21}$$

for $t\in\left[t_k,t_k+\Delta\right)$. Using Section 2.3, the existence of the controller implies the existence of a continuously differentiable Lyapunov function. This further implies that the Lyapunov function and its derivatives are Lipschitz continuous, hence Eq. (3c). Additionally, the Lyapunov function must satisfy Eq. (5). Thus, Eq. (21) simplifies to

$$\dot{V}\left(x(t),u\left(t_k\right)\right) \leq L_x'\left|x(t)-x\left(t_k\right)\right| - \alpha V\left(x\left(t_k\right)\right) \tag{22}$$

for $t\in\left[t_k,t_k+\Delta\right)$. We can further simplify this using the integral triangle inequality as in Eq. (10) to obtain:

$$\dot{V}\left(x(t),u\left(t_k\right)\right) \leq L_x' M_F \Delta - \alpha V\left(x\left(t_k\right)\right) \tag{23}$$

Following the same logic as Theorem 1, the sample and hold assumption results in incomplete convergence to the origin. As a result, the condition described by Eq. (6c) must be satisfied. This condition yields

$$\rho_s < V\left(x\left(t_k\right)\right) \tag{24}$$

$$-\alpha V\left(x\left(t_k\right)\right) < -\alpha\rho_s \tag{25}$$

which results in

$$\dot{V}\left(x(t),u\left(t_k\right)\right) \leq L_x' M_F \Delta - \alpha\rho_s \tag{26}$$

Following the logic of Theorem 1, we say that for sufficiently small Δ and for all $x\left(t_k\right)\in\Omega_\rho\backslash\Omega_{\rho_s}$ we can ensure

$$\dot{V}\left(x(t),u\left(t_k\right)\right) \leq -\epsilon_w \tag{27}$$

where ϵ_w is some positive constant, $t\in\left[t_k,t_k+\Delta\right)$. As such, we can ensure that the control Lyapunov function will decay with time, ultimately causing the system to converge to a small region around the origin. □

The choice between these constraints is up to the control system designer, but both ensure convergence of the closed-loop state to a small region around the origin proportional to the sampling time. It should be noted, however, that a proof of this form does not exist for the Neural Network controller, as strict constraint guarantees are not involved.

Remark 9. If the Lyapunov constraint of the form shown in Eq. (5) is used, the extent to which the control is stabilizing is determined by the magnitude of α. Large values can be beneficial for rapid stabilization or for stabilizing systems when large sampling times are needed, but this comes at the risk of over-constraining the system. Of the possible control actions that can be used, a subset of these will be valid for a given V and α. As α increases, this subset becomes smaller as more control action will be needed to provide the stronger stabilizing control. Thus, there will be fewer valid control action solutions. Ideally, the consequence of this will be a reduction in the optimality of the MPC. Since control actions are bounded, the reality is that for a given V and α, the requested rate of stabilization can be too large for any control action to satisfy it. Therefore, it is important to choose an α that is sufficiently large to guarantee stability while being small enough not to over-constrain the optimization problem.

4.2.3. Closed-loop stability under NN controller implementation

Theorem 4 below establishes closed-loop stability of the process under control in the form shown in the block diagram on the right of Fig. 1.

Theorem 4. *For a nonlinear system described by Eq. (1), assuming the existence of an explicit feedback controller $\Phi(x)$ that renders the nonlinear system exponentially stable with respect to the origin if applied continuously for any initial condition satisfying $V\left(x\left(t_0\right)\right)\leq\rho$, and a neural network controller $\Phi_{nn}\left(x\left(t_k\right)\right)$ trained to fit the MPC solution described by Eq. (4)*

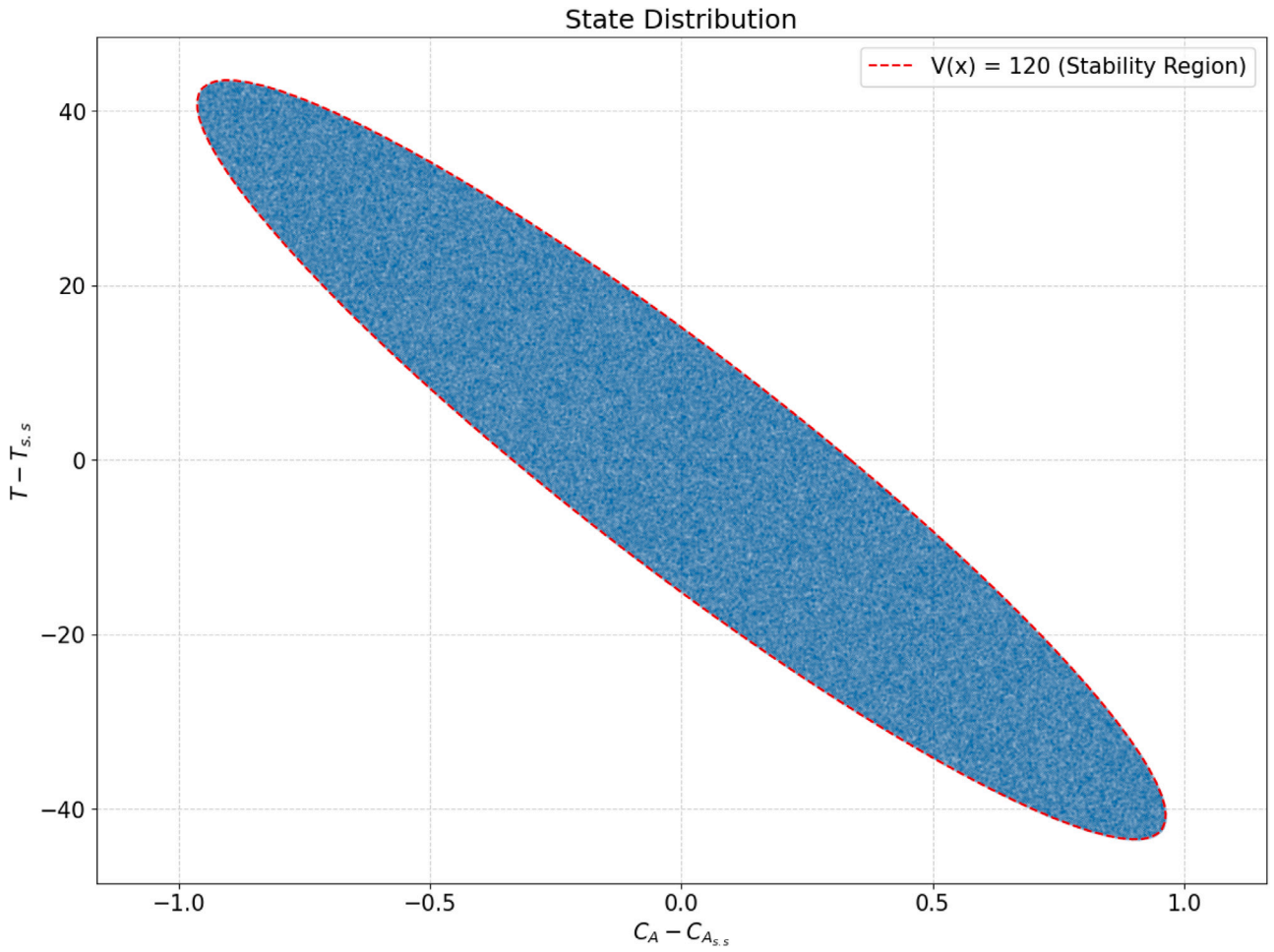

Fig. 2. State distribution of the generated data used for NN controller training.

where Eq. (4b) is of the form shown in Eq. (1). Then, a constraint enforcer implementation of the form:

$$u = \begin{cases} \boldsymbol{\Phi}_{nn}\left(x\left(t_k\right)\right), & \text{if } \dot{V}\left(x\left(t_k\right), \boldsymbol{\Phi}_{nn}\left(x\left(t_k\right)\right)\right) \leq \dot{V}\left(x\left(t_k\right), \boldsymbol{\Phi}\left(x\left(t_k\right)\right)\right) \\ \boldsymbol{\Phi}\left(x\left(t_k\right)\right), & \text{if } \dot{V}\left(x\left(t_k\right), \boldsymbol{\Phi}_{nn}\left(x\left(t_k\right)\right)\right) > \dot{V}\left(x\left(t_k\right), \boldsymbol{\Phi}\left(x\left(t_k\right)\right)\right) \end{cases} \tag{28}$$

applied to a process as shown in Fig. 1 implies the existence of a positive constant ϵ_w such that if the following conditions are satisfied, the sample-and-hold implementation of u ensures the convergence of the closed-loop state to a small region around the origin denoted $\Omega_{\rho_{\min}}$ and determined using Eqs. (6a)–(6d).

Proof.

Case 1. Consider the case where $\dot{V}\left(x\left(t_k\right), \boldsymbol{\Phi}_{nn}\left(x\left(t_k\right)\right)\right) > \dot{V}\left(x\left(t_k\right), \boldsymbol{\Phi}\left(x\left(t_k\right)\right)\right)$. In this case, the neural network controller does not produce a control action that would ensure convergence of the closed-loop system state to a small region around the origin; however, the constraint enforcer compensates for this by flipping the control action to instead utilize the control provided by the fallback controller. It is assumed that the fallback controller is an explicit feedback controller that renders the nonlinear system exponentially stable with respect to the origin if applied continuously for any initial state satisfying $V\left(x\left(t_0\right)\right) \leq \rho$. Thus, the fallback controller satisfies Theorem 1 which implies Eq. (16). Thus, the fallback controller ensures that the closed-loop system state converges to a small region around the origin when applied in a sample-and-hold manner.

Case 2. Consider the case where $\dot{V}\left(x\left(t_k\right), \boldsymbol{\Phi}_{nn}\left(x\left(t_k\right)\right)\right) \leq \dot{V}\left(x\left(t_k\right), \boldsymbol{\Phi}\left(x\left(t_k\right)\right)\right)$. In this case, the proof follows the same logic as in Theorem 2. Here, the fallback controller is a controller that satisfies Theorem 1, which implies Eq. (16). Thus, the fallback controller ensures that the closed-loop system state converges to a small region around the origin when applied in a sample-and-hold manner. Consequently, the NN controller satisfies

$$\dot{V}\left(x\left(t_k\right), \boldsymbol{\Phi}_{nn}\left(x\left(t_k\right)\right)\right) \leq \dot{V}\left(x\left(t_k\right), \boldsymbol{\Phi}\left(x\left(t_k\right)\right)\right) \leq -\epsilon_w \tag{29}$$

which ensures that application of the NN controller yields convergence of the closed-loop system state to a small region around the origin. □

Remark 10. Although the NN can be applied directly to the system without any fallback control if desired, doing so will no longer preserve the stability guarantees described above. Miscellaneous techniques may exist that improve the NN's ability to satisfy these constraints even in the case of poor mapping to a reference MPC, but unless these techniques strictly enforce the constraints that are necessary, then the system is not guaranteed to be stabilized by the NN's control.

Remark 11. We note that the Lyapunov time-derivative is not computed for a discrete-time system, as the Lyapunov function is formulated with respect to a system described by Eq. (1), i.e, a continuous-time system. Thus, the resulting derivative is of the form $\dot{V} = 2xP\dot{x}^\top$ and can be defined explicitly for a given set of state measurements and control signals. The construction of such a Lyapunov function is a general design problem for process control and is thus not detailed in the present paper; in practice, quadratic Lyapunov functions have

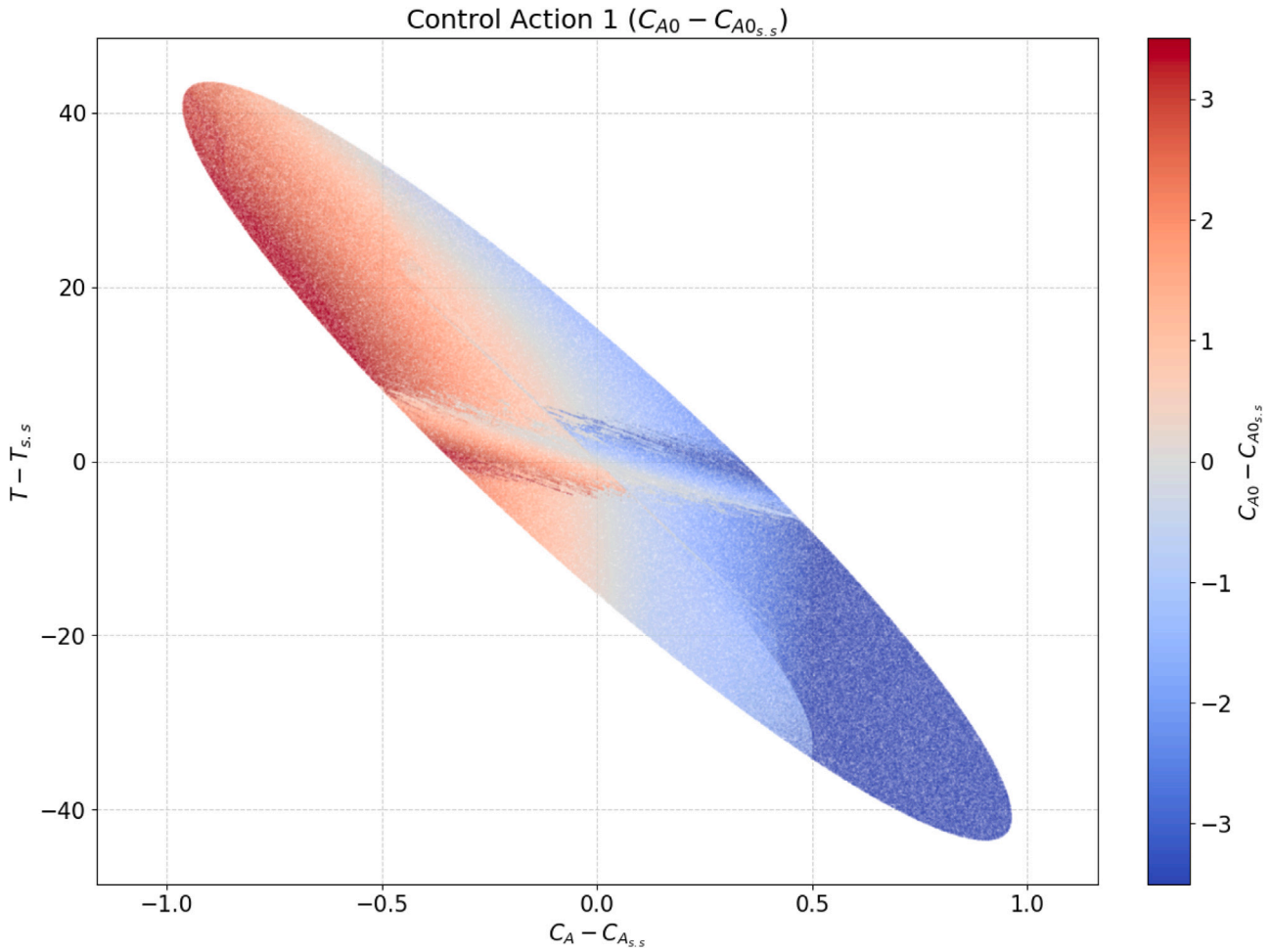

Fig. 3. State distribution of the generated data color coded by the value of the first control action $C_{A0} - C_{A0_{s.s}}$.

been found to work well. The proofs from Section 4 demonstrate how the sample-and-hold implementation impacts the upper bound of this derivative for the closed-loop system under the stabilizing reference controller and provides a robustness margin with respect to bounded disturbances.

5. Application to a chemical process example

To demonstrate the performance of the proposed NN control framework, we apply the framework to a benchmark nonlinear chemical process example. The objective of the MPC is to stabilize the system to a predefined steady state by minimizing a quadratic cost function of the deviation variable form of the state variables with respect to this steady state. The objective of the resulting NN is to provide solutions that roughly matches what the MPC would provide for any given point in the state space. This would demonstrate the ability for a low-dimensional system to be accurately modeled using neural networks, thus enabling performance gains with minimal control quality loss.

5.1. Process model and control problem

The chosen benchmark nonlinear chemical process example is a perfectly mixed continuous stirred tank reactor (CSTR). The CSTR converts a generalized reactant A to product B via a second-order irreversible exothermic elementary reaction $A \rightarrow B$ based on the following second-order non-isothermal elementary rate law $r_A = k_0 \exp\left(-\frac{E}{RT}\right) C_A^2$. In practice, the temperature of the reactor would be controlled through a jacket, but for modeling simplicity, heat transfer is modeled as a direct heating or cooling input term ($\dot{Q}$). Pure species A of variable concentration is fed into the reactor at a fixed rate; thus, the dynamics of this model consist of the following pair of first-order ODEs:

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = \frac{F}{V}(C_{A0} - C_A) - k_0 \exp\left(-\frac{E}{RT}\right) C_A^2 \tag{30a}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 \exp\left(-\frac{E}{RT}\right) C_A^2 + \frac{\dot{Q}}{\rho_L C_p V} \tag{30b}$$

In this system, F, V, T_0, k_0, ΔH, ρ_L, C_P, and E are defined as constants as detailed in Table 1 along with the reference unstable steady state values. The reactant concentration (C_A) and the reactor temperature (T) are chosen to be the state variables, whereas the heat input rate ($\dot{Q}$) and inlet concentration of species A (C_{A0}) are chosen to be the manipulated inputs. The state variables and manipulated inputs are converted into deviation variable form relative to the chosen unstable steady state, meaning we denote the state variables as $x = \left[C_A - C_{As}, T - T_s\right]$ and the input variables as $u = \left[C_{A0} - C_{A0s}, \dot{Q} - \dot{Q}_s\right]$. The input variables are bounded as follows:

$$-3.5 \leq u_1 \leq 3.5 \qquad \left[\mathrm{kmol\,m^{-3}}\right] \tag{31a}$$

$$-5 \times 10^5 \leq u_2 \leq 5 \times 10^5 \qquad \left[\mathrm{kJ\,h^{-1}}\right] \tag{31b}$$

All simulations begin at a random point within the state space bounded by the level set $V = 120$ at $t_k = 0$ and optimize a quadratic cost function as shown in Eq. (32)

$$L(\tilde{x}(t), u(t)) = \tilde{x}(t) Q_x \tilde{x}(t)^\top + u(t) Q_u u(t)^\top \tag{32}$$

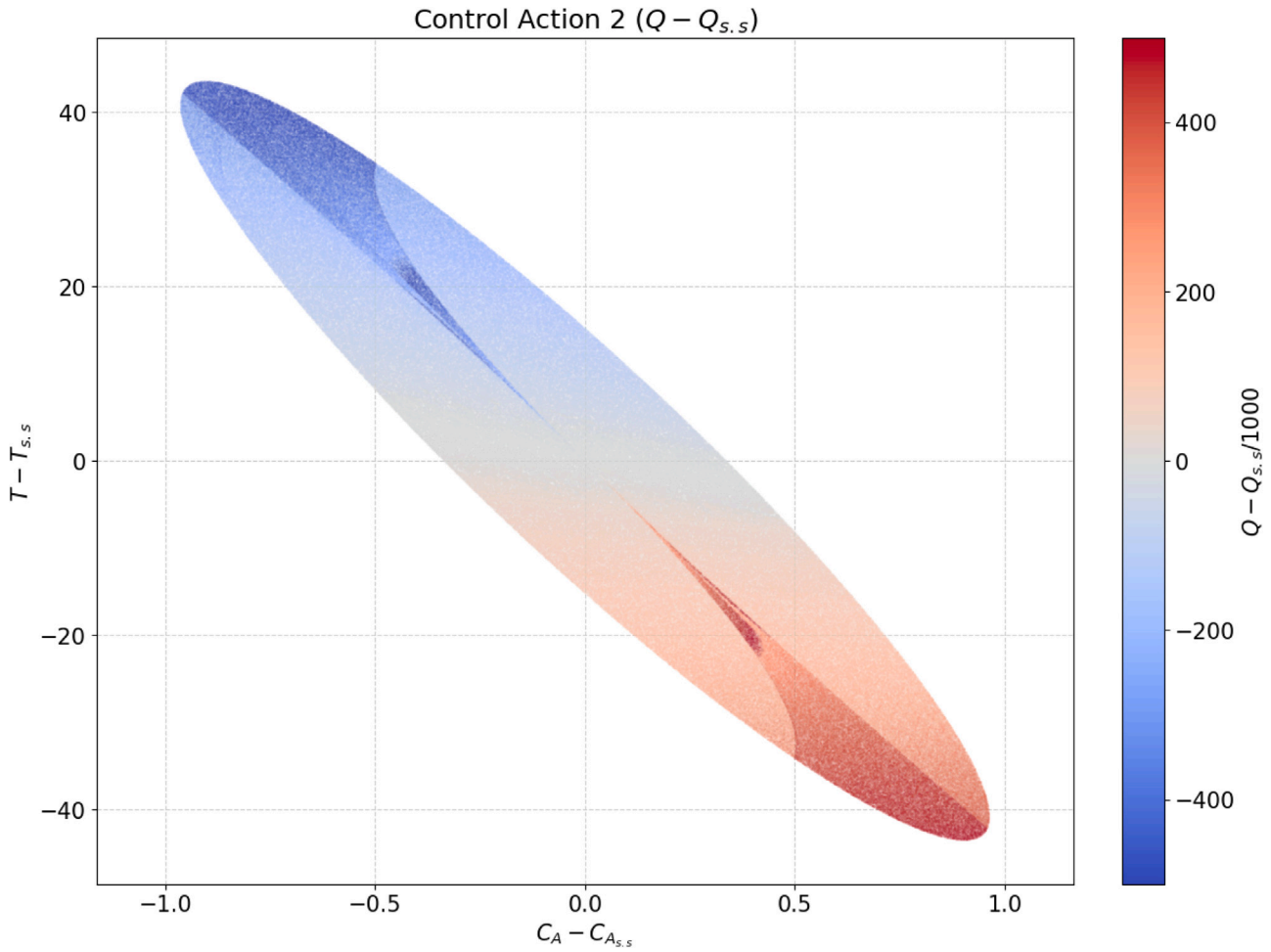

Fig. 4. State distribution of the generated data color coded by the value of the second control action $Q - Q_{s.s}$.

where the weight matrices are defined as

$$Q_x = \begin{bmatrix} 1000 & 0 \\ 0 & 1 \end{bmatrix} \tag{33}$$

$$Q_u = \begin{bmatrix} 10 & 0 \\ 0 & 10^{-8} \end{bmatrix} \tag{34}$$

5.2. Stability analysis

The reference stabilizing controller used to determine the stability region is a set of P controllers for C_{A0} and $\dot{Q}$ with gains of $K_{C_{A0}} = 2$ and $K_Q = 5,000$ respectively. The control Lyapunov function for this system is of the form $V(x) = x^\top Px$ where P is a positive definite matrix defined as:

$$P = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{35}$$

Stability can be guaranteed within a level set of this Lyapunov function, denoted Ω_ρ, with a value of $\rho = 120$. This level set was determined by simulating the system with the reference stabilizing controller. Starting at a large V, 100 initial conditions were sampled along the level set boundary. For each initial condition, the closed-loop system under the reference controller was simulated for 10 sampling times, and $\dot{V}$ was checked at each. Once $\dot{V}$ returned negative values for all 10 time steps for all 100 points, the boundary of the stability region of the stabilizing controller was deemed found. To validate, level sets smaller than this, in increments of 1, are also checked to ensure that all subsequent level sets satisfy the stabilizing controller assumption. Using this procedure, we selected $\rho = 120$ as a conservative estimate of the stability region. Because this example is entirely first-principles based, there is no modeling error, and due to the small step size used, the small region around the origin in which we cannot guarantee negative $\dot{V}$ is found to be of the order of $V < 10^{-23}$, and is thus negligibly small. To be conservative, we terminate the simulation once the current trajectory enters a level set for which $V \leq 1$.

As a final measure of caution regarding stability, we slightly modify the stabilizing constraint to be of the form

$$\dot{V}\left(x\left(t_k\right), u\left(t_k\right)\right) + 0.065 \leq \min\left(0, \dot{V}\left(x\left(t_k\right), \Phi\left(x\left(t_k\right)\right)\right)\right) \tag{36}$$

Table 1
Parameter values for the chemical process example.

Name	*Label*	*Value*	*Units*
Flow Rate	F	5	$m^3\,h^{-1}$
Reactor Volume	V	1	m^3
Pre-exponential Factor	k_0	8.46×10^6	$m^3\,kmol^{-1}\,h^{-1}$
Activation Energy	E	5×10^4	$kJ\,kmol^{-1}$
Gas Constant	R	8.314	$kJ\,kmol^{-1}\,K^{-1}$
Liquid Density	ρ_L	1000	$kg\,m^{-3}$
Enthalpy of Reaction	ΔH	-1.15×10^4	$kJ\,kmol^{-1}$
Inlet Temperature	T_0	300	K
s.s Heat Input Rate	$\dot{Q}_s$	0	$kJ\,h^{-1}$
s.s Inlet Concentration	C_{A0_s}	4	$kmol\,m^{-3}$
s.s Concentration	C_{As}	1.954	$kmol\,m^{-3}$
s.s Temperature	T_s	401.9	K
Specific Heat	C_p	0.231	$kJ\,kg^{-1}\,K$

s.s stands for steady-state.

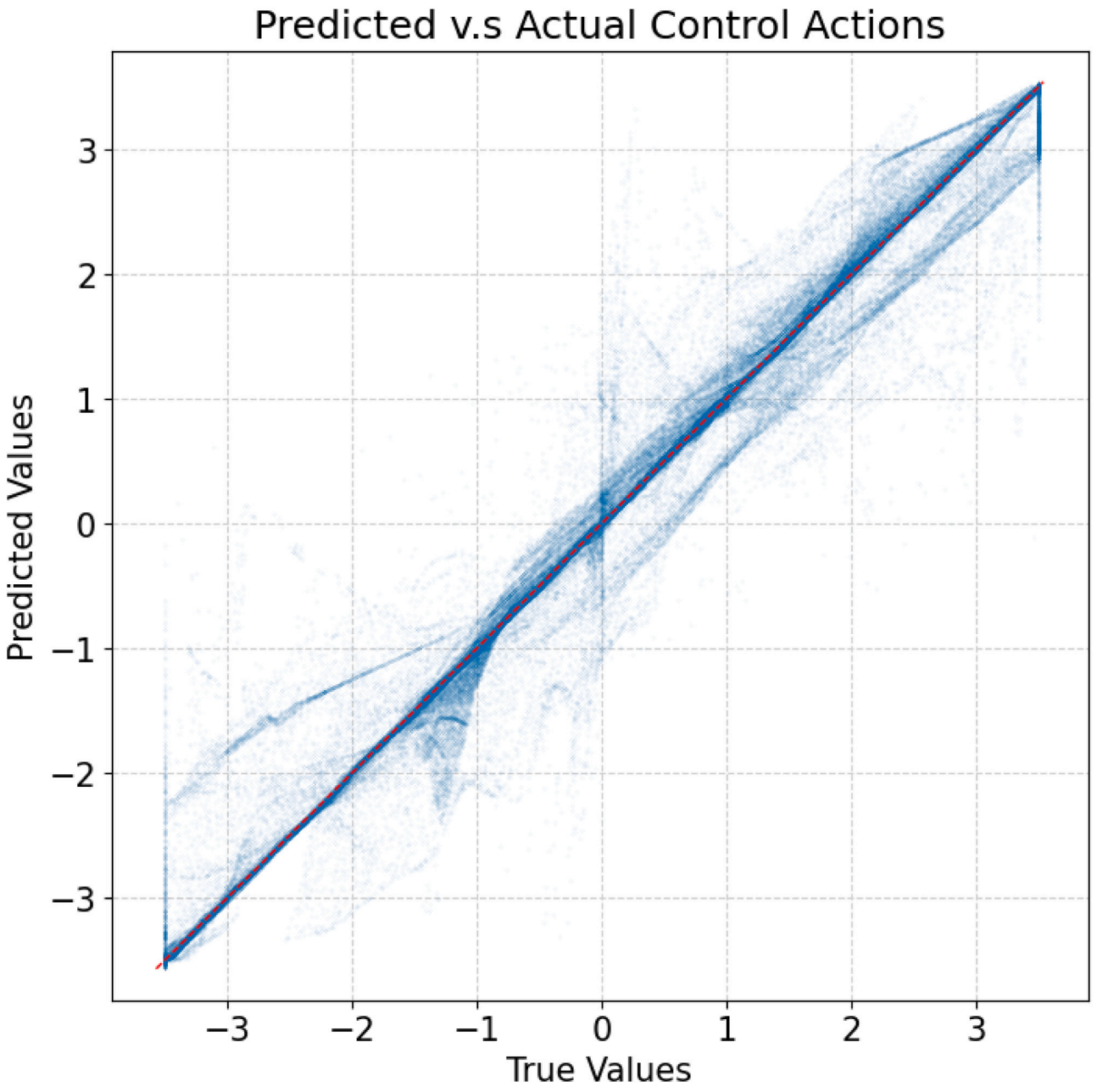

Fig. 5. Predicted vs. actual (true) values for the first control action $C_{A0} - C_{A0_{s,s}}$.

This prevents the possibility of the reference control generating non-negative time derivatives for the Lyapunov function while also forcing the MPC control to be more stabilizing to a specified magnitude at minimum. The bias term exists because the stabilizing control is not robustly verified, and so there is a potential risk in violating the assumptions for the controller, which are necessary for stability of the system. In other words, we artificially enforce a lower bound for ϵ_w to be $0.065 \leq \epsilon_w$ as a precautionary measure.

Remark 12. Using conservative estimates of both regions is important, as it is infeasible to thoroughly test all level sets. During this portion of the design process, the level sets are checked in increments of 1, which leaves room for intermediate level sets to violate the constraint. To avoid this, we used a smaller subset of the stability region that was found in order to minimize the possibility of this influencing the system. Similarly, because our testing was sufficient for whole numbers, $V < 1$ is conservatively chosen to over-estimate the small region around the origin in order to avoid the same possibilities.

Remark 13. ρ_s was found to be of the order of 10^{-23} by using a modified form of the methodology described above. Starting at the level set where $\rho = 1$, in increments of 0.1, until $\rho = 0.1$ was reached, we followed the same methodology as above. This process was repeated where the terms were all decreased by an order of magnitude until eventually a point where $\dot{V} > 0$ was found.

5.3. Control system parameters

The reference controllers used a sampling time of $\Delta_{PI,\text{reference}} = 7.2\,\text{s}$. The MPC optimizes the system over a horizon of length $N = 60$ with the same sampling period of $\Delta = 7.2\,\text{s}$, which corresponds to optimizing for the next 432 s. Of the resulting 60 control action solutions, we only apply the first before resolving the MPC (i.e., receding horizon

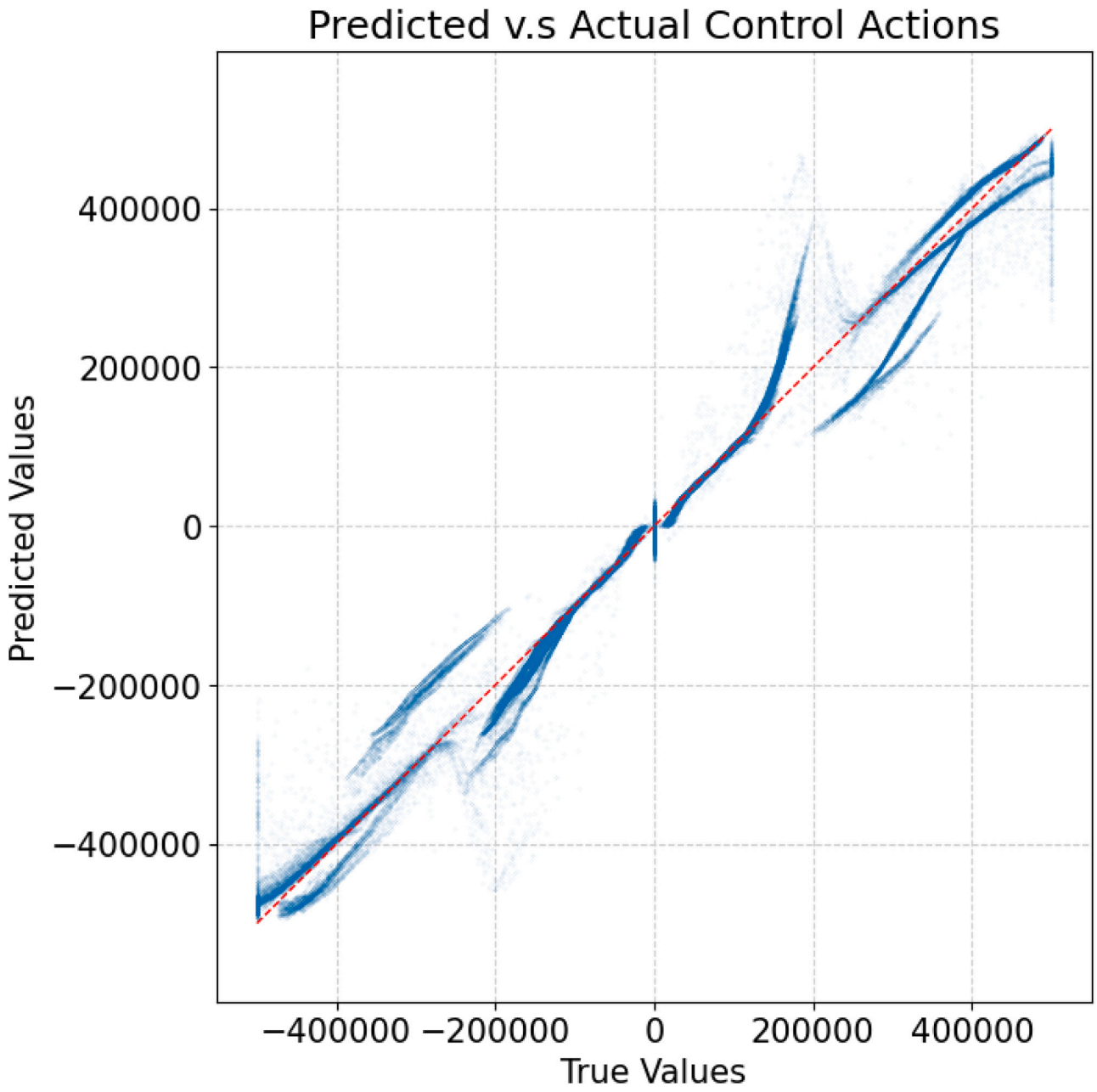

Fig. 6. Predicted vs. actual (true) values for the second control action $Q - Q_{s,s}$.

implementation). In an attempt to reduce the computational burden, integration of the model used in LMPC is done via the forward Euler method with a fixed step size of 0.36 s for which numerical stability is ensured.

5.4. Data generation

In our case, the operating region is explicitly known to be all states bounded by the level set $V = 120$. Thus, we generate data by sampling random states within this region. These random states are paired with their corresponding optimal control input for the same time instant by solving MPC and storing the first control action. The MPC was solved in Python using the SciPy package's minimize function. Specifically, the function was designed to use the SLSQP method with a max iteration count of 1000, and an 'ftol' and 'eps' value of 1e−4 (Virtanen et al., 2020). In order to prevent data contamination, if the MPC fails to solve for any reason, the sample is discarded. This process is parallelized for performance and is terminated after roughly 1 million points are generated. The resulting state distribution plot is shown in Fig. 2, which demonstrates the density and diversity of the data within our operating region.

As a low-dimensional model, it is also convenient to visualize the control distribution in the state-space plot by color coding the state distribution plot based on the control action value. These plots are shown in Figs. 3 and 4. Although most of the state space can be visually seen as numerically smooth, there are regions with rapid changes in the optimal action. Such regions can pose issues when training, as smooth data is preferred.

Remark 14. Because ρ_s is of the order of 10^{-23}, it is effectively negligible. Thus, data sampling within the state-space was allowed to include points within $\rho \leq 1$. Due to the stability constraints in the MPC, any points that would violate Eq. (16), such as those within ρ_s, are discarded anyways.

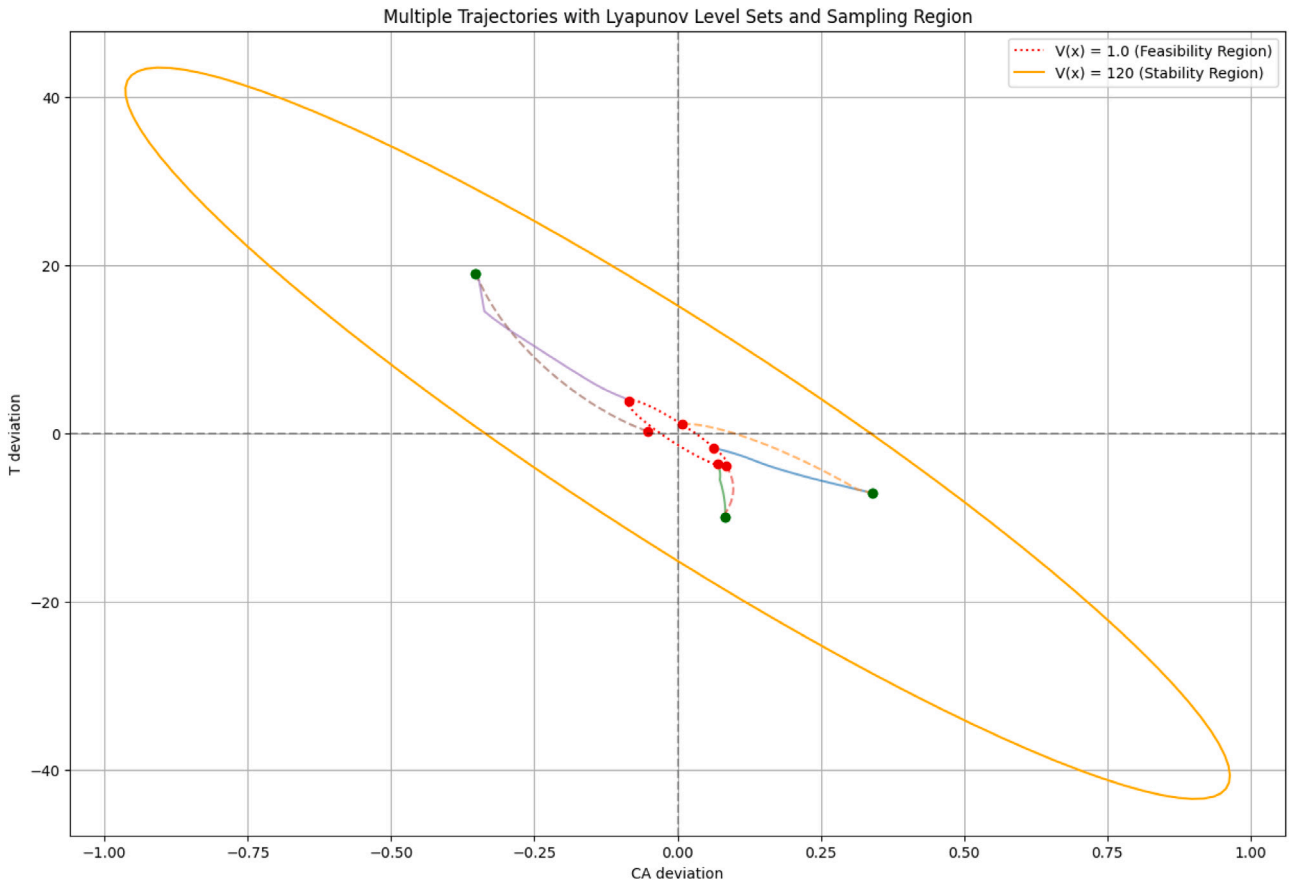

Fig. 7. Closed-loop state trajectory examples for the process controlled under MPC (dashed line) and NN with Proportional control as fallback control (solid line).

5.5. Neural network training

The neural network used is a Feedforward Neural Network, where the inputs and outputs are scaled using a StandardScaler. The StandardScaler is fit on the training set, which is composed of 80% of the generated samples, chosen randomly, with the remaining samples being used as the test set. This model was trained with the Adam optimizer on Mean Squared Error (MSE) loss using a flat learning rate of 0.001, utilizing early stopping as the terminating condition. Early stopping was set to 200 epochs. The neural network was designed with 4 hidden layers of size 128, 64, 32, and 16, respectively. Hyperparameter tuning yielded marginally better loss, but for the sake of simplicity, this result was discarded. The resulting Actual vs Predicted plots are shown in Figs. 5 and 6. These results correspond to the training results, which yielded an MSE of roughly 0.02. Of the two, the inlet concentration control action better maps the training data, whereas the heat input does not show as good mapping.

5.6. Closed-loop simulation results

To demonstrate the closed-loop behavior of this controller, we split the state space into rings. Starting from the stability region boundary, we sample the region between this boundary and the adjacent level set that is 1 unit smaller. We do this for all rings between the boundary and the minimal region $V = 1$ in increments of 1. To get decent sampling per ring, we used 10 randomly generated samples and simulated the trajectory for each sample from the initial point until the trajectory either enters the minimal region $V \leq 1$ or completes a full horizon worth of control actions. Data from each trajectory is gathered and presented in Fig. 8. Sample trajectories are shown in Fig. 7, where the term steps refers to the number of sampling times that have elapsed as well as the number of control actions that have been applied thus far.

The sample trajectories for the NN controller demonstrate a rough similarity to the MPC results. The trajectories visibly deviate over the duration of the trajectory, but when averaged out over the various rings, both controllers exhibit similar trends regarding the number of steps needed to reach the minimal region. Notably, we can see that on average, the MPC and NN share very similar steps needed, whereas MPC tends to have higher maximum steps needed in some parts of the ring. Most importantly, the time-complexity of the NN over the full trajectory is 3 orders of magnitude smaller than the designed MPC, and is only 1 order of magnitude larger than a simple P controller, which demonstrates higher step requirements on average. This enables the use of the NN with roughly 100x smaller sampling times before the NN-based controller design begins to have feasibility concerns as the computational time reaches the order of the sampling time.

For the implementation used in this example, it is also possible to modify the MPC implementation to maximize its performance, ignoring the feasibility implications. In the given system, one can lower the sampling time and extend the horizon length to maximize the MPC's performance. Doing so in real-world applications will be impossible, as the computational time for solving the MPC risks exceeding the sampling time; however, the NN is trained using offline MPC data applied to an internal model of the system. Thus, the feasibility of the controller in real-world applications is irrelevant, as the optimal control for a given state with the new sampling time and horizon length will be the same regardless of if the control gets applied. With the performance benefits, an NN can then be trained on this data to approximate it within the necessary sampling time limits. This enables the potential for higher quality control through this framework than otherwise possible with traditional MPC, but a good fit is needed for this benefit to be important.

In this example, the neural network that was used to fit the LMPC control action data was subjected to a certain, albeit small, error, and so

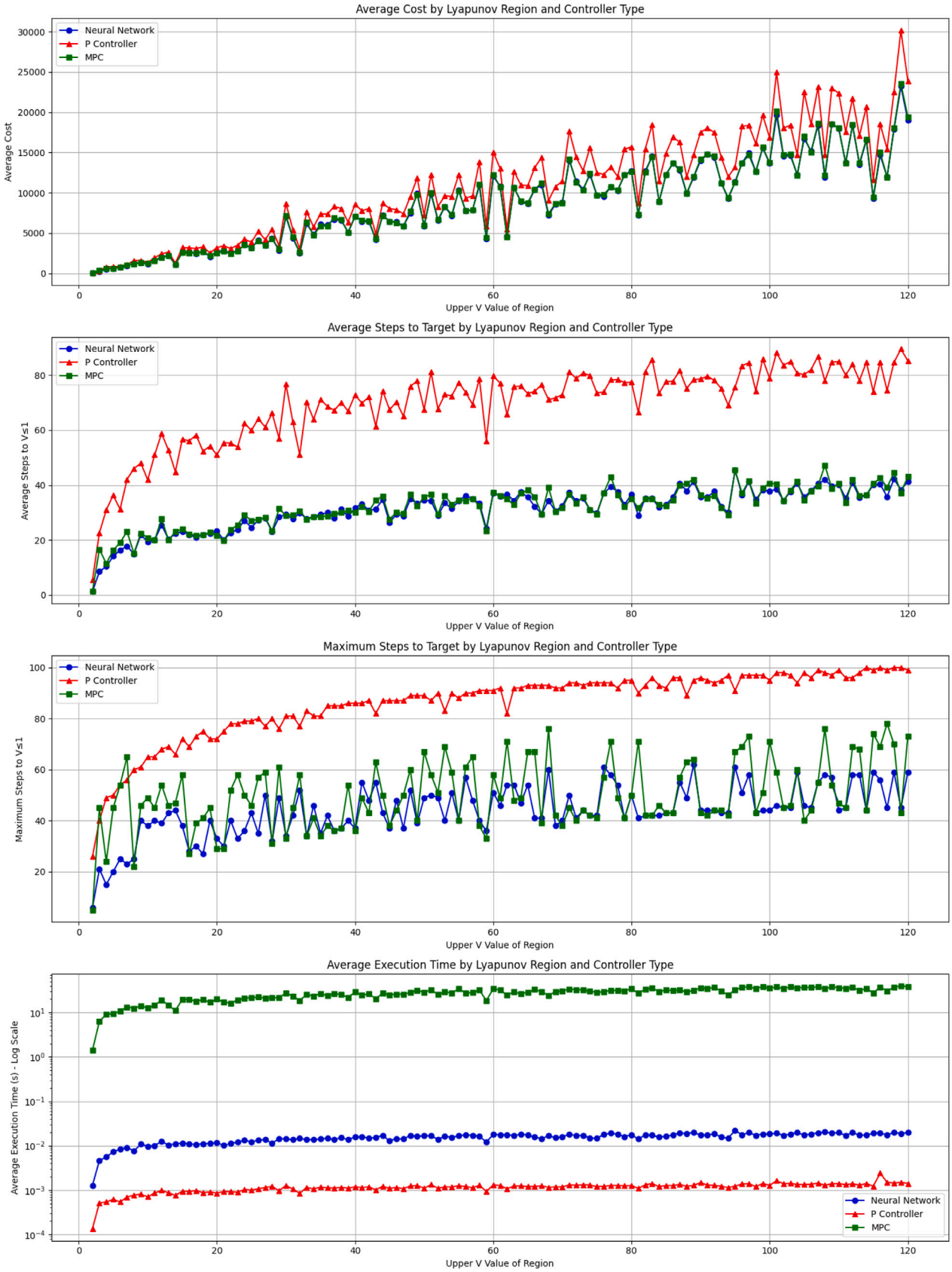

Fig. 8. Performance results of state trajectories compared by the control system type and the level set region in which the initial state lies. Performance in this context includes the average cost, steps, and computation time, as well as the maximum steps.

when analyzing the rate at which the fallback proportional controller was needed for the NN to maintain the specified stability guarantees, we found that roughly 2.6% of operating points required the use of the fallback control. Fig. 7 demonstrates how the NN, when applied for the same initial point as the MPC, produces a trajectory that does not track the MPC's trajectory. The bulk of the control actions generated by the NN are valid with regard to the stability constraint. Thus, the NN framework demonstrates that even in the absence of a close fit to the training data, the existence of the fallback control enables enhanced performance while maintaining stability guarantees.

6. Conclusion

The work proposes a framework for approximating optimization-based model predictive control using neural networks while maintaining stability guarantees through the use of externally enforced Lyapunov stability constraints (expressed in terms of the negative definiteness of the time derivative of a Lyapunov function) and stabilizing fallback control. Randomly sampled points in the operating region are fed into a Lyapunov-based MPC to generate optimal control inputs. These points are paired together and stored as training data. After any desired data pre-processing and filtering is done, a neural network is trained to fit the data. The resulting neural network acts as a nonlinear feedback controller and a substitute for the MPC, where constraints are checked after computation of the NN's output (i.e., control actions). If the Lyapunov stability constraints are violated, the stabilizing fallback controller is used to ensure that the system maintains its stability guarantees. As a result, the NN substitute enables approximate optimal control with a potential for several orders of magnitude improvement in the speed of real-time calculation of the control actions when applied to a benchmark nonlinear chemical process example. This framework enables the use of approximate optimal control in environments where computation of the corresponding MPC is infeasible due to control action computation time exceeding the sampling time. Additionally, the framework applies these guarantees through external constraints, which do not hinder the flexibility of the NN design at all.

CRediT authorship contribution statement

Arthur Khodaverdian: Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Dhruv Gohil:** Software, Investigation. **Panagiotis D. Christofides:** Writing – original draft, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

Financial support from the National Science Foundation, United States, CBET-2227241, is gratefully acknowledged.

References

Adabag, E., Atal, M., Gerard, W., Plancher, B., 2024. MPCGPU: Real-time nonlinear model predictive control through preconditioned conjugate gradient on the GPU. In: Proceedings of International Conference on Robotics and Automation. Yokohama, Japan, pp. 9787–9794.

Ahn, K., Mhammedi, Z., Mania, H., Hong, Z.W., Jadbabaie, A., 2022. Model predictive control via on-policy imitation learning. arXiv preprint arXiv:2210.09206.

Alhajeri, M.S., Ren, Y.M., Ou, F., Abdullah, F., Christofides, P.D., 2024. Model predictive control of nonlinear processes using transfer learning-based recurrent neural networks. Chem. Eng. Res. Des. 205, 1–12.

Alora, J.I., Pabon, L.A., Köhler, J., Cenedese, M., Schmerling, E., Zeilinger, M.N., Haller, G., Pavone, M., 2023. Robust nonlinear reduced-order model predictive control. In: Proceedings of 62nd Conference on Decision and Control. Marina Bay Sands, Singapore, pp. 4798–4805.

Alsmeier, H., Savchenko, A., Findeisen, R., 2024. Neural horizon model predictive control - increasing computational efficiency with neural networks. In: Proceedings of the American Control Conference. Toronto, Canada, pp. 1646–1651.

Åström, K., Hägglund, T., 2001. The future of PID control. Control Eng. Pract. 9, 1163–1175.

Bonzanini, A.D., Paulson, J.A., Graves, D.B., Mesbah, A., 2020. Toward safe dose delivery in plasma medicine using projected neural network-based fast approximate NMPC. IFAC-PapersOnLine 53, 5279–5285.

Gill, P.E., Murray, W., Picken, S.M., Wright, M.H., 1979. The design and structure of a fortran program library for optimization. ACM Trans. Math. Soft. 5 (3), 259–283.

Gonzalez, C., Asadi, H., Kooijman, L., Lim, C.P., 2024. Neural networks for fast optimisation in model predictive control: A review. arXiv preprint arXiv:2309.02668.

Gordon, D.C., Winkler, A., Bedei, J., Schaber, P., Pischinger, S., Andert, J., Koch, C.R., 2024. Introducing a deep neural network-based model predictive control framework for rapid controller implementation. In: Proceedings of American Control Conference. Toronto, Canada, pp. 5232–5237.

Kapoor, S., Vaidya, P.M., 1986. Fast algorithms for convex quadratic programming and multicommodity flows. In: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing. Berkeley, California, USA, pp. 147–159.

Khodaverdian, A., Gohil, D., Christofides, P.D., 2025. Enhancing cybersecurity of nonlinear processes via a two-layer control architecture. Digit. Chem. Eng. 15, 100233.

Kraft, D., 1988. A software package for sequential quadratic programming. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht, Wiss. Berichtswesen d. DFVLR.

Lucia, S., Karg, B., 2018. A deep learning-based approach to robust nonlinear model predictive control. IFAC-PapersOnLine 51, 511–516.

Macmurray, J., Himmelblau, D., 1995. Modeling and control of a packed distillation column using artificial neural networks. Comput. Chem. Eng. 19, 1077–1088.

Meng, D., Chu, H., Tian, M., Gao, B., Chen, H., 2024. Real-time high-precision nonlinear tracking control of autonomous vehicles using fast iterative model predictive control. IEEE Trans. Intell. Veh. 9, 3644–3657.

Mhaskar, P., El-Farra, N.H., Christofides, P.D., 2006. Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control. Syst. Contr. Lett. 55, 650–659.

Pardalos, P.M., Vavasis, S.A., 1991. Quadratic programming with one negative eigenvalue is NP-hard. J. Global Optim. 1, 15–22.

Patel, R., Bhartiya, S., Gudi, R.D., 2025. Neural network-based model predictive control framework incorporating first-principles knowledge for process systems. Ind. Eng. Chem. Res. 64 (18), 9287–9302.

Peng, Y., Yan, H., Rao, K., Yang, P., Lv, Y., 2024. Distributed model predictive control for unmanned aerial vehicles and vehicle platoon systems: a review. Intell. Robot. 4, 293–317.

Qin, S.J., Badgwell, T.A., 2003. A survey of industrial model predictive control technology. Control Eng. Pract. 11, 733–764.

Tomasetto, M., Braghin, F., Manzoni, A., 2025. Latent feedback control of distributed systems in multiple scenarios through deep learning-based reduced order models. Comput. Methods Appl. Mech. Engrg. 442, 118030.

Vilanova, R., Visioli, A. (Eds.), 2012. PID control in the third millennium: Lessons learned and new approaches. In: Advances in Industrial Control, Springer London, p. XIV, 602.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors, 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods 17, 261–272.

Wang, R., Li, H., Xu, D., 2022. Learning model predictive control law for nonlinear systems. In: Proceedings of International Symposium on Autonomous Systems. Hangzhou, China, pp. 1–6.

Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019a. Machine learning-based predictive control of nonlinear processes. Part I: Theory. AIChE J. 65, e16729.

Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019b. Machine-learning-based predictive control of nonlinear processes. Part II: Computational implementation. AIChE J. 65, e16734.

Xi, Y.G., Li, D., Lin, S., 2013. Model predictive control — Status and challenges. Acta Automat. Sinica 39, 222–236.

Yaren, T., Kizir, S., 2025. Real-time nonlinear model predictive control of a robotic arm using spatial operator algebra theory. J. Field Robot. (in Press).

Zarrouki, B., Nunes, J., Betz, J., 2023. R^2NMPC: A real-time reduced robustified nonlinear model predictive control with ellipsoidal uncertainty sets for autonomous vehicle motion control. arXiv preprint arXiv:2311.06420.