mathematics

MDPI

Article

Machine Learning-Based Model Predictive Control of Two-Time-Scale Systems

Aisha Alnajdi [1], Fahim Abdullah [2], Atharva Suryavanshi [2] and Panagiotis D. Christofides [1,2,*]

[1] Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095, USA; aishaaln2@gmail.com
[2] Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095, USA; fa2@g.ucla.edu (F.A.); atharvasurya99@g.ucla.edu (A.S.)
* Correspondence: pdc@seas.ucla.edu; Tel.: +1-(310)-794-1015

Abstract: In this study, we present a general form of nonlinear two-time-scale systems, where singular perturbation analysis is used to separate the dynamics of the slow and fast subsystems. Machine learning techniques are utilized to approximate the dynamics of both subsystems. Specifically, a recurrent neural network (RNN) and a feedforward neural network (FNN) are used to predict the slow and fast state vectors, respectively. Moreover, we investigate the generalization error bounds for these machine learning models approximating the dynamics of two-time-scale systems. Next, under the assumption that the fast states are asymptotically stable, our focus shifts toward designing a Lyapunov-based model predictive control (LMPC) scheme that exclusively employs the RNN to predict the dynamics of the slow states. Additionally, we derive sufficient conditions to guarantee the closed-loop stability of the system under the sample-and-hold implementation of the controller. A nonlinear chemical process example is used to demonstrate the theory. In particular, two RNN models are constructed: one to model the full two-time-scale system and the other to predict solely the slow state vector. Both models are integrated within the LMPC scheme, and we compare their closed-loop performance while assessing the computational time required to execute the LMPC optimization problem.

Keywords: two-time-scale systems; machine learning; recurrent neural networks; long short-term memory recurrent neural networks; feedforward neural network; process control; model predictive control; nonlinear systems; singular perturbations

MSC: 93B45; 93C10; 93C70

Citation: Alnajdi, A.; Abdullah, F.; Suryavanashi, A.; Christofides, P. D. Machine Learning-Based Model Predictive Control of Two-Time-Scale Systems. *Mathematics* **2023**, *11*, 3827. https://doi.org/10.3390/math11183827

Academic Editors: Teng Huang, Qiong Wang and Yan Pang

Received: 11 August 2023
Revised: 1 September 2023
Accepted: 4 September 2023
Published: 6 September 2023

1. Introduction

Various applications in the field of chemical engineering involve systems that exhibit different time scales. Instances of such systems include biochemical processes, catalytic reactors, and distillation columns. Furthermore, other scientific sectors, such as power electronics, communication networks, and biological systems, also feature systems with dynamics evolving in disparate time scales. Specifically, in the context of chemical engineering, researchers have utilized the method of singular perturbation to decouple two-time-scale systems into reduced-order subsystems, each associated with a distinct time scale (e.g., [1]). This approach simplifies the analysis of the ordinary differential equations governing the system, allowing for the design of a suitable well-conditioned control law that stabilizes the system. In the context of control system design, model predictive control (MPC) is widely recognized as one of the leading and most convenient approaches employed in stabilizing different types of nonlinear systems. MPC is essentially an optimization problem formulated with a well-defined objective function aimed at enhancing the performance of the system while meeting constraints associated with the system's physical structure and closed-loop stability. The fact that MPC accounts for multiple inputs, outputs,

and constraints within its framework is what makes it a suitable choice for designing control systems to stabilize chemical processes. However, in singularly perturbed systems, the presence of distinct time scales can lead to challenges when employing MPC without accounting for the state evolution in the different time scales. Neglecting this aspect can result in degradation of the closed-loop performance, long time consumption, or, in more critical situations, instability and stiffness in the control system due to issues with the controller's effectiveness and the presence of ill-conditioning [2].

However, if the time-scale multiplicity is accounted for, MPC has proven to be an efficient control strategy when it comes to situations involving time-scale separation. For instance, in [3], the design of a composite control system was presented, which included two MPC designs: one for the slow subsystem and one for the fast subsystem. Using stability analysis and the theory of singular perturbations, the authors investigated and guaranteed the closed-loop stability for the full, nonlinear two-time-scale system under Lyapunov-based tracking MPC. To additionally consider process economics, a similar strategy was proposed in [4], which focused on the design of a composite controller involving two MPC problems. Specifically, for the stability of the fast subsystem (and full two-time-scale system as a result), a Lyapunov-based tracking MPC was used for the fast states based on the error between the fast states and the slow manifold using a quadratic cost function, while a Lyapunov-based economic MPC was used for the slow subsystem to optimize the economic considerations and achieve any other desirable closed-loop stability properties using a non-quadratic, economic cost function. The authors of [3,4] focused on first-principles-based controllers, and in [5], a two-time-scale system was separated into slow and fast subsystems using a singular perturbation strategy and then modeled using only data from the system via the method of sparse identification for nonlinear dynamics. Subsequently, an MPC was designed based on the reduced-order slow subsystem modeled using sparse identification, and closed-loop stability guarantees were derived. Furthermore, numerical simulations were used to demonstrate the reduced computational cost of the reduced-order sparse-identified model.

Due to the vast advancements in the chemical industry and the multitude of complex reactions occurring in real-life chemical process applications, engineers encounter challenges when constructing first-principles models for these chemical processes. As a result, in recent years, machine learning (ML) models have been utilized to approximate the dynamics of chemical processes. These models are highly beneficial when designed carefully, as they offer a reliable and efficient alternative to classical first-principles models and can then be incorporated into MPC frameworks. Many works have been conducted in this field. For instance, the authors of [6] offered essential insights and a fundamental theory regarding the integration of machine learning techniques into MPC schemes. Their work focused on designing a recurrent neural network (RNN) model to approximate the nominal nonlinear system and integrated the model within an MPC framework to stabilize the system. In [7], polynomial nonlinear autoregressive with exogenous inputs (NARX) models were used to build nonlinear MPC, with the relevant polynomial terms to retain selected via sparse regression. The proposed MPC was applied to a multi-input multi-output chemical reactor, and an algebraic modeling language was used to reduce computational time.

Another research direction termed "approximate MPC" involves the replacement of the MPC optimization problem with a closed-form expression that approximates the MPC control actions without having to solve an optimization problem. This is meant to address the computational challenges inherent in classical MPC at the expense of lower accuracy and generalizability. In [8], an approximate MPC was designed for heating, ventilation, and air-conditioning (HVAC) systems due to the stringent computational resources available for building control systems. Specifically, the MPC optimization problem was imitated using a recurrent neural network with a structure of a nonlinear autoregressive network with exogenous inputs trained with data from 30 days of operation in a closed loop under MPC. While both MPC and approximate MPC greatly reduced the cooling consumption of

the HVAC system when tested, approximate MPC solved for the control actions over 100 times faster throughout the testing period, with only a 12% degradation in performance compared to MPC. An alternate research direction to reduce the computational burden of MPC is the design of "explicit MPC", in which the optimal control actions are computed offline using multiparametric programming methods, allowing the online component of MPC to only have to search a table of linear gains and compute a single function evaluation. A summary of studies investigating explicit MPC is provided in [9]. Machine learning has also been investigated in the context of explicit MPC. In [10], for example, an explicit control law was learned for discrete-time linear time-invariant systems using machine learning. Specifically, the key challenge addressed was that of high dimensionality, which has been attempted to be solved in the existing literature by identifying suboptimal polytope partitions of the state space and designing control laws based on such regions. Ref. [10] focused on tackling high dimensionality in a similar manner but by using reinforcement learning and additionally investigated constraint satisfaction and feasibility guarantees for explicit MPC. Specifically, by using deep neural networks with rectified linear units as the activation function and a modified policy gradient algorithm based on prior knowledge, the objectives were met and demonstrated via three numerical examples of varying levels of dimensionality and complexity. A potential limitation of the approximate and explicit MPC approaches is that they are practically applicable to only smaller-scale systems, even though they are applicable to systems with very fast sampling times [9]. However, for general nonlinear systems and nonlinear MPC, due to the highly nonconvex optimization problem, methods such as the use of suboptimal polytopes [10] may not be readily extended to the nonlinear case.

While ML techniques have shown great success when incorporating them into MPC schemes, the practical application of machine learning-based MPC schemes to real-life processes poses significant challenges. The fact that the ML model is developed using a finite number of data samples makes it challenging to assess the accuracy of the model when considering generalized scenarios. Therefore, researchers have developed methods to quantify the error of the process model used in MPC and study its effects on closed-loop performance and stability. For example, ref. [11] proposed the use of Bayesian neural networks (BNN) to quantify the plant-model mismatch and subsequently create adaptive scenarios in real time for scenario-based MPC using the BNN. Based on the simulation results from a cold atmospheric plasma system, the proposed approach outperformed scenario-based MPC without adaptive scenarios and adaptive scenario-based MPC with Gaussian process regression. On more theoretical fronts, researchers have also studied the concept of generalization error bounds, which are crucial to study to ensure the construction of reliable models that are capable of performing accurate predictions on unseen or new data. Specifically, a low generalization error bound guarantees the effectiveness and high performance of the ML model and, consequently, the MPC scheme utilizing the designed ML model. To address this, ref. [12] derived a generalization error bound for a fully connected RNN (FCRNN) model, discussed the main factors that affect the bound, and integrated the FCRNN model into an MPC scheme to stabilize a nonlinear process. Moreover, the authors of [13] focused on deriving a generalization error bound for a feedforward neural network (FNN) that was used to construct a control Lyapunov-barrier function. The authors of [14] studied the generalization error bound for both a partially connected RNN (PCRNN) and an LSTM-RNN and subsequently carried out a comparison study between the generalization performance of an FCRNN and a PCRNN. Additionally, all the aforementioned works incorporated the designed machine learning models into Lyapunov-based MPC schemes to demonstrate the ability of the LMPCs to stabilize a chemical process. To the best of our knowledge, the application of generalization error bounds to neural networks modeling two-time-scale systems has not yet been investigated.

In light of the above considerations, in this article, we use the theory of statistical machine learning to construct the generalization error bounds for neural networks modeling two-time-scale systems. Moreover, the computational time for a neural network-based

MPC is known to be an important practical consideration. However, the computational time for an RNN-based MPC with and without decomposition into lower-order subsystems has not yet been studied. Therefore, we introduce two LMPC frameworks, one designed based on an RNN model that predicts the slow state vector and the other designed based on an RNN model that models the full two-time-scale system, and compare the two LMPC schemes in terms of the closed-loop stability and computational time. The rest of this article is structured as follows. Section 2 introduces the essential preliminaries, as well as the general class of two-time-scale systems considered in this work. Section 3 discusses the generalization error bounds of neural networks modeling two-time-scale systems. In Section 4, we present a machine learning-based Lyapunov-based MPC using an RNN that predicts the slow state vector, followed by a closed-loop stability analysis and the corresponding results. In Section 5, a chemical process example is used to demonstrate the effectiveness of the designed controller. Section 6 summarizes the key findings obtained in this study.

2. Preliminaries

2.1. Notations

The Euclidean norm of a vector is denoted by $|\cdot|$. The transpose of the vector x is given by x^T. Given a matrix B, its Frobenius norm is denoted as $\|B\|_F$. Moreover, the infinity norm of the 1-norms of the columns of B is denoted as $\|B\|_{1,\infty} = \max_j \sum_i |B_{i,j}|$. The expression $L_F V(x)$ denotes the standard Lie derivative $L_F V(x) := \frac{\partial V(x)}{x} f(x)$. Set subtraction is denoted by '\', i.e., $A \backslash B := \{x \in \mathbb{R}^n | x \in A, x \notin B\}$. A function $f(x)$ is of class $\mathcal{C}^1$ if it is continuously differentiable in its domain. A continuous function $\alpha : [0, a) \to [0, \infty)$ belongs to class $\mathcal{K}$ if it is strictly increasing and is zero only when evaluated at zero. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be L-Lipschitz continuous if there exists $L \geq 0$, such that $|f(a) - f(b)| \leq L|a - b|$ for all $a, b \in \mathbb{R}^n$. A continuous function $\beta : [0, a) \times [0, \infty) \to [0, \infty)$ belongs to class $\mathcal{KL}$ if, for each constant value of t, the function $\beta(\cdot, t)$ is of class $\mathcal{K}$, and for each constant value of s, the function $\beta(s, \cdot)$ is decreasing and approaches zero as $s \to \infty$. The probability that the event A will occur is denoted as $\mathbb{P}(A)$. Additionally, the expected value of a random variable X is denoted as $\mathbb{E}[X]$.

2.2. Class of Systems

The following family of ordinary differential equations describes the general class of two-time-scale continuous-time nonlinear systems with k states considered in this work:

$$\dot{x} = f_1(x, z, u, \epsilon) \tag{1a}$$
$$\epsilon \dot{z} = f_2(x, z, \epsilon) \tag{1b}$$

where $x \in \mathbb{R}^n$ is the slow state vector, $z \in \mathbb{R}^p$ is the fast state vector, and $n + p = k$. $u \in \mathbb{R}^{q_1}$ is the bounded manipulated input vector. The input vector u is constrained by $u \in U := \{|u_i| \leq u_{1_{\max,i}}, i = 1, ..., q_1\}$. We assume the vector functions $f_1(x, z, u, \epsilon)$ and $f_2(x, z, \epsilon)$ to be sufficiently smooth vector functions in $\mathbb{R}^n$ and $\mathbb{R}^p$, respectively. Moreover, we assume that the closed-loop stability region of the nonlinear system in Equation (1) is defined by the region Ω_{ρ_F}, where $\Omega_{\rho_F} := \{x \in D | V(x, z) \leq \rho_F\}$, where $\rho_F > 0$ and D is an open neighborhood around the origin. The speed ratio of the slow to the fast dynamics of the system is represented by the small positive parameter ϵ. In Equation (1b), we observe that the speed ratio ϵ pre-multiplies the derivative of the fast state vector z, which enables us to utilize a well-known strategy called "reduced-order modeling" via singular perturbations. In this approach, we will be able to decompose the system in Equation (1) into two different reduced-order subsystems. We follow the strategy thoroughly illustrated in [2].

The following equations are obtained by setting $\epsilon = 0$ in Equation (1):

$$\dot{\bar{x}} = f_1(\bar{x}, \bar{z}, \bar{u}, 0) \tag{2a}$$
$$0 = f_2(\bar{x}, \bar{z}, 0) \tag{2b}$$

where $\bar{x}$ and $\bar{z}$ are the slow state vector and the fast state vector associated with the system in Equation (1) under the case where $\epsilon = 0$, respectively. Additionally, the following assumption is considered to be fundamental in the theory of singular perturbations.

Assumption 1. *Equation (2b) has an isolated solution, which is determined by*

$$\bar{z} = \bar{f}_2(\bar{x}) \tag{3}$$

where $\bar{z}$ is a quasi-steady state for the fast state z, and the function $\bar{f}_2 : \mathbb{R}^n \to \mathbb{R}^p$ is continuously differentiable.

We substitute Equation (3) into Equation (2a) to obtain the reduced-order slow subsystem,

$$\dot{\bar{x}} = f_1(\bar{x}, \bar{f}_2(\bar{x}), \bar{u}, 0) \tag{4}$$

Furthermore, to obtain the dynamics of the reduced-order fast subsystem, we introduce a fast timescale $\tau = t/\epsilon$ and a new coordinate $\bar{\bar{z}} = z - \bar{z}$ to re-write Equation (1b) with respect to the newly introduced variables $\bar{\bar{z}}$ and τ. Then, the fast subsystem can be expressed as the derivative of $\bar{\bar{z}}$ with respect to τ while setting $\epsilon = 0$,

$$\frac{\mathrm{d}\bar{\bar{z}}}{\mathrm{d}\tau} = f_2(\bar{x}, \bar{\bar{z}} + \bar{f}_2(\bar{x}), 0) \tag{5}$$

Starting at t_0, the initial conditions of the state vectors x and z are given by the vectors x_0 and z_0, respectively. Based on Equations (1)–(5), and by utilizing the basic theoretical concepts of two-time-scale systems in [2], the following equation will hold for all $t \in [t_p, T]$, where $t_p > t_0$

$$z(t) = \bar{z}(t) + \mathcal{O}(\epsilon). \tag{6}$$

where $\bar{z}(t)$ reflects the slow transient of z, and $\mathcal{O}(\epsilon)$ is an error of order epsilon. A variable, $z(t)$, is $\mathcal{O}(\epsilon)$, where ϵ is a positive constant if there exists a positive constant $\bar{k}$ (independent of ϵ), such that $|z(t)| \leq \bar{k}\epsilon$. Furthermore, if we were to consider the time interval where $t \in [t_0, T]$, Equation (6) would be slightly modified, such that the approximation of the state z is given by the following:

$$z(t) = \bar{z}(t) + \bar{\bar{z}}(t) + \mathcal{O}(\epsilon), \tag{7}$$

where $\bar{z}(t)$ and $\bar{\bar{z}}(t)$ are the slow and fast transients of z, respectively. Additionally, the approximation of state x by $\bar{x}$ is given by the following equation for all $t \in [t_0, T]$

$$x(t) = \bar{x}(t) + \mathcal{O}(\epsilon). \tag{8}$$

The fast subsystem in Equation (5) needs to satisfy certain stability properties for the above closeness of the solution estimates to hold true, which are described by the following assumption.

Assumption 2. *The equilibrium $\bar{\bar{z}}(\tau) = 0$ in Equation (5) demonstrates uniform asymptotic stability in x_0 and t_0. In addition, $z_0 - \bar{z}(t_0)$ resides within its domain of attraction. As a result, for all $\tau \geq 0$, $\bar{\bar{z}}(\tau)$ exists.*

If Assumption 2 holds true, then

$$\lim_{\tau \to \infty} \bar{\bar{z}}(\tau) = 0, \tag{9}$$

This implies that, at some time $t_p > t_0$, z will approach close to its quasi-steady state $\bar{z}$. The local stability of the equilibrium of the fast subsystem holds if the following verifiable condition holds.

Assumption 3. *All the eigenvalues of* $\frac{\partial f_2}{\partial z}$*, computed for* $\epsilon = 0$*, along* $\bar{x}(t), \bar{z}(t)$*, exhibit real parts less than a constant negative value, i.e.,*

$$\Re\, \lambda \left\{ \frac{\partial f_2}{\partial z} \right\} \leq -c < 0. \tag{10}$$

Utilizing the aforementioned assumptions, we establish the well-known "Tikhonov's theorem" in the following theorem.

Theorem 1 (c.f. Theorem 3.1 in [2])**.** *Consider that Assumptions 2 and 3 hold true. Then, for all* $t \in [t_0, T]$*, Equations (7) and (8) are valid. In addition, there exists a specific time instant* $t_p \geq t_0$*, such that Equation (6) is valid for all* $t \in [t_p, T]$*.*

Several sources such as [15] are useful for analyzing slightly varied formulations of Theorem 1 and reviewing its proof. At this point, we have investigated several essential concepts of two-time-scale systems. We will assume that the error between x and $\bar{x}$ is not greater than $\mathcal{O}(\epsilon)$. This assumption allows us to simplify the analysis by approximating $\bar{x}$ as x. Similarly, the same principle applies to z and $\bar{z}$, with our focus on capturing the slow transient of the fast state dynamics z, that is, $\bar{z}$. Therefore, based on this assumption, we express the reduced-order slow subsystem in Equation (4) as the following throughout this manuscript,

$$\dot{x} = F(x, u) := f(x) + g(x)u, \qquad x(t_0) = x_0, \tag{11}$$

where $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth vector functions of dimensions $n \times 1$ and $n \times q_1$, respectively. Without loss of generality, we assume that the initial time $t_0 = 0$ and that $f(0) = 0$; hence, the steady state of the nonlinear system in Equation (11) is the origin. Finally, we will assume that the fast subsystem is globally asymptotically stable, which is necessary for establishing the stability of the closed-loop system under model predictive control using machine learning models in Section 4.

Assumption 4. *The origin of the closed-loop fast subsystem described by Equation (5) exhibits global asymptotic stability uniformly in* x*. This implies that there exists a function* $\beta_{\bar{z}}$ *of class* $\mathcal{KL}$*, such that for any* $\bar{\bar{z}}(0) \in \mathbb{R}^p$*,*

$$|\bar{\bar{z}}(t)| \leq \beta_{\bar{z}}\left(|\bar{\bar{z}}(0)|, \frac{t}{\epsilon} \right) \qquad \forall t \geq 0 \tag{12}$$

2.3. Stabilizability Assumption via Control Lyapunov Function

Regarding the dynamics of the slow subsystem described in Equation (11), we assume that there exists a locally Lipschitz feedback controller, $\Phi(x) \in U$, which renders the origin of the slow subsystem in Equation (11) asymptotically stable, i.e., there exists a $\mathcal{C}^1$ Lyapunov function $V(x)$ that is continuously differentiable and meets the following set of inequalities:

$$a_1(|x|) \leq V(x) \leq a_2(|x|), \tag{13a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \leq -a_3(|x|), \tag{13b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq a_4(|x|) \tag{13c}$$

where a_1, a_2, a_3, and a_4 are class $\mathcal{K}$ functions for all $x \in \mathbb{R}^n \subset D$. Additionally, given that the system $F(x, u)$ has a Lipschitz property and the input u is constrained by the set U, then

there exist positive constants M, L_x, and $L_x^{'}$, such that the subsequent inequalities hold for all $x, x' \in D$, and $u \in U$:

$$|F(x,u)| \leq M \tag{14a}$$

$$|F(x,u) - F(x',u)| \leq L_x|x - x'| \tag{14b}$$

$$\left|\frac{\partial V(x)}{\partial x}F(x,u) - \frac{\partial V(x')}{\partial x}F(x',u)\right| \leq L_x^{'}|x - x'| \tag{14c}$$

Moreover, the region $\Omega_\rho := \{x \in D | V(x) \leq \rho\}, \rho > 0$, is defined as the stability region for the slow subsystem in Equation (11).

Now we discuss the incorporation of machine learning models, particularly neural networks, in process systems engineering. The first step in developing any type of neural network model is generating a comprehensive data set that effectively captures the input–output relation. This data set is used to train the network and enables it to learn patterns present in the data and make accurate predictions. To generate a data set that captures the dynamics of the nonlinear system in Equation (1) within the region Ω_ρ, we follow the data generation technique described in [6]. The first step is to carry out various open-loop simulations of the nonlinear system in Equation (1), covering a wide range of initial conditions (i.e., $x_0 \in \Omega_\rho$ and $z_0 \in \Omega_\rho$) and valid input signals under sample-and-hold implementation (i.e., the input is applied to the system as piecewise constant functions, $u = u(t_k) \in U, \forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$, Δ is the sampling period). To integrate the nonlinear system in Equation (1), the well-established forward Euler approach is used with a sufficiently small integration time step $h_c \ll \Delta$. As a result, an extensive data set containing the dynamics of the system is generated, which is then utilized to train a neural network employing the Keras library.

2.4. Recurrent Neural Networks

Consider designing a recurrent neural network (RNN) model that predicts only the slow state vector x in Equation (1), or in other words, to approximate the dynamics of the slow subsystem in Equation (11). The network is trained using m data points $(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$ of T-time length, where $i = 1, ..., m$ and $t = 1, ..., T$. $\mathbf{x}_{i,t} \in \mathbb{R}^{d_x}$ and $\mathbf{y}_{i,t} \in \mathbb{R}^{d_y}$ are the RNN input and output, respectively. Additionally, d_x and d_y are the dimensions of the RNN input and output, respectively. It is worth mentioning that the notations used for the RNN inputs and outputs are represented in bold to establish a clear distinction between the RNN notations and the notations used to describe the nonlinear two-time-scale system in Equation (1). The RNN input $\mathbf{x}_{i,t}$ comprises the current slow state measurement, along with the manipulated inputs applied through the time steps $t = 1, ..., T$, where the manipulated inputs are generated randomly within the set U. The RNN output $\mathbf{y}_{i,t}$ comprises the slow state predicted through the time steps $t = 1, ..., T$. As a result, the RNN model is designed to make predictions of the slow states over one sampling period with $T = \frac{\Delta}{h_c}$ time steps. The data set utilized to develop the RNN model is constructed by generating m independent data sequences obtained from an underlying distribution over $\mathbb{R}^{d_x \times T} \times \mathbb{R}^{d_y \times T}$.

With the aim of simplification, we introduce a single-hidden-layer RNN model, given that $\mathbf{h}_i \in \mathbb{R}^{d_h}$ are the hidden states and can be evaluated as follows:

$$\mathbf{h}_{i,t} = \sigma_h(U\mathbf{h}_{i,t-1} + W\mathbf{x}_{i,t}) \tag{15}$$

where the element-wise nonlinear activation function is denoted by σ_h. $U \in \mathbb{R}^{d_h \times d_h}$ is the weight matrix associated with the hidden states, whereas $W \in \mathbb{R}^{d_h \times d_x}$ is the weight matrix associated with the input vector. Furthermore, the following equation evaluates the output layer $y_{i,t}$ of the RNN model:

$$\mathbf{y}_{i,t} = \sigma_y(\mathcal{V}\mathbf{h}_{i,t}) \tag{16}$$

where the element-wise activation function of the output layer is denoted by σ_y, and $\mathcal{V} \in \mathbb{R}^{d_y \times d_h}$ is the weight matrix associated with the output layer. In this particular appli-

cation, the input of the RNN model will be the current slow state measurement, $x \in \mathbb{R}^n$, as well as the manipulated input $u \in \mathbb{R}^{q_1}$ for the next sampling period. The output of the RNN model will be the predicted slow states x for at least one sampling period ahead. The basic structure of the RNN is represented in Figure 1.

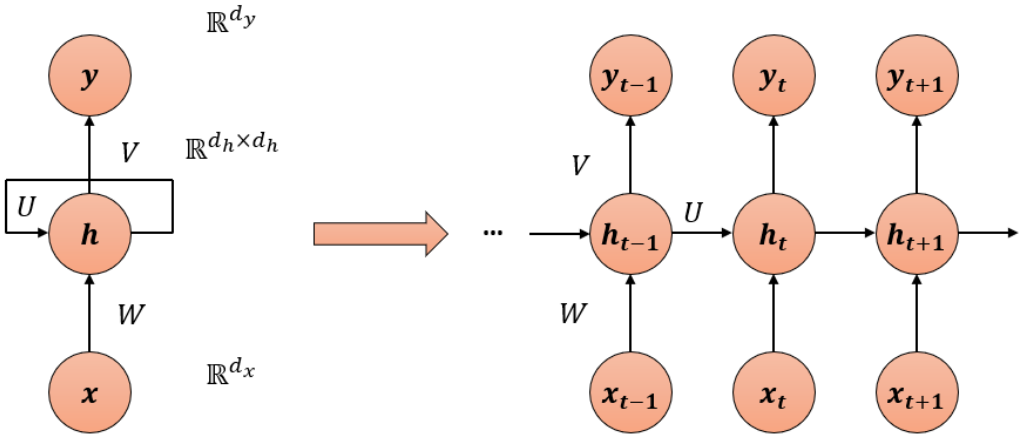

Figure 1. Recurrent neural network structure.

Let $\check{\mathbf{y}}$ be the predicted value and $\mathbf{y}$ be the actual value. We can then define the loss function $L(\mathbf{y}, \check{\mathbf{y}})$ that computes the squared difference between the actual and predicted values. More precisely, we will consider the L_2 (i.e., mean squared error) loss function. Without loss of generality, we establish the following standard assumptions that are required for the study of the generalization error bounds and are reviewed for completeness:

Assumption 5. *The inputs of the RNN are bounded (i.e.,* $\|\mathbf{x}_{i,t}\| \leq B_X$ *for all* $i = 1, ..., m$ *and* $t = 1, ..., T$*).*

Assumption 6. *The Frobenius norms of the weight matrices are bounded (i.e.,* $\|\mathcal{V}\|_F \leq B_{\mathcal{V},F}$, $\|W\|_F \leq B_{W,F}$, $\|U\|_F \leq B_{U,F}$*).*

Assumption 7. σ_h *is a positive homogeneous and 1-Lipschitz continuous nonlinear activation function (i.e.,* $\sigma_h(\alpha z) = \alpha\sigma_h(z)$ *holds* $\forall \alpha \geq 0$ *and* $z \in \mathbb{R}$*).*

Assumption 8. *The data sets used for training, testing, and validation are drawn from the same distribution.*

Remark 1. *Although the RNN model is developed to approximate the dynamics of the slow subsystem described by Equation (11), we emphasize the fact that the data used to train this RNN model are constructed using the nonlinear two-time-scale system in Equation (1). The reason is that in most practical scenarios, engineers are limited to working with data obtained from the original nonlinear two-time-scale system in Equation (1). Additionally, it is possible to design an RNN model that predicts the full two-time-scale dynamics of Equation (1) using full-state measurements.*

Remark 2. *Assumption 5 takes into account the bounded nature of the RNN inputs. This aligns with the observation that the states x and the inputs u are restricted within certain limits, where* $x \in \Omega_\rho$ *and* $u \in U$*. Assumption 6 requires that the weight matrices of the RNN are bounded. This requirement can be met while training the RNN since the search for the optimal RNN parameters is limited to a finite class of neural network hypotheses. Assumption 8 indicates that the RNN model constructed using data derived from industrial operations will be utilized in the exact same process under the condition that the data distribution remains unchanged. It is important to point out that in the context of assessing the generalization performance for machine learning models, Assumption 8 is regarded as essential. Several well-known activation functions can be used to*

construct the RNN model that satisfy Assumption 7; the rectified linear unit (ReLu) activation function is one such example.

2.5. Feedforward Neural Networks

Using the RNN model described in Section 2.4, we are able to predict the slow state vector x. However, it is important to devise a methodology to predict the values of the fast states. In line with the framework proposed in [5], we consider the design of a feedforward neural network (FNN) that predicts the fast states using the previously predicted slow states from the RNN model. More precisely, the FNN input will be the slow states x, and its output will be the fast states z. The FNN model is trained using m independent data points from an underlying distribution over $\mathbb{R}^{d_x^F} \times \mathbb{R}^{d_y^F}$, where d_x^F and d_y^F are the dimensions of the FNN input and output, respectively. In this particular application, given that the FNN input is the slow state, $x \in \mathbb{R}^n$, and its output is the fast state $z \in \mathbb{R}^p$, we have $d_x^F = n$ and $d_y^F = p$. The general form of an FNN model can be formulated as follows:

$$\mathbf{y}^F = \sigma_d(Q_d\sigma_{d-1}(Q_{d-1}\sigma_{d-2}(...\sigma_1(Q_1\mathbf{x}^F)))) \tag{17}$$

where d is the total number of FNN layers, $\mathbf{y}^F \in \mathbb{R}^{d_y^F}$ is the predicted output of the FNN, and $\mathbf{x}^F \in \mathbb{R}^{d_x^F}$ is the FNN's input. For each FNN layer l, where $l = 1, \ldots, d$, the weight parameter matrix is denoted as Q_l, and the activation function is represented as σ_l. The depth of the network is represented by the number of layers d. However, the width of the network is the maximal number of neurons in a hidden layer and is represented by $h_{\max}$. Given that h_l denotes the number of neurons in the l^{th} layer, the width of the network can be expressed as $h_{\max} = \max_{l=1,\ldots,d}\{h_l\}$. The general FNN structure is shown in Figure 2.

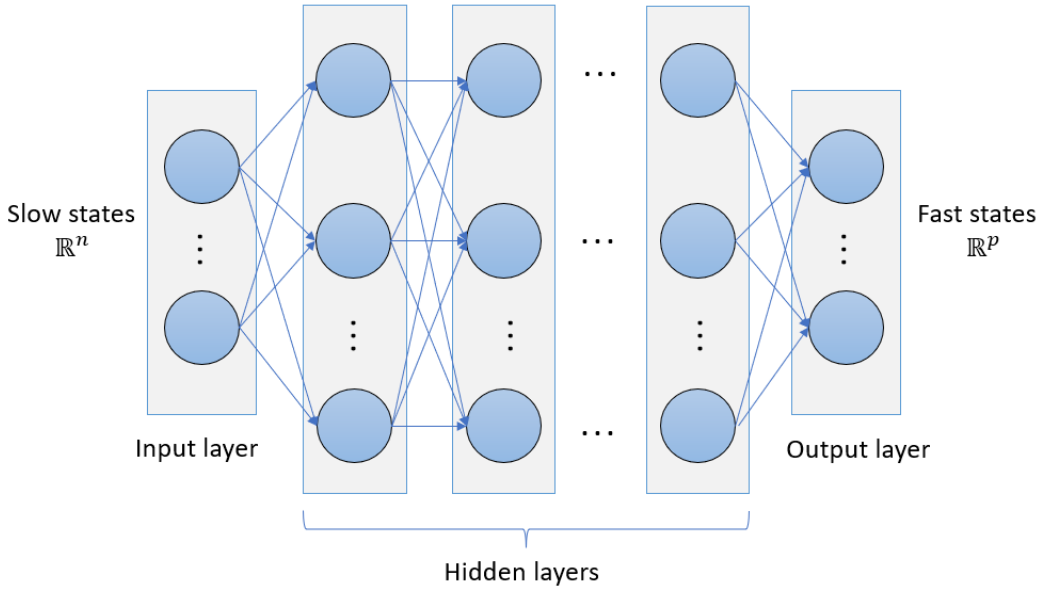

Figure 2. Feedforward neural network structure.

If $\check{\mathbf{y}}^F$ is the predicted value and $\mathbf{y}^F$ is the actual value, we can define the loss function $L(\mathbf{y}^F, \check{\mathbf{y}}^F)$ that computes the squared difference between the actual and the predicted values. More precisely, we will consider the L_2 (i.e., mean squared error) loss function. Similar to the discussion on RNNs, we establish the following assumptions for the developed FNN model:

Assumption 9. *The inputs of the FNN are bounded (i.e., $\|\mathbf{x}_i^F\| \leq B_X$ for all $i = 1, ..., m$).*

Assumption 10. *The norms of the weight matrices are bounded in the sense that the maximal 1-norm of the rows of the weight matrices in the hidden layers and output is bounded (i.e., $\|Q\|_{1,\infty} \leq B_Q$).*

Assumption 11. *The activation function σ_l is 1-Lipschitz continuous and satisfies $\sigma_l(0) = 0$, where $l = 1, ..., d$ (e.g., $\tanh(\cdot)$).*

Assumption 9 is an implication of the assumption that the RNN output (slow states) is bounded. Assumptions 9–11 follow the same reasoning as Assumptions 5–7, as explained in Remark 2. Building upon the previous assumptions, it should be noted that Assumption 8 remains applicable to the FNN model.

3. Generalization Error Bounds of Neural Networks Modeling Two-Time-Scale Systems

The primary goal of constructing any neural network model is to make accurate predictions. Therefore, analyzing the generalization error bounds for various types of neural networks modeling different types of nonlinear systems enables us to improve the construction and design of these network models. Typically, the assessment of neural networks is conducted using a finite set of training samples. Hence, it is essential to investigate the generalization error bound for the neural network model, which allows us to evaluate the performance of the model in accurately predicting outcomes for new, unseen data. In other words, considering new data from the same distribution, the generalization error bound quantifies the ability of a neural network model to make accurate predictions for unseen data that the neural network has not encountered before.

In this work, we consider an RNN that predicts the dynamics of the slow states. Then, an FNN is utilized to predict the fast states using the predicted slow states as the input for the FNN. The generalization error bounds of the neural networks modeling two-time-scale systems are investigated. Particularly, in this section, we discuss the generalization error bounds of both the RNN (which predicts the slow states' dynamics) and the FNN (which predicts the fast states). The generalization error bound is derived by utilizing the basic concepts of statistical machine learning theory. Hence, in the upcoming subsection, we outline the fundamental concepts and definitions used to derive the generalization error bounds. Additionally, it is worth mentioning that in the following subsection, the same principles that apply to $\mathbf{x}$,$\mathbf{y}$, and $\breve{\mathbf{y}}$ also extend to $\mathbf{x}^F$,$\mathbf{y}^F$, and $\breve{\mathbf{y}}^F$, respectively. In the same vein, the principles that apply to d_x and d_y are also true for d_x^F and d_y^F, respectively. Let $\mathcal{H}$ be the hypothesis class of RNN functions $h(\cdot)$ that map the d_x-dimensional input $\mathbf{x} \in \mathbb{R}^{d_x}$ to the d_y-dimensional output $\breve{\mathbf{y}} \in \mathbb{R}^{d_y}$. Similarly, let $\mathcal{H}^F$ be the hypothesis class of FNN functions $h^F(\cdot)$ that map the d_x-dimensional input $\mathbf{x}^F \in \mathbb{R}^{d_x}$ to the d_y-dimensional output $\breve{\mathbf{y}}^F \in \mathbb{R}^{d_y}$. We also note that in the following subsection, the principles that apply to $\mathcal{H}$ and $h(\cdot)$ are also valid for $\mathcal{H}^F$ and $h^F(\cdot)$, respectively. It should be noted that the dimensions d_x and d_y differ based on whether the network is an RNN or an FNN.

3.1. Generalization Error Bound Preliminaries

Definition 1. *Consider a function, h, which makes predictions of the output $\mathbf{y}$ corresponding to the input $\mathbf{x}$, along with an underlying distribution D. The generalization error or the expected loss is formulated as follows:*

$$\mathbb{E}[L(h(\mathbf{x}), \mathbf{y})] = \int_{X \times Y} L(h(\mathbf{x}), \mathbf{y})\, \rho(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{y}. \tag{18}$$

where $\rho(\mathbf{x}, \mathbf{y})$ represents the joint probability distribution for $\mathbf{x}$ and $\mathbf{y}$, whereas X and Y represent the vector space for all possible inputs and outputs, respectively, and L denotes the loss function.

We introduce the following definition of the empirical error as an approximation of the expected loss due to the fact that the joint probability distribution ρ is unknown in many scenarios.

Definition 2. *Considering a data set consisting of m data samples* $S = (s_1, ..., s_m)$*, with each* $s_i = (\mathbf{x}_i, \mathbf{y}_i)$*, the empirical error or risk can be expressed as*

$$\hat{\mathbb{E}}_S[L(h(\mathbf{x}), \mathbf{y})] = \frac{1}{m}\sum_{i=1}^{m} L(h(\mathbf{x}_i), \mathbf{y}_i) \tag{19}$$

We note that the m data samples are gathered from the same data distribution. In this work, the loss function $L(\mathbf{y}, \check{\mathbf{y}})$ is chosen as the mean squared error (i.e., L_2 loss function). Furthermore, the generalization error bounds are derived using "Rademacher complexity", a widely recognized concept in the field of machine learning theory that is used to determine the complexity and richness of a class of functions. The Rademacher complexity is defined as follows:

Definition 3. *Given a data set* $S = \{s_1, ..., s_m\}$ *with m samples, and a hypothesis class* $\mathcal{F}$ *of real-valued functions, the empirical Rademacher complexity of* $\mathcal{F}$ *can be defined as follows:*

$$\mathcal{R}_S(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^{m} \epsilon_i f(s_i)\right] \tag{20}$$

where $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_m)^T$ *consists of the Rademacher random variables* ϵ_i*. These variables are independent and identically distributed (i.i.d.), satisfying the condition that* $\mathbb{P}(\epsilon_i = -1) = \mathbb{P}(\epsilon_i = 1) = 0.5$*.*

Given that $\mathcal{G}_t$ is the class of loss functions associated with the function class $\mathcal{H}$ and is defined as follows

$$\mathcal{G}_t = \{g_t : (\mathbf{x}, \mathbf{y}) \to L(h(\mathbf{x}), \mathbf{y}), h \in \mathcal{H}\}, \tag{21}$$

where $\mathbf{x}$ and $h(\mathbf{x})$ are the model's input and output, respectively, whereas $\mathbf{y}$ denotes the actual value of the output, the upper bound of the generalization error can be computed using the Rademacher complexity $\mathcal{R}_S(\mathcal{G}_t)$, as shown in the following Lemma.

Lemma 1 (c.f. Theorem 3.3 in [16])**.** *Given a data set* $S = (s_i)$*,* $i = 1, ..., m$*, that consists of m i.i.d. data samples, where* $s_i = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$ *for an RNN and* $s_i = (\mathbf{x}_i, \mathbf{y}_i)$ *for an FNN, with probability* $1 - \delta$*, the following inequality holds for all* $g_t \in \mathcal{G}_t$ *over the data samples S:*

$$\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i) + 2\mathcal{R}_S(\mathcal{G}_t) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{22}$$

For a comprehensive proof of Lemma 1, interested readers can refer to [12,16]. By observing Equation (22), clearly, the upper bound of the generalization error relies on three terms. The first term is the empirical loss ($\frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i)$). The second term is the Rademacher complexity ($2\mathcal{R}_S(\mathcal{G}_t)$), whereas the third term depends on the data set size m, along with the confidence level δ. At this stage, the goal is to be able to effectively quantify the generalization error bound using predefined and measurable values. This can be achieved by imposing a further upper bound on the Rademacher complexity term. Having established the background and basic concepts of the generalization error bound, our next discussion focuses on examining the generalization error bounds for neural networks modeling two-time-scale systems.

3.2. RNN Generalization Error Bound

In this subsection, we investigate the generalization error bound for the RNN model that predicts the slow dynamics of the two-time-scale system defined in Equation (1). We start by introducing the following Lemma, which upper-bounds the Rademacher complexity in terms of the RNN parameters.

Lemma 2. *Let $\mathcal{H}_{k,t}$, $k = 1, ..., d_y$ represent the class of real-valued functions associated with the k^{th} component of the RNN output at the t^{th} time step, with the activation functions and weight matrices satisfying Assumptions 5–8. Given a data set $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}, i = 1, ..., m$, consisting of m i.i.d. data samples, the following inequality holds for the Rademacher complexity:*

$$\mathcal{R}_S(\mathcal{H}_{k,t}) \leq \frac{M(\sqrt{2\log(2)t}+1)B_X}{\sqrt{m}} \tag{23}$$

where $M = B_{\mathcal{V},F} B_{W,F} \frac{B_{U,F}^{t}-1}{B_{U,F}^{-1}}$, and B_X is the upper bound for the RNN inputs.

For a comprehensive and detailed proof of Lemma 2, interested readers may refer to [12]. By applying the results derived and proven in [12], the following Lemma specifically addresses the generalization error bound for the RNN model that predicts the slow dynamics of the two-time-scale system defined in Equation (1).

Lemma 3. *Consider the general class of the two-time-scale continuous-time nonlinear systems described in Equation (1) under the assumption that the slow and fast states in Equation (1) are stable and the perturbation parameter ϵ is sufficiently small. An RNN satisfying Assumptions 5–8 is constructed to predict the slow states of the system at the t^{th} time step. Given the L_r–Lipschitz loss function that belongs to the family of loss functions $\mathcal{G}_t$ associated with the RNN function of class $\mathcal{H}_t$ and a data set $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, ..., m$, with m i.i.d. data samples, the following inequality holds true with a probability of at least $1 - \delta$ over S for all $t \in [t_0, T]$:*

$$\mathbb{E}[g_t^{RNN}(\check{x}, x)] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\check{x}_i, x_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{24}$$

where $M = B_{\mathcal{V},F} B_{W,F} \frac{B_{U,F}^{t}-1}{B_{U,F}^{-1}}$, and B_X is the upper bound for the RNN inputs.

Equation (24) indicates that the generalization error bound of the RNN that predicts the slow state vector x depends on a number of factors, such as the number of training samples m, the time length t of the RNN's inputs, the upper bound on the input vector B_X, and the complexity hypothesis class in terms of the weight matrices M, as well as the empirical loss ($\frac{1}{m}\sum_{i=1}^{m} g_t(\check{x}_i, x_i)$). We emphasize that the actual slow state is denoted as x, and the slow state predicted by the RNN model is $\check{x}$. Additionally, based on Equation (8), the generalization error bound in Equation (24) holds for all $t \in [t_0, T]$ under the assumption that the slow and fast states are stable, and the perturbation parameter ϵ is sufficiently small.

Now we discuss the implementation of the RNN's generalization error bound for a specific loss function. A locally Lipschitz continuous loss function is used to optimize the RNN's weights and biases. To be more precise, we employ the MSE loss function of the form,

$$L = \frac{1}{m}\sum_{i=1}^{m}(\check{\mathbf{y}}_i, \mathbf{y}_i)^2, \tag{25}$$

where $\check{\mathbf{y}}$ and $\mathbf{y}$ are the predicted and actual values, respectively. Since the loss function L is locally Lipschitz continuous, it satisfies the following inequality,

$$|L(\mathbf{y}, \check{\mathbf{y}}_\mathbf{2}) - L(\mathbf{y}, \check{\mathbf{y}}_\mathbf{1})| \leq L_r|\check{\mathbf{y}}_\mathbf{2} - \check{\mathbf{y}}_\mathbf{1}| \tag{26}$$

where L_r is the local Lipschitz constant for the loss function L. Since the RNN model predicts the slow states of Equation (1), the loss function is computed over the actual and

predicted slow states, x and $\check{x}$, respectively. Moreover, the expected loss of L is upper-bounded by the following inequality, with a probability of at least $1-\delta$:

$$\mathbb{E}[L(\check{x},x)] \leq \frac{1}{m}\sum_{i=1}^{m} L(\check{x}_i,x_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{27}$$

Given that the MSE loss function L computes the error between the RNN's output x and the RNN's predicted output $\check{x}$, the upper bound on $|\check{x}-x|$ can be written as follows:

$$|\check{x}-x| \leq \sqrt{\frac{1}{m}\sum_{i=1}^{m} L(\check{x}_i,x_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right)} \tag{28}$$

3.3. FNN Generalization Error Bound

In this subsection, we investigate the generalization error bound for the FNN model that predicts the fast states of the two-time-scale system defined in Equation (1). We start by introducing the following Lemma, which upper-bounds the Rademacher complexity in terms of the FNN parameters.

Lemma 4 (c.f. Theorem 2 in [17])**.** *Given a class of scalar-valued functions $\mathcal{H}_k^F$ and a neural network with depth d, where $\|Q_1\|_{1,\infty} \leq B_Q$ for all $l = 1,...,d$ and Assumptions 8–11 are satisfied, the following inequality holds for the Rademacher complexity:*

$$\mathcal{R}_S(\mathcal{H}_k^F) \leq \frac{2B_X(B_Q)^d\sqrt{d+1+\log(d_x)}}{\sqrt{m}} \tag{29}$$

For a comprehensive and detailed proof of Lemma 4, interested readers may refer to [17]. By applying the results derived and proven in [13], the following Lemma specifically addresses the generalization error bound for the FNN model that predicts the fast states using the slow states of the two-time-scale system defined in Equation (1).

Lemma 5. *Consider the general class of two-time-scale continuous-time nonlinear systems described in Equation (1) under the assumption that the slow and fast states are stable, and the perturbation parameter ϵ is sufficiently small. An FNN satisfying Assumptions 8–11 is constructed to predict the fast states using the slow states. Given the L_r–Lipschitz loss functions associated with the vector-valued FNN hypothesis class $\mathcal{H}^F$ and a data set S consisting of m i.i.d. data samples, the following inequality holds true, with a probability of at least $1-\delta$ over S for all $t \in [t_p, T]$, where t_p is defined in Theorem 1:*

$$\mathbb{E}[g_t^{FNN}(\check{z},z)] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\check{z}_i,z_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{B_X(B_Q)^d\sqrt{d+1+\log(d_x)}}{\sqrt{m}}\right) \tag{30}$$

Equation (30) indicates that the generalization error bound of the FNN modeling the fast subsystem depends on several factors, such as the number of training samples m, the upper bound on the input vector B_X, the complexity hypothesis class in terms of the upper bound of the weight matrices B_Q, and the number of layers d, as well as the empirical loss ($\frac{1}{m}\sum_{i=1}^{m} L(\check{z}_i,z_i)$). We emphasize that the actual fast state is denoted as z, and the fast state predicted by the FNN model is $\check{z}$. Therefore, the generalization error bound of Equation (30) holds for all $t \in [t_p, T]$ under the assumption that the slow and fast states are stable, and the perturbation parameter ϵ is sufficiently small.

Now we discuss the implementation of the FNN's generalization error bound specifically on the MSE loss function. A locally Lipschitz continuous loss function such as the MSE is used to optimize the FNN's weights and biases. The MSE loss function satisfies the inequality of Equation (26). Since the FNN model predicts the fast states of Equation (1),

the loss function is computed over the actual and predicted fast states, z and $\check{z}$, respectively. Moreover, the expected loss of L is upper-bounded by the following inequality, with a probability of at least $1-\delta$:

$$\mathbb{E}[L(\check{z},z)] \leq \frac{1}{m}\sum_{i=1}^{m} L(\check{z}_i,z_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{B_X (B_Q)^d \sqrt{d+1+\log(d_x)}}{\sqrt{m}}\right) \tag{31}$$

Given that the MSE loss function L computes the error between the RNN's output z and the RNN's predicted output $\check{z}$, the upper bound on $|\check{z}-z|$ can be written as follows:

$$|\check{z}-z| \leq \sqrt{\frac{1}{m}\sum_{i=1}^{m} L(\check{z}_i,z_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{B_X (B_Q)^d \sqrt{d+1+\log(d_x)}}{\sqrt{m}}\right)} \tag{32}$$

4. Machine Learning-Based LMPC Using an RNN That Approximates the Slow Subsystem

Having investigated the generalization error bounds for neural networks modeling two-time-scale systems, the rest of this manuscript focuses on the design and performance of LMPC using RNN models with and without time-scale decomposition. Specifically, the objective is to design an RNN to predict only the slow state vector x, following the approach outlined in Section 2.4, and then incorporate the designed RNN model within an LMPC scheme to show that it is sufficient to stabilize the full two-time-scale system in Equation (1). In this section, we present a stability analysis for the proposed LMPC framework. For simplicity, in this section, we present the RNN model that predicts the slow state vector x as a continuous-time nonlinear system of the following form:

$$\dot{\hat{x}} = F_{nn}(\hat{x},u) := A\hat{x} + \Theta^T \hat{z} \tag{33}$$

where $\hat{x} \in \mathbb{R}^n$ is the RNN's state vector and $u \in \mathbb{R}^{q_1}$ is the manipulated input vector. The weight matrices are denoted as A and Θ. The diagonal coefficient matrix A has negative diagonal entries. Θ is defined as $\Theta = [\theta_1,...,\theta_n] \in \mathbb{R}^{(q_1+n)\times n}$, with entries $\theta_i = b_i[w_{i1},...,w_{i(q_1+n)}], i = 1,...,n$, given that w_{ij} represents the weight associated with the connection between the ith neuron and the jth output, with i ranging from 1 to n and j ranging from 1 to (q_1+n). The vector $\hat{z} = [\hat{z}_1,...,\hat{z}_n,\hat{z}_{n+1},...,\hat{z}_{n+q_1}] = [\sigma(\hat{x}_1)...\sigma(\hat{x}_n),u...u_{q_1}] \in \mathbb{R}^{n+q_1}$ consists of both the network states $\hat{x}$ and inputs u. Additionally, a nonlinear activation function $\sigma(\cdot)$ is applied to the network states $\hat{x}$ when constructing the vector $\hat{z}$. It is worth mentioning that the weight matrices and activation functions satisfy Assumptions 5–8. For simplicity, we assume a single-hidden-layer RNN model represented by Equation (33), which does not explicitly include bias terms. Nevertheless, it should be emphasized that the results obtained in this section are not limited to single-hidden-layer RNN models, but can be generalized to include deep RNN models that consist of multiple hidden layers.

4.1. Lyapunov-Based Control Using an RNN Model

Taking into account the RNN model designed to predict the values of the slow states, we assume that there exists a stabilizing control law $\Phi_{nn}(x) \in U$ (e.g., a P-controller, a Lyapunov-based controller) that can render the origin of the RNN model in Equation (33) asymptotically stable in an open neighborhood around the origin denoted as $\hat{D}$. Hence, there exists a continuously differentiable Lyapunov function $\hat{V}: \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ that satisfies the following inequalities:

$$\hat{a}_1(|x|) \leq \hat{V}(x) \leq \hat{a}_2(|x|) \tag{34a}$$

$$\frac{\partial \hat{V}(x)}{\partial x} F_{nn}(x,\Phi_{nn}(x)) \leq -\hat{a}_3(|x|) \tag{34b}$$

$$\left|\frac{\partial \hat{V}(x)}{\partial x}\right| \leq \hat{a}_4(|x|) \tag{34c}$$

where $\hat{a}_i$ are class $\mathcal{K}$ functions for all $x \in \mathbb{R}^n \subset \hat{D}$. The Lyapunov function $\hat{V}$ can be designed using a quadratic formulation, which satisfies the required criteria, or, alternately, using learning-based approaches, as conducted using a linear parameter-varying (LPV) framework in [18]. In the simulation example presented in our work below, a quadratic Lyapunov function is used, and extensive open-loop and closed-loop simulations are conducted to verify that the asymptotic stability assumption is met. The Lyapunov level set that ensures the stability of the RNN model is defined as $\Omega_{\hat{\rho}} := \{x \in \hat{D} \mid \hat{V}(x) \leq \hat{\rho}\}$, where $\hat{\rho} > 0$. This implies that the closed-loop stability region of the RNN model in Equation (33) is denoted as $\Omega_{\hat{\rho}}$. Moreover, there exist positive constants M_{nn} and L_{nn}, such that the following inequalities are satisfied for all $x, x' \in \Omega_{\hat{\rho}}$, and $u \in U$:

$$|F_{nn}(x,u)| \leq M_{nn} \tag{35a}$$

$$\left|\frac{\partial \hat{V}(x)}{\partial x}F_{nn}(x,u) - \frac{\partial \hat{V}(x')}{\partial x}F_{nn}(x',u)\right| \leq L_{nn}|x-x'| \tag{35b}$$

The feedback controller $u = \Phi_{nn}(x) \in U$ can stabilize the dynamics of the slow subsystem in Equation (11) under a sufficiently small modeling error between the nominal slow subsystem and the RNN model. This result is shown in the following proposition.

Proposition 1. *Consider the RNN model in Equation (33) that satisfies the stabilizability conditions of Equation (34) and is rendered asymptotically stable around the origin under the control law $u = \Phi_{nn}(x) \in U$ for all $x \in \Omega_{\hat{\rho}}$. Then, the origin of the slow subsystem in Equation (11) is asymptotically stable if there exists a positive real number ν_m, where $\nu_m < \hat{a}_3(|x|)/\hat{a}_4(|x|)$, such that the modeling error between the slow subsystem in Equation (11) and the RNN model in Equation (33) (i.e., $\nu = |F(x,u) - F_{nn}(x,u)|$) is upper-bounded by ν_m for all $x \in \Omega_{\hat{\rho}}$.*

Proof. We follow the strategy outlined in [5] to prove Proposition 1. The main objective is to show that, under the control law $u = \Phi_{nn}(x) \in U$, the slow subsystem in Equation (11) is asymptotically stable around the origin for all $x \in \Omega_{\hat{\rho}}$ if $\Phi_{nn}(x)$ renders the RNN designed to approximate the slow subsystem asymptotically stable with a bounded modeling error between the nominal slow subsystem and the RNN model. In other words, we show that $\dot{\hat{V}}(x) \leq 0$ for the slow subsystem in Equation (11) under the controller $u = \Phi_{nn}(x) \in U$ based on the RNN model. We utilize the inequalities in Equations (34b) and (34c) and compute $\dot{\hat{V}}(x)$ as follows:

$$\begin{aligned}
\dot{\hat{V}} &= \frac{\partial \hat{V}(x)}{\partial x}F(x,\Phi_{nn}(x)) \\
&= \frac{\partial \hat{V}}{\partial x}(F_{nn}(x,\Phi_{nn}(x)) + F(x,\Phi_{nn}(x)) - F_{nn}(x,\Phi_{nn}(x)) \\
&\leq -\hat{a}_3(|x|) + \hat{a}_4(|x|)(F(x,\Phi_{nn}(x)) - F_{nn}(x,\Phi_{nn}(x))) \\
&\leq -\hat{a}_3(|x|) + \nu_m \hat{a}_4(|x|)
\end{aligned} \tag{36}$$

By assigning $\nu_m < \frac{\hat{a}_3(|x|)}{\hat{a}_4(|x|)}$, we obtain $\dot{\hat{V}}(x) \leq -\tilde{a}_3(|x|) \leq 0$, where $\tilde{a}_3(|x|) = -\hat{a}_3(|x|) + \nu_m \hat{a}_4(|x|) > 0$ since $\hat{a}_3$ and $\hat{a}_4$ are known functions chosen in such a way as to ensure that the above result holds. To show that if the general class $\mathcal{K}$ functions are chosen as $\hat{a}_3 = a_3|x|$ and $\hat{a}_4 = a_4|x|$, where a_3 and a_4 are constants, then $\nu_m = \frac{a_3}{a_4}$. Hence, we guarantee the closed-loop stability of the slow subsystem in Equation (11) around the origin under the control law $\Phi_{nn}(x) \in U$ for all $x \in \Omega_{\hat{\rho}}$. □

The RNN model designed in Equation (33) is incorporated into an LMPC scheme, where the control actions computed by the LMPC are implemented in a sample-and-hold fashion. Moreover, this sample-and-hold implementation of the LMPC control law $u = \Phi_{nn}(x) \in U$ establishes certain properties. These properties are studied in the following two propositions. Specifically, the following proposition shows that the error between the state of the slow subsystem in Equation (11) and the state predicted by the RNN model in Equation (33) is bounded.

Proposition 2. *Consider the RNN model in Equation (33) and the slow subsystem in Equation (11), starting with the same initial condition $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$. There exists a class $\mathcal{K}$ function $f_w(\cdot)$ and a positive constant κ, such that the following inequalities are satisfied for all $x, \hat{x} \in \Omega_{\hat{\rho}}$:*

$$|x(t) - \hat{x}(t)| \leq f_w(t) := \frac{\nu_m}{L_x}\left(e^{L_x t} - 1\right) \tag{37a}$$

$$\dot{\hat{V}}(x) \leq \dot{\hat{V}}(\hat{x}) + \hat{a}_4(\hat{a}_1^{-1}(\hat{\rho}))|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{37b}$$

Proof. Considering that $e(t) = x(t) - \hat{x}(t)$ represents the error vector between the state of the slow subsystem in Equation (11) and the state of the RNN model in Equation (33), the bound for the time derivative of $e(t)$ is given as follows:

$$\begin{aligned} |\dot{e}(t)| &= |F(x,u) - F_{nn}(\hat{x},u)| \\ &\leq |F(x,u) - F(\hat{x},u)| + |F(\hat{x},u) - F_{nn}(\hat{x},u)| \end{aligned} \tag{38}$$

Using Equation (14b), for all $x, \hat{x} \in \Omega_{\hat{\rho}}$, the first term in Equation (38) can be bounded as follows:

$$\begin{aligned} |F(x,u) - F(\hat{x},u)| &\leq L_x|x(t) - \hat{x}(t)| \\ &\leq L_x|x(t) - \hat{x}(t)| \end{aligned} \tag{39}$$

The term $|F(\hat{x},u) - F_{nn}(\hat{x},u)|$ in Equation (39) denotes the modeling error, and it is upper-bounded by ν_m for all $\hat{x} \in \Omega_{\hat{\rho}}$. Hence, the term $\dot{e}(t)$ can be further bounded by utilizing the bound of the modeling error and the bound of Equation (39), as follows:

$$\begin{aligned} |\dot{e}(t)| &\leq L_x|x(t) - \hat{x}(t)| + \nu_m \\ &\leq L_x|e(t)| + \nu_m \end{aligned} \tag{40}$$

Taking into account the zero initial condition (i.e., $e(0) = 0$), we integrate the inequality in Equation (40) and obtain the following upper bound for the error vector for all $x, \hat{x} \in \Omega_{\hat{\rho}}$:

$$|e(t)| = |x(t) - \hat{x}(t)| \leq \frac{\nu_m}{L_x}\left(e^{L_x t} - 1\right) \tag{41}$$

Equation (37b) can be derived using the Taylor series expansion of $\dot{\hat{V}}(x)$ around $\hat{x}$ for all $x, \hat{x} \in \Omega_{\hat{\rho}}$ as follows:

$$\dot{\hat{V}}(x) \leq \dot{\hat{V}}(\hat{x}) + \frac{\partial \dot{\hat{V}}(\hat{x})}{\partial x}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{42}$$

where κ is a positive real number. Furthermore, Equations (34a) and (34b) are used to further upper-bound $\dot{\hat{V}}(x)$ as follows:

$$\dot{\hat{V}}(x) \leq \dot{\hat{V}}(\hat{x}) + \hat{a}_4(\hat{a}_1^{-1}(\hat{\rho}))|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{43}$$

□

The following proposition demonstrates that the closed-loop state of the slow subsystem described in Equation (11) is maintained within the stability region $\Omega_{\hat{\rho}}$ at all times.

Additionally, utilizing the Lyapunov-based controller $u = \Phi_{nn}(x) \in U$ through sample-and-hold implementation ensures that the closed-loop state of the slow subsystem can be ultimately bounded within a small region around the origin $\Omega_{\rho_{\min}}$.

Proposition 3 (c.f Proposition 3 in [5]). *Consider the slow subsystem in Equation (11) under the controller $\Phi_{nn}(\hat{x}) \in U$ that satisfies the conditions in Equation (34) and is implemented in a sample-and-hold fashion (i.e., $\Phi_{nn}(\hat{x}(t_k)), \forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$) to stabilize the RNN model in Equation (33). Then, there exist $\epsilon_w > 0$, $\Delta > 0$ and $\hat{\rho} > \rho_{\min} > \rho_{nn} > \rho_s$ that satisfy*

$$-\hat{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_{nn}M_{nn}\Delta \leq -\epsilon_s \tag{44a}$$

$$-\tilde{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_x^{'}M_F\Delta \leq -\epsilon_w \tag{44b}$$

and

$$\rho_{nn} := \max\{\hat{V}(\hat{x}(t+\Delta)) \mid \hat{x}(t) \in \Omega_{\rho_s}, u \in U\} \tag{45a}$$

$$\rho_{\min} \geq \rho_{nn} + -\hat{a}_4(\hat{a}_1^{-1}(\hat{\rho}))f_w(\Delta) + \kappa\big(f_w(\Delta)\big)^2 \tag{45b}$$

such that, for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, there exists a class $\mathcal{KL}$ function β_x and a class $\mathcal{K}$ function $\bar{\gamma}$, such that the following inequality is satisfied:

$$|x(t)| \leq \beta_x(|x(0)|, t) + \bar{\gamma}(\rho_{\min}) \tag{46}$$

and the state $x(t)$ of the nominal slow subsystem in Equation (11) is bounded in $\Omega_{\hat{\rho}}$ for all times and ultimately bounded in $\Omega_{\rho_{\min}}$.

Proof. Assuming that $x(t_k) = \hat{x}(t_k)$, the first part of establishing this proof involves showing that $\hat{V}(\hat{x})$ is decreasing under the control law $u(t) = \Phi_{nn}(x(t_k)) \in U$ for $t \in [t_k, t_{k+1})$, where $x(t_k)$ and $\hat{x}(t_k)$ are the state of the slow subsystem in Equation (11) and the state of the RNN model in Equation (33), respectively. The time derivative of $\hat{V}(\hat{x})$ for all $t \in [t_k, t_{k+1})$ is calculated as follows:

$$\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) &= \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))) \\
&= \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}\big(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k))\big) \\
&\quad + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}\big(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))\big) \\
&\quad - \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}\big(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k))\big)
\end{aligned} \tag{47}$$

By utilizing the inequalities in Equations (34a) and (34b), we further bound $\dot{\hat{V}}(\hat{x}(t))$ as follows:

$$\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) \leq &-\hat{a}_3(\hat{a}_2^{-1}(\rho_s)) + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{nn}(\hat{x}(t), \Phi_{nn}(\hat{x}(t_k))) \\
&- \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{nn}(\hat{x}(t_k), \Phi_{nn}(\hat{x}(t_k)))
\end{aligned} \tag{48}$$

By applying the Lipschitz inequalities in Equation (35), we proceed with bounding $\dot{\hat{V}}(\hat{x}(t))$ as follows:

$$\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) &\leq -\hat{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_{nn}|\hat{x}(t) - \hat{x}(t_k)| \\
&\leq -\hat{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_{nn}M_{nn}\Delta
\end{aligned} \tag{49}$$

Therefore, if Equation (44a) is satisfied, the following inequality holds for all $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(x(t)) \leq -\epsilon_s \tag{50}$$

Upon integrating the aforementioned differential equation over the time interval $t \in [t_k, t_{k+1})$, it is derived that $\hat{V}(\hat{x}(t_{k+1})) \leq \hat{V}(\hat{x}(t_k)) - \epsilon_s \Delta$. Therefore, when Equation (44a) is satisfied, this leads to the fact that $\dot{\hat{V}}(x(t))$ is negative for any $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. As a result, utilizing the control law $u = \Phi_{nn}(\hat{x})$ in a sample-and-hold fashion ensures that the closed-loop state of the RNN model in Equation (33) is bounded within the region $\Omega_{\hat{\rho}}$ and moves toward the origin. Nevertheless, it should be observed that Equation (50) may not hold true in cases where $x(t_k) = \hat{x}(t_k) \in \Omega_{\rho_s}$, which indicates that the state $\hat{x}(t_k)$ may leave the region Ω_{ρ_s} within one sampling period. Hence, we introduce the region $\Omega_{\rho_{nn}}$ in Equation (45a) to guarantee that the closed-loop state $\hat{x}(t_k)$ of the RNN model will be bounded in the region $\Omega_{\rho_{nn}}$ within one sampling period for all $t \in [t_k, t_{k+1})$, $u \in U$ and $\hat{x}(t_k) \in \Omega_{\rho_s}$. Consider the case where $\hat{x}(t_{k+1})$ leaves the region Ω_{ρ_s}. Then, the controller $u = \Phi_{nn}(x(t_{k+1}))$ reactivates to derive the states into the region Ω_{ρ_s}, and Equation (50) will be satisfied again at $t = t_{k+1}$. Up to this point, it can be concluded that the state of the RNN system in Equation (33) is ultimately bounded in $\Omega_{\rho_{nn}}$ for all $x_0 \in \Omega_{\hat{\rho}}$.

The second part of this proof is to demonstrate that the controller $u = \Phi_{nn}(x) \in U$, implemented in a sample-and-hold fashion, effectively bounds the states of the slow subsystem in Equation (11) in $\Omega_{\hat{\rho}}$ and can ultimately derive the states to a small region around the origin. This requires showing that $\hat{V}(x)$ for the slow subsystem in Equation (11) is decreasing under the control law $u(t) = \Phi_{nn}(x(t_k))$ for $t \in [t_k, t_{k+1})$ and $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. The derivative of $\hat{V}(x(t))$ with respect to time is computed as follows:

$$\begin{aligned}
\dot{\hat{V}}(x(t)) &= \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k))) \\
&= \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k))) \\
&\quad + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k))) \\
&\quad - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)))
\end{aligned} \tag{51}$$

Using the inequality in Equation (36), which holds for all $x \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, the first term in Equation (51) can be further bounded as follows:

$$\begin{aligned}
\dot{\hat{V}}(x(t)) \leq & -\tilde{a}_3(\hat{a}_2^{-1}(\rho_s)) + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{nn}(x(t_k)), \xi) \\
& - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{nn}(x(t_k)), 0)
\end{aligned} \tag{52}$$

where $\tilde{a}_3(\cdot)$ was previously defined in the proof of Proposition 1. By using the Lipschitz condition stated in Equation (14), we derive the following upper bound for $\dot{\hat{V}}(x(t))$:

$$\begin{aligned}
\dot{\hat{V}}(x(t)) \leq & \tilde{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_x^{'} |x(t) - x(t_k)| \\
\leq & \tilde{a}_3(\hat{a}_2^{-1}(\rho_s)) + L_x^{'} M_F \Delta
\end{aligned} \tag{53}$$

Therefore, if Equation (44b) holds true, the following inequality is satisfied for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and for all $t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(x(t)) \leq -\epsilon_w \tag{54}$$

By integrating the above differential equation over the time interval $t \in [t_k, t_{k+1})$ between any two arbitrary points within the previous time interval, it can be shown that for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_S}$, the following results are obtained:

$$\hat{V}(x(t_{k+1})) \leq V(x(t_k)) - \epsilon_w \Delta \tag{55}$$

$$\hat{V}(x(t)) \leq \hat{V}(x(t_k)), \forall t \in [t_k, t_{k+1}) \tag{56}$$

Based on Equation (54), it can be observed that $\dot{\hat{V}}(x(t))$ is negative for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_S}$. As a result, the state of the slow subsystem in Equation (11) remains continuously bounded within the region $\Omega_{\hat{\rho}}$ and can be ultimately driven toward the origin in each sampling period by implementing the control law $u = \Phi_{nn}(x)$. In addition, if $x(t_k) \in \Omega_{\rho_s}$, the state of the RNN model in Equation (33) is bounded within the region $\Omega_{\rho_{nn}}$ for one sampling period. This outcome has been demonstrated in the first part of the proof. Taking onto account the bounded modeling error between the state of the RNN, as described in Equation (33), and the state of the slow subsystem in Equation (11), a compact set $\Omega_{\rho_{\min}} \supset \Omega_{\rho_{nn}}$ exists and satisfies Equation (45b). The set $\Omega_{\rho_{\min}}$ is introduced to ensure that the state of the slow subsystem in Equation (11) is bounded within the region $\Omega_{\rho_{\min}}$ during one sampling period, provided that the state of the RNN represented by Equation (33) is bounded within the region $\Omega_{\rho_{nn}}$. In the case where the state $x(t)$ enters the set $\Omega_{\rho_{\min}} \backslash \Omega_{\rho_S}$, it has been shown that Equation (56) is satisfied. Therefore, under the control law $u = \Phi_{nn}(x)$, the state $x(t)$ will be driven toward the origin in the next sampling period, ultimately bounding the state of the slow subsystem within a small region around the origin $\Omega_{\rho_{\min}}$. Hence, considering the continuity property of the Lyapunov function $\hat{V}$, it follows that there exists a class $\mathcal{KL}$ function β_x and a class $\mathcal{K}$ function $\bar{\gamma}$, such that if $x_0 \in \Omega_{\hat{\rho}}$, then $x(t) \in \Omega_{\hat{\rho}}$ for all $t \leq t_0$:

$$|x(t)| \leq \beta_x(|x(0)|, t) + \bar{\gamma}(\rho_{\min}) \tag{57}$$

□

4.2. Machine Learning-Based LMPC Formulation

In this subsection, we introduce the machine learning-based LMPC formulation that utilizes the RNN model designed in Equation (33) to predict future states over a certain future horizon. Furthermore, the LMPC is essentially an optimization problem consisting of an objective function and a number of constraints. This optimization problem is solved repeatedly to compute the optimal input trajectory. The following shows the formulation of the machine learning-based LMPC optimization problem:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L_{MPC}(\tilde{x}(t), u(t)) dt \tag{58a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{58b}$$

$$u(t) \in U, \ \forall \ t \in [t_k, t_{k+N}) \tag{58c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{58d}$$

$$\dot{\hat{V}}(x(t_k), u) \leq \dot{\hat{V}}(x(t_k), \Phi_{nn}(x(t_k))), \text{ if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{58e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{nn}, \ \forall \ t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega_{\rho_{nn}} \tag{58f}$$

where the predicted slow state trajectory is denoted as $\tilde{x}$. The set of piecewise constant functions with period Δ is denoted as $S(\Delta)$, and the number of sampling periods in the prediction horizon is denoted by N. The optimal control action $u^*(t)$ is calculated by repeatedly solving the machine learning-based LMPC optimization problem over the entire prediction horizon $t \in [t_k, t_{k+N})$. Then, the optimal control action $u^*(t_k)$ computed at the first sampling period is transmitted by the controller to be applied to the process. Subsequently, the resulting real-time state, $x(t_k)$, is fed back to the machine learning-based

LMPC optimization problem to compute the optimal input trajectory for the next sampling time. The optimization problem aims to minimize the time integral of $L_{MPC}(\tilde{x}(t), u(t))$ over the prediction horizon, which is represented in the cost function in Equation (58a). As for the constraints of the optimization problem, the first constraint demonstrated in Equation (58b) is essentially the RNN model utilized to approximate the states of the slow subsystem's dynamics. Equation (58c) represents the input constraints that may be applied to the process over the entire prediction horizon. The initial condition needed to solve Equation (58b) is essentially the slow state measurement at $t = t_k$, which is specified in the constraint in Equation (58d). In the case where $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$, the constraint Equation (58e) ensures that the closed-loop state converges toward the origin. Furthermore, if the state $x(t_k)$ enters the region $\Omega_{\rho_{nn}}$, it is necessary to guarantee that the states predicted by the RNN model remain bounded within the region $\Omega_{\rho_{nn}}$ throughout the entire prediction horizon. This is accomplished by employing Equation (58f).

4.3. Closed-Loop Stability

Assuming that certain conditions are met, the following theorem establishes the closed-loop stability of a singularly perturbed system described by Equation (1) when the machine learning-based LMPC of Equation (58) is implemented.

Theorem 2. *Consider the singularly perturbed system in Equation (1) in a closed loop, with the optimal control action u^* calculated using the machine learning-based LMPC in Equation (58) based on the controller $\Phi_{nn}(x)$ that meets the conditions of Equation (34). Additionally, assuming that Propositions 1–3 and Assumptions 1 and 4 hold true, then there exist class $\mathcal{KL}$ functions β_x and β_z, a pair of positive real numbers (δ, d), and $\epsilon^* > 0$, such that if $\max\{|x(0), |z(0)|\} \leq \delta$ and $\epsilon \in (0, \epsilon^*]$, then, for all $t \geq 0$,*

$$|x(t)| \leq \beta_x(|x(0)|, t) + \bar{\gamma}(\rho_{\min}) + d \tag{59}$$

$$|z(t)| \leq \beta_z\left(|z(0)|, \frac{t}{\epsilon}\right) + d \tag{60}$$

Proof. We follow the proof analysis in [5]. Moreover, we consider that by substituting the optimal control action u^* into Equation (1), the closed-loop system can be represented as follows:

$$\dot{x} = f_1(x, z, u^*, \epsilon) \tag{61a}$$
$$\epsilon \dot{z} = f_2(x, z, \epsilon) \tag{61b}$$

We set $\epsilon = 0$ and obtain the following,

$$\dot{x} = f_1(x, z, u^*, 0) \tag{62a}$$
$$0 = f_2(x, z, 0) \tag{62b}$$

where the assumption that the error between z and $\bar{z}$ is not greater than $\mathcal{O}(\epsilon)$ enables us to simplify the analysis by approximating $\bar{z}$ as z. By solving Equation (62b) for z, we obtain a unique, isolated root $z = \bar{f}(x)$. We substitute the root into Equation (62a) and obtain the following for the reduced-order slow subsystem,

$$\dot{x} = f_1(x, \bar{f}_2(x), u^*, 0) \tag{63}$$

By observing the LMPC equation in Equation (58), in the case where $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$, Equation (58e) is activated, such that the Lyapunov function $\hat{V}$ under the calculated control law u is decreased. According to the finding in Proposition 3 and considering a sufficiently small modeling error, the state of the slow subsystem in Equation (61a) will gradually converge to the origin and enter the region $\Omega_{\rho_{nn}}$ within a finite number of sampling time

steps. Equation (58f) is activated once the state $x(t_k)$ enters the region $\Omega_{\rho_{nn}}$, where it ensures that the predicted state is bounded within $\Omega_{\rho_{nn}}$ throughout the prediction horizon. Additionally, by ensuring that the modeling error and the sampling time are sufficiently small, Proposition 3 establishes that the actual state in Equation (61a) can be maintained within a slightly expanded set $\Omega_{\rho_{min}}$, encompassing $\Omega_{\rho_{nn}}$. Hence, LMPC guarantees that for any initial state $x_0 \in \Omega_{\hat{\rho}}$, the state $x(t)$ of the closed-loop slow subsystem described by Equation (61a) is maintained bounded within the region $\Omega_{\hat{\rho}}$ at all times and fulfills the bound of Equation (57) that was established in Proposition 3.

Additionally, by introducing a fast time scale $\tau = t/\epsilon$, a new coordinate $\bar{\bar{z}} = z - \bar{f}_2(x)$, and considering $\epsilon = 0$, the closed-loop fast subsystem is given by:

$$\frac{d\bar{\bar{z}}}{d\tau} = f_2(x, \bar{\bar{z}} + \bar{f}_2(x), 0) \tag{64}$$

According to Assumption 4, global asymptotic stability is achieved for the origin of the closed-loop fast subsystem in Equation (64), satisfying Equation (12) for any $\bar{\bar{z}}(0) \in \mathbb{R}^p$. As a result, the closed-loop system in Equation (61) meets all the assumptions required for Theorem 1 in [19] to hold true. Hence, there exist class $\mathcal{KL}$ functions β_x and β_z, a pair of positive real numbers (δ, d), and $\epsilon^* > 0$, such that if $\max\{|x(0), |z(0)|\} \leq \delta$ and $\epsilon \in (0, \epsilon^*]$, the closed-loop states of the slow and fast systems are bounded by Equations (59) and (60). □

Remark 3. *It is possible to define the general class of two-time-scale systems with an additional input associated with the fast subsystem as follows:*

$$\dot{x} = f_1(x, z, u, \epsilon) \tag{65a}$$

$$\epsilon \dot{z} = f_2(x, z, u_2, \epsilon) \tag{65b}$$

where $u_2 \in \mathbb{R}^{q_2}$ is the bounded manipulated input vector associated with the fast subsystem. The input vector u_2 is constrained by $u_2 \in U_2 := \{|u_i| \leq u_{2_{\max,i}}, i = 1, ..., q_2\}$. However, if there exists a stabilizing control law $u_2 = h_{stable}(x, z)$ that stabilizes the dynamics of the fast subsystem, then the general class of two-time-scale systems defined by Equation (65) can be reduced to the class of systems presented in Equation (1), which is the primary focus of this work. Furthermore, there are relevant works that have addressed the stability analysis of classes of systems defined in Equation (65), which share similar concepts but incorporate certain modifications. Interested readers may refer to [3] to further explore this subject.

5. Example of Application to a Chemical Process

In this section, we apply the proposed reduced-order machine learning strategy to a chemical process example. Specifically, we build two ML models—a reduced-order ML model to approximate the slow subsystem, and another ML model to capture the dynamics of the full two-time-scale system. Subsequently, each ML model is incorporated into an LMPC scheme, and the performance of the controllers is compared in terms of the closed-loop properties and computational efficiencies.

We consider a perfectly mixed, non-isothermal CSTR, in which an irreversible and endothermic reaction that transforms chemical A to product B ($A \rightarrow B$) occurs. Figure 3 depicts the CSTR, where C_A represents the concentration of chemical A, and T_r represents the temperature of the reactor's contents. C_{A0} denotes the inlet molar concentration of chemical A, which is fed into the reactor at flow rate F and temperature T_{A0}. Assuming that the vessel maintains a constant holdup, V_r denotes the volume of liquid present in the reactor. Due to the occurrence of an endothermic reaction within the reactor, a heating jacket with volume V_j is used to provide the energy as required. T_{j0} is the inlet temperature of the heat-transfer fluid provided to the jacket with a flow rate F_j. The reactor's contents and the heat-transfer fluid maintain constant densities, denoted as ρ_m and ρ_j, respectively. In addition, they both have constant heat capacities, denoted as $c_{p,m}$ and $c_{p,j}$, respectively. The enthalpy of the reaction is ΔH_r, the heat-transfer coefficient is U, and the heat-transfer

contact area between the reactor and the jacket is A_r. Given that k_0 is the pre-exponential constant, R is the ideal gas constant, and E is the activation energy of the reaction, the time-varying rate constant of the reaction is denoted by k and is given by the following equation:

$$k = k_0 \exp\left(\frac{-E}{RT_r}\right) \tag{66}$$

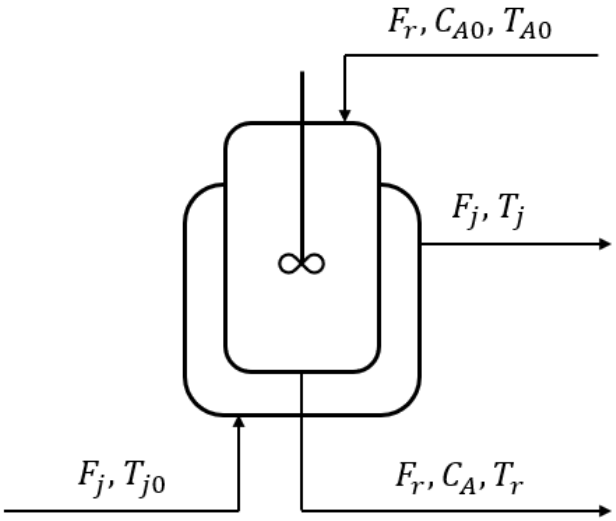

Figure 3. The continuous-stirred tank reactor with jacket.

The first-principles equations of the CSTR are described by the following dynamic material and energy balance equations:

$$V_r \frac{dC_A}{dt} = F_r(C_{A0} - C_A) - k_0 \exp\left(\frac{-E}{RT_r}\right) C_A V_r \tag{67a}$$

$$V_r \frac{dT_r}{dt} = F_r(T_{A0} - T_r) + \frac{-\Delta H_r}{\rho_m C_{p,m}} k_0 \exp\left(\frac{-E}{RT_r}\right) C_A V_r + \frac{UA_r}{\rho_m C_{p,m}}(T_j - T_r) \tag{67b}$$

$$V_j \frac{dT_j}{dt} = F_j(T_{j0} - T_j) - \frac{UA_r}{\rho_j C_{p,j}}(T_j - T_r) \tag{67c}$$

where the values of the process parameters are listed in Table 1. Additionally, we consider the constant $\epsilon = \frac{V_j}{V_r}$ to be the singular perturbation parameter. Hence, the system of Equation (67) can be written in the standard singularly perturbed form of Equation (1), which implies that the concentration of chemical A (C_A) and the temperature of the reactor (T_r) are the slow states, whereas the temperature of the heating jacket (T_j) can be considered as the fast state. The input of the system in Equation (67) is the feed concentration of chemical A (C_{A0}). The control objective is to drive the system to its stable steady state $(C_{As}, T_{rs}, T_{js}) = (2.54 \text{ kmol m}^{-3}, 274.4 \text{ K}, 303.3 \text{ K})$, corresponding to the steady-state input value of $C_{A0s} = 3.75 \text{ kmol m}^{-3}$, while the input is permitted to vary within the range $\Delta C_{A0} = [-3.5, 3.5] \text{ kmol m}^{-3}$. To shift the steady state of the system in Equation (67) to the origin, we express the states and the input in terms of deviation variables and rewrite the system in Equation (67). Specifically, the deviation variables are defined as follows:

$$\Delta C_A = C_A - C_{As} \tag{68a}$$

$$\Delta T_r = T_r - T_{rs} \tag{68b}$$

$$\Delta T_j = T_j - T_{js} \tag{68c}$$

$$\Delta C_{A0} = C_{A0} - C_{A0s} \tag{68d}$$

Therefore, to write the CSTR system of Equation (67) in the standard form of Equation (1), we define $x^T = [\Delta C_A \quad \Delta T_r]$, $z = \Delta T_j$, and the input $u = \Delta C_{A0}$. The explicit Euler method

is utilized to simulate the CSTR system, i.e., Equation (67) is numerically integrated with a sufficiently small time step $h_c = 10^{-4}$ h. The open-source software package IPOPT [20] is employed to solve the ML-based LMPC optimization problem in Equation (58), with a sampling time of $\Delta = 0.03$ h.

Table 1. Notations and parameter values of the CSTR.

$V_r = 1.0\ \text{m}^3$	$E = 8.0 \times 10^3\ \text{kcal kg}^{-1}$
$V_j = 0.08\ \text{m}^3$	$U = 1000.0\ \text{kcal h}^{-1}\text{m}^{-2}\text{K}^{-1}$
$A_r = 6.0\ \text{m}^3$	$k_0 = 3.36 \times 10^6\ \text{h}^{-1}$
$C_{A0s} = 3.75\ \text{kmol m}^{-3}$	$\rho_m = 900.0\ \text{kg m}^{-3}$
$C_{As} = 2.54\ \text{kmol m}^{-3}$	$\rho_j = 800.0\ \text{kg m}^{-3}$
$R = 1.987\ \text{kcal kmol}^{-1}\text{K}^{-1}$	$c_{p,m} = 0.231\ \text{kcal kg}^{-1}\text{K}^{-1}$
$F_r = 3.0\ \text{m}^3\text{h}^{-1}$	$c_{p,j} = 0.200\ \text{kcal kg}^{-1}\text{K}^{-1}$
$F_j = 20.0\ \text{m}^3\text{h}^{-1}$	$\Delta H_r = 5.4 \times 10^4\ \text{kcal mol}^{-1}$
$T_{rs} = 274.4$ K	$T_{j0} = 357.5$ K
$T_{js} = 303.3$ K	$T_{A0} = 310.0$ K

5.1. Data Generation and Development of RNN Models

The first step in building the RNN models was to generate a comprehensive data set that captured the dynamics of the system in Equation (67). To implement the explicit Euler algorithm, we first needed to initialize the process model using a set of initial conditions $(C_A(0), T_r(0), T_j(0))$, where $C_A(0) \in [0, 9]$ mol/m^3, $T_r(0) \in [280, 370]$ K and $T_j(0) \in [300, 390]$ K. We conducted open-loop simulations by integrating the CSTR system defined by Equation (67) using a sufficiently small time step h_c for 10^6 random initial conditions within the specified ranges. We applied random input signals $C_{A0} \in [0.5, 7.5]$ mol/m^3 over a period of one sampling period Δ, which yielded a data set of 10^6 trajectories, each with a length equal to Δ. The data set was then divided into three sets—training, validation, and testing. The training of the models was performed using the open-source library Keras [21]. The first RNN model was designed to predict the dynamics of the full two-time-scale system in Equation (67) (i.e, it predicts the dynamics of the slow and fast states). In other words, it utilizes the current state measurements $x(t_k)$ and $z(t_k)$, and the manipulated input for the subsequent sampling period $u(t) \in [t_k, t_{k+1})$ to predict the slow and fast state measurements of $x(t)$ and $z(t)$ $\forall t \in [t_k, t_{k+1}]$. For simplicity, we denote the model that predicts the dynamics of the full two-time-scale system in Equation (67) as RNN_F, where the subscript "F" denotes "full" and not "fast". This model was designed using a single layer of 20 long short-term memory recurrent neural network (LSTM-RNN) units, and the validation and testing errors achieved were 1.434×10^{-6} and 1.432×10^{-6}, respectively. In addition, the total number of learning parameters for training this model was 2063. We note that the network was designed in a step-wise manner. Initially, we started with a simple structure, consisting of a single-layer network with one LSTM-RNN unit but the validation error was high. Gradually, we increased the network's complexity by adding more LSTM-RNN units. This progression was continued until we achieved a satisfactory level of validation loss. In this particular case, the optimal outcome was achieved using a single-layer network with 20 LSTM-RNN units. Further increasing the number of LSTM-RNN units beyond 20 resulted in an increase in the validation loss. Therefore, RNN_F was constructed using 20 LSTM-RNN units. On the other hand, the second RNN, denoted as RNN_S, was designed to predict the dynamics of only the slow states in Equation (67). Specifically, this RNN utilizes the current slow state measurement $x(t_k)$ and the manipulated input for the subsequent sampling period $u(t) \in [t_k, t_{k+1})$ to predict the slow state measurements of $x(t)$ $\forall t \in [t_k, t_{k+1}]$. This model was designed using a single layer of five LSTM-RNN units corresponding to a total of 192 learning parameters following the same step-wise manner strategy used to design RNN_F, with a validation error of 2.67×10^{-4} and a testing error of 2.68×10^{-4}. The results of both models are

summarized in Table 2. The low testing errors for both models indicate that the modeling error, as defined in Proposition 1, is sufficiently small over a wide range of open-loop trajectories. Due to having fewer learning parameters, the RNN_S model is significantly less complex compared to the RNN_F model, possibly leading to a lower computational cost and making it a more practical choice for control applications. Therefore, our aim is to investigate whether the LMPC scheme utilizing RNN_S can sufficiently meet the desired control objective more efficiently compared to the LMPC scheme using RNN_F, which is demonstrated via closed-loop simulations of the CSTR in Equation (67) under both LMPC designs. However, before proceeding, it is essential to define the Lyapunov and objective functions used in both LMPC schemes. For the RNN_F-based LMPC, since the controller design was based on the dynamics of the full two-time-scale system in Equation (67), the Lyapunov function is defined as:

$$V_F(x,z) = \begin{bmatrix} x - x_s \\ z - z_s \end{bmatrix}^T P_F \begin{bmatrix} x - x_s \\ z - z_s \end{bmatrix} \tag{69}$$

where the matrix P_F is given by

$$P_F = \begin{bmatrix} 91.6 & 2.9 & 4.3 \\ 2.9 & 0.8 & 0.2 \\ 4.3 & 0.2 & 0.7 \end{bmatrix}$$

The objective function for the RNN_F-based LMPC is given by $L_F(x,z,u) = |x|^2_{Q_{1F}} + |z|^2_{Q_{2F}} + |u|^2_{Q_{3F}}$, where $Q_{1F} = \begin{bmatrix} 20 & 0 \\ 0 & 0.05 \end{bmatrix}$, $Q_{2F} = 6$, and $Q_{3F} = 0.1$. On the other hand, the RNN_S-based LMPC was designed based on the dynamics of the slow subsystem of the CSTR, specifically Equations (67a) and (67b) expressed in the standard form. Therefore, the Lyapunov function of the slow subsystem used in the RNN_S-based LMPC is given by:

$$V(x) = (x - x_s)^T P (x - x_s) \tag{70}$$

where the matrix P is defined as

$$P = \begin{bmatrix} 84.9 & 1.2 \\ 1.2 & 0.4 \end{bmatrix}$$

The objective function is given by $L(x,u) = |x|^2_{Q_1} + |u|^2_{Q_2}$, where $Q_1 = \begin{bmatrix} 20 & 0 \\ 0 & 0.05 \end{bmatrix}$ and $Q_2 = 0.05$.

Remark 4. *In this particular application, we used long short-term memory recurrent neural network (LSTM-RNN) units to construct the models. LSTM-RNNs are a special type of RNN known for their ability to overcome the common problem of vanishing/exploding gradients in RNNs, resulting in generally better performance. For more information about LSTM-RNNs, interested readers may refer to [22]. However, we can always utilize classical RNNs that are well-suited for this particular application by choosing appropriate structures and carefully tuning the hyperparameters to obtain the desired objective, as well as an acceptable level of performance.*

Table 2. Specifications of the constructed recurrent neural network models.

RNN Model	RNN_F	RNN_S
Number of units used	20	5
Testing error	1.432×10^{-6}	2.68×10^{-4}
Validation error	1.434×10^{-6}	2.67×10^{-4}
Number of learning parameters	2063	192

5.2. Simulation Results

In this subsection, we conduct closed-loop simulations of the CSTR system described by Equation (67) under the two LMPC schemes, one with each RNN model. We first assess the ability of both LMPCs to achieve the control objective adequately compared to LMPCs that utilize the respective first-principles systems as their process models. Subsequently, we compare the RNN-based LMPCs to each other in terms of closed-loop performance and the computational time required for the simulations.

To establish the satisfactory performance of both RNN-based LMPCs, we compared their closed-loop performance to first-principles-based LMPCs, with the latter serving as the baseline for the best possible performance achievable under LMPC. Specifically, the LMPC in Equation (58) was considered in two scenarios, with the process model of Equation (58e) being different between each scenario:

- Scenario 1: The LMPC utilizing RNN_F as the process model was compared to an LMPC employing the first-principles model in Equation (67), denoted as FP_F, as its process model.
- Scenario 2: The LMPC utilizing RNN_S as the process model was compared to an LMPC employing the first-principles slow subsystem in Equations (67a) and (67b), denoted as FP_S, as its process model. In this case, for the first-principles-based LMPC, we note that the full CSTR system in Equation (67) was integrated, but only the slow states C_A and T_r from Equations (67a) and (67b) were used in calculating the LMPC cost function of Equation (58a) and the Lyapunov function V.

For both scenarios, we started the simulations from the initial condition $IC_{\text{main}} = (\Delta C_A(0), \Delta T_r(0), \Delta T_j(0)) = (1, 30, 40)$. The LMPC prediction horizon was set to $N = 3$, and the remaining controller parameters were described in Section 5.1. The state and input trajectories under the LMPCs of scenarios 1 and 2 are shown in Figures 4 and 5, respectively. To quantitatively compare the performance of the different LMPCs, the time integral of the cost function of the LMPC, $\int_{t=0}^{t_f} L(x(\tau), u(\tau))\, \mathrm{d}\tau$, was calculated over the entire simulation duration, $t_f = 3$ h. For scenario 1, the cost function values were 3458 and 3485 for the FP_F-based LMPC and the RNN_F-based LMPC, respectively, whereas for scenario 2, the values of the cost function were 176 and 179 for the FP_S-based LMPC and the RNN_S-based LMPC, respectively. For both scenarios, we noticed that the value of the cost function when utilizing the corresponding RNN model closely aligned with that achieved when employing the respective first-principles model, demonstrating the reliable predictive performance of the designed RNN models and their ability to drive the system to the steady state when incorporated into an LMPC scheme.

Next, we demonstrated the computational efficiency of the RNN_S-based LMPC due to its lower complexity while still achieving the desired controller performance. For IC_{main}, the computational time for the RNN_F-based LMPC shown in Figure 4 was 5578 s , whereas, for the RNN_S-based LMPC shown in Figure 5, it was significantly lower at 2170 s, which was 61% lower. This substantial difference in computational time can make RNN_S a more practical choice for real-time application of MPC, where computations must be completed within a sampling period to be sent to the actuator. Hence, to rigorously investigate the impact of integrating RNN_F and RNN_S into an LMPC framework in terms of stability and computational demand, we conducted closed-loop simulations from several different initial conditions in addition to IC_{main}.

The initial conditions chosen for further investigation are outlined in Table 3. Ten different initial conditions (IC_i where $i = 1, ..., 10$) within the operating region were selected, along with IC_{main}. For each initial condition, we simulated the CSTR system described in Equation (67) under the RNN_F-based LMPC, as well as the RNN_S-based LMPC. The state and input profiles under each LMPC for four representative initial conditions are depicted in Figures 6–9, which indicate that both LMPC schemes successfully drove the process in Equation (67) to its steady state. The closed-loop behavior was similar for the remaining six initial conditions in terms of convergence to the steady state. Additionally, Table 3 provides the computational times required in terms of CPU time for the entire simulation duration of $t_f = 3$ h when applying either LMPC scheme for the 11 different initial conditions. For all the tested initial conditions, the computational time required to execute the RNN_S-based LMPC was less than that required for the RNN_F-based LMPC. Among all the initial conditions in Table 3, the largest relative difference in computational time was observed for IC_3, with a percentage difference of 92% between the two LMPC schemes. Hence, the incorporation of RNN_S in the LMPC framework has been demonstrated to be more practical and computationally efficient compared to employing RNN_F, without compromising stability and closed-loop performance.

The computational time of an MPC optimization problem can be influenced by several factors. As mentioned in [23], MPC computational times are generally affected by the horizon length N, the number of constraints in the MPC optimization problem, and the number of states and control actions of the system. The computational time is also directly related to the complexity of the process model. Hence, given the lower computational times across all 11 initial conditions tested, incorporating RNN_S into an LMPC framework is a more practical approach compared to the use of RNN_F. In the study of [24], it was observed that the CPU time of the IPOPT optimization problem increased as the complexity of the problem increased. Additionally, several works have discussed computational complexity evaluations of neural networks. For instance, the authors of [25] offered a comprehensive and systematic analysis for quantifying and comparing the computational complexity of neural network layers. The authors proposed and computed three computational metrics per layer for different types of neural networks and presented the exact mathematical expressions and derivations, providing a deep understanding of various networks' complexities. Upon analyzing the results derived in [25], it is evident that a neural network's complexity is proportional to its hyperparameters. For example, factors like the number of neurons, the number of hidden units, the input time sequence size, the number of output neurons, and other parameters associated with the structure, size, and design of the network, all of which affect the complexity and performance of a network, as well as the time required to solve the ML-based MPC optimization problem. As RNN_S consisted of only 5 LSTM-RNN units while RNN_F consisted of 20 LSTM-RNN units, the complexity of RNN_F was much higher than that of RNN_S, leading to a correspondingly complex MPC optimization problem. Hence, over the entire simulation duration of $t_f = 3$ h, the total CPU time required to solve all 100 MPC optimization problems over the 100 sampling periods was significantly less for the RNN_S-based LMPC than the RNN_F-based LMPC.

Remark 5. *Although the state and input trajectories in Figure 6 have not completely settled, the process is stabilized under both the RNN_F-based LMPC and the RNN_S-based LMPC. However, the process requires simulation for a longer duration, exceeding 3 h, for the states and input to ultimately converge to their steady-state values and remain close to the origin. The reason for limiting the simulation time is to ensure fair and clear comparisons between the computational times for both controllers from various initial conditions, as listed in Table 3.*

Remark 6. *In this particular application, RNN_S was designed to utilize the current slow state measurement and the manipulated input for the next sampling period. As a result, the output of RNN_S will be the predicted slow states x for only one sampling period ahead (1Δ). Consequently, in this application, since the prediction horizon was chosen to be an integer multiple of the sampling*

time Δ *(i.e.,* $N = 3$ *implies that* $t_{k+N} = t_k + 3\Delta$ *in Equation (58a)), based on the design of* RNN_S*,* RNN_S *is required to be invoked at least three times to be able to make predictions for the entire prediction horizon. Additionally, an alternative method involves designing an RNN model, denoted as* RNN_{new}*, with the same goal of predicting the slow states. However,* RNN_{new} *utilizes not only the current slow state measurements and the manipulated input for the next sampling period but also the current fast state measurements to predict the slow states one sampling period* Δ *ahead, i.e.,* $x(t_{k+1})$*. To obtain predictions for the entire prediction horizon (*$N = 3$*), an FNN takes as its input the output of* RNN_{new} *and predicts the fast state* z *at the next sampling period, i.e., the FNN predicts* $z(t_{k+1})$ *from* $x(t_{k+1})$*, following the design described in Section 2.5. Subsequently, for the next sampling period, the outputs of both* RNN_{new} *and the FNN are given as inputs to* RNN_{new}*, enabling it to predict the slow states* $x(t_{k+2})$ *for the next sampling period in the prediction horizon.*

Table 3. Computational times for the RNN_F-based LMPC and the RNN_S-based LMPC over the simulation duration of $t_f = 3$ h starting from various initial conditions within the operating region.

Index	Initial Condition	Computational Time (s)	
	$(\Delta C_A(0), \Delta T_r(0), \Delta T_j(0))$	RNN_F-Based LMPC	RNN_S-Based LMPC
IC_{main}	(1, 30, 40)	5578	2170
IC_1	(−1, 50, 40)	16,059	1807
IC_2	(−1, −10, −3)	4801	2667
IC_3	(−3, 30, 5)	31,884	2417
IC_4	(−2, −10, 100)	17,896	1921
IC_5	(1, 90, 10)	6078	1990
IC_6	(1, 20, 60)	5795	2404
IC_7	(3, −6, 20)	21,231	2411
IC_8	(1, 50, 50)	5161	2225
IC_9	(−2, 30, 60)	10,275	2194
IC_{10}	(−1, 10, 80)	18,146	2106

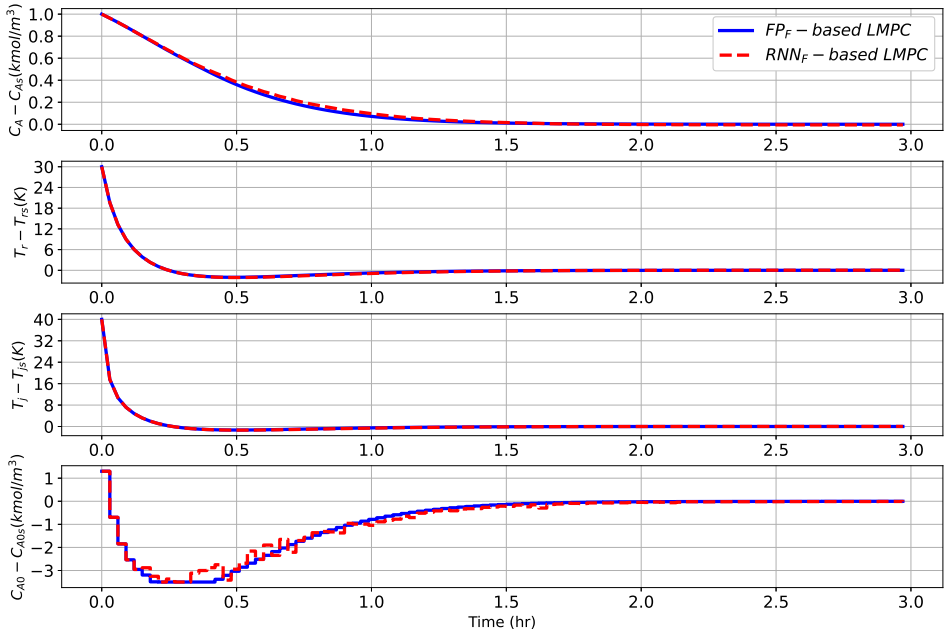

Figure 4. States and input trajectories of the CSTR under the Lyapunov-based MPC scheme using the first-principles model of the full process (FP_F-based LMPC, blue line) and RNN_F (RNN_F-based LMPC, red dashed line).

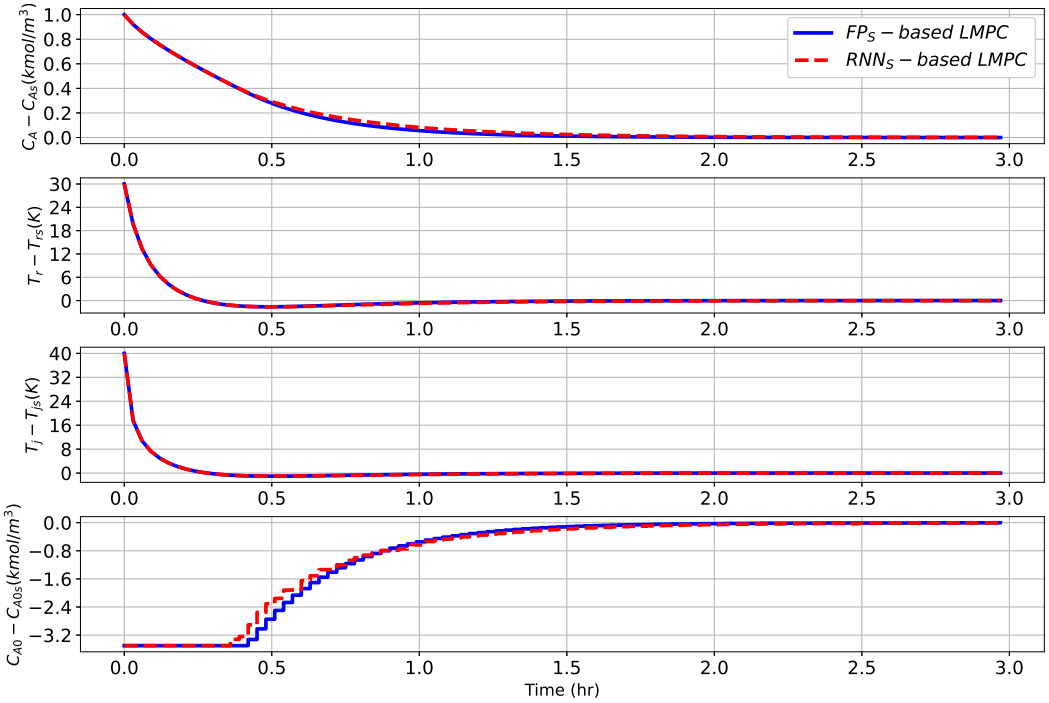

Figure 5. States and input trajectories of the CSTR under the Lyapunov-based MPC scheme using the first-principles model of the slow-subsystem (FP_S-based LMPC, blue line) and RNN_S (RNN_S-based LMPC, red dashed line).

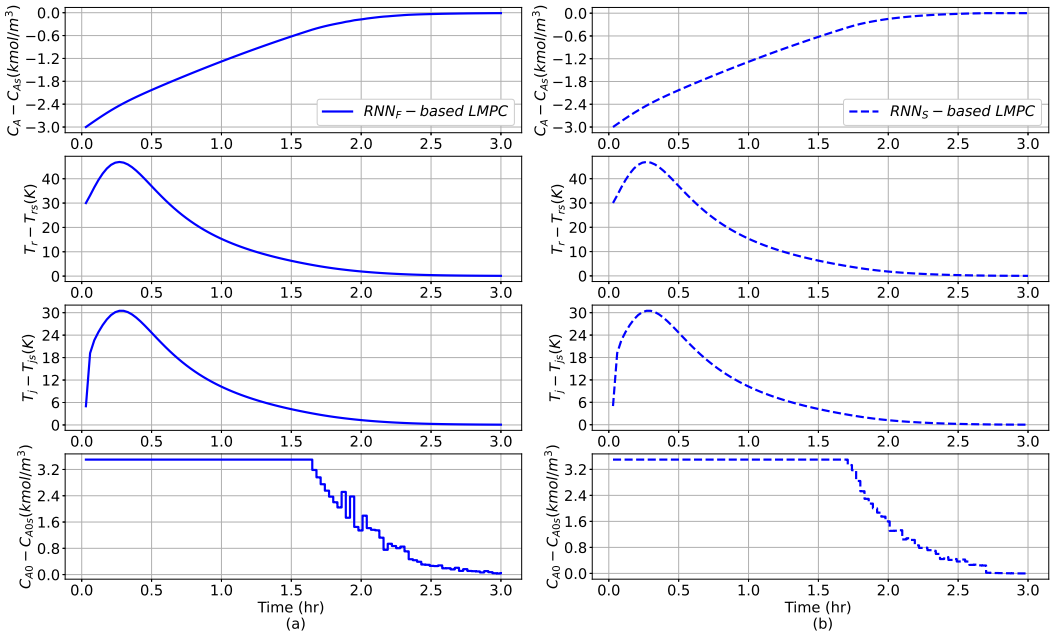

Figure 6. Considering the initial condition $IC_3 = (-3, 30, 5)$, (**a**) illustrates the time-varying profiles of the states and the input under the RNN_F-based LMPC (solid line), whereas (**b**) shows the time-varying profiles of the states and the input under the RNN_S-based LMPC (dashed line).

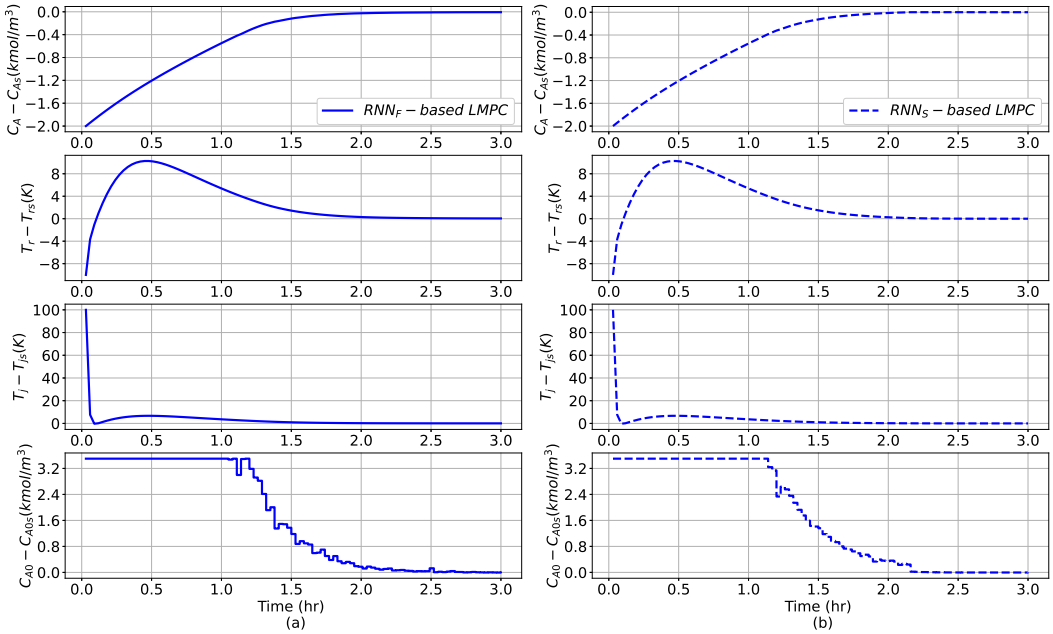

Figure 7. Considering the initial condition $IC_4 = (-2, -10, 100)$, (**a**) illustrates the time-varying profiles of the states and the input under the RNN_F-based LMPC (solid line), whereas (**b**) shows the time-varying profiles of the states and the input under the RNN_S-based LMPC (dashed line).

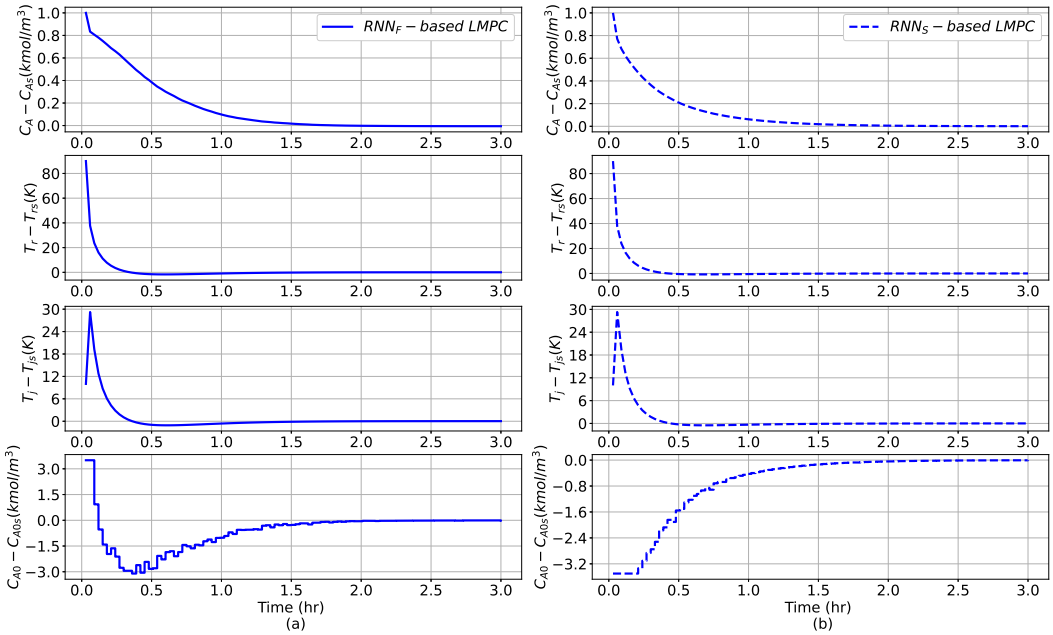

Figure 8. Considering the initial condition $IC_5 = (1, 90, 10)$, (**a**) illustrates the time-varying profiles of the states and the input under the RNN_F-based LMPC (solid line), whereas (**b**) shows the time-varying profiles of the states and the input under the RNN_S-based LMPC (dashed line).

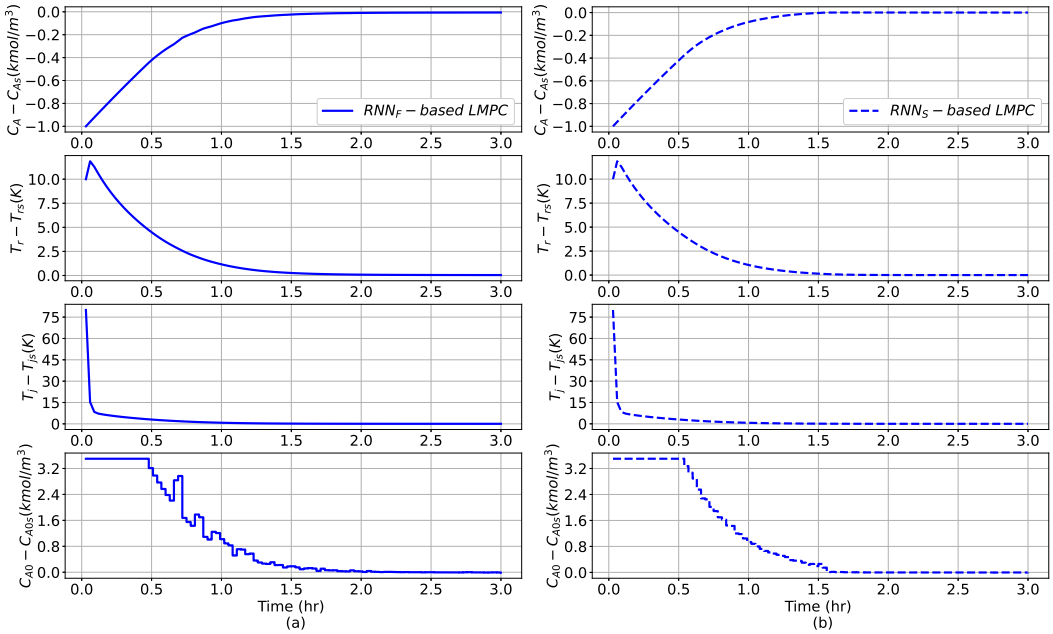

Figure 9. Considering the initial condition $IC_{10} = (-1, 10, 80)$, (**a**) illustrates the time-varying profiles of the states and the input under the RNN_F-based LMPC (solid line), whereas (**b**) shows the time-varying profiles of the states and the input under the RNN_S-based LMPC (dashed line).

6. Conclusions

In this work, we introduced a general class of nonlinear two-time-scale systems, where the two distinct time scales can be decoupled into slow subsystem and fast subsystem dynamics using singular perturbation analysis. Machine learning was employed to approximate the dynamics of both subsystems. In particular, an RNN was used to predict the slow state vector or, in other words, to approximate the dynamics of the slow subsystem, whereas an FNN was used to approximate the dynamics of the fast subsystem. To draw inferences about the performance of the neural network models on unseen data, the generalization error bounds for both neural networks when modeling two-time-scale systems were derived. Subsequently, the RNN modeling of the slow states (RNN_S) was incorporated into an LMPC framework and sufficient conditions to guarantee closed-loop stability were derived under the sample-and-hold implementation of the LMPC. Finally, a chemical process example was used to demonstrate the efficiency of the LMPC scheme using an RNN to predict the slow states compared to first-principles-based LMPC schemes and an LMPC scheme using an RNN to model the full two-time-scale system. Through closed-loop simulations, while both RNN-based LMPCs were able to achieve the desired control objective of driving the system to the steady state, due to the significantly lower complexity of the RNN approximating only the slow subsystem, the LMPC based on RNN_S required significantly less computational time in all simulations compared to the LMPC using the RNN modeling the full singularly perturbed system, which supported the use of RNN_S in real-time MPC applications.

Author Contributions: Conceptualization, A.A., F.A., A.S. and P.D.C.; methodology, A.A., F.A., A.S. and P.D.C.; software, A.A., F.A. and A.S.; validation, A.A., F.A. and A.S.; formal analysis, A.A., F.A. and A.S.; investigation, A.A., F.A. and A.S.; resources, P.D.C.; data curation, A.A., F.A. and A.S.; writing—original draft preparation, A.A., F.A. and A.S.; writing—review and editing, A.A., F.A., A.S. and P.D.C.; supervision, P.D.C.; project administration, P.D.C.; funding acquisition, P.D.C. All authors have read and agreed to the published version of the manuscript.

Funding: Financial support from the National Science Foundation, CBET-CBET-2140506, and the Department of Energy, through the Office of Energy Efficiency and Renewable Energy (EERE) under the Advanced Manufacturing Office Award Number DEEE0007613, is gratefully acknowledged. The first author acknowledges the support of Kuwait University via the KU scholarship program.

Data Availability Statement: Data is available upon request to the corresponding author.

Conflicts of Interest: The authors declare that they have no conflicts of interest regarding the publication of this research article.

References

1. Christofides, P.D.; Daoutidis, P. Feedback control of two-time-scale nonlinear systems. *Int. J. Control* **1996**, *63*, 965–994.
2. Kokotović, P.; Khalil, H.K.; O'reilly, J. *Singular Perturbation Methods in Control: Analysis and Design*; SIAM: Philadelphia, PA, USA, 1999.
3. Chen, X.; Heidarinejad, M.; Liu, J.; Christofides, P.D. Composite fast-slow MPC design for nonlinear singularly perturbed systems. *AIChE J.* **2012**, *58*, 1802–1811. [CrossRef]
4. Ellis, M.; Heidarinejad, M.; Christofides, P.D. Economic model predictive control of nonlinear singularly perturbed systems. *J. Process Control* **2013**, *23*, 743–754. [CrossRef]
5. Abdullah, F.; Wu, Z.; Christofides, P.D. Sparse-identification-based model predictive control of nonlinear two-time-scale processes. *Comput. Chem. Eng.* **2021**, *153*, 107411. [CrossRef]
6. Wu, Z.; Tran, A.; Rincon, D.; Christofides, P.D. Machine Learning-Based Predictive Control of Nonlinear Processes. Part II: Computational Implementation. *AIChE J.* **2019**, *65*, e16734. [CrossRef]
7. Nikolakopoulou, A.; Braatz, R.D. Polynomial NARX-based nonlinear model predictive control of modular chemical systems. *Comput. Chem. Eng.* **2023**, *177*, 108272. [CrossRef]
8. Yang, S.; Wan, M.P.; Chen, W.; Ng, B.F.; Dubey, S. Experiment study of machine-learning-based approximate model predictive control for energy-efficient building control. *Appl. Energy* **2021**, *288*, 116648. [CrossRef]
9. Alessio, A.; Bemporad, A. A survey on explicit model predictive control. In *Nonlinear Model Predictive Control: Towards New Challenging Applications*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 345–369.
10. Chen, S.; Saulnier, K.; Atanasov, N.; Lee, D.D.; Kumar, V.; Pappas, G.J.; Morari, M. Approximating explicit model predictive control using constrained neural networks. In Proceedings of the 2018 Annual American Control Conference (ACC), Milwaukee, WI, USA, 27–29 June 2018; pp. 1520–1527.
11. Bao, Y.; Chan, K.J.; Mesbah, A.; Velni, J.M. Learning-based adaptive-scenario-tree model predictive control with improved probabilistic safety using robust Bayesian neural networks. *Int. J. Robust Nonlinear Control* **2023**, *33*, 3312–3333. [CrossRef]
12. Wu, Z.; Rincon, D.; Gu, Q.; Christofides, P.D. Statistical machine learning in model predictive control of nonlinear processes. *Mathematics* **2021**, *9*, 1912. [CrossRef]
13. Chen, S.; Wu, Z.; Christofides, P.D. Statistical Machine-Learning-based Predictive Control Using Barrier Functions for Process Operational Safety. *Comput. Chem. Eng.* **2022**, *163*, 107860. [CrossRef]
14. Alhajeri, M.S.; Alnajdi, A.; Abdullah, F.; Christofides, P.D. On generalization error of neural network models and its application to predictive control of nonlinear processes. *Chem. Eng. Res. Des.* **2023**, *189*, 664–679. [CrossRef]
15. Hoppensteadt, F. Properties of solutions of ordinary differential equations with small parameters. *Commun. Pure Appl. Math.* **1971**, *24*, 807–840. [CrossRef]
16. Mohri, M.; Rostamizadeh, A.; Talwalkar, A. *Foundations of Machine Learning*; MIT Press: Cambridge, MA, USA, 2018.
17. Golowich, N.; Rakhlin, A.; Shamir, O. Size-independent sample complexity of neural networks. In Proceedings of the Conference on Learning Theory, Stockholm, Sweden, 5–9 July 2018; pp. 297–299.
18. Bao, Y.; Abbas, H.S.; Velni, J.M. A learning- and scenario-based MPC design for nonlinear systems in LPV framework with safety and stability guarantees. *Int. J. Control* 2023, *in press*. [CrossRef]
19. Christofides, P.D.; Teel, A.R. Singular perturbations and input-to-state stability. *IEEE Trans. Autom. Control* **1996**, *41*, 1645–1650. [CrossRef]
20. Wächter, A.; Biegler, L.T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* **2006**, *106*, 25–57. [CrossRef]
21. Chollet, F. Keras. 2015. Available online: https://keras.io (accessed on 11 August 2023).
22. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
23. Rawlings, J.B.; Mayne, D.Q.; Diehl, M. *Model Predictive Control: Theory, Computation, and Design*; Nob Hill Publishing: Madison, WI, USA, 2017; Volume 2.
24. Pöri, L. Comparison of Two Interior Point Solvers in Model Predictive Control Optimization. Master's Thesis, Aalto University, Espoo, Finland, 2016.
25. Freire, P.J.; Srivallapanondh, S.; Napoli, A.; Prilepsky, J.E.; Turitsyn, S.K. Computational complexity evaluation of neural network applications in signal processing. *arXiv* **2022**, arXiv:2206.12191.