Computers and Chemical Engineering 204 (2026) 109396

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Uniting neural network-based control and model predictive control: Application to a large-scale nonlinear process

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:
Lyapunov stability
Approximate model predictive control
Neural networks
Stability guarantees
Closed-loop performance
Nonlinear processes

ABSTRACT

This work proposes a method to overcome the issue of nonlinear model predictive control (MPC) requiring practically infeasible computation times for large-scale systems. In particular, the use of Neural Networks (NN) to approximate nonlinear MPC calculated control actions in a real-time closed-loop implementation with externally enforced stability guarantees is explored. Using Lyapunov-based stability constraints, the reduced computational complexity of NNs paired with the ability to train using MPC that would be infeasible to apply in real-time systems (due to the use of a large prediction horizon to ensure good closed-loop performance) enables the training of an NN-based approximate control policy that directly substitutes MPC. With a stabilizing fallback controller available, this NN controller enables real-time stabilizing control of high-dimensional nonlinear systems. To demonstrate this, Aspen Plus Dynamics, a dynamic chemical process simulation software, is used to create a large-scale nonlinear chemical process example. Using an NN trained off of an offline MPC using a first-principles model and a large prediction horizon, a comprehensive study of the resulting closed-loop behavior is carried out to evaluate the closed-loop stability, performance, and robustness properties of the approach.

1. Introduction

Controlling large-scale nonlinear chemical processes poses a unique design challenge due to the need to address the possibility of highly nonlinear behavior and interacting variables within the process network. For systems of this scale, the best option for designing a reliable and effective controller is to work within a model predictive control (MPC) framework. MPCs are unique due to their explicit use of a predictive model accounting for process dynamic behavior. Unlike traditional controllers, such as Proportional–Integral–Derivative control, which operate on a single input and regulate a single output without any consideration of the interactive dynamics between process variables, MPC uses an internal model of the process dynamics to forecast the future trajectory of the process states for any given set of control actions. With this trajectory, MPC can optimize control actions based on the approximated impact they will have, which enables a tunable controller with generalizable applicability (Qin and Badgwell, 2003).

A major consequence of explicitly calculating these control action trajectories is the impact on computation time. For linear convex cases (i.e., linear process model and convex cost and constraints), MPC is solvable via convex quadratic programming, which is known to have polynomial-time complexity with some forms reaching $\mathcal{O}\left(L^2n^4\right)$ where L is the bit-length of the input and n is the number of variables (Peng et al., 2024). Certain novel optimization schemes can slightly reduce this complexity, but for nonlinear constrained examples that arise in nonlinear chemical processes, the optimization problem's complexity becomes NP-hard (Kapoor and Vaidya, 1986; Pardalos and Vavasis, 1991). Generally, these nonlinear programming (NLP) problems use an iterative solution method to calculate a local optimum, which complicates the derivation of any explicit Big O notation for time complexity. Some methods, such as the Sequential Least Squares Programming (SLSQP) method, are composed of subproblems that are of known complexity (in SLSQP's case, these subproblems include linear least squares problems and are of $\mathcal{O}\left(n^2\right)$ or $\mathcal{O}\left(n^3\right)$) (Gill et al., 1979; Kraft, 1988). Knowing this, it may be reasonably assumed that MPC operates with polynomial-time complexity at best, which results in poor scaling. Because of this poor scaling, MPC risks becoming infeasible to solve in real-time for large-scale nonlinear processes when a long horizon is used to improve closed-loop performance, as it may take longer to solve than the controller time (Xi et al., 2013).

The handling of this scaling issue is a topic that recent research aims to address; however, the current approaches are mixed in how this is

* 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.compchemeng.2025.109396
Received 13 August 2025; Received in revised form 4 September 2025; Accepted 5 September 2025
Available online 12 September 2025

done. The two main approaches can be categorized as either a 'Reduction' approach or an 'Optimization' approach. The reduction approach focuses on reducing the dimension of the process model used in MPC without significant performance losses, which does not directly address the scaling issue but instead attempts to alleviate it by artificially lowering the scale (Tomasetto et al., 2025; Alora et al., 2023; Zarrouki et al., 2023). The optimization approach aims to directly reduce the computational time of the MPC; however, these optimizations have thus far failed to directly address the time complexity scaling problem (Meng et al., 2024; Yaren and Kizir, 2025; Adabag et al., 2024). The primary issue with the optimization approach is that reducing computation time by a flat order of magnitude does not change the general behavior of the computation time scaling with polynomial complexity. In other words, both major focuses simply delay the onset of this scaling problem.

A more recent development in this topic has been the application of neural networks. Much of the research on neural network applications has focused on the optimization approach, where an NN replaces a component of the MPC problem. Most notably, NNs that model the system dynamics have shown promise as replacements for existing first-principles models in complex systems, as they can simultaneously improve the real-world accuracy of the predictions for the system dynamics while also being faster to compute (Wu et al., 2019; Gordon et al., 2024; Patel et al., 2025; Alsmeier et al., 2024; Ren et al., 2022). As a data-driven approach, some highly complex systems demonstrate better modeling through NNs than through first-principles models (Macmurray and Himmelblau, 1995). Despite the success of NNs in this context, the iterative nature of MPCs will still result in an overall polynomial-time complexity.

Expanding on this line of logic, if neural networks can improve the computation time of dynamics modeling with potentially better accuracy than first-principles models, then how will a neural network perform if it is instead trained to approximate the entire model predictive controller? This question has been a point of exploration in novel research, which has shown the potential for this method to be viable (Lucia and Karg, 2018; Bonzanini et al., 2020; Ahn et al., 2022). The main challenges with this research thus far have been how the network can be designed to operate in a way that strictly abides by constraints and how the method scales (Gonzalez et al., 2024). Constraint satisfaction is achievable through either mapping of outputs, further optimizations of outputs, or by externally enforcing stabilizing constraints with a fallback stabilizing controller (Khodaverdian et al., 2025b). Other viable options include the use of constraints within the loss function, which aids in the satisfaction of constraints during training prior to any strict enforcement (Wang et al., 2022).

Although this neural network-based approach shows promising improvements to the computation time relative to model predictive control, the scalability and the resulting impact on time complexity remain underexplored. Thus, the present work explores the case where a large-scale system cannot run long-horizon model predictive control (thereby ensuring good closed-loop performance) within its controller sampling time. Specifically, a supervised learning approach with externally enforced Lyapunov-based stability constraints is used to design a neural network-based control process that can be trained offline using this practically infeasible model predictive controller (which has a long prediction horizon) in order to approximate it in real-time. If instead of the long-horizon MPC, a short-horizon form of the same MPC is used as the stabilizing backup controller, then the NN-based controller will ideally achieve performance that is similar to the long-horizon MPC, which could not be utilized in real-time due to its long calculation time exceeding the process sampling time. Aspen Plus Dynamics enables the use of a large process model with high complexity. The performance of the resulting NN-based control system is examined for this benchmark process to demonstrate the viability of the method when applied to a large-scale process.

2. Preliminaries

2.1. Notation

We denote the transpose of a vector x as $x^\top$. $\mathbb{R}$ denotes the set of real numbers. Subtraction of set B from set A produces the elements present in A but absent from B and is written as $A \backslash B$. We represent functions using $f(\cdot)$ where the choice of f is arbitrary.

2.2. Class of systems

This study examines nonlinear multiple-input multiple-output (MIMO) continuous-time systems governed by nonlinear first-order ordinary differential equations (ODEs) of the form:

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

Here, the state vector $x = \left[x_1, x_2, \ldots, x_n\right]^\top \in \mathbb{R}^n$ represents the process state variables, which are measured at every sampling time t_k (i.e., a state feedback control scenario). The control input vector $u = \left[u_1, u_2, \ldots, u_m\right]^\top \in \mathbb{R}^m$ represents the applied control actions, with $0 < m \leq n$. To account for actuator physical limits, each control action 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 inequality constraint defines the bounded subset of valid control actions, denoted $U \subset \mathbb{R}^m$. The functions $f(\cdot)$ and $g_i(\cdot)\ \forall\ i = 1,2,\ldots,m$ from Eq. (1) 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 $F(0,0) = 0$). The initial time is defined as zero ($t_0 = 0$). Finally, the set $S(\Delta)$ denotes the assortment of piecewise constant functions characterized by a period of Δ.

2.3. Stabilizability assumption

We assume an explicit feedback control law $u(x) = \Phi(x) \in U$ exists that can render the origin of Eq. (1) exponentially stable. This assumption is referred to as the stabilizability assumption. Specifically, we assume the existence of 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:

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

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

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

with positive constants c_1, c_2, c_3, and c_4. Combined with the previously mentioned input bounds, the stabilizability assumption ensures the existence of positive constants M_F, L_x, and L'_x that ensure, for all $x, x' \in D$ and $u \in U$, that the following inequalities are satisfied:

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

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

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

2.4. Lyapunov-based model predictive control

A Lyapunov-based MPC (LMPC) that ensures closed-loop stability within an explicitly defined region is formulated as follows (Mhaskar et al., 2006):

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

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

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

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

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

where Δ is the controller sampling interval. Δ in this context matches both the time between state measurements as well as the hold time for the control actions that will be applied in a sample-and-hold manner. N is the number of intervals in the prediction/control horizon, i.e., the horizon length. Eq. (4a) is the cost function to be optimized, whereas Eq. (4b) is the model that enables the simulation of process dynamics and is initialized by Eq. (4d) for any arbitrary time frame t_k. In this case, a quadratic form of the cost function and a first-principles form of the process dynamics model are used. The stability constraint (Eq. (4e)) leverages Section 2.3 to ensure that the LMPC's control action drives the state towards the origin at least as quickly as the reference stabilizing controller $\boldsymbol{\Phi}(x)$. Alternatively, one may use:

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

This form of the constraint enforces negativity for $\dot{V}$ proportional to V using a positive constant α, enabling stability without the need for an explicit reference controller.

Remark 1. LMPC has flexible support for cost functions. While a quadratic form is used here, a particularly beneficial option is an economic-based cost function that can vary with $\tilde{x}$, u, and even time, as was used in our prior work (Khodaverdian et al., 2025a).

Remark 2. As detailed in Section 2.3, continuous application of $\boldsymbol{\Phi}(x)$ ensures exponential stability, but practically, sample-and-hold implementation is needed to account for the computation and signal transport induced delays. The consequence of this is that practical implementations of $\boldsymbol{\Phi}(x)$ will limit guarantees to convergence of the closed-loop system to a small neighborhood around the origin, as will be detailed in Section 4.2.

Remark 3. The LMPC formulation is used for two identical LMPC designs where the only variation is the horizon length. One LMPC is a long-horizon LMPC where real-time application is infeasible due to the computation time of the LMPC being longer than the sampling time of the system. This long-horizon LMPC is used to generate training data. The other LMPC is a short-horizon LMPC where real-time application is feasible. Unlike the long-horizon LMPC, this short-horizon LMPC is designed with a sufficiently small horizon such that the computation time is never larger than the sampling time. This LMPC is used as a stabilizing fallback controller where the quality of the control action will suffer without the loss of closed-loop stability guarantees.

Remark 4. Stability constraints are only applied at t_k. Due to the use of a receding horizon approach, only the initial control action is necessary for closed-loop stability guarantees to be satisfied. Enforcing these constraints along the full horizon can enhance the cost function minimization, but comes with a large computational burden, making it a difficult option to consider in real-time control. Although the offline LMPC is not applied in real-time, both the LMPC used for training and the LMPC used for real-time stabilizing fallback control are designed to be identical, with the exception of the horizon length.

3. Neural network design

3.1. Data generation

Supervised learning is most effective when using a comprehensive, varied dataset that covers the target operating region; however, high-dimensional process control often does not have a well-defined operating region, making it difficult to prepare a sufficient dataset for consistent network performance in real-world applications. As a data-driven process, it is important to attempt to handle this problem in a manner that can be applied more generally.

One such method that is viable for systems that use model predictive control is to use the LMPC design as an offline process. The LMPC uses an internal physics-based model of the process and can be simulated offline in a closed-loop manner to generate approximate trajectories for a given set of constraints, costs, and process dynamics. This enables on-demand data generation. Additionally, because the process is simulated offline, it is possible to take an LMPC that is otherwise infeasible to apply in a real-time closed-loop system and use it to generate higher-quality data. Model predictive controllers can be tuned to provide higher-quality optimal control at the expense of computation time. While this is a convenient way to generate a dataset of high-quality state-control pairs, the higher computation time also limits the rate at which training data is generated. Our approach aims to get a mix of high-quality and high-quantity data. In order to prevent the risk of data contamination, data generation is only done on a singular LMPC formulation where only valid solutions are kept.

3.2. Controller neural network model type and training

The controller implementation framework ensures closed-loop stability and any robustness guarantees against measurement noise or model uncertainty through fallback control to a stabilizing controller that is capable of enforcing these guarantees. If constraints cannot be satisfied by the fallback controller, such as excessive robustness demands, then the NN-controller will not have guaranteed satisfaction of these constraints. In other words, the NN-controller has no inherent guarantees and relies on fallback control to ensure these guarantees hold. Thus, the framework enables a broad design basis. The neural network and its training can be of any desired form so long as there exists some way in which state measurements can be converted to approximated optimal control actions. If this NN is poorly constructed, either through a suboptimal framework, poorly tuned hyperparameters, or poor training, the end result will be a gradual shift towards the applied control action being mostly comprised of the fallback controller's solution. In order to avoid this, it is important that the NN demonstrates good performance prior to implementation; otherwise, the quality of the control action with respect to the desired cost function will suffer.

4. Guaranteeing stability for neural network-based control

This section covers how the NN-based approximate implementation of MPC with stability guarantees functions.

4.1. Closed-loop implementation

Fig. 1 depicts the practical implementation of the NN controller as a direct substitute for a given MPC design. Note the existence of both a fallback controller and a constraint enforcer block. The fallback controller is a key design aspect, as it is needed in order for any guarantees to exist. The constraint enforcer block contains the logic as to how the constraints are enforced for the given system. Some constraints, such as bounds on the control actions, are enforceable by simply clipping the solved control actions. Other constraints, such as the stabilizing constraints, have complex conditional requirements that need to be checked using sensor and control signals. This conditional

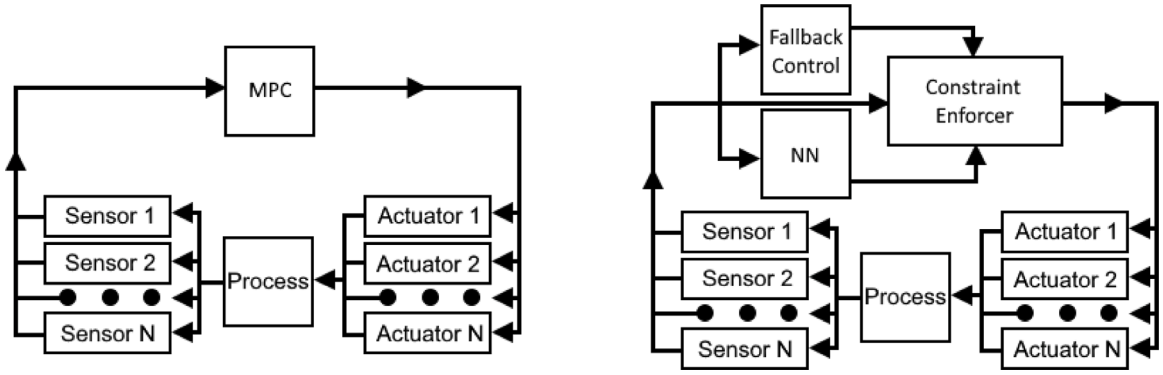

Fig. 1. Model predictive controller (Left) application block diagram for a general process vs. neural network-based controller with validation and fallback stabilizing controller (Right). Failsafe control is implied to be within either the fallback control block, constraint enforcer block, or both, as desired.

computation is done, and the decision as to whether or not the fallback control is applied is made. In other words, the constraint enforcer does the necessary checks to determine which constraints are violated, and either directly modifies the control action to satisfy the constraints or replaces the control action with that of the fallback controller.

A notable aspect of the framework is that the only necessary information for the NN is the sensor signals at a given reference time frame $x\left(t_k\right)$. Additional information, such as time-series data, can potentially improve the controller's performance, but is not necessary. Thus, a minimal realization of this NN-controller is one of

$$\boldsymbol{\Phi}_{nn} : \mathbb{R}^n \to \mathbb{R}^m \tag{6}$$

$$u\left(t_k\right) := \boldsymbol{\Phi}_{nn}\left(x\left(t_k\right)\right) \tag{7}$$

Remark 5. For complex systems, it is beneficial to have a short-horizon form of the existing MPC as the fallback controller. This minimizes the scaling problem while still guaranteeing stability, but comes at the risk of poor control quality. Because MPC is a numerical approach, there are many cases where, even if a solution exists, the framework might not be able to solve it properly. In such cases, the MPC will fail, requiring its own fallback controller, referred to as the failsafe controller. Thus, the existence of stabilizing fallback control is a core requirement for any such form of control with stabilizing guarantees.

Remark 6. A benefit of this framework is the support for modified MPC formats. An example that enables a cyber-secure implementation of MPC is the two-layer framework. This framework does not directly transmit the MPC's optimal control trajectory; instead, it uses these control actions with the internal process model to estimate the state trajectory, and transmits this trajectory with encryption. Through homomorphic encryption, this trajectory can be used as set-points for a linear controller, enabling computations to be done without decrypting the signal, thereby obfuscating the control signal. This gives improved cybersecurity at the cost of poorer quality control due to linear control. In the context of the NN-controller, either the control trajectory can be used to analytically make the estimated state trajectory, or the NN-controller can be trained to encompass this step and will generate the estimated state trajectory directly.

Remark 7. Specific design details for MPC and the NN are left unspecified here because these are process-dependent design parameters that are not influenced by the framework. The framework's novelty is the external constraint enforcer block, which simplifies the design process by not requiring complex training or architectural features while simultaneously supporting these complex features without loss of its guarantees.

4.2. Closed-loop stability under LMPC

Section 2.3 presents the stabilizability assumption for a continuous-time controller; however, practical implementation requires a sample-and-hold implementation for the controller. Thus, the closed-loop properties need to be re-evaluated under sample-and-hold implementation of the controller. As will be shown below, instead of closed-loop exponential stability, the controller will guarantee convergence of the closed-loop state to a small region around the origin whose size is proportional to the sampling time.

Theorem 1. *Consider an MPC described by Eq. (4) using either Eq. (4e) or Eq. (5) as the stabilizability assumption with Eq. (4b) describing 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 state satisfying $V\left(x\left(t_0\right)\right) \le \rho$, there exists positive constants ϵ_w and α such that if the following conditions are satisfied, the sample-and-hold implementation of $u(t)$ ensures the convergence of the closed-loop state to a small region around the origin denoted $\Omega_{\rho_{\min}}$ and determined as follows:*

$$L_x' M_F \Delta - \alpha \rho_s \le -\epsilon_w \tag{8a}$$

$$\rho_{\min} = \max\left\{V\left(x\left(t_k+\Delta\right) | V\left(x\left(t_k\right)\right) \le \rho_s\right)\right\} \tag{8b}$$

$$\rho_s < V\left(x\left(t_k\right)\right) \tag{8c}$$

$$\rho_s < \rho_{\min} < \rho \tag{8d}$$

Proof.

Case 1 (*Eq. (4e) as the Lyapunov Constraint*)**.** The sample-and-hold implementation results in states evolving without new control actions over $t \in \left[t_k, t_k+\Delta\right)$; thus, the time derivative of the Lyapunov function for any given point in this interval is:

$$\dot{V}\left(x(t), u\left(t_k\right)\right) = \frac{\partial V(x(t))}{\partial x} F\left(x(t), u\left(t_k\right)\right) \tag{9}$$

Adding and subtracting the derivative at t_k:

$$\begin{aligned}\dot{V}\left(x(t), u\left(t_k\right)\right) = &\frac{\partial V(x(t))}{\partial x} F\left(x(t), u\left(t_k\right)\right) \\ &- \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} F\left(x\left(t_k\right), u\left(t_k\right)\right) \\ &+ \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} F\left(x\left(t_k\right), u\left(t_k\right)\right)\end{aligned} \tag{10}$$

Using Lipschitz continuity of V and stabilizability assumptions Eqs. (2b) and (3c), we simplify with Eq. (4e):

$$\dot{V}\left(x(t),u\left(t_k\right)\right) \leq L'_x \left|x(t)-x\left(t_k\right)\right| - c_3\left|x\left(t_k\right)\right|^2 \tag{11}$$

By the integral triangle inequality:

$$\left|x(t)-x\left(t_k\right)\right| \leq \int_{t_k}^{t} \left|F\left(x(\tau),u\left(t_k\right)\right)\right| \mathrm{d}\tau \leq M_F \Delta \tag{12}$$

Thus:

$$\dot{V}\left(x(t),u\left(t_k\right)\right) \leq L'_x M_F \Delta - c_3\left|x\left(t_k\right)\right|^2 \tag{13}$$

For small $|x|$, the $L'_x M_F \Delta$ term may dominate. Thus, stability requires Eq. (8c). Paired with Eq. (3b), this yields:

$$\dot{V}\left(x(t),u\left(t_k\right)\right) \leq L'_x M_F \Delta - \frac{c_3}{c_2}\rho_s \leq L'_x M_F \Delta - \alpha\rho_s \tag{14}$$

For $x\left(t_k\right) \in \Omega_\rho \backslash \Omega_{\rho_s}$ and sufficiently small Δ, Eq. (8a) ensures:

$$\dot{V}\left(x(t),u\left(t_k\right)\right) \leq -\epsilon_w \tag{15}$$

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 denoted $\Omega_{\rho_{\min}}$ defined by Eq. (8b)

Case 2 (*Eq. (5) as the Lyapunov Constraint*)**.** Following similar steps as before:

$$\dot{V}\left(x(t),u\left(t_k\right)\right) = \frac{\partial V(x(t))}{\partial x} F\left(x(t),u\left(t_k\right)\right) \tag{16}$$

Using Eq. (5), this simplifies as before:

$$\dot{V}\left(x(t),u\left(t_k\right)\right) \leq L'_x \left|x(t)-x\left(t_k\right)\right| - \alpha V\left(x\left(t_k\right)\right) \leq L'_x M_F \Delta - \alpha\rho_s \tag{17}$$

Thus, the result is the same as the prior case. □

There are two valid forms of the Lyapunov constraint whose stability guarantees are demonstrated in the above proof, but both guarantee convergence of the closed-loop system to a small region around its origin whose size is proportional to the sampling time of the system. Notably, neither of these conditions explicitly utilizes the NN-control, whereas both require the existence of a reference stabilizing controller, which emphasizes the fact that the NN-control itself lacks constraint guarantees.

Remark 8. Eq. (5) provides an alternate form of the Lyapunov constraint that can potentially simplify the design process of the LMPC due to the lack of needing to explicitly find the reference stabilizing controllers. The consequence of this is that the existence of a solution that satisfies this form of the constraint is dependent on how large α is. Thus, the solution depends on α. Specifically, as α increases, the magnitude of $\dot{V}$ proportionally increases. If the control actions were unbounded, this would not be an issue, but because the system has bounded control actions, it is possible that such a large magnitude $\dot{V}$ is not achievable within the control bounds. This is a case in which the LMPC problem is over-constrained to the point where a valid solution does not exist. In reality, α only needs to be large enough to counteract the impact of the sample-and-hold implementation on the upper-bound of the $\dot{V}$. Increasing it further than this point will reduce ρ_s, which is not necessary beyond a certain degree. Thus, it is important to choose an α that is sufficiently small to prevent overconstraining while still being sufficiently large to guarantee stability.

Remark 9. It is important to emphasize that this proof is independent of the horizon length of the LMPC. The proof is written in general form and thus any applied sample-and-hold control action that satisfies the stability constraint will ensure stability as defined. In other words, the only requirement for ensuring convergence to a small region around the origin is to enforce the stability constraint for t_k so long as the LMPC is resolved for subsequent control actions. An alternative to this would be to enforce the stability constraint for all control actions in the horizon, but this would significantly impact computation time.

4.2.1. Closed-loop stability under NN controller

The framework used in this paper is not a pure LMPC-based approach, and instead is of the form shown on the right half of Fig. 1. This control system conditionally applies control actions depending on satisfaction of the Lyapunov constraint. Theorem 2 demonstrates how this framework also guarantees closed-loop convergence to a small region around the origin.

Theorem 2. *Consider the following:*

- *An LMPC described by Eq. (4) using Eq. (5) as the stabilizability assumption with Eq. (4b) describing a nonlinear system of the form described by Eq. (1).*
- *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 $V\left(x\left(t_0\right)\right) \leq \rho$.*
- *A neural network-based controller whose control actions are clipped to satisfy the bounds defined in Eq. (4c) ($\Phi_{nn}\left(x\left(t_k\right)\right) \in U$).*

For a process of the form shown on the right of Fig. 1 with a constraint enforcer implemented as

$$u\left(t_k\right) = \begin{cases} \Phi_{nn}\left(x\left(t_k\right)\right), & \text{if } \dot{V}\left(x\left(t_k\right),\Phi_{nn}\left(x\left(t_k\right)\right)\right) \leq -\alpha V\left(x\left(t_k\right)\right) \\ \Phi_{LMPC}\left(t_k\right), & \text{if } \dot{V}\left(x\left(t_k\right),\Phi_{nn}\left(x\left(t_k\right)\right)\right) > -\alpha V\left(x\left(t_k\right)\right) \end{cases} \tag{18}$$

where

$$\Phi_{LMPC}\left(t_k\right) = \begin{cases} u_{LMPC}\left(t_k\right), & \text{if } \dot{V}\left(x\left(t_k\right),u\left(t_k\right)\right) \leq -\alpha V\left(x\left(t_k\right)\right) \\ \Phi\left(x\left(t_k\right)\right), & \text{if } \dot{V}\left(x\left(t_k\right),u\left(t_k\right)\right) > -\alpha V\left(x\left(t_k\right)\right) \end{cases} \tag{19}$$

such that $u, u_{LMPC}, \Phi, \Phi_{nn}, \Phi_{LMPC} \in U$, there exists positive constants ϵ_w and α such that is Eqs. (8a)–(8d) are satisfied, application of $u\left(t_k\right)$ in a sample-and-hold manner to the closed-loop system guarantees convergence to a small region around the origin ($\Omega_{\rho_{\min}}$) for any reference time frame t_k.

Proof.

Case 1 ($\dot{V}\left(x\left(t_k\right),\Phi_{nn}\left(x\left(t_k\right)\right)\right) > -\alpha V\left(x\left(t_k\right)\right)$)**.** This case contains two further subcases to consider. In the first, the LMPC fails to find a solution. In such a case, the LMPC solution is no longer guaranteed to satisfy the constraints, and so some form of controller needs to be applied that is capable of handling these rarer failure modes. For this case, the reference stabilizing controller, $\Phi\left(x\left(t_k\right)\right)$, is used as the final layer of fallback control, referred to as failsafe control. As discussed in Section 2.3, this reference controller guarantees satisfaction of the constraints and is explicit, meaning there is no chance of failure. The proof for this controller in the case of a sample-and-hold implementation follows Theorem 1 with the only change being in the initial formulation of $\dot{V}$.

$$\begin{aligned} \dot{V}\left(x(t),\Phi\left(x\left(t_k\right)\right)\right) = &\frac{\partial V(x(t))}{\partial x} F\left(x(t),\Phi\left(x\left(t_k\right)\right)\right) \\ &- \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} F\left(x\left(t_k\right),\Phi\left(x\left(t_k\right)\right)\right) \\ &+ \frac{\partial V\left(x\left(t_k\right)\right)}{\partial x} F\left(x\left(t_k\right),\Phi\left(x\left(t_k\right)\right)\right) \end{aligned} \tag{20}$$

As discussed in Theorem 1, the Lipschitz property applies for any $u \in U$, and the remaining term is already in the valid form, and so the remaining proof follows exactly as shown in Theorem 1.

For the remaining case where the LMPC solution is valid, the proof is exactly as shown in Theorem 1.

Case 2 ($\dot{V}\left(x\left(t_k\right),\Phi_{nn}\left(x\left(t_k\right)\right)\right) \leq -\alpha V\left(x\left(t_k\right)\right)$)**.** This case once again follows the form from Theorem 1 with the only difference being the

initial formulation.

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

Once again, the Lipschitz continuity applies for any such $u \in U$, and it is assumed that Φ_{nn} is the NN-controller's solution after clipping the control action to fit the control bounds. Because $\dot{V}\left(x\left(t_k\right),\Phi_{nn}\left(x\left(t_k\right)\right)\right)$ satisfies $\dot{V}\left(x\left(t_k\right),\Phi_{nn}\left(x\left(t_k\right)\right)\right) \leq -\alpha V\left(x\left(t_k\right)\right)$, the expression simplifies to the same form shown in Eq. (17). Thus, the remaining proof follows exactly as shown in Theorem 1. □

Remark 10. There are a multitude of techniques available that enable the NN-controller to better approximate the LMPC, including its constraints, as was briefly discussed in Section 1. If the current framework is applied without any fallback controller, then the burden of ensuring constraint satisfaction falls on the techniques applied to the NN-controller. Unless such a technique is capable of demonstrating stability guarantees for the general NLP case of a nonlinear system with nonlinear constraints and nonlinear cost, the system does not have guarantees for stability from the NN-controller alone.

5. Application to a chemical process example

To explore how this framework performs when applied to a large-scale system, a benchmark nonlinear chemical process example is constructed using Aspen Plus Dynamics. For this example, an LMPC is built using a quadratic cost function for both the state and control variables. The resulting NN-controller is trained to approximate this LMPC. The final process is controlled using the NN-controller with a short-horizon form of the LMPC as the fallback control, and Proportional control as the final failsafe. Using this control framework, the resulting closed-loop behavior is analyzed to determine the performance tradeoff.

5.1. Process description

The process involves three elementary second-order irreversible exothermic reactions. The reactions involve ethylene (E), benzene (B), ethylbenzene (EB), and diethylbenzene (DEB), where the main reaction is labeled as "primary" below:

$$C_2H_4 + C_6H_6 \longrightarrow C_8H_{10} \quad \text{(primary)} \tag{22a}$$

$$C_2H_4 + C_8H_{10} \longrightarrow C_{10}H_{14} \tag{22b}$$

$$C_6H_6 + C_{10}H_{14} \longrightarrow 2\,C_8H_{10} \tag{22c}$$

The remaining two reactions are considered to be side reactions. These three reactions occur in two separate non-isothermal, well-mixed continuous stirred tank reactors (CSTR) that are ordered in series.

5.2. Aspen Plus Dynamics model development

The goal when building the Aspen Plus Dynamics model is to have a high-fidelity model of the process that can best capture the real-world dynamics. To do this, we start by using Aspen Plus to build the baseline of the process and determine the steady-states for operation or initialization as follows:

1. Properties:
   (a) Setup:
      i. Valid Phases: Liquid-Only
      ii. Free Water: No
   (b) Components (CAS numbers):
      i. 71-43-2 (benzene)
      ii. 141-93-5 (diethylbenzene)
      iii. 74-85-1 (ethylene)
      iv. 100-41-4 (ethylbenzene)
      v. 110-54-3 (Hexane)[Solvent]
   (c) Methods:
      i. Base Method: PSRK
      ii. Free-water method: STEAM-TA
      iii. Water solubility: 3
      iv. Use true components: True
2. Simulation:
   (a) Main Flowsheet: Refer to Fig. 2
   (b) Reactions: Refer to Eq. (22) and Table 1. Irreversible elementary power-law reactions are used.
   (c) Pressure: All feed streams are at 20 bars. All Pumps set their outlet pressure to 15 bars. All reactors operate at a fixed pressure of 15 bars. Valves 1, 2, 3, and 4 drop the pressure of their outlets by 5, 5, 2, and 14 bars, respectively.
3. Steady-States (T_1 and T_2 are the temperatures of CSTR 1 and 2 from Fig. 2, respectively):
   (a) Desired Steady State: $T_1 = 350\,\text{K}$, $T_2 = 400\,\text{K}$
   (b) Cold Steady State: $T_1 = 300\,\text{K}$, $T_2 = 300\,\text{K}$
   (c) Hot Steady State: $T_1 = 450\,\text{K}$, $T_2 = 500\,\text{K}$

Then, this model is converted to an Aspen Plus Dynamics process as follows:

1. Open 'Dynamics' tab.
2. Enable 'Dynamic Mode' and run once.
3. Run 'Pressure Checker' to ensure no pressure issues remain.
4. Run 'Pressure Driven' to generate the Aspen Plus Dynamics software.
5. In Aspen Plus Dynamics, delete the newly generated Temperature PID controllers.

Remark 11. The desired steady state in this example, as shown in Table 1, was chosen because ethylbenzene is the desired product. Relative to lower temperature steady states, higher temperature steady states tend to provide higher concentrations of this species because these reactions are irreversible, and so conversion rates increase as temperature increases. It was noted during the formulation of the example that exceeding the chosen steady state temperatures posed an increased risk of thermal runaway. Thus, the desired steady state is selected, not only due to it being the optimal point for safe ethylbenzene production, but also because it provided a strong reason to control the process (to avoid thermal runaway).

Remark 12. 'Liquid-Only' is enforced by having a relatively low temperature for the relatively high operating pressures. High temperatures are a particularly troubling behavior with this assumption, even if the temperatures could be controlled, because Aspen Plus Dynamics decreases the fluid's density instead of changing phase. Through testing, it is found that increasing the volume of the reactors, without increasing the fluid volume, counteracts this issue by giving buffer room for the fluid level to rise before overflow becomes an issue, as is the case at extreme temperatures.

Table 1
Aspen model constants and steady-state values.

$C_{B_{o1}} = 2\,\text{kmol m}^{-3}$	$C_{B_{o2}} = 2\,\text{kmol m}^{-3}$	$V_1 = V_2 = 60\,\text{m}^3$
$C_{E_{o1}} = 7\,\text{kmol m}^{-3}$	$C_{E_{o2}} = 6\,\text{kmol m}^{-3}$	$T_{1o} = T_{2o} = 300\,\text{K}$
$E_1 = 71\,160\,\text{kJ kmol}^{-1}$	$E_2 = 83.680\,\text{kJ kmol}^{-1}$	$E_3 = 62.760\,\text{kJ kmol}^{-1}$
$k_1 = 1.528 \times 10^6\,\text{m}^3\,\text{kmol}^{-1}\,\text{s}^{-1}$	$k_2 = 2.778 \times 10^4\,\text{m}^3\,\text{kmol}^{-1}\,\text{s}^{-1}$	$k_3 = 0.4167\,\text{m}^3\,\text{kmol}^{-1}\,\text{s}^{-1}$
$F_1 = 43.2\,\text{m}^3\,\text{h}^{-1}$	$F_2 = 47.87\,\text{m}^3\,\text{h}^{-1}$	$R = 8.314\,\text{kJ kmol}^{-1}\,\text{K}^{-1}$
$\rho_1 = 639.153\,\text{kg m}^{-3}$	$\rho_2 = 607.504\,\text{kg m}^{-3}$	$C_p = 2.411\,\text{kJ kg}^{-1}\,\text{K}^{-1}$
$\Delta H_1 = -1.04 \times 10^5\,\text{kJ kmol}^{-1}$	$\Delta H_2 = -1.02 \times 10^5\,\text{kJ kmol}^{-1}$	$\Delta H_3 = -5.5 \times 10^2\,\text{kJ kmol}^{-1}$
$T_{1c} = 300\,\text{K}$	$T_{1s} = 350\,\text{K}$	$T_{1h} = 450\,\text{K}$
$T_{2c} = 300\,\text{K}$	$T_{2s} = 400\,\text{K}$	$T_{2h} = 500\,\text{K}$
$C_{B_{1c}} = 6.956\,\text{kmol m}^{-3}$	$C_{B_{1s}} = 5.73\,\text{kmol m}^{-3}$	$C_{B_{1h}} = 4.133\,\text{kmol m}^{-3}$
$C_{DEB_{1c}} = 3.1 \times 10^{-8}\,\text{kmol m}^{-3}$	$C_{DEB_{1s}} = 3.93 \times 10^{-5}\,\text{kmol m}^{-3}$	$C_{DEB_{1h}} = 4.2 \times 10^{-4}\,\text{kmol m}^{-3}$
$C_{E_{1c}} = 1.9576\,\text{kmol m}^{-3}$	$C_{E_{1s}} = 0.954\,\text{kmol m}^{-3}$	$C_{E_{1h}} = 1.1444 \times 10^{-2}\,\text{kmol m}^{-3}$
$C_{EB_{1c}} = 4.24 \times 10^{-2}\,\text{kmol m}^{-3}$	$C_{EB_{1s}} = 0.956\,\text{kmol m}^{-3}$	$C_{EB_{1h}} = 1.636\,\text{kmol m}^{-3}$
$C_{B_{2c}} = 6.435\,\text{kmol m}^{-3}$	$C_{B_{2s}} = 4.23\,\text{kmol m}^{-3}$	$C_{B_{2h}} = 3.055\,\text{kmol m}^{-3}$
$C_{DEB_{2c}} = 2.8 \times 10^{-8}\,\text{kmol m}^{-3}$	$C_{DEB_{2s}} = 2.16 \times 10^{-4}\,\text{kmol m}^{-3}$	$C_{DEB_{2h}} = 4.5 \times 10^{-4}\,\text{kmol m}^{-3}$
$C_{E_{2c}} = 1.961\,\text{kmol m}^{-3}$	$C_{E_{2s}} = 0.171\,\text{kmol m}^{-3}$	$C_{E_{2h}} = 2.5875 \times 10^{-3}\,\text{kmol m}^{-3}$
$C_{EB_{2c}} = 3.87 \times 10^{-2}\,\text{kmol m}^{-3}$	$C_{EB_{2s}} = 1.64\,\text{kmol m}^{-3}$	$C_{EB_{2h}} = 1.361\,\text{kmol m}^{-3}$
$Q_{1c} = -56.4623\,\text{kW}$	$Q_{1s} = -412.254\,\text{kW}$	$Q_{1h} = 221.16\,\text{kW}$
$Q_{2c} = -56.2121\,\text{kW}$	$Q_{2s} = -733.54\,\text{kW}$	$Q_{2h} = 2573.87\,\text{kW}$

'o' subscript stands for inlet/feed stream values.
's' subscript stands for 'desired' steady-state values, i.e., the point representing the origin.
'c' subscript stands for the cold steady-state values.
'h' subscript stands for the hot steady-state values.
Values shown are rounded for readability.

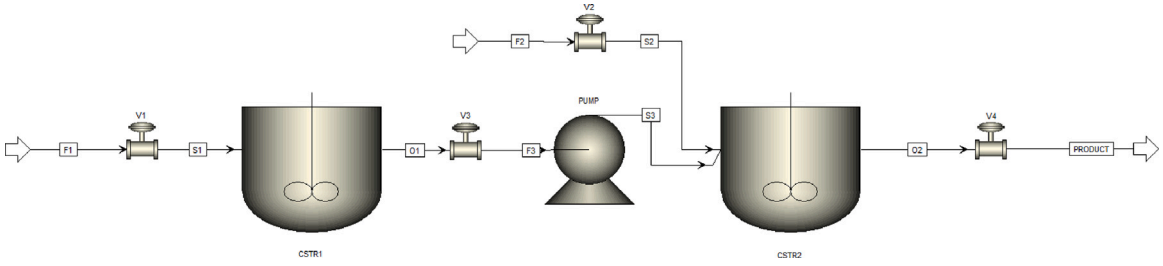

Fig. 2. Aspen Plus Dynamics model flow sheet.

Remark 13. The Temperature PID controllers typically set in Aspen are removed because the control signal for this system is the heating rate to each reactor, which gives control over the temperatures. How exactly this heating is applied is up to the user to decide. In our case, we use Python to calculate the control action and Aspen Plus Dynamics to apply these actions to the system for a fixed time duration (i.e, sample-and-hold control implementation). This is achieved through a custom script using Aspen Plus Dynamics Visual Basic's scripting support to read and write data, such as the system's state measurements, to a folder that Python can also read and write from. This handshake between the two software applications enables states to be calculated by Aspen and controls to be calculated by Python, with the needed information being exchanged in the folder as '.txt' files.

5.3. First-principles model development

The process dynamics model that the LMPC will use is a first-principles model for the considered process that is built using mass and energy balances. The resulting system of ODEs representing the dynamical model of the first CSTR is as follows:

$$\frac{dC_{B_1}}{dt} = \frac{F_1 C_{Bo_1} - F_{out_1} C_{B_1}}{V_1} - r_{1,1} - r_{1,3} \tag{23a}$$

$$\frac{dC_{DEB_1}}{dt} = \frac{-F_{out_1} C_{DEB_1}}{V_1} + r_{1,2} - r_{1,3} \tag{23b}$$

$$\frac{dC_{E_1}}{dt} = \frac{F_1 C_{Eo_1} - F_{out_1} C_{E_1}}{V_1} - r_{1,1} - r_{1,2} \tag{23c}$$

$$\frac{dC_{EB_1}}{dt} = \frac{-F_{out_1} C_{EB_1}}{V_1} + r_{1,1} - r_{1,2} + 2r_{1,3} \tag{23d}$$

$$\frac{dT_1}{dt} = \frac{T_{1_o} F_1 - T_1 F_{out_1}}{V_1} + \sum_{i=1}^{3} \frac{-\Delta H_i}{\rho_1 C_p} r_{1,i} + \frac{Q_1}{\rho_1 C_p V_1} \tag{23e}$$

Similarly, the system of ODEs representing the dynamical model of the second CSTR is as follows:

$$\frac{dC_{B_2}}{dt} = \frac{F_2 C_{Bo_2} + F_{out_1} C_{B_1} - F_{out_2} C_{B_2}}{V_2} - r_{2,1} - r_{2,3} \tag{24a}$$

$$\frac{dC_{DEB_2}}{dt} = \frac{F_{out_1}C_{DEB_1} - F_{out_2}C_{DEB_2}}{V_2} + r_{2,2} - r_{2,3} \tag{24b}$$

$$\frac{dC_{E_2}}{dt} = \frac{F_2C_{Eo_2} + F_{out_1}C_{E_1} - F_{out_2}C_{E_2}}{V_2} - r_{2,1} - r_{2,2} \tag{24c}$$

$$\frac{dC_{EB_2}}{dt} = \frac{F_{out_1}C_{EB_1} - F_{out_2}C_{EB_2}}{V_2} + r_{2,1} - r_{2,2} + 2r_{2,3} \tag{24d}$$

$$\frac{dT_2}{dt} = \frac{T_{2_o}F_2 - T_1F_{out_1} - T_2F_{out_2}}{V_2} + \sum_{i=1}^{3}\frac{-\Delta H_i}{\rho_2 C_p}r_{2,i} + \frac{Q_2}{\rho_2 C_p V_2} \tag{24e}$$

where the 'o' subscript denotes the outlet values and the reaction rates are second-order elementary with non-isothermal reaction rate constants:

$$r_{n,1} = k_1 \exp\left[\frac{-E_1}{RT_n}\right] C_{E_n}C_{B_n} \tag{25a}$$

$$r_{n,2} = k_2 \exp\left[\frac{-E_2}{RT_n}\right] C_{E_n}C_{EB_n} \quad n = 1, 2 \text{ (Reactor Index)} \tag{25b}$$

$$r_{n,3} = k_3 \exp\left[\frac{-E_3}{RT_n}\right] C_{DEB_n}C_{B_n} \tag{25c}$$

Remark 14. The specific heat capacities (C_p) for both reactor volumes are assumed to be the same in the first-principles model of the process. No unique notation is used to distinguish between the two in an attempt to improve the readability of the equations.

5.4. Control problem

The state variables that are considered are the concentrations of these 4 components, as well as the temperature in each of the two reactors. For simplicity, the state variables are defined in deviation variable form with respect to a desired steady state. Steady-state values of these terms are denoted with the 's' subscript. The resulting state vector is of the form:

$$\begin{aligned} x = [&C_{B_1} - C_{B_{1s}}, C_{DEB_1} - C_{DEB_{1s}}, C_{E_1} - C_{E_{1s}}, C_{EB_1} - C_{EB_{1s}}, T_1 - T_{1s}, \\ &C_{B_2} - C_{B_{2s}}, C_{DEB_2} - C_{DEB_{2s}}, C_{E_2} - C_{E_{2s}}, C_{EB_2} - C_{EB_{2s}}, T_2 - T_{2s}] \end{aligned} \tag{26}$$

where the 1 and 2 subscripts represent the reactor numbering. Similarly, the control actions are represented in deviation variable form but only consist of the heating/cooling rate of the individual reactors:

$$u = [u_1, u_2] = [Q_1 - Q_{1s}, Q_2 - Q_{2s}] \tag{27a}$$

$$-2.5 \times 10^3 \le u_1 \le 5 \times 10^3 - Q_{1s} \quad \text{[kW]} \tag{27b}$$

$$-2.5 \times 10^3 \le u_2 \le 5 \times 10^3 - Q_{2s} \quad \text{[kW]} \tag{27c}$$

These bounds were selected after consideration of feasible limits for reactors of the provided scale. Specifically, the lower bounds are made by looking at the cooling rate needed in order to keep the first-principles model of the system at the desired steady state, applying a safety factor of roughly 10, and then selecting the smallest magnitude to be shared. The upper bounds were chosen such that the system should be capable of reaching the desired temperatures in roughly one hour, assuming no reactions. To provide additional safety room, this was approximated as heating the reactor volumes worth of water ($60\,m^3$) by 80 K in 1 h.

The overall objective of this controller is to bring the closed-loop system to a small region around its origin defined by $V < 2$ in the most cost-effective manner as dictated by the quadratic objective function, at which point the state is deemed sufficiently close to the steady state and any further convergence can be achieved through simpler means with minimal impact on the objective. The corresponding Lyapunov function that satisfies the properties discussed in Section 2.3 is a quadratic Lyapunov function of the form $V = x^\top Px$ with P defined as follows:

$$P = \text{diag}(500, 10, 200, 2500, 0.25, 1000, 1, 1000, 500, 0.5) \tag{28}$$

Because of the deviation variable notation, this origin represents the desired unstable steady state. This state was validated as an unstable steady state through testing in Aspen Plus Dynamics. The process is initialized to a point very close to the origin, at which point the simulation is left to run in the Dynamics mode with no further changes to the manipulated inputs. This open-loop simulation was left to run for 6 h worth of simulated process time, after which it was observed that the state evolved away from the origin towards a new steady state, indicating that the origin is an unstable steady state. Using this method, three steady states were chosen: the hot, cold, and desired steady states detailed in Table 1.

All simulations start ($t_k = 0$) at either the hot or cold steady states and are driven towards a small region around the desired steady state via a quadratic cost function.

$$L(\tilde{x}(t), u(t)) = \tilde{x}^\top(t)\,Q_x\tilde{x}(t) + u^\top(t)\,Q_u u(t) \tag{29}$$

where Q_x and Q_u are the weight matrices for the state and control variables, respectively. These matrices are diagonal and are defined as follows:

$$Q_x = \text{diag}(5, 5, 5, 650, 2.5, 25, 2, 25, 100, 6) \tag{30}$$

$$Q_u = \text{diag}\left(5 \times 10^{-5}, 1 \times 10^{-4}\right) \tag{31}$$

5.5. Stability analysis

A set of proportional controllers with gains shown below are used as reference stabilizing controllers for this process:

$$K = \text{diag}(400, 250) \tag{32}$$

The Lyapunov stability constraint is of the form shown in Eq. (5) where $\alpha = 1.5 \times 10^{-5}$.

Remark 15. The primary reason for not using Eq. (4e) as the Lyapunov constraint is due to the high dimensionality of the system. There is no direct way to guarantee that the controller is stabilizing for all potential conditions that the LMPC may encounter within our operating region; therefore, we bypass the risk of a poorly designed reference controller by using the alternate form of the constraint. This alternative form, as discussed before, does not directly need this stabilizing controller to be known. As such, a small value of α is used to counteract any potential over-constraining risks, and the reference controller is reserved as a final failsafe in case the LMPC system fails to solve for any reason.

5.6. Control system parameters

The sampling time for this system is $\Delta = 5$ s. Two LMPCs are considered, one called the long-horizon LMPC with a horizon length of $N = 250$ and one called the short-horizon LMPC with a horizon length of $N = 20$. The only difference between the two LMPCs is the fact that the long-horizon LMPC optimizes the trajectory over 1250 s while the short-horizon LMPC optimizes its trajectory over 100 s. The consequence of this is that the long-horizon LMPC has improved cost optimization; however, the computational burden of this long-horizon results in the computation time exceeding the sampling time, making the long-horizon LMPC infeasible for real-time control. Instead, the short-horizon LMPC is used for real-time control, but at the expense of poorer closed-loop performance.

Numerical integration of the LMPC model is carried out using a Radau collocation method with a collocation degree of 5 and an integration step size of $h_c = 0.05$ s. The LMPC problem is solved using IPOPT with SPRAL as the linear solver and the `mu_strategy` argument set to `adaptive`. Additionally, IPOPT's tolerance and acceptable tolerance are set to 1×10^{-8} and 1×10^{-6}, respectively. The process of building the LMPC is done using the `do_mpc` package, which utilizes 64-bit precision floats due to its use of the `Casadi` package (Wächter and Biegler, 2006; Fiedler et al., 2023; Andersson et al., 2019).

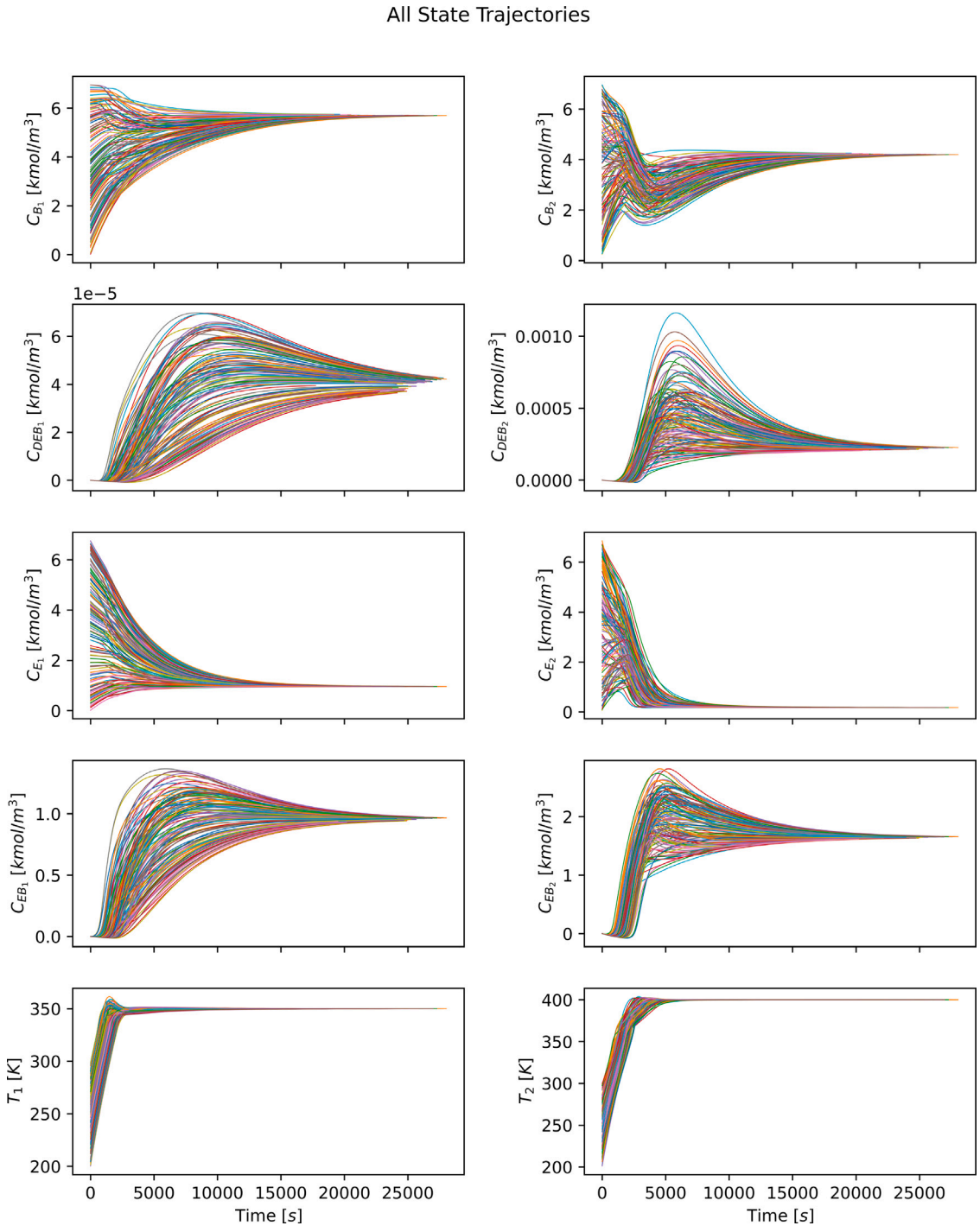

Fig. 3. Generated closed-loop trajectories used for training of the neural network before any filtering.

Remark 16. IPOPT attempts to solve the optimization problem to a defined tolerance, but in some systems, the iterative process gets stuck in an oscillatory loop and fails to converge within the desired tolerances. The acceptable tolerance argument defines a larger tolerance value in an attempt to capture these oscillatory cases and treat them as 'acceptable' solutions. These are solutions that satisfy the constraints, but fail to optimize the cost to the desired tolerances, and the decision on how to handle these solutions is left to the engineer designing the LMPC. This term was included due to the highly nonlinear dynamics of the system as a precaution against cases where excessively tight tolerances might lead to 'failed' solutions that are not truly failures.

5.7. Data generation

Data generation for a system with a 10-dimensional input is a challenge that requires compromise due to the infeasibility of densely sampling the state space. For a system with 10 inputs with n unique values per input, the total number of combinations of these inputs into

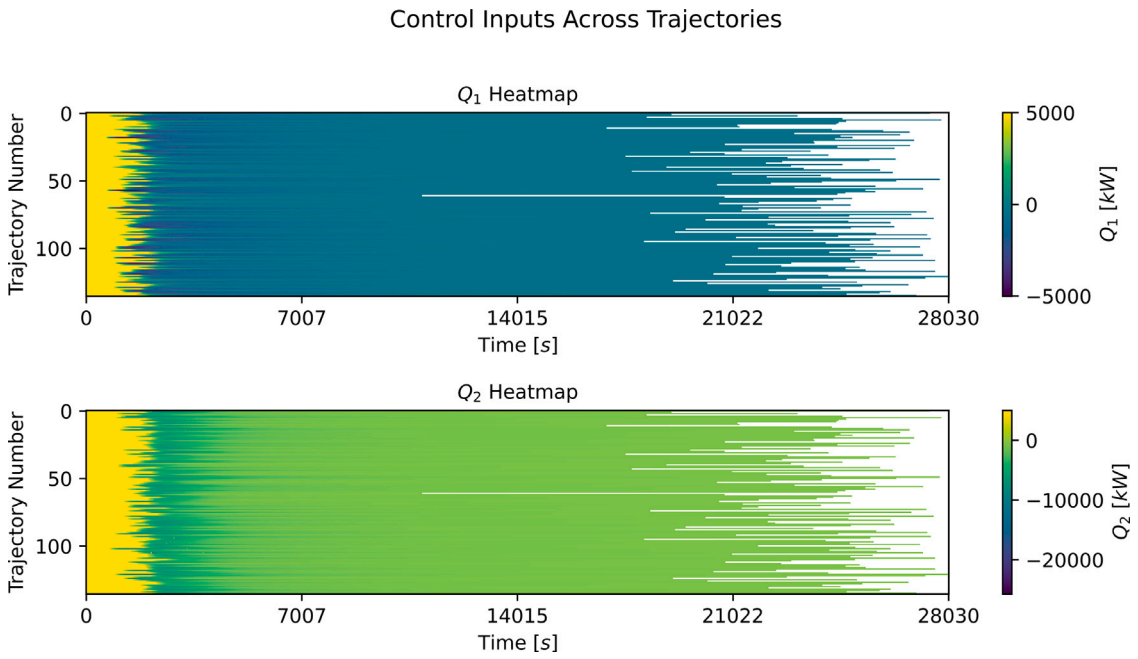

Fig. 4. Heat-map of LMPC control actions per-trajectory used for training of the neural network. All trajectories start at a unique initial point and terminate after satisfying $V \leq 2$.

unique points in the state space is n^{10}. Thus, it can be seen that if a brute force approach were applied in which, given some lower and upper bound for each state variable, an even distribution of $n \geq 2$ points is sampled, then the resulting number of samples to generate and train with grows excessively. For example, if the variables range from 0 to 1, then 5 samples corresponding to steps of size 0.25 will result in a total of 10 million samples to generate and process. Simply doubling the number of samples per variable, in this case, increases the value to 10 billion. Thus, a brute force approach is not considered.

Instead, data is generated in a physics-informed manner. It is known that the process's dynamics function is based on one primary reaction, which involves ethylene and benzene. Additionally, it is known that all reactions are exothermic, irreversible, and that cooling is limited. Thus, data is generated by artificially lowering the dimensionality of the system. Instead of varying 10 state variables, the data is generated by varying 6. These states correspond to the concentration of benzene, ethylene, and the temperature of each CSTR. This gives a range of starting points, after which we simulate the first-principles model in a closed-loop manner using the long-horizon form of the LMPC until the state enters a small level set of the Lyapunov function chosen as $V \leq 2$.

Specifically, the concentrations for ethylene and benzene for both reactors are randomly generated between 0 and 7 with the same units specified in Table 1. The temperature for both reactors is randomly generated between 200 and 300 K. To speed up data generation, the data generation was parallelized, resulting in a total of 5 179 308 samples across 1136 unique trajectories. These trajectories are visualized in Fig. 3 where it can be seen how the state dynamics evolve over time as the system converges to a small region around the origin. Some of the samples are points where the LMPC failed to solve, in which case the failsafe proportional control was applied, but these samples are not used in the final dataset. Because the NN-controller is designed to only take the current state as the input, time-series data is not needed; thus, these trajectories are effectively converted into a list of state points with a control input paired with them. The resulting control actions are difficult to visualize in a readable form, so they are presented as a heatmap in Fig. 4.

The data should ideally be smooth to ensure good model training and diverse enough to cover the region in which we should expect our states to lie. As seen in Fig. 5, the ethylene and benzene distributions are well sampled across various temperatures, but this is not true for the remaining states, as their concentrations are poorly distributed except for points surrounding the origin. The data is visually smooth, as the color-coded control action gradients demonstrate somewhat consistent behaviors, although there is a potential abundance of max-heating control actions. Most notably, there is a severe lack of data for high temperature values, which may pose problems during runtime. A careful look at the histograms for the state and control variables, as shown in Figs. 6 and 7, reveals that despite the bulk of temperature readings being centered around the origin, the concentrations for diethylbenzene and ethylbenzene demonstrate good variety. This suggests that these two concentrations have less impact on the behavior of the trajectory than the two concentrations that were chosen to be randomly sampled. Thus, the logic behind the data generation is supported by the results.

Remark 17. The concentration range is selected to be reasonable based on the values of the feed stream concentrations detailed in Table 1. The temperature range is selected after trial and error. Due to the potential for high initial concentrations of reactive components in a fully exothermic system with limited cooling, it is observed that for high initial temperatures, the LMPC struggles to prevent thermal runaway. Thus, low-temperature initial values are selected as the overshooting risk is eliminated. This reduces the failure rate of the LMPC at the expense of data above the desired steady-state temperatures being effectively nonexistent.

Remark 18. Due to computational limits, despite a sampling time of the long-horizon LMPC being 5 s, the computation time per-solution exceeds this. This is not an issue during data generation, as the closed-loop process is simulated offline here, meaning the effective delay

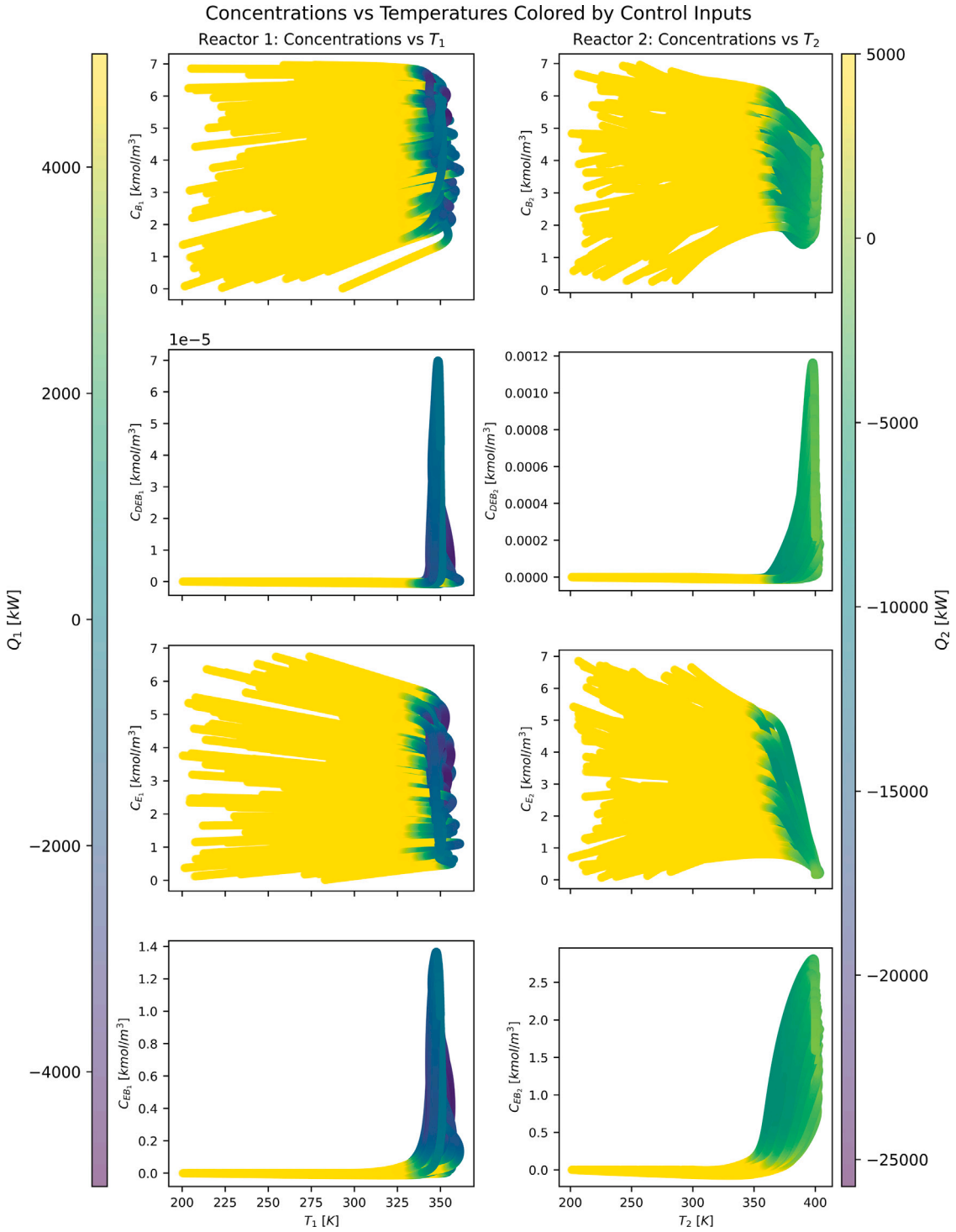

Fig. 5. Sampled state space visualization color-coded by heat input. The temperature of each respective reactor is used as the reference variable to make properties of the 10-dimensional state space somewhat observable.

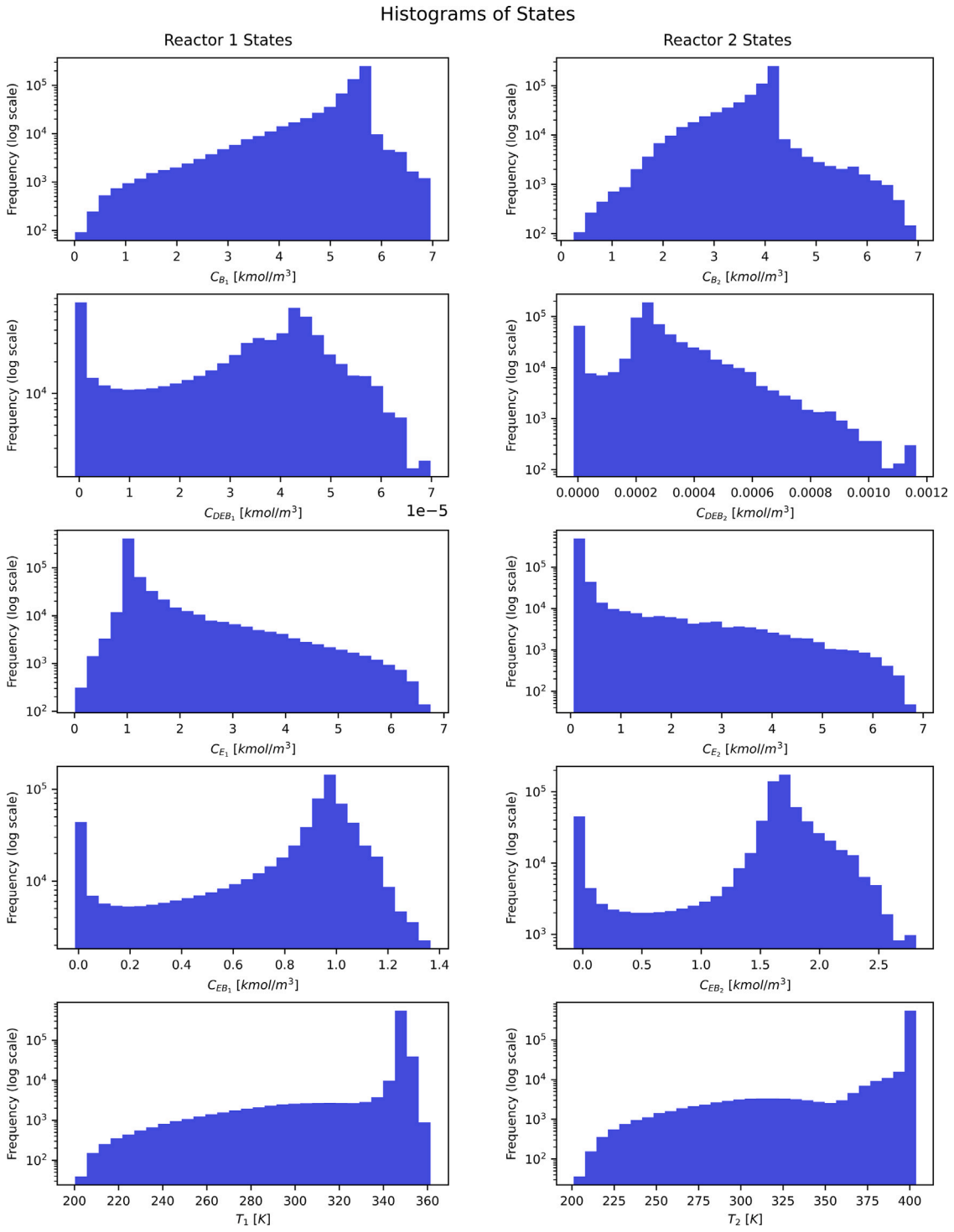

Fig. 6. Log-scale Histogram of the state variables during data generation.

between sensor readings and their resulting optimal control actions does not exist. This is a key benefit of the data-driven approach, as it enables the generation of higher-quality data than what would be feasible to apply in real-time control. Assuming the NN-controller is designed such that the control actions it generates closely match those of the LMPC, the resulting control would be better than the shorter-horizon LMPC counterpart that would have been necessary for real-time control.

Remark 19. $V \leq 2$ is used as the cutoff for the training data. In other words, $\rho_s = 2$ is assumed. This value was chosen to be conservative as a

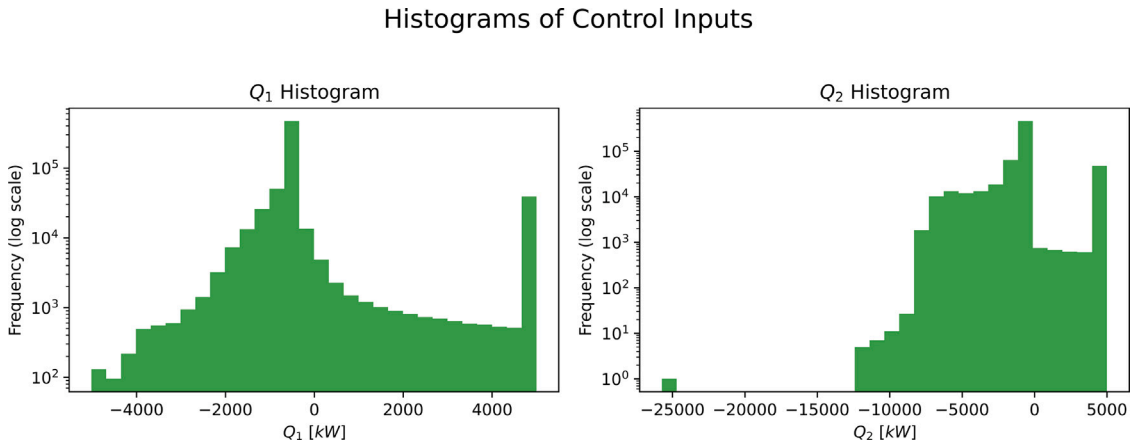

Fig. 7. Log-scale Histogram of the control variables during data generation.

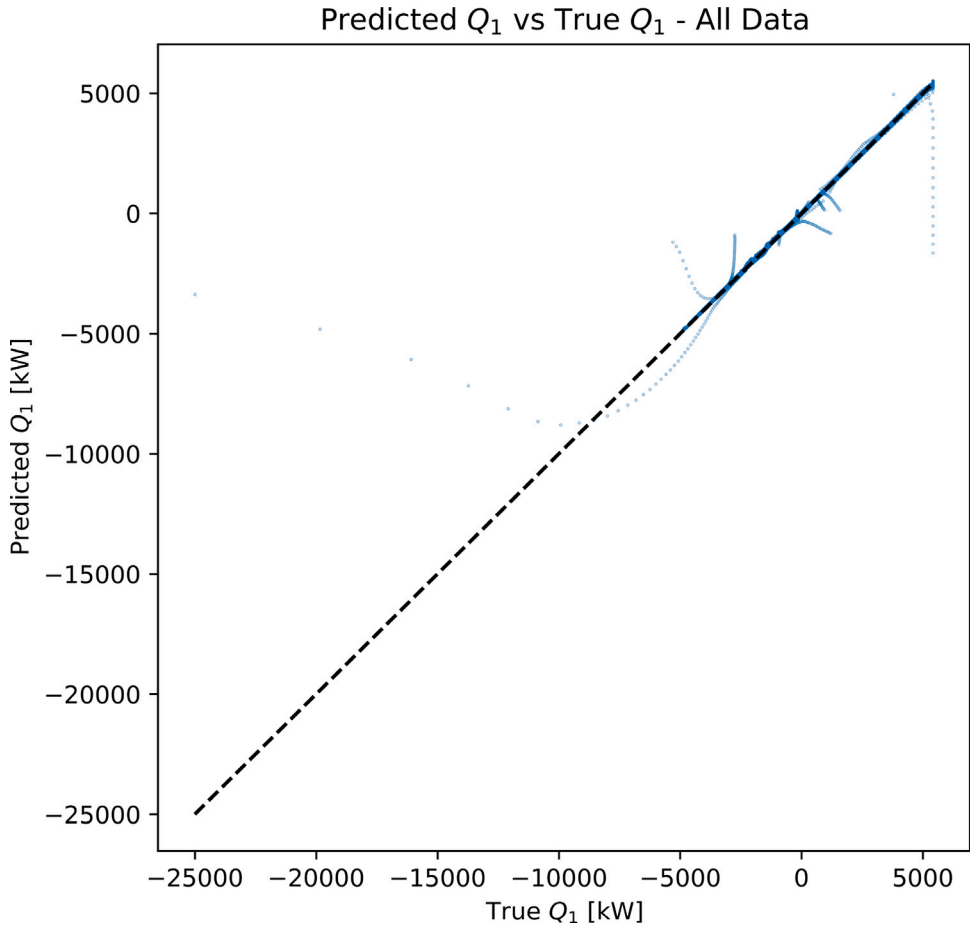

Fig. 8. Predicted vs. actual (true) values for the first control action ($Q_1 - Q_{1s}$).

precaution in case the value of α was chosen to be too small, such that the small region around the origin would interfere with data generation near the origin. To avoid any chance of this occurring, a large cutoff is chosen.

5.8. Neural network training

Before training, the data is filtered. Any point along the closed-loop trajectories that make up the training data where any of the temperatures exceed 100 K deviation from the origin is discarded, as are any duplicate values, NaN values, and values where $V \leq 2$. Temperatures beyond 100 K deviation from the origin are deemed to be outside the reasonable operating region of the system. The remaining filters are used to purge duplicate or meaningless data. The data is then split into Training, Validation, and Testing sets by first splitting the data into a Train/Test set with an 80/20 split and then further splitting this

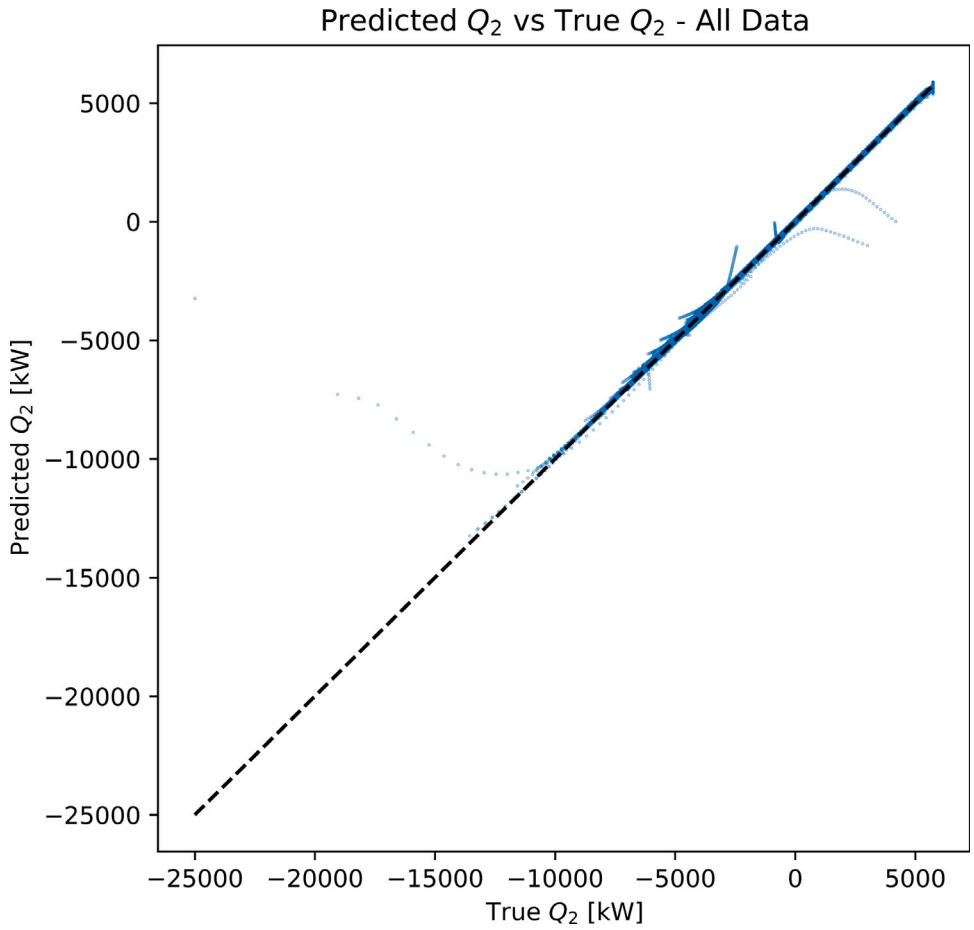

Fig. 9. Predicted vs. actual (true) values for the second control action ($Q_2 - Q_{2_s}$).

Training set into a Train/Validation set with a 90/10 split for a total Train/Val/Test split of 0.72/0.08/0.2.

Some aspects of the Neural Network were fixed during the design process. The loss function was chosen to be Mean Squared Error (MSE), the optimizer was chosen to be AdamW, and the scheduler was chosen to be ReduceLROnPlateau with a factor of 0.9, patience of 2, and `eps` of 0.

The neural network was built and tuned using Optuna to determine the best architecture and set of hyperparameters. The Optuna study was designed to minimize the best-case validation loss while optimizing the following parameters: learning rate, weight decay, architecture type (either feedforward or residual), layer count, size of each individual layer, whether or not to use Batch Normalization or Dropout (on a per-layer basis), and batch sizing. Batch sizing was decided to be 1024 after running a smaller Optuna study on a subset of the data. The remaining terms were decided during a larger Optuna study of 1000 trials. Each study ran for a maximum of 100 epochs with an early stopping patience of 5 with pruning enabled.

In general, the data set (and how representative it is with respect to the operating region) used to train the NN controller as well as the structure and type of the NN used significantly influence the fit of the NN to the LMPC behavior and the resulting closed-loop performance. In the present application, the best NN controller was a Residual Neural Network with 2 hidden layers of size 512 and no BatchNormalization or Dropout. The resulting learning rate was approximately 3.74×10^{-4} with a weight decay of approximately 2.29×10^{-3}. Some analysis was done to check the quality of this model before any decision was made on further training. First, plots on the predicted vs. actual control actions are constructed based on the predicted control action the neural network generates (after clipping) for all data (i.e, including training, validation, and test sets) and are shown in Figs. 8 and 9. From these plots, two things are made clear. First, there is a lack of data where the system needs to apply strong cooling. Second, the neural network's predictions fit extremely well to the generated data. To determine how we can expect these predictions to vary in quality over various temperatures, an additional two plots are made that look at the residuals (actual value - predicted value) with respect to the temperature at which the chosen CSTR is currently operating. As shown in Figs. 10 and 11, there is minimal variation in the NN's prediction error across the temperature ranges. Notably, the noise seems to elevate as the temperature approaches the origin. The y-axis on both plots is seemingly not fit due to the existence of the occasional extremely noisy result. Such points are accounted for by virtue of the fallback control systems, and thus, it was deemed unnecessary to further train this model.

5.9. Closed-loop simulation results

The closed-loop behavior of this controller is analyzed by simulating the process starting at two distinct steady states specified in Table 1. The cold steady state represents a type of real-world case

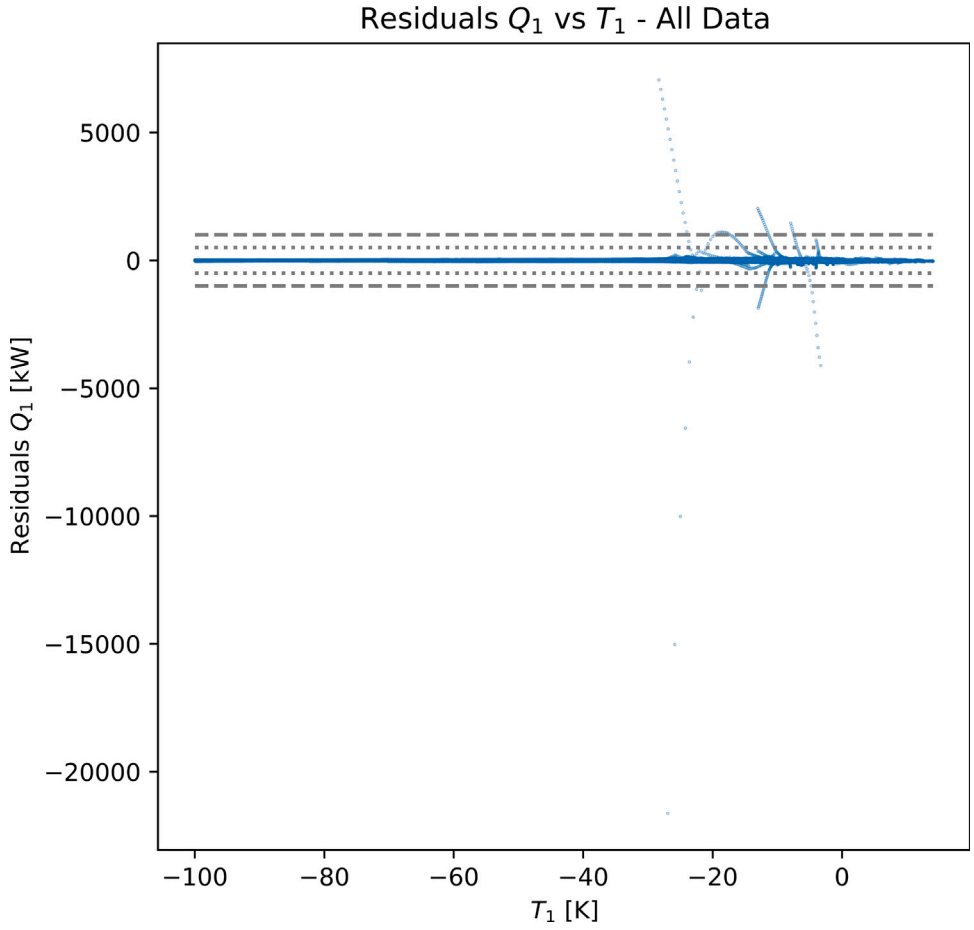

Fig. 10. Residual ($Q_{1,LMPC} - Q_{1,nn}$) vs. temperature values for the first CSTR.

where the data used during training is somewhat comparable to the data experienced during inference. The hot steady state represents the opposite—data that does not exist during training in any form—and exists to demonstrate the architecture's robustness to poor model behavior while maintaining stability guarantees. Starting at these steady states, the closed-loop system is simulated by solving control actions in Python, and solving the state evolution using Aspen Plus Dynamics, which yielded the trajectories shown in Fig. 12. As expected from the training data's results, the cold steady state's trajectory had no issues reaching the small region around the origin without needing any fallback control. Similarly, it is seen that the initialization at the hot steady state caused problems for the controller, requiring a large number of fallback control actions provided by the short-horizon LMPC. Notably, the system still managed to stabilize towards the desired steady state even though there was a case of proportional failsafe control being needed. This demonstrates the framework's ability to maintain stability guarantees even if the trained model poorly fits to data encountered during inference. Additionally, we can look at the results for the long-horizon LMPC for these same initial steady state points as a reference to see how well the NN-controller matches this controller. These results are seen in Fig. 13. Fig. 14 combines the two controllers' trajectories on the same plots for easier comparison. It can be seen that even the LMPC struggles in the hot steady state case due to the difficult dynamics of this example, requiring failsafe proportional control at multiple points. Despite this, the NN-controller is seen to perform worse than the LMPC in the hot steady state case while maintaining near identical performance in the cold steady state case. This demonstrates the NN-controller's ability to generalize to unseen data so long as this data is not exceedingly different from its training data. Prior discussion highlighted the fact that the data generation done for this example failed to sufficiently sample high-temperature points, and it is likely that this lack of data is the primary cause for this poor performance in the hot steady state case. Regardless, the closed-loop results demonstrate the framework's robustness while maintaining stability guarantees.

With regard to the computational benefit of the NN controller compared to the short-horizon LMPC and the long-horizon LMPC, the NN controller demonstrates significant improvements in computation time. Specifically, the long-horizon LMPC took, on average, roughly 15.8 s to solve, which is more than 3 times longer than the sampling time of the process. The short-horizon LMPC took, on average, roughly 0.58 s, which allows it to be applied in real-time due to it being roughly 10 times smaller than the sampling time of the process. The NN controller took, on average, roughly 17×10^{-4} s to solve. This is roughly 300 times faster than the short-horizon LMPC, and roughly 10,000 times faster than the long-horizon LMPC. These results indicate a potential for NN-controllers to demonstrate similar control quality to LMPC without being impacted by the polynomial scaling behavior of NLP frameworks.

Remark 20. The long-horizon LMPC is used to generate Fig. 13 despite it being previously discussed how such an LMPC cannot be

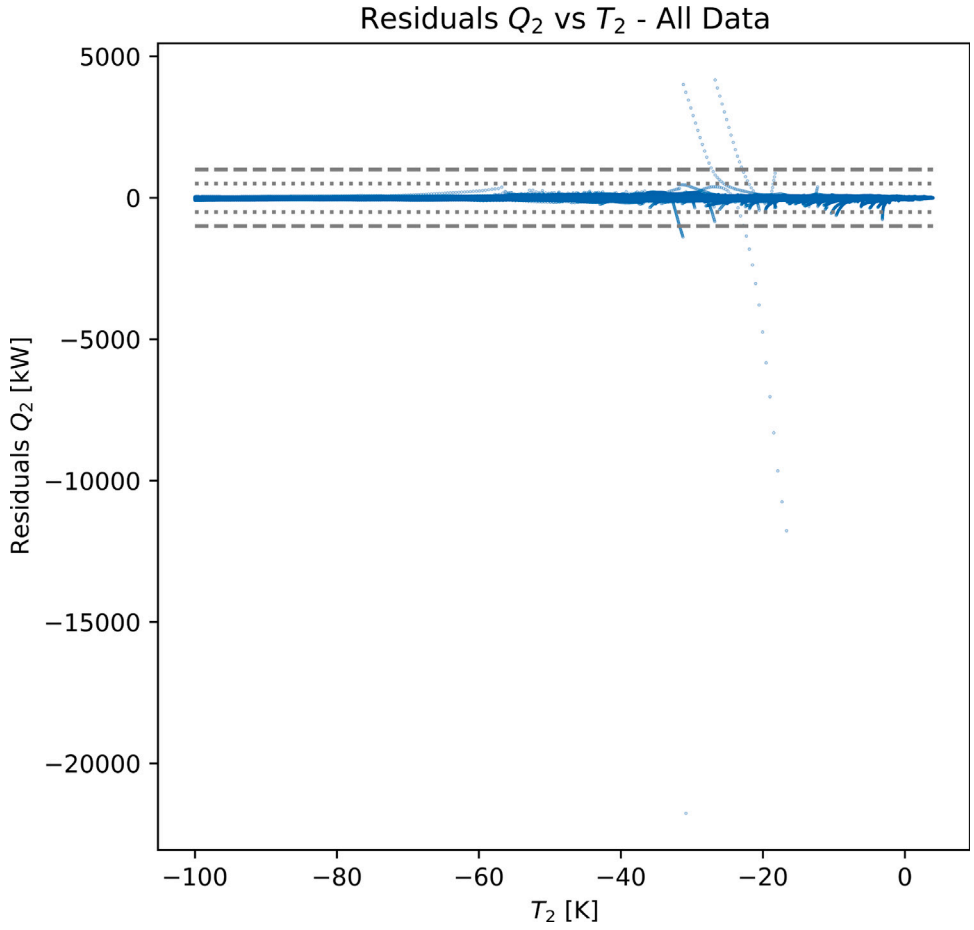

Fig. 11. Residual ($Q_{2,LMPC} - Q_{2,nn}$) vs. temperature values for the first CSTR.

applied in real-time. Since this example is done via simulation, such a limitation is irrelevant, as the Aspen Plus Dynamics model is paused while Python calculates the control action. Practically, the long-horizon LMPC cannot be applied in real-time, and this figure is only provided with the intent to demonstrate that the NN-controller did a good job fitting to the long horizon LMPC control actions, such that it is capable of handling unseen data that is closely related to its training data without significant deviation from the optimal control action trajectory.

Remark 21. Fig. 12 demonstrates a potential problem with the use of fallback control without any form of management for the change in control actions between individual steps via the jittery segments where the enforcer is swapping between NN control and the LMPC fallback rapidly. This is a problem that can cause feasibility issues in practice and must be handled through the fallback controller's design. One such method is to add an additional constraint in the LMPC where the change in the control action between samples is bounded.

Remark 22. With respect to the impact of measurement noise in the training procedure of the NN controller and its closed-loop implementation, it is important to clarify the following. The NN controller, Φ_{nn}, is calculated off-line using long-horizon LMPC solutions, and thus, measurement noise is not present during its training. That said, there is mismatch between the process model used to make forward state evolution predictions within LMPC and the actual process dynamics, and this is why in our application an ASPEN model is used to simulate the process and a first-principles model is used in the LMPC to generate data in order to evaluate the robustness of our approach. Of course, during the implementation of the NN controller, Φ_{nn}, there is measurement noise but its impact is no different compared to impact of noise on another feedback control system in that the feedback control system continues to maintain closed-loop stability in the presence of sufficiently small levels of measurement noise.

6. Conclusion

This work applies a data-driven method for approximating model predictive control in a manner that retains stability guarantees through external enforcement of negative definiteness for a Lyapunov function by means of fallback control to an overall controller that is capable of guaranteeing such constraints. Specifically, the proposed NN controller framework ensures both guaranteed closed-loop stability (due to the use of the fall-back short horizon LMPC) and very good closed-loop performance as the NN controller is trained using data from an LMPC with a long horizon that can only be solved off-line. Data is generated through physics-informed sampling and state simulation using a first-principles-based approximation of the process model. Using a Lyapunov-based model predictive controller (LMPC) that cannot be used in a real-time system due to its computation time exceeding the process's sampling time (owing to the use of a long prediction horizon

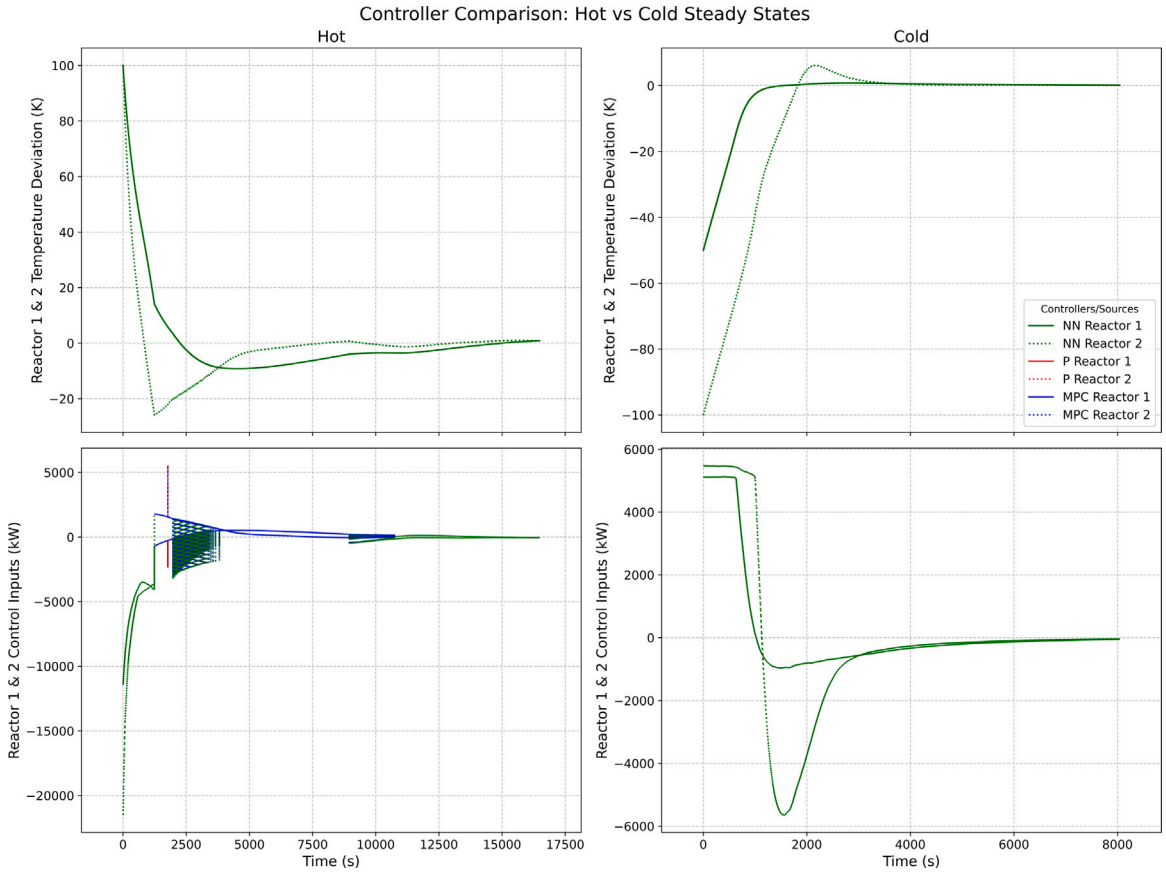

Fig. 12. Closed-loop trajectories of reactor temperatures and heat inputs under NN-controller with fallback control when applied to the Aspen Plus Dynamics model.

to enhance closed-loop performance), state-control action pairs are generated for points along various trajectories. These trajectories are filtered to only include valid LMPC solutions and are split into datasets to train a neural network (NN) to predict the control action based on the current (measured) state values. This NN functions as a replacement for the LMPC. In order to guarantee stability, a shorter horizon form of the LMPC that is capable of running in real-time with reduced performance is used as fallback control, with a reference stabilizing proportional controller used as the final failsafe controller in case the LMPC fails to solve. This pair of controllers is managed by an enforcer that checks the time derivative of a Lyapunov function and determines if it is sufficiently negative to ensure stability based on the desired form of the constraint. If satisfied, the NN is used, but if not, fallback control is used. This framework enables the system to operate with slightly reduced control quality while retaining the stability guaranteed for a process that would otherwise not be feasible to control using long-horizon LMPC. This design is additionally appealing due to the lack of compromises made to ensure stability, which enables broader support for different NN or LMPC designs. The approach was demonstrated using a large-scale nonlinear process modeled and controlled within an Aspen simulation environment. This application allowed addressing numerous practical questions on the implementation of the proposed approach. While the lessons learned from the ASPEN example are specific to this application, we believe they provide insights into how to handle similar issues in the context of other applications.

CRediT authorship contribution statement

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

Acknowledgments

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

Data availability

Data will be made available on request.

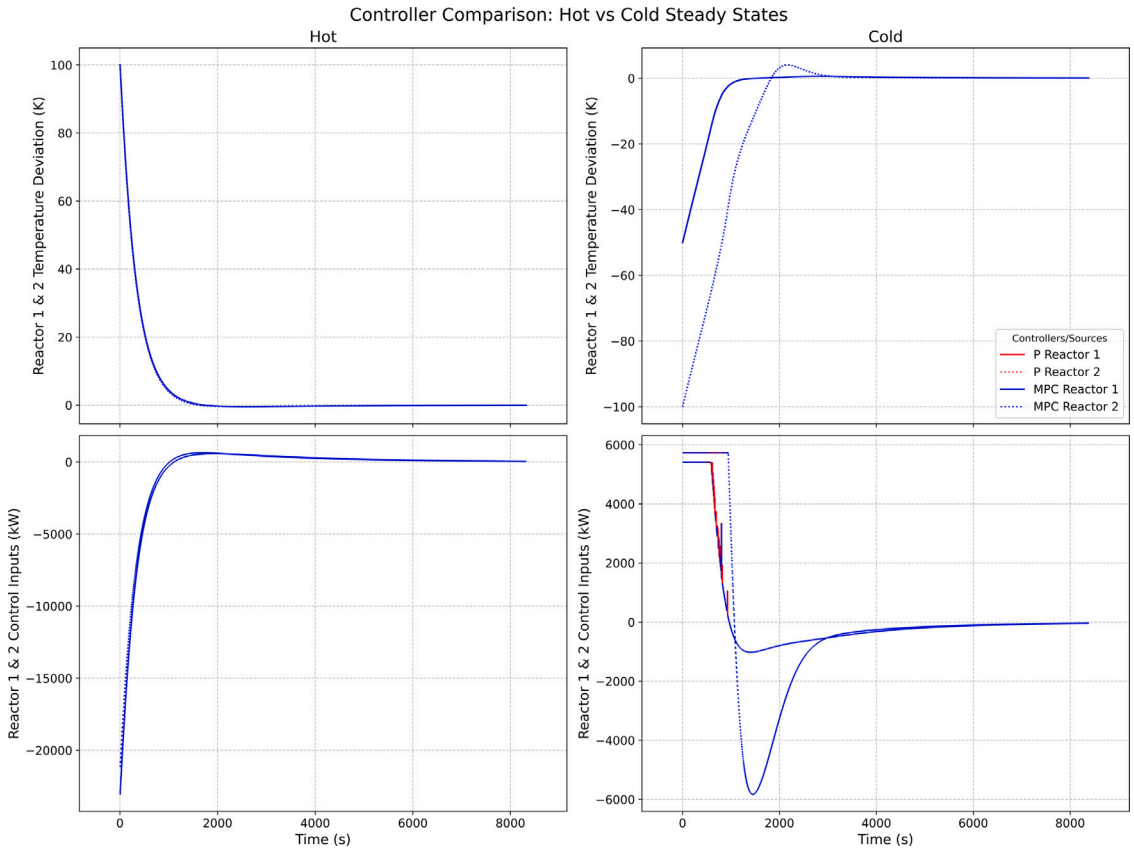

Fig. 13. Closed-loop trajectories of reactor temperatures and heat inputs under LMPC with fallback control when applied to the Aspen Plus Dynamics model.

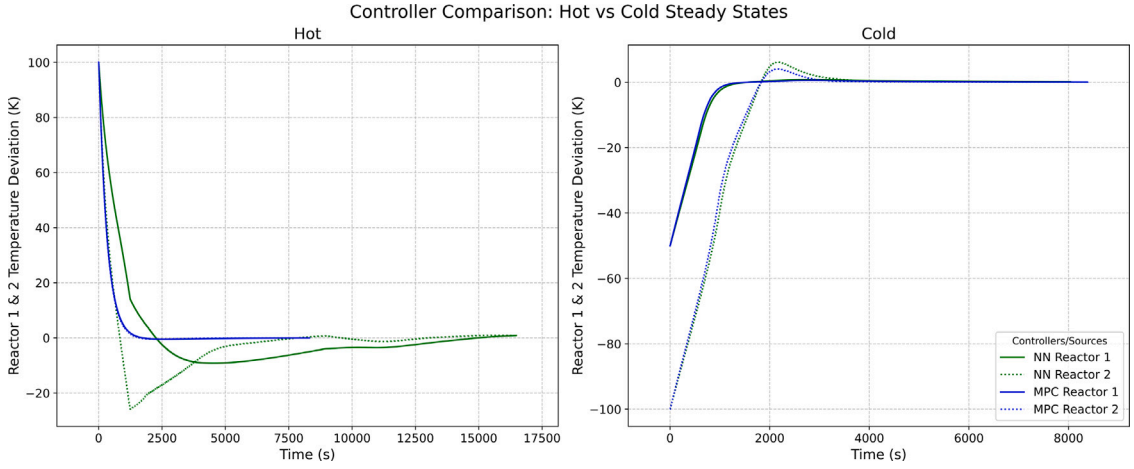

Fig. 14. Closed-loop trajectories of reactor temperatures under the LMPC and the NN-controller with fallback control when applied to the Aspen Plus Dynamics model.

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.

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.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M., 2019. CasADi – A software framework for nonlinear optimization and optimal control. Math. Program. Comput. 11 (1), 1–36.

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 Pap. (ISSN: 2405-8963) 53, 5279–5285.

Fiedler, F., Karg, B., L"uken, L., Brandner, D., Heinlein, M., Brabender, F., Lucia, S., 2023. Do-mpc: Towards FAIR nonlinear and robust model predictive control. Control Eng. Pract. 140, 105676.

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. (ISSN: 0098-3500) 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., 2025a. Enhancing cybersecurity of nonlinear processes via a two-layer control architecture. Digit. Chem. Eng. 15, 100233.

Khodaverdian, A., Gohil, D., Christofides, P.D., 2025b. Neural network implementation of model predictive control with stability guarantees. Digit. Chem. Eng. 16, 100262.

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 Pap. (ISSN: 2405-8963) 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.

Ren, Y.M., Alhajeri, M.S., Luo, J., Chen, S., Abdullah, F., Wu, Z., Christofides, P.D., 2022. A tutorial review of neural network modeling approaches for model predictive control. Comput. Chem. Eng. 165, 107956.

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. (ISSN: 0045-7825) 442, 118030.

Wächter, A., Biegler, L.T., 2006. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. 106 (1), 25–57, (preprint).

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., 2019. Machine-learning-based predictive control of nonlinear processes. Part I: Theory. AIChE J. 65, e16729.

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.