Online Machine Learning Modeling and Predictive Control of Nonlinear Systems With Scheduled Mode Transitions

Cheng Hu, Yuan Cao, Zhe Wu*

June 20, 2022

Abstract

This work develops a model predictive control (MPC) scheme using online learning of recurrent neural network (RNN) models for nonlinear systems switched between multiple operating regions following a prescribed switching schedule. Specifically, an RNN model is initially developed offline to model process dynamics using the historical operational data collected in a small region around a certain steady-state. After the system is switched to another operating region under a Lyapunov-based MPC with suitable constraints to ensure satisfaction of the prescribed switching schedule policy, RNN models are updated using real-time process data to improve closed-loop performance. A generalization error bound is derived for the updated RNN models using the notion of regret, and closed-loop stability results are established for the switched nonlinear system under RNN-based MPC. Finally, a chemical process example with the operation schedule that requires switching between two steady-states is used to demonstrate the effectiveness of the proposed RNN-MPC scheme.

Keywords: Online Machine Learning; Recurrent Neural Networks; Generalization Error; Model Predictive Control; Nonlinear Systems

Introduction

Over the past decades, modeling large-scale, complex nonlinear dynamic systems has received considerable attention in process systems engineering. While nonlinear first-principles process modeling

*Cheng Hu and Zhe Wu are with the Department of Chemical and Biomolecular Engineering, National University of Singapore, 117585, Singapore. Yuan Cao is with Department of Statistics and Actuarial Science and Department of Mathematics, The University of Hong Kong, Hong Kong. Emails: hucheng@nus.edu.sg; yuancao@hku.hk; wuzhe@nus.edu.sg. Corresponding author: Zhe Wu.

provides a direct way to model nonlinear dynamic systems based on the underlying physio-chemical phenomena, it is cumbersome and difficult to implement in complex industrial processes which are not well-understood. Machine learning modeling approaches have gained increasing attention in recent years, since they are able to model nonlinear processes efficiently with big data obtained from industrial/simulation data. In addition to computer science, machine learning methods have been adopted by researchers in the fields of classical engineering to tackle classification and regression problems ([8,33]).

Among many machine learning modeling methods, recurrent neural networks (RNNs) have been increasingly utilized in machine learning-based control of chemical processes due to their ability to model nonlinear dynamic systems using time-series data. For example, RNNs were used to derive a nonlinear prediction model for model predictive control (MPC) that stabilizes nonlinear chemical processes at their steady-states in.[33,34] One key challenge for the practical implementation of MPC is the accuracy of the nonlinear prediction model. As discussed in,[33,34] the RNN model was trained offline using the data collected from the entire operating region in order to fully capture the dynamics of nonlinear systems. However, the requirement for training data over the entire operating region may not be realizable since real-world chemical processes are often operated within a small region around the steady-state for safe and stable operation, and thus, only very limited data around the steady-state is available for training in practice. Additionally, since machine learning models are generally developed offline using historical data from past normal operations that do not involve model uncertainty, the resulting machine learning models may not be able to accurately predict real-time process dynamics in the presence of model uncertainty. To that end, online learning provides a promising solution to improve machine learning models using real-time data for better control performance under MPC. Compared to traditional offline machine learning algorithms that learn a model using the entire training dataset at once, online learning algorithms can instantly and efficiently update the model to capture the latest process dynamics when the learner receives new training data sequentially.

Online learning algorithms have been widely used to develop machine learning models for large-scale problems with a tremendous amount of data, since the training process is more computationally efficient than batch algorithms ([23]). In addition to the considerations of computational efficiency, online learning of machine learning models has demonstrated its benefits in improving model prediction and closed-loop performance in many real-time control systems ([2,11,24,27,31,35]). However, an ongoing challenge for the practical implementation of online learning models in real-world chemical processes is their generalization performance on unseen data, for which a fundamental understand-

ing needs to be developed. Generalization error bound is commonly used in statistical machine learning to quantitatively characterize the generalization performance of machine learning models. Many recent efforts have focused on the generalization analysis of neural networks in classification and regression problems. For example, a sample complexity bound was derived for feedforward neural networks in.[14] A generalization error bound was developed for RNNs that approximate the dynamics of nonlinear systems with a single output in.[15] The generalization error bound for RNNs was established for multiclass classification problems in.[9] Additionally, the generalization error bounds were developed for over-parameterized deep neural networks trained using stochastic gradient descent in[3,4] and gradient descent in,[5] respectively. More recently, a generalization error bound was derived for the offline learning RNNs that model a general class of nonlinear systems, and probabilistic closed-loop stability results were developed for the MPC using RNN models in.[30,32]

The objective of an online learner is to minimize regret, which is defined as the difference between the cumulative loss of an online learner and that of the single best learner chosen in hindsight given all the data. Online-to-batch conversion is a common technique in machine learning theory to evaluate the generalization performance of online learning algorithms, which has been extensively studied in.[6,7,17,20] Additionally, by using online-to-batch conversion techniques, many common learning problems that fit more naturally in the batch learning setting can be addressed through online learning algorithms to develop models in a more memory- and time-efficient manner ([12]). It is demonstrated that the generalization performance of the online learning models is competitive with the best model chosen in hindsight using all the data provided that the regret of the online learning algorithms grows sublinearly. Therefore, many stochastic optimization methods have been developed showing that online convex optimization problems such as AdaGrad,[13] RMSProp,[28] Adam,[18] and AMSGrad,[25] have a sublinear regret bound. Additionally, as online learning algorithms will constantly generate new machine learning models as data comes in a sequential order, how to make the best use of all the models collected over time also plays an important role in improving the generalization performance of online learning models. Ensemble learning that uses multiple learners obtained to solve the same problem has demonstrated its better predictive performance than that can be achieved by any of the constituent learning algorithms alone ([21]). However, at this stage, the generalization performance of ensemble online learning models has not been sufficiently studied yet. Moreover, despite an increasing number of literature investigating generalization error bound and regret analysis of online learning algorithms, stability analysis for the MPC using online learning of RNN models has not been investigated.

Motivated by the above, this work utilizes an online learning algorithm to update RNN models

using real-time process data and incorporates online learning models into the MPC design for the switched nonlinear systems that are required to operate in different operating regions at prescribed switching times. Closed-loop stability analysis is performed accounting for the generalization error bound derived for online learning models. Specifically, in section "Preliminaries", the notations, class of nonlinear systems, stabilizability assumptions, and the general structure of RNNs are introduced. In section "Online Learning of Recurrent Neural Networks", a brief recap of the generalization error bound for offline learning RNN models is presented, followed by the derivation of a generalization error bound for online learning models based on regret analysis. In section "RNN-based LMPC of Switched Nonlinear Systems", Lyapunov-based MPCs using RNN models are developed for the nonlinear system operated in a fixed operating region and switched different operating regions, respectively. Subsequently, the implementation strategies for the integration of online learning within RNN-based MPC are presented with the closed-loop stability analysis. Finally, the proposed RNN-based MPC scheme is applied to a chemical process to demonstrate its effectiveness in section "Application to a Chemical Process Example".

Preliminaries

Notation

We use $\|A\|_F$ and $|\cdot|$ to denote the Frobenius norm of A and the Euclidean norm of a vector, respectively. The transpose of x is denoted by x^T. The notation $L_f V(x)$ is used to denote the standard Lie derivative $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$. The operator "\" denotes set subtraction, i.e., $A \backslash B := \{x \in \mathbb{R}^n \mid x \in A, x \notin B\}$. A function $f(\cdot)$ is of class $\mathcal{C}^1$ if it is continuously differentiable. A continuous function $\alpha : [0, a) \to [0, \infty)$ belongs to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be L-Lipschitz, $L \geq 0$, if $|f(a) - f(b)| \leq L|a - b|$ for all $a, b \in \mathbb{R}^n$. $\mathbb{P}(A)$ denotes the probability that event A will occur. The expected value of a random variable X is denoted by $\mathbb{E}[X]$. Given two sequences a_n and b_n, we have $a_n = \mathcal{O}(b_n)$ if $\limsup_{n\to\infty} |a_n/b_n| < \infty$.

Class of nonlinear systems

Consider the class of continuous-time nonlinear systems described by the following state-space form:

$$\dot{x} = F(x, u) := f(x) + g(x)u, \; x(t_0) = x_0 \tag{1}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^{n_u}$ are the state vector and the control input vector. The control input is constrained by $u \in U := \{u_{min} \leq u \leq u_{max}\} \subset \mathbb{R}^{n_u}$, where u_{min} and u_{max} denote the minimum and maximum value vectors of the input constraint. Throughout this manuscript, it is assumed that the

initial time t_0 is set to zero ($t_0 = 0$). Additionally, we assume that the vector function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ and the matrix function $g(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n\times n_u}$ are sufficiently smooth. The nonlinear system of Eq. (1) is assumed to have multiple steady-states x_{s_k} under $u = u_s$ (i.e., $\dot{x}_{s_k} = f(x_{s_k}) + g(x_{s_k})u_s \equiv 0$), where $k \in \ell = \{1, 2, \ldots, p\}$ and p is the number of steady-states. The nonlinear system of Eq. (1) is said to operate in mode k if it is required to be stabilized at the steady-state x_{s_k}, $\forall k \in \ell$. Additionally, we use t_k^{in} and t_k^{out} to represent the time when the k^{th} mode is switched in and out, respectively. When the nonlinear system is operated in mode k, i.e., $t \in [t_k^{in}, t_k^{out})$, we assume that the state measurements are available every sampling time $t_q = t_k^{in} + q\Delta$, $q = 0, 1, \cdots, N_k$, where Δ is the sampling period and N_k is the number of sampling periods within t_k^{in} and t_k^{out}.

Stabilization via control Lyapunov function

Consider the nonlinear system of Eq. (1) with multiple steady-states x_{s_k}, $k \in \ell$. Using the deviation variables $z_k := x - x_{s_k}$, $k \in \ell$, the nonlinear system of Eq. (1) can be written in the deviation form of $\dot{z}_k = F_k'(z_k, u)$ such that $F_k'(0, u) = 0$ under $u = u_s$. We assume that for each steady-state x_{s_k}, $k \in \ell$, there exists a stabilizing feedback controller $u_k = \Phi_k(z_k) \in U$ such that the origin of the nonlinear system of $\dot{z}_k = F_k'(z_k, u)$ is rendered exponentially stable. This stabilizability assumption implies that for each steady-state x_{s_k}, $k \in \ell$, there exists a $\mathcal{C}^1$ control Lyapunov function $V_k(z_k)$ such that the following inequalities hold for all z_k in an open neighborhood D_k around the origin:

$$c_{1_k}|z_k|^2 \leq V_k(z_k) \leq c_{2_k}|z_k|^2 \tag{2a}$$

$$\frac{\partial V_k(z_k)}{\partial z_k} F_k'(z_k, \Phi_k(z_k)) \leq -c_{3_k}|z_k|^2 \tag{2b}$$

$$\left|\frac{\partial V_k(z_k)}{\partial z_k}\right| \leq c_{4_k}|z_k| \tag{2c}$$

where $c_{1_k}, c_{2_k}, c_{3_k}$, and c_{4_k}, $k \in \ell$, are positive constants. We use a level set of Lyapunov function described by $\Omega_{\rho_k} := \{z_k \in D_k \mid V_k(z_k) \leq \rho_k\}, \rho_k > 0$, $k \in \ell$, to denote an estimate of the closed-loop stability region for the nonlinear system of $\dot{z}_k = F_k'(z_k, u)$ under $u_k = \Phi_k(z_k) \in U$. Additionally, due to the smoothness of $f(\cdot)$ and $g(\cdot)$ and the boundedness of u, for all $z_k, z_k' \in \Omega_{\rho_k}, u \in U$, $k \in \ell$, there exist positive constants M_k, L_k, L_k' such that the following inequalities hold:

$$|F_k'(z_k, u)| \leq M_k \tag{3a}$$

$$|F_k'(z_k, u) - F_k'(z_k', u)| \leq L_k |z_k - z_k'| \tag{3b}$$

$$\left|\frac{\partial V_k(z_k)}{\partial z_k} F_k'(z_k, u) - \frac{\partial V_k(z_k')}{\partial z_k} F_k'(z_k', u)\right| \leq L_k' |z_k - z_k'| \tag{3c}$$

Recurrent neural networks

In this work, we consider a general RNN model with the following form to approximate the dynamics of the nonlinear system of Eq. (1):

$$\mathbf{h}_t = \sigma_h \left(Q\mathbf{h}_{t-1} + W\mathbf{x}_t \right), \quad \mathbf{y}_t = \sigma_y \left(V\mathbf{h}_t \right) \tag{4}$$

where $\mathbf{x}_t \in \mathbb{R}^{d_x}$, $\mathbf{y}_t \in \mathbb{R}^{d_y}$, and $\mathbf{h}_t \in \mathbb{R}^{d_h}$ denote the input, the output, and the hidden state of the RNN model at the t^{th} time step, respectively, where $t = 1, \ldots, T_{nn}$. All vectors of the RNN model are written in boldface to differentiate the notation of input, output, and state between the RNN model of Eq. (4) and the nonlinear system of Eq. (1). $W \in \mathbb{R}^{d_h \times d_x}$, $V \in \mathbb{R}^{d_y \times d_h}$, and $Q \in \mathbb{R}^{d_h \times d_h}$ are the weight matrices for the input layer, the output layer, and the hidden layer, respectively. σ_h is the nonlinear element-wise activation function for the hidden layer (e.g., ReLU). The activation function of the output layer is denoted by σ_y; a typical choice of σ_y is a linear unit for regression problems. The datasets for offline training of RNNs can be generated following the simulation-based data generation method in.[33] Specifically, extensive open-loop simulations of the nonlinear system of Eq. (1) are carried out under various initial conditions $(x_0 - x_{s_k}) \in \Omega_{\rho_k}$ and control actions $u \in U$. Each simulation runs for a fixed period of time, e.g., one sampling period, where the control action u is implemented in a sample-and-hold fashion (i.e., $u(t) = u\left(t_q\right), \forall t \in [t_q, t_{q+1})$, where $t_{q+1} := t_q + \Delta$, and Δ is the sampling period), and the system of Eq. (1) is solved using the explicit Euler method with a sufficiently small integration time step $\bar{h}_c < \Delta$. Based on open-loop simulation data, the RNN model is developed to predict future states $x(t)$, $\forall t \in [t_q, t_{q+1})$, using the current state measurement $x\left(t_q\right)$ and the control input $u(t)$, $t \in [t_q, t_{q+1})$. Therefore, the RNN input $\mathbf{x} = [x(t_q)\ u(t)]$ consists of the state at the current time step $t = t_q$ and the control action that will be applied for $[t_q, t_{q+1})$, and the predicted states $x(t)$, $t \in [t_q, t_{q+1})$ including those in the integration time steps are the RNN output $\mathbf{y}$. Since the control input $u(t)$ is implemented in a sample-and-hold fashion, the RNN input $\mathbf{x}$ remains unchanged for one sampling period. Additionally, $\mathbf{x}_t$ and $\mathbf{y}_t$ represent the RNN input and output at the t^{th} time step, where each time step corresponds to one integration time step in this work, and thus, the total number of time steps is $T_{nn} = \frac{\Delta}{\bar{h}_c}$. We consider the mean squared error (MSE) as the loss function $L(h\left(\mathbf{x}_t\right), \bar{\mathbf{y}}_t)$ for the RNN model, where $\mathbf{y}_t = h(\mathbf{x}_t)$ and $\bar{\mathbf{y}}_t$ are the predicted output and the true output values, respectively, and $h(\cdot)$ denotes the RNN model chosen from the hypothesis class $\mathcal{H}$ that maps an input $\mathbf{x}_t \in \mathbb{R}^{d_x}$ to an output $\mathbf{y}_t \in \mathbb{R}^{d_y}$. We have the following assumptions on the development of RNN models:

Assumption 1. *The RNN inputs are bounded, i.e.,* $|\mathbf{x}_t| \leq B_X$, *for all* $t = 1, \ldots, T_{nn}$.

Assumption 2. *The Frobenius norms of all the weight matrices are bounded as follows:*

$$\|W\|_F \leq B_{W,F}, \|V\|_F \leq B_{V,F}, \|Q\|_F \leq B_{Q,F} \tag{5}$$

Assumption 3. *The datasets for training RNN models are drawn from the same distribution.*

Remark 1. The above assumptions are standard in the machine learning framework. Specifically, Assumption 1 requires the boundedness of the RNN input, which follows directly from the fact that the state and the control input of the nonlinear system of Eq. (1) are bounded by $(x - x_{s_k}) \in \Omega_{\rho_k}$ and $u \in U$. The boundedness of the RNN weight matrices in Assumption 2 is consistent with the practical training process and also implies that the predicted output is bounded. Assumption 3 implies that the offline and online RNN models developed from historical and real-time process data will be applied to the same process with the same data distribution.

Online Learning of Recurrent Neural Networks

Since the evaluation of any learning algorithm is based on finite training samples only, the training error measured on the training data may not provide sufficient information on its predictive capacity on unseen data. Therefore, generalization error is generally used in statistical learning theory to measure how accurately a neural network model learned from training data can generalize to new data that has not been seen previously. Therefore, the aim of many learning algorithms is to obtain an upper bound on the generalization error. In this section, the notion of generalization error and empirical error is first introduced, followed by a brief discussion on the generalization error bound derived for the offline learning RNN models. Then, we develop a generalization error bound for online learning models based on regret bound analysis.

Generalization error bound for offline learning

Given a data distribution $\mathcal{D}$ and a function h that predicts the output y for the input x, the generalization error is defined as follows:

$$\mathbb{E}[L(h(x), y)] = \int_{X \times Y} L(h(x), y)\rho(x, y)dxdy \tag{6}$$

where $\rho(x, y)$ represents the joint probability distribution for x and y, and X, Y denote the vector space for all possible inputs and outputs, respectively. Since the joint probability distribution ρ is unknown in most cases, we can instead calculate the empirical error based on m labeled samples $(x_1, y_1), \cdots, (x_m, y_m)$ drawn from the same distribution $\mathcal{D}$ to approximate the generalization error in Eq. (6). The empirical error is defined as follows:

$$\hat{\mathbb{E}}_S[L(h(x), y)] = \frac{1}{m}\sum_{i=1}^{m} L(h(x_i), y_i) \tag{7}$$

Offline training has been widely used to develop machine learning models using historical process data by minimizing the empirical error of Eq. (7). The following lemma was developed in[32] to derive a generalization error bound for offline-trained RNN models.

Lemma 1 (c.f. Theorem 1 in[32]). Let $\mathcal{G}_t = \{g_t : (\mathbf{x}, \mathbf{y}) \to L(h(\mathbf{x}), \mathbf{y}), h \in \mathcal{H}\}$ be the loss function class associated with the RNN hypothesis class $\mathcal{H}$ that predicts the RNN output at the t^{th} time step, with 1-Lipschitz continuous and positive-homogeneous activation functions as well as weight matrices satisfying Assumptions 1-3. Given m sequences of T_{nn}-time-length independent and identically distributed (i.i.d.) data points $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T_{nn}}$, $i = 1, \ldots, m$ and $t = 1, \ldots, T_{nn}$, with probability at least $1 - \delta$ over S, the following inequality holds:

$$\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] \leq \frac{1}{m} \sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \mathcal{O}\left(L_r d_y \frac{M B_X (1 + \sqrt{2\log(2)t})}{\sqrt{m}} \right) \tag{8}$$

where $M = B_{W,F} B_{V,F} \frac{1-(B_{Q,F})^t}{1-B_{Q,F}}$, and $B_{W,F}$, $B_{V,F}$, $B_{Q,F}$, B_X are the bounds of RNN weight matrices and RNN inputs defined in Assumptions 1-3. L_r is the local Lipschitz constant for the MSE loss function and d_y is the dimension of the RNN output.

Remark 2. Lemma 1 demonstrates that the generalization error of offline learning RNN models is bounded by the three terms on the right-hand side (RHS) of Eq. (8). Specifically, the generalization error depends on the training error (first term), confidence level δ and sample size m (second term), as well as the bounds of RNN weight matrices/inputs and the time length t of RNN inputs (last term). Interested readers may refer to[32] for detailed discussion and derivations of Eq. (8).

Online machine learning

As discussed in the data generation method for training RNN models in,[33] extensive open-loop simulations of the nonlinear system of Eq.(1) are conducted by sweeping over all possible values of $(x - x_{s_k}) \in \Omega_{\rho_k}$ and $u_k \in U$ for $k \in \ell$ in order to capture the dynamics of the nonlinear system of Eq.(1) accurately. However, this dataset generation method may not be feasible in practice, as chemical processes are often required to operate in a small region around the steady-state x_{s_k} to ensure safe and stable operation. When the RNN model is trained offline using a limited dataset collected around the steady-state x_{s_k}, the RNN model may not well capture the nonlinear dynamics within the entire operating region. Furthermore, real-world chemical processes are often required to execute mode transitions for considerations of economics and safety, i.e., the processes are switched from the current operating region around x_{s_k} to a subsequent operating region around another steady-state x_{s_f} for some $k, f \in \ell$. This mode transition may lead to an undesired predictive performance of the RNN model trained offline since it lacks the data around x_{s_f}. To address this issue, online learning can be used to update RNN models during operation using real-time process data. In this section, we will use statistical learning theory and online learning algorithms to develop the generalization error bound for online learning of RNN models. We first introduce the following

lemma that will be used in the derivation of the generalization error bound for online learning.

Lemma 2. (Azuma's inequality, c.f. Theorem D.7 in[23]) Let $V_1, \ldots, V_m$ be a martingale difference sequence with respect to a sequence of random variables $X_1, \ldots, X_m$. Assume that for all $i > 0$, there exist a constant $c_i \geq 0$ and a random variable Z_i that is a function of $X_1, \ldots, X_{i-1}$, such that $Z_i \leq V_i \leq Z_i + c_i$. Then, for all m and $\epsilon > 0$, the following inequalities hold:

$$\mathbb{P}\left[\sum_{i=1}^{m} V_i \geq \epsilon\right] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{m} c_i^2}\right), \ \mathbb{P}\left[\sum_{i=1}^{m} V_i \leq -\epsilon\right] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{m} c_i^2}\right) \tag{9}$$

Consider a set of $\mathcal{T}$ labeled samples collected in $\mathcal{T}$ rounds (x_1, y_1), ..., $(x_\mathcal{T}, y_\mathcal{T})$ that are drawn from a distribution $\mathcal{D}$, and the samples are sequentially processed by an online algorithm $\mathcal{A}$ to generate a sequence of hypotheses h_1, ..., $h_\mathcal{T}$. The algorithm incurs a loss $L(h_\tau(x_\tau), y_\tau)$ in each round $\tau = 1, \ldots, \mathcal{T}$, and the regret of the algorithm $\mathcal{A}$ after $\mathcal{T}$ rounds is defined as follows:

$$\mathrm{Reg}_\mathcal{A}(\mathcal{T}) = \sum_{\tau=1}^{\mathcal{T}} L\left(h_\tau\left(x_\tau\right), y_\tau\right) - \sum_{\tau=1}^{\mathcal{T}} L\left(h^\star\left(x_\tau\right), y_\tau\right) \tag{10}$$

where $h^\star$ denotes the optimal model from a hypothesis class $\mathcal{H}$ that achieves the minimum cumulative loss after $\mathcal{T}$ rounds (i.e., $h^\star = \arg\min_{h \in \mathcal{H}} \sum_{\tau=1}^{\mathcal{T}} L\left(h\left(x_\tau\right), y_\tau\right)$. $h^\star$ can only be obtained in hindsight after receiving all samples. The generalization error of a hypothesis $h \in \mathcal{H}$ is defined by its expected value of the loss function $R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(h(x), y)]$.

Generalization error bound for online learning

Following the idea in,[19] we develop a more general generalization error bound for online learning using online-to-batch conversion based on the existing results of,[6,23] which connects the regret of the online algorithm with its generalization error.

Theorem 1. *Given a set of labeled samples $S = \left(\left(x_1, y_1\right), \ldots, \left(x_\mathcal{T}, y_\mathcal{T}\right)\right)$ drawn i.i.d. from a distribution $\mathcal{D}$. Consider a loss function $L(\cdot, \cdot)$ that is convex with respect to its first argument and is bounded by M for some $M \geq 0$, $h_1, \ldots, h_\mathcal{T}$ being a sequence of hypotheses generated by an online algorithm $\mathcal{A}$ processing samples S sequentially, and $\boldsymbol{\lambda} = [\lambda_1 \ \ldots \ \lambda_\mathcal{T}]^T$ being a weight vector belonging to an unit simplex, i.e., $\Omega_\mathcal{T} =: \left\{\boldsymbol{\lambda} \in \mathbb{R}^\mathcal{T} \mid \sum_{\tau=1}^{\mathcal{T}} \lambda_\tau = 1 \text{ and } \lambda_\tau \geq 0 \text{ for } \tau = 1, \ldots, \mathcal{T}\right\}$. Let h be the ensemble of all the hypotheses through weighted sum, i.e., $h = \sum_{\tau=1}^{\mathcal{T}} \lambda_\tau h_\tau$, and $h^\star$ be the optimal hypothesis from a hypothesis class $\mathcal{H}$. Then, with probability at least $1 - \delta$, the following inequalities hold:*

$$R\left(\sum_{\tau=1}^{\mathcal{T}} \lambda_\tau h_\tau\right) \leq \sum_{\tau=1}^{\mathcal{T}} \lambda_\tau L\left(h_\tau\left(x_\tau\right), y_\tau\right) + M|\boldsymbol{\lambda}|\sqrt{2\log\frac{1}{\delta}} \tag{11}$$

$$R\left(\sum_{\tau=1}^{\mathcal{T}}\lambda_\tau h_\tau\right) \leq \frac{\mathrm{Reg}_{\mathcal{A}}(\mathcal{T})}{\mathcal{T}} + \sum_{\tau=1}^{\mathcal{T}}\lambda_\tau L\left(h^\star\left(x_\tau\right), y_\tau\right) + \sum_{\tau=1}^{\mathcal{T}} M|\lambda_\tau - \frac{1}{\mathcal{T}}| + M|\boldsymbol{\lambda}|\sqrt{2\log\frac{1}{\delta}} \tag{12}$$

Proof. For each round $\tau = 1, \ldots, \mathcal{T}$, we define $V_\tau = \lambda_\tau\left(R(h_\tau) - L\left(h_\tau\left(x_\tau\right), y_\tau\right)\right)$, which is a martingale difference sequence since $\mathbb{E}\left[V_\tau\right] = \lambda_\tau\left(R\left(h_\tau\right) - \mathbb{E}\left[L\left(h_\tau\left(x_\tau\right), y_\tau\right)\right]\right) = \lambda_\tau\left(R\left(h_\tau\right) - R\left(h_\tau\right)\right) = 0$, based on the fact that a stochastic series V is a martingale difference sequence if its expectation with respect to the past is zero. Furthermore, $-\lambda_\tau M \leq V_\tau \leq \lambda_\tau M$ holds since the loss function L is bounded by M. Thus, by applying Azuma's inequality in Lemma 2, for all $\epsilon > 0$, we have $\mathbb{P}\left[\sum_{\tau=1}^{\mathcal{T}} V_\tau \geq \epsilon\right] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{\tau=1}^{\mathcal{T}}(2\lambda_\tau M)^2}\right)$. Setting $\epsilon = M|\boldsymbol{\lambda}|\sqrt{2\log\frac{1}{\delta}}$, the following result holds with probability at least $1-\delta$:

$$\sum_{\tau=1}^{\mathcal{T}}\lambda_\tau R\left(h_\tau\right) \leq \sum_{\tau=1}^{\mathcal{T}}\lambda_\tau L\left(h_\tau\left(x_\tau\right), y_\tau\right) + M|\boldsymbol{\lambda}|\sqrt{2\log\frac{1}{\delta}} \tag{13}$$

Eq. (11) is obtained by substituting the inequality $R\left(\sum_{\tau=1}^{\mathcal{T}}\lambda_\tau h_\tau\right) \leq \sum_{\tau=1}^{\mathcal{T}}\lambda_\tau R\left(h_\tau\right)$ derived from Jensen's inequality into Eq. (13). Next, to prove Eq. (12), we expand $\mathcal{L} := \sum_{\tau=1}^{\mathcal{T}}\lambda_\tau L(h_\tau(x_\tau), y_\tau) - \sum_{\tau=1}^{\mathcal{T}}\lambda_\tau L(h^\star(x_\tau), y_\tau)$ as follows.

$$\begin{aligned}\mathcal{L} &= \sum_{\tau=1}^{\mathcal{T}}(\lambda_\tau - \frac{1}{\mathcal{T}})\left(L\left(h_\tau\left(x_\tau\right), y_\tau\right) - L\left(h^\star\left(x_\tau\right), y_\tau\right)\right) + \frac{1}{\mathcal{T}}\sum_{\tau=1}^{\mathcal{T}}\left(L\left(h_\tau\left(x_\tau\right), y_\tau\right) - L\left(h^\star\left(x_\tau\right), y_\tau\right)\right) \\ &\leq \sum_{\tau=1}^{\mathcal{T}} M|\lambda_\tau - \frac{1}{\mathcal{T}}| + \frac{\mathrm{Reg}_{\mathcal{A}}(\mathcal{T})}{\mathcal{T}}\end{aligned} \tag{14}$$

where the above inequality is derived from the fact that the loss function L is bounded by M and the definition of regret in Eq. (10). Using Eq. (14), we have

$$\sum_{\tau=1}^{\mathcal{T}}\lambda_\tau L\left(h_\tau\left(x_\tau\right), y_\tau\right) \leq \frac{\mathrm{Reg}_{\mathcal{A}}(\mathcal{T})}{\mathcal{T}} + \sum_{\tau=1}^{\mathcal{T}}\lambda_\tau L\left(h^\star\left(x_\tau\right), y_\tau\right) + \sum_{\tau=1}^{\mathcal{T}} M|\lambda_\tau - \frac{1}{\mathcal{T}}| \tag{15}$$

Therefore, by replacing $\sum_{\tau=1}^{\mathcal{T}}\lambda_\tau L\left(h_\tau\left(x_\tau\right), y_\tau\right)$ in Eq. (11) by the three terms in the RHS of Eq. (15), Eq. (12) is obtained, and this completes the proof of Theorem 1. □

Remark 3. Note that in,[6,23] the generalization error bound was derived only for the weights $\lambda_\tau = 1/\mathcal{T}$ for all $\tau = 1, \ldots, \mathcal{T}$. However, in this work, we consider a more general scenario in which weights can be optimized for the ensemble model to achieve the best prediction performance. Therefore, the results derived in[6,23] can be considered a special case of the generalization error bounds derived in Theorem 1. Specifically, when $\lambda_\tau = 1/\mathcal{T}$ for all $\tau = 1, \ldots, \mathcal{T}$, the inequalities of Eqs. (11) and

(12) in Theorem 1 are reduced to the inequalities of Eqs. (8.27) and (8.28) in Theorem 8.15.[23] Additionally, the generalization error bounds proposed in Theorem 1 hold for any arbitrary weight vector belonging to a unit simplex, which allows more freedom to search for a better generalization performance in practice. In the next section, we will demonstrate that the weights for each model generated by an online learning algorithm $\mathcal{A}$ can be optimized using a linear programming problem.

Remark 4. It is noted from Eq. (11) that the generalization error bound for an ensemble hypothesis $h = \sum_{\tau=1}^{\mathcal{T}} \lambda_\tau h_\tau$ consists of two terms: the first term in the RHS of Eq. (11) represents the cumulative training loss in $\mathcal{T}$ rounds. The second term in the RHS of Eq. (11) is an error function associated with the bound M for the loss function, the weight vector $\boldsymbol{\lambda}$, and the confidence δ. To further show that the generalization performance of the ensemble model $h = \sum_{\tau=1}^{\mathcal{T}} \lambda_\tau h_\tau$ converges to the optimal model $h^\star$ from the hypothesis class $\mathcal{H}$ in hindsight, Eq. (12) is derived to connect regret with generalization error. Specifically, the generalization error can be bounded by regret (the first term), the loss suffered by the optimal model $h^\star$ (the second term), and the error functions with respect to $M, \boldsymbol{\lambda}, \delta$, and $\mathcal{T}$ (the third and last terms). The third and last terms in the RHS of Eq. (12) are readily known once a weight vector $\boldsymbol{\lambda}$ and a confidence level δ are chosen, which can be sufficiently small with an appropriate choice of $\boldsymbol{\lambda}$, i.e., for $\lambda_\tau = 1/\mathcal{T}$ for all $\tau = 1, \ldots, \mathcal{T}$, the third term equals zero and the last term becomes $M\sqrt{\frac{2\log\frac{1}{\delta}}{\mathcal{T}}}$ that converges to zero as $\mathcal{T} \to \infty$. Therefore, if an online algorithm $\mathcal{A}$ ensures that its regret is a sublinear function of $\mathcal{T}$, i.e., $\mathrm{Reg}_{\mathcal{A}}(\mathcal{T}) = \mathcal{O}(\sqrt{\mathcal{T}})$, the regret term $\frac{\mathrm{Reg}_{\mathcal{A}}(\mathcal{T})}{\mathcal{T}}$ in Eq. (12) converges to zero as $\mathcal{T} \to \infty$, and the loss suffered by an ensemble hypothesis $h = \sum_{\tau=1}^{\mathcal{T}} \lambda_\tau h_\tau$ can be sufficiently close to the minimum loss achieved by the optimal hypothesis $h^\star$ using the entire dataset S.

Remark 5. While Adam is commonly used in training neural networks, it has been shown in[25] that Adam may fail to converge to the optimal solution in some optimization problems due to fundamental flaws in the convergence analysis of Adam. To address the non-convergence issue of Adam caused by the use of the exponential moving average of the past squared gradients, AMSGrad is proposed in,[25] where the key difference of AMSGrad is that it uses the maximum of the past squared gradients. Some recent works have shown that AMSGrad can achieve convergence with a sublinear rate of regret bound for online optimization problems ([1,10,36]). In this work, AMSGrad will be used for the online learning algorithm.

Ensemble of online predictors

Machine learning modeling with a limited dataset is challenging, since a single machine learning model may not provide perfect predictions over the entire operating region. Therefore, ensemble learning has been utilized to generate ensemble models that improve predictive capability by com-

bining the predictions from multiple machine learning models. Compared to a single model, the prediction accuracy can be improved via ensemble learning due to the following reasons ([26]). First, a single model with a sufficiently small training error may perform poorly in a region with insufficient training data, while ensemble models obtained by aggregating multiple machine learning models can decrease the risk of choosing a single flawed model. Second, a single learner may get stuck in locally optimal solutions since the learning algorithm is known to be a non-convex optimization problem in general, while ensemble learning using multiple learners can reduce the risk of obtaining a local optimum. Third, the optimal hypothesis may be outside the search space of a finite hypothesis class for any single model, while ensemble learning that combines different models can extend the search space, and thus, achieves a better fit to the underlying data distribution. Many effective methods for generating an ensemble model have been proposed in the past decades, and three most popular ensemble methods are Bagging, Boosting, and Stacking. In this work, we follow the idea of ensemble learning to develop an ensemble model that combines the predictions of multiple online learning models. Specifically, the ensemble model denoted by $h = \sum_{\tau=1}^{\mathcal{T}} \lambda_\tau h_\tau$ is built based on the following three-step procedure. *Step 1*: Generate a sequence of hypotheses $h_1, \ldots, h_{\mathcal{T}}$ using an online algorithm $\mathcal{A}$. *Step 2*: Choose a weight vector $\boldsymbol{\lambda}$ by solving the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\lambda} \in \Omega_{\mathcal{T}}} \sum_{\tau=1}^{\mathcal{T}} \lambda_\tau L\left(h_\tau\left(x_\tau\right), y_\tau\right) \\ &s.t. \ \sum_{\tau=1}^{\mathcal{T}} |\lambda_\tau - \frac{1}{\mathcal{T}}| \leq \alpha \end{aligned} \tag{16}$$

where $\alpha \geq 0$ is a hyperparameter that constrains the difference between λ_τ and $1/\mathcal{T}$, and can be predetermined through a validation process. *Step 3*: Develop an ensemble hypothesis $h = \sum_{\tau=1}^{\mathcal{T}} \lambda_\tau h_\tau$. It should be noted that the training loss $L\left(h_\tau\left(x_\tau\right), y_\tau\right)$ for each hypothesis h_τ can be obtained at the end of each round. The optimization problem of Eq. (16) considers an objective function subject to two constraints (i.e., $\boldsymbol{\lambda} \in \Omega_T$ and $\sum_{\tau=1}^{\mathcal{T}} |\lambda_\tau - \frac{1}{\mathcal{T}}| \leq \alpha$), where $\boldsymbol{\lambda}$ is the weight vector to be optimized. Additionally, Eq. (12) demonstrates that the weight λ_τ should be chosen to be sufficiently close to $\frac{1}{\mathcal{T}}$ to ensure that the generalization performance of an ensemble model can converge to that of the optimal model $h^\star$. Therefore, an inequality constraint $\sum_{\tau=1}^{\mathcal{T}} |\lambda_\tau - \frac{1}{\mathcal{T}}| \leq \alpha$ is imposed to avoid the extreme scenario where the weight of a single hypothesis is assigned to be 1.

Development of RNN models using online learning

In this work, an RNN model is initially developed offline using the historical data collected around a certain steady-state, and will be updated using real-time process data based on the online learning algorithm. Specifically, given m sequences of T_{nn}-time-length i.i.d. data points $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$,

$i = 1, \ldots, m$ and $t = 1, \ldots, T_{nn}$, where T_{nn} is the number of time steps to be predicted by the RNN model, an RNN model is initially trained offline to approximate the nonlinear system of Eq. (1) under sample-and-hold implementation of the control actions $u(t) = u\left(t_q\right), \forall t \in [t_q, t_{q+1})$, where $t_{q+1} := t_q + \Delta$. The explicit Euler method is used to integrate the nonlinear system of Eq. (1) with a sufficiently small integration time step $\bar{h}_c < \Delta$. The RNN inputs $\mathbf{x}_{i,t}$ and outputs $\mathbf{y}_{i,t}$ are normalized via standardization, and the statistics information such as distribution mean and standard deviation will be saved for later data processing in online learning. The initial RNN model is constructed using the normalized dataset to predict the future states for the next sampling period with total time steps $T_{nn} = \frac{\Delta}{\bar{h}_c}$.

Subsequently, we apply online-to-batch conversion by using online algorithms in the batch setting to update the RNN models. Specifically, instead of using randomly initialized weights to update RNN model, the weights of the previous RNN model are used as the initial guess for the current RNN model. The updated RNN models are developed using only the most recent process data in a rolling window, which is collected from the real-time process operation. It should be noted that the collected process data is normalized via standardization using the distribution mean and standard deviation obtained from the initial offline-trained RNN model since the real-time data and offline training data are of the same distribution. The above process for updating RNN models is repeated once new process data is collected. The advantage of using previous RNN models as the initial guess for online learning is that some of the nonlinear dynamics captured by the old datasets can be transferred to the updated RNN models. Instead of training a new RNN model from scratch using the entire dataset, we only utilize the most recent process data to update the RNN model, which can significantly reduce the computation time for updating RNN models. However, due to the lack of sufficient online training samples, it is not guaranteed that the online learning RNN models can capture the dynamics of the nonlinear system of Eq. (1) throughout the operating region. Therefore, RNN models need to be iteratively updated until the desired prediction accuracy is achieved such that the MPC using RNN models can stabilize the nonlinear system at the given steady-state.

RNN-based LMPC of Switched Nonlinear Systems

In this section, we develop a Lyapunov-based MPC (LMPC) scheme using online learning RNN models to analyze the closed-loop stability of the nonlinear system of Eq. (1) under scheduled mode transitions between multiple steady-states. Specifically, an RNN model is initially trained using the dataset collected around an initial steady-state, and is incorporated into LMPC to stabilize the nonlinear system of Eq. (1) at the initial steady-state. The RNN model will be constantly improved using the online learning algorithm as dynamic data becomes available during the mode transitions.

We will demonstrate that under the RNN-MPC scheme, closed-loop state of the nonlinear system of Eq. (1) is maintained in the stability region for each mode during the switching interval and enters the stability region of a subsequent mode at the switching time, and ultimately converges to a small terminal set after the terminal mode is activated.

Lyapunov-based control using RNN models

To simplify the discussion of stability properties for RNN-MPC, we use the following continuous-time nonlinear system to represent the single-hidden-layer RNN model of Eq. (4):

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) \tag{17}$$

where $\hat{x} \in \mathbb{R}^n$ is the state vector of the RNN model and $u \in \mathbb{R}^{n_u}$ is the control input vector. Consider the RNN model of Eq. (17) with multiple steady-states x_{s_k}, for all $k \in \ell$. By using the deviation variables $\hat{z}_k := \hat{x} - x_{s_k}$ all $k \in \ell$, the RNN model of Eq. (17) can be rewritten in the deviation form of $\dot{\hat{z}}_k = F'_{nn_k}(\hat{z}_k, u)$. We assume for each steady-state x_{s_k}, $k \in \ell$, there exists a stabilizing feedback controller $u_k = \Phi_{nn_k}(z_k) \in U$ such that the steady-state of the RNN model of Eq. (17) is rendered exponentially stable. This stabilizability assumption implies that for each steady-state x_{s_k}, $k \in \ell$, there exists a $\mathcal{C}^1$ control Lyapunov function $\hat{V}_k(z_k)$ such that the following inequalities hold for all z_k in an open neighborhood $\hat{D}_k$ around the origin:

$$\hat{c}_{1_k}|z_k|^2 \leq \hat{V}_k(z_k) \leq \hat{c}_{2_k}|z_k|^2, \tag{18a}$$

$$\frac{\partial \hat{V}_k(z_k)}{\partial z_k} F'_{nn_k}(z_k, \Phi_{nn_k}(z_k)) \leq -\hat{c}_{3_k}|z_k|^2, \tag{18b}$$

$$\left| \frac{\partial \hat{V}_k(z_k)}{\partial z_k} \right| \leq \hat{c}_{4_k}|z_k|, \tag{18c}$$

where $\hat{c}_{1_k}, \hat{c}_{2_k}, \hat{c}_{3_k}$, and $\hat{c}_{4_k}$, $k \in \ell$, are positive constants. Additionally, due to the smoothness of $F_{nn}(\hat{x}, u)$ and the boundedness of u, for all $z_k, z'_k \in \Omega_{\rho_k}, u \in U$, $k \in \ell$, for all $z_k, z'_k \in \Omega_{\hat{\rho}_k}, u \in U$, $k \in \ell$, there exist positive constants $M_{nn_k}, L_{nn_k}, L'_{nn_k}$ such that the following inequalities hold:

$$|F'_{nn_k}(z_k, u)| \leq M_{nn_k} \tag{19a}$$

$$\left|F'_{nn_k}(z_k, u) - F'_{nn_k}(z'_k, u)\right| \leq L_{nn_k}|z_k - z'_k| \tag{19b}$$

$$\left| \frac{\partial \hat{V}_k(z_k)}{\partial z_k} F'_{nn_k}(z_k, u) - \frac{\partial \hat{V}_k(z'_k)}{\partial z_k} F'_{nn_k}(z'_k, u) \right| \leq L'_{nn_k}|z_k - z'_k| \tag{19c}$$

We follow the construction method in[33] to characterize the closed-loop stability regions $\Omega_{\hat{\rho}_k}$ for the RNN model of Eq. (17), $k \in \ell$, starting from which stabilization at the steady-state x_{s_k} can be achieved under the controller $u_k = \Phi_{nn_k}(z_k) \in U$. Specifically, using Eq. (18), we first search the entire state space to characterize a set of states $\hat{\phi}_{u_k}$ where Eq. (18) is satisfied under the controller $u_k = \Phi_{nn_k}(z_k) \in U$. A level set of Lyapunov function inside $\hat{\phi}_{u_k}$, i.e., $\Omega_{\hat{\rho}_k} := \left\{ z_k \in \hat{\phi}_{u_k} \mid \hat{V}_k(z_k) \leq \hat{\rho}_k \right\}, \hat{\rho}_k > 0$, $k \in \ell$, is characterized as an estimate of the closed-loop stability region for the RNN model around each steady-state. It should be noted that while the RNN models are updated online using real-time data of the process variables, the stability regions $\Omega_{\hat{\rho}_k}$, $k \in \ell$, are characterized using the initial RNN model obtained offline, and remain the same for all times. As a result, when the RNN models are updated online, the controller $u_k = \Phi_{nn_k}(z_k) \in U$ designed based on the initial RNN model may not be able to ensure stability for all states within $\Omega_{\hat{\rho}_k}$ using updated models. This issue will be addressed in section "Integration of online learning with RNN-MPC". Finally, while the closed-loop stability regions are defined with respect to the states in deviation variable form, in the following text, we will use $x \in \Omega_{\hat{\rho}_k}$ to represent that the state x is inside the stability region $\Omega_{\hat{\rho}_k}$ around the steady-state x_{s_k}, with a slight abuse of notation.

LMPC for nonlinear systems operated in a fixed mode

In this section, we consider the nonlinear system of Eq. (1) operated in a fixed mode k for some $k \in \ell$ at all times (i.e., $t \in [t_k^{in}, t_k^{out})$ when $t_k^{out} = \infty$). We present an RNN-MPC design that guarantees closed-loop stability of the nonlinear system of Eq. (1) in probability provided that the RNN models achieve a sufficiently small modeling error. Specifically, the next two propositions from[32] are first presented to demonstrate that the nonlinear system of Eq. (1) can be stabilized under sample-and-hold implementation of the controller $u_k = \Phi_{nn_k}(z_k) \in U$, $k \in l$ in probability if the modeling error is sufficiently small.

Proposition 1. *(c.f. Proposition 2 in[32]) Consider the same initial condition $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}_k}$ for the nonlinear system of Eq. (1) and the RNN model of Eq. (17). If the initial offline-trained RNN model and the online-updated RNN models both achieve a sufficiently small modeling error that satisfies $|F(x,u) - F_{nn}(x,u))| \leq E_M$, where E_M denotes the generalization error bound (i.e., the RHS of Eq. (8) and of Eq. (12) for offline and online learning models, respectively). Then, for each steady-state x_{s_k}, $k \in \ell$, there exist a positive constant κ and a class $\mathcal{K}$ function $f_k(\cdot)$ such that for all $x, \hat{x} \in \Omega_{\hat{\rho}_k}$, with probability at least $1-\delta$, the following inequalities hold:*

$$|x(t) - \hat{x}(t)| \leq f_k(t) := \frac{E_M}{L_k}(e^{L_k t} - 1) \tag{20a}$$

$$\hat{V}_k(x - x_{s_k}) \leq \hat{V}_k(\hat{x} - x_{s_k}) + \frac{\hat{c}_{4_k}\sqrt{\hat{\rho}_k}}{\sqrt{\hat{c}_{1_k}}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{20b}$$

Proof. The proof of Proposition 1 follows the proof of Proposition 2 in[32] and is omitted here. The only difference is that the proof in[32] considers the modeling error E_M for the offline learning models using the generalization error of Eq. (8), while in this work, in addition to the initial offline-trained RNN model, we also account for the modeling error derived for online learning of RNN models. □

Proposition 2. *(c.f. Proposition 3 in[32])Consider the nonlinear system of Eq.* (1) *under the controller* $u_k = \Phi_{nn_k}(\hat{x} - x_{s_k}) \in U$ *that stabilizes the RNN model of Eq.* (17) *and meets the conditions of Eq.* (18)*. If the modeling error is constrained by* $|F(x,u) - F_{nn}(x,u))| \leq E_M \leq \gamma|x - x_{s_k}|$*, where* γ_k *and* γ *are positive real numbers that satisfy* $\gamma_k < \hat{c}_{3_k}/\hat{c}_{4_k}$ *and* $\gamma = min\{\gamma_k, k \in \ell\}$*, and there exist* $\Delta > 0$*,* $\hat{\rho}_k > \rho_{min_k} > \rho_{nn_k} > \rho_{s_k}$ *and* $\epsilon_k > 0$*, for* $k \in \ell$*, such that the following inequalities hold:*

$$-\frac{\tilde{c}_{3_k}}{\hat{c}_{2_k}}\rho_{s_k} + L'_k M_k \Delta \leq -\epsilon_k \tag{21a}$$

$$\rho_{nn_k} := \max\{\hat{V}_k(\hat{x}(t+\Delta) - x_{s_k}) \mid u \in U,\ \hat{x}(t) \in \Omega_{\rho_{s_k}}\} \tag{21b}$$

$$\rho_{min_k} \geq \rho_{nn_k} + \frac{\hat{c}_{4_k}\sqrt{\hat{\rho}_k}}{\sqrt{\hat{c}_{1_k}}} f_k(\Delta) + \kappa(f_k(\Delta))^2 \tag{21c}$$

where $\tilde{c}_{3_k} = -\hat{c}_{3_k} + \hat{c}_{4_k}\gamma < 0$ *and* $f_k(t) = \frac{E_M}{L_k}(e^{L_k t} - 1)$*, then under the implementation of control inputs in a sample-and-hold fashion, i.e.,* $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k}), \forall t \in [t_q, t_{q+1})$*, where* $t_{q+1} := t_q + \Delta$*, for any* $x(t_q) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{s_k}}$*, with a probability at least* $1 - \delta$*, the following inequality holds:*

$$\hat{V}(x(t) - x_{s_k}) \leq \hat{V}(x(t_q) - x_{s_k}),\ \forall t \in [t_q, t_{q+1}) \tag{22}$$

and the state $x(t)$ *of the nonlinear system of Eq.* (1) *remains inside* $\Omega_{\hat{\rho}_k}$ *for all times and ultimately converges to* $\Omega_{\rho_{min_k}}$*.*

Proof. The proof of Proposition 2 follows the proof of Proposition 3 in[32] that is developed for a single steady-state of the nonlinear system of Eq. (1). To account for the existence of multiple steady-states in this work, the coefficient of modeling error condition, γ, in[32] is modified to be the minimum of $\gamma_k := \hat{c}_{3_k}/\hat{c}_{4_k}$, $k \in \ell$ such that the modeling error condition $|F(x,u) - F_{nn}(x,u))| \leq E_M \leq \gamma|x - x_{s_k}|$ is met for all the steady-states of the system of Eq. (1). Although the complete proof of Proposition 2 is omitted, we present the key steps that show the difference between this proof and the one in.[32] To simplify the notation, we use the state in deviation form and show that the state $z_k(t) = x(t) - x_{s_k}$ of the system of $\dot{z}_k = F'_k(z_k, u)$ converges to a small set $\Omega_{\rho_{s_k}}$ around the steady-state x_{s_k} under

the controller $u_k = \Phi_{nn_k}(z_k) \in U$. Specifically, we first derive the time derivative of $\hat{V}_k$ for any $z_k(t_q) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{s_k}}$ under $u_k = \Phi_{nn_k}(z_k) \in U$:

$$\dot{\hat{V}}_k(z_k(t)) = \dot{\hat{V}}_k(z_k(t_q)) + (\dot{\hat{V}}_k(z_k(t)) - \dot{\hat{V}}_k(z_k(t_q))) \tag{23}$$

Following the proof in,[32] Eq. (23) can be expanded as $\dot{\hat{V}}_k(z_k(t)) = \dot{\hat{V}}_k(z_k(t_q)) + L'_k M_k \Delta$ using Eq. (3), where $\dot{\hat{V}}_k(z_k(t_q))$ can be further expanded using Eq. (18b) and Eq. (18c) as follows:

$$\begin{aligned} \dot{\hat{V}}_k(z_k(t_q)) =& \frac{\partial \hat{V}_k(z_k(t_q))}{\partial z_k} F'_k\left(z_k(t_q), \Phi_{nn_k}(z_k(t_q))\right) \\ \leq& -\hat{c}_{3_k}|z_k(t_q)|^2 + \hat{c}_{4_k}|z_k(t_q)| \cdot \left|F'_k(z_k(t_q), \Phi_{nn_k}(z_k(t_q))) - F'_{nn_k}\left(z_k(t_q), \Phi_{nn_k}(z_k(t_q))\right)\right| \end{aligned} \tag{24}$$

Note that the term $\left|F'_k(z_k(t_q), \Phi_{nn_k}(z_k(t_q))) - F'_{nn_k}(z_k(t_q), \Phi_{nn_k}(z_k(t_q)))\right|$ represents the modeling error between the nonlinear system of Eq. (1) and the RNN model of Eq. (17) in their deviation form, and it is equivalent to the modeling error $|F(x,u) - F_{nn}(x,u))|$ represented by the state x in the original state space. Since the generalization error of the offline-trained RNN model depends on the training sample size m, by choosing the number of samples m greater than the minimum sample size $m_N(\delta, |z_k|)$ such that the modeling error constraint E_M can be rendered less than $\gamma|z_k|$. Therefore, based on Eq. (18a), for any $z_k(t_q) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{s_k}}$, with probability at least $1-\delta$, Eq. (24) is further bounded by:

$$\begin{aligned} \dot{\hat{V}}_k(z_k(t_q)) &\leq -\hat{c}_{3_k}|z_k(t_q)|^2 + \hat{c}_{4_k}|z_k(t_q)|\gamma|z_k(t_q)| \\ &\leq -\hat{c}_{3_k}|z_k(t_q)|^2 + \hat{c}_{4_k}|z_k(t_q)|\gamma_k|z_k(t_q)| \end{aligned} \tag{25}$$

where the second line is derived based on the definition of γ, i.e., $\gamma = min\{\gamma_k, k \in \ell\}$. Since $\gamma_k < \hat{c}_{3_k}/\hat{c}_{4_k}$, we can further derive $\dot{\hat{V}}_k(z_k(t_q)) \leq -\tilde{c}_{3_k}|z_k(t_q)|^2 \leq -\tilde{c}_{3_k}\frac{\rho_{s_k}}{\hat{c}_{2_k}}$ by defining $\tilde{c}_{3_k} = \hat{c}_{3_k} - \hat{c}_{4_k}\gamma_k > 0$, and using Eq. (24) for any $z_k(t_q) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{s_k}}$. Therefore, the time derivative of $\hat{V}_k$ of Eq. (23) satisfies the following inequality $\forall k \in \ell$ with a probability of at least $1-\delta$ if Eq. (21a) is satisfied for any $z_k(t_q) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{s_k}}$ and $t \in [t_q, t_{q+1})$.

$$\begin{aligned} \dot{\hat{V}}_k(z_k(t)) &\leq -\tilde{c}_{3_k}\frac{\rho_{s_k}}{\hat{c}_{2_k}} + L'_k M_k \Delta \\ &\leq -\epsilon_k \end{aligned} \tag{26}$$

Eq. (26) implies that the time derivative of $\hat{V}_k$ can be rendered negative with a probability of at least $1-\delta$ for any $z_k(t_q) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{s_k}}$ and $t \in [t_q, t_{q+1})$. As a result, the value of Lyapunov function decreases every sampling period (Eq. (22)), and thus, the state $z_k(t)$ is bounded in $\Omega_{\hat{\rho}_k}$ for all times and moves towards $\Omega_{\rho_{s_k}}$ under $u_k = \Phi_{nn_k}(z_k) \in U$. Additionally, since Eq. (26) may not hold when $z_k(t_q) = \hat{z}_k(t_q) \in \Omega_{\rho_{s_k}}$, we can show that the state $\hat{z}_k(t)$ of the RNN model and the state

$z_k(t)$ of the nonlinear system of Eq. (1) will be bounded in $\Omega_{\rho_{nn_k}}$ and $\Omega_{\rho_{min_k}}$ within one sampling period according to the definitions of $\Omega_{\rho_{nn_k}}$ and $\Omega_{\rho_{min_k}}$ in Eq. (21b) and Eq. (21c), respectively, where $\Omega_{\rho_{min_k}}$ is designed to be a superset of $\Omega_{\rho_{nn_k}}$ that accounts for the modeling error within one sampling period. As a result, with a certain probability, the state of Eq. (1) is maintained in $\Omega_{\hat{\rho}_k}$ for all times and ultimately converges to $\Omega_{\rho_{min_k}}$ under the controller $u_k = \Phi_{nn_k}(z_k) \in U$. □

Remark 6. Proposition 2 demonstrates that under a sample-and-hold implementation of the controller $u_k = \Phi_{nn_k}(x - x_{s_k}) \in U$, the state of the nonlinear system of Eq. (1) can be driven towards $\Omega_{\rho_{s_k}}$, and ultimately bounded in a terminal set $\Omega_{\rho_{min_k}}$ around the steady-state x_{s_k}, provided that the modeling error of the offline learning RNN model satisfies the constraint $|F(x,u) - F_{nn}(x,u))| \leq E_M \leq \gamma|x - x_{s_k}|$. While stability is guaranteed (in a probability sense) using the initial offline learning RNN model, the control performance may not be optimal in terms of convergence to the steady-state since the modeling error E_M may not be sufficiently small due to insufficient training samples. As a result, a large terminal set $\Omega_{\rho_{min_k}}$ may have to be chosen to ensure the boundedness of the state of the nonlinear system of Eq. (1) by accounting for the modeling error E_M of the initial RNN model. Therefore, to improve closed-loop performance under $u_k = \Phi_{nn_k}(x - x_{s_k}) \in U$, online learning will be used to reduce the modeling error E_M such that the closed-loop state can ultimately be bounded in a new terminal set that is closer to the steady-state.

Remark 7. Note that the results in Proposition 2 are derived for offline learning RNN models since the stabilizability conditions of Eq. (18) and the closed-loop stability regions $\Omega_{\hat{\rho}_k}$ are developed only for the initial RNN model. Therefore, once the RNN models are updated online using real-time data, no stability guarantees can be derived for the new RNN models using the controller $u_k = \Phi_{nn_k}(x - x_{s_k}) \in U$. One potential solution to address this issue is to update stability regions and stabilizing controllers as well using the latest RNN model. However, the characterization of new stability regions may be computationally intractable if the RNN model is updated frequently. Therefore, in the next section, we propose a practically feasible solution to ensure the closed-loop stability of the system of Eq. (1) with the closed-loop stability regions remaining the same.

We consider the scenario where the nonlinear system is operated in a fixed mode around a certain steady-state x_{s_k}. The LMPC design using RNN models for the nonlinear system of Eq. (1) operated in a fixed mode k is formulated as the following optimization problem:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_q}^{t_{q+N}} L_{MPC}(\tilde{x}(t), u(t)) dt \tag{27a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{27b}$$

$$u(t) \in U,\ \forall\ t \in [t_q, t_{q+N}) \tag{27c}$$

$$\tilde{x}(t_q) = x(t_q) \tag{27d}$$

$$\dot{\hat{V}}_k(x(t_q) - x_{s_k}, u) \leq \dot{\hat{V}}_k(x(t_q) - x_{s_k}, \Phi_{nn_k}(x(t_q) - x_{s_k})),\ \text{if } x(t_q) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{nn_k}} \tag{27e}$$

$$\hat{V}_k(\tilde{x}(t) - x_{s_k}) \leq \rho_{nn_k},\ \forall\ t \in [t_q, t_{q+N}),\ \text{if } x(t_q) \in \Omega_{\rho_{nn_k}} \tag{27f}$$

where $\tilde{x}$, N, and $S(\Delta)$ denote the predicted state obtained from the RNN model, the fixed finite prediction horizon, and the class of piecewise constant functions with sampling period Δ, respectively. The objective of the RNN-MPC of Eq. (27) is to minimize the cost function of Eq. (27a) and subject to the constraints of Eqs. (27b)-Eq. (27f). Specifically, Eq. (27b) uses the RNN model of Eq. (17) to predict the state evolution. Eq. (27c) defines the constraints for control input over $t \in [t_q, t_{q+N})$. Eq. (27d) defines the initial state $\tilde{x}(t_q)$, which is equal to the state measurements at each sampling step. Eqs. (27e)-(27f) are the two Lyapunov-based constraints used to ensure the closed-loop stability of the nonlinear system of Eq. (1). The optimal solution calculated by the optimization problem of Eq. (27) is denoted by $u_k^\star(t \mid t_q)$, $t \in [t_q, t_{q+N})$. Only the first control action is implemented to the systems in a sample-and-hold fashion, i.e., $u_k(t) = u_k^\star(t \mid t_q), t \in [t_q, t_{q+1})$, and the LMPC will be solved again with the new measurements at the next sampling time. The analysis of the closed-loop stability of the nonlinear system of Eq. (1) under the MPC of Eq. (27) using offline learning RNN models can be found in.[32]

LMPC for nonlinear systems switched between different modes

In this section, we consider the nonlinear system of Eq. (1) with switching modes according to a prescribed switching schedule defined by switching times, i.e., the system is operated in the current mode k for $t \in [t_k^{in}, t_k^{out})$ and is switched to a subsequent mode f for some $k, f \in \ell$ at $t = t_k^{out} = t_f^{in}$. The following proposition is developed to ensure that the closed-loop state under sample-and-hold implementation of the controller $u_k = \Phi_{nn_k}(x - x_{s_k}) \in U$ enters the stability region of the subsequent mode f at $t = t_k^{out} = t_f^{in}$.

Proposition 3. *Consider the nonlinear system of Eq.* (1) *and the RNN model of Eq.* (17) *under the controller* $u_k = \Phi_{nn_k}(x - x_{s_k}) \in U$ *that meets the conditions in Propositions 1 and 2. Given* $t_k^{in} \leq t < t_k^{out} = t_f^{in}$ *and* $(x(t_k^{in}) - x_{s_k}) \in \Omega_{\hat{\rho}_k}$, *if there exist* $\hat{\rho}_k > 0$, $\epsilon_k > 0$, $N_k > 0$, *and* $\Delta > 0\ \forall k \in \ell$ *such that*

$$\hat{c}_{2_f}\left(\sqrt{\frac{\hat{\rho}_k - \epsilon_k N_k \Delta}{\hat{c}_{1_k}}} + \left|x_{s_k} - x_{s_f}\right|\right) \leq \hat{\rho}_f, \tag{28}$$

then $(x(t_f^{in}) - x_{s_f}) \in \Omega_{\hat{\rho}_f}$.

Proof. It has been proven in Eq. (26) that $\dot{\hat{V}}_k(x(t)-x_{s_k}) \leq -\epsilon_k$ holds for all $(x(t)-x_{s_k}) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{s_k}}$ and $t \in [t_q, t_{q+1})$. Using the above inequality recursively for all sampling periods within the switching interval $t \in [t_k^{in}, t_k^{out})$, we have the following inequality: $\hat{V}_k(x(t_k^{out}) - x_{s_k}) \leq \hat{V}_k(x(t_k^{in}) - x_{s_k}) - \epsilon_k N_k \Delta$. Since $(x(t_k^{in}) - x_{s_k}) \in \Omega_{\hat{\rho}_k}$, it follows that $\hat{V}_k(x(t_k^{out}) - x_{s_k}) \leq \hat{\rho}_k - \epsilon_k N_k \Delta$. From Eq. (18a) we can obtain $|x(t_k^{out}) - x_{s_k}| \leq \sqrt{\frac{\hat{\rho}_k - \epsilon_k N_k \Delta}{\hat{c}_{1_k}}}$. Using the triangular inequality $\left|x(t_k^{out}) - x_{s_f}\right| \leq |x(t_k^{out}) - x_{s_k}| + \left|x_{s_k} - x_{s_f}\right|$ and Eq. (18a) for the Lyapunov function at the subsequent mode f and $t_k^{\text{out}} = t_f^{\text{in}}$, it follows that $\hat{V}_f(x(t_f^{in}) - x_{s_f}) \leq \hat{\rho}_f$ if Eq. (28) holds, which implies that $(x(t_f^{in}) - x_{s_f}) \in \Omega_{\hat{\rho}_f}$. □

The LMPC using RNN models for the nonlinear system of Eq. (1) with switching modes is formulated as follows:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_q}^{t_k^{out}} L_{MPC}(\tilde{x}(t), u(t)) dt \tag{29a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{29b}$$

$$u(t) \in U, \ \forall \ t \in [t_q, t_k^{out}) \tag{29c}$$

$$\tilde{x}(t_q) = x(t_q) \tag{29d}$$

$$\dot{\hat{V}}_k(x(t_q) - x_{s_k}, u) \leq \dot{\hat{V}}_k(x(t_q) - x_{s_k}, \Phi_{nn_k}(x(t_q) - x_{s_k})), \text{if } x(t_q) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{nn_k}} \tag{29e}$$

$$\hat{V}_k(\tilde{x}(t) - x_{s_k}) \leq \rho_{nn_k}, \ \forall \ t \in [t_q, t_k^{out}), \ \text{if } x(t_q) \in \Omega_{\rho_{nn_k}} \tag{29f}$$

$$\hat{V}_f(\tilde{x}(t_k^{out}) - x_{s_f}) \leq \hat{\rho}_f \tag{29g}$$

The notations follow those in Eq. (27). In contrast to the RNN-MPC of Eq. (27) for the nonlinear system of Eq. (1) operated in a fixed mode, the RNN-MPC of Eq. (29) for the nonlinear systems with switching modes has two major differences. First, the RNN-MPC problem of Eq. (29) is implemented with a shrinking prediction horizon, which is calculated by the difference between the switching out time t_k^{out} and the current time t_q, while the RNN-MPC problem of Eq. (27) uses a fixed predicted horizon N. Second, the constraint of Eq. (29g) is imposed by the RNN-MPC problem of Eq. (29) to ensure that the closed-loop system state can enter the stability region $\Omega_{\hat{\rho}_f}$ of the subsequent mode f at the time t_k^{out} when the system is switched out from the current mode k. Closed-loop stability analysis for the nonlinear system of Eq. (1) under the RNN-MPC of Eq. (29) will be provided in section "Closed-loop stability under RNN-MPC".

Remark 8. Proposition 3 demonstrates that the closed-loop state is guaranteed to enter the stability region $\Omega_{\hat{\rho}_f}$ of the subsequent mode f at the switching time $t = t_k^{out} = t_f^{in}$ if the constraint of

Eq. (28) holds. Therefore, the feasibility of the MPC optimization problem of Eq. (29) depends on the feasibility of the transition constraint of Eq. (28). This implies that the mode transition should not be carried out if the constraint of Eq. (28) is not feasible at $t = t_f^{\text{out}}$ in order to maintain closed-loop stability. An underlying assumption for Proposition 3 is that the stability regions of two consecutive switching modes (i.e., $\Omega_{\hat{\rho}_k}$ and $\Omega_{\hat{\rho}_f}$) should have a non-empty intersection. Additionally, the time interval between t_k^{in} and t_k^{out} is assumed to be $N_k\Delta$, which implies that the mode transition is executed at the sampling time instant.

Integration of online learning with RNN-MPC

To improve the generalization performance of RNN models, the RNN models will be updated online using real-time state measurements. The updated RNN models will be incorporated in RNN-MPC to replace the previous RNN models (i.e., F_{nn} in Eq. (29b)) to provide a better prediction of future states. However, since the RNN models obtained through the online learning algorithm may take more than $\mathcal{T}_N$ rounds to converge to the best model using the entire dataset (see Eq. (12)), where $\mathcal{T}_N$ is the minimum number of rounds that the online learning algorithm takes to meet the modeling error constraint $E_M \leq \gamma|x - x_{s_k}|$, an implementation strategy needs to be developed to determine whether to apply the updated RNN model in RNN-MPC at each sampling step. As discussed in previous sections, the closed-loop system may lose stability properties using online learning RNN models since the stability regions are characterized under the initial RNN model (i.e., Eq. (18) may not hold for the new RNN models). Specifically, when incorporating online learning RNN models in the RNN-MPC of Eq. (29), only the prediction model of Eq. (29b) is updated, and the RNN model used in the contractive constraint of Eq. (29e) remains unchanged in order to guarantee closed-loop stability. In this case, the constraints of Eq. (29f) and Eq. (29g) that depend on the future states $\tilde{x}$ will use the updated RNN model of Eq. (29b), while the contractive constraint of Eq. (29e) will use the initial RNN model for all times. As a result, feasibility is no longer guaranteed for the RNN-MPC of Eq. (29) using the updated RNN models under the controller $u_k = \Phi_{nn_k}(x - x_{s_k}) \in U$.

To simplify the discussion, we assume that the nonlinear system of Eq. (1) is switched between different modes for some $k, f \in \ell$ under the RNN-MPC of Eq. (29) during the operation period $t \in [0, t')$, while for $t \geq t'$, the nonlinear system of Eq. (1) operates in a terminal mode $z \in \ell$ under the RNN-MPC of Eq. (27). Based on the prescribed switching schedule policy, the implementation strategy (Algorithm 1) for online learning with RNN-MPC is presented as follows. *Step 1*: An RNN model is initially trained offline using the open-loop simulation data for the nonlinear system of Eq. (1). Following the construction method in,[33] the stabilizing controller Φ_{nn_k} and the stability regions $\Omega_{\hat{\rho}_k}$ for each mode $k \in \ell$ are characterized using the initial RNN model. *Step 2*: Given

any state measurement $x(t_q) \in \Omega_{\hat{\rho}_k}$ at $t = t_q$, $\forall k \in \ell$, the nonlinear system of Eq. (1) is operated under the RNN-MPC of Eq. (29) (if $t < t'$) or Eq. (27) (if $t \geq t'$) with real-time process data collected sequentially. The optimal control action $u_k^\star(t)$ is calculated by the RNN-MPC of Eq. (27) or Eq. (29) using the latest RNN model at each sampling time. If the RNN-MPC of Eq. (27) or Eq. (29) does not have a feasible solution, the stabilizing controller $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k}) \in U$ is applied to the nonlinear system of Eq. (1) at the current mode k. *Step 3*: The RNN models are updated using the previous RNN models and the most recent process data. The generalization error of updated RNN models is evaluated using the testing set in each round. This new RNN model is used in RNN-MPC to replace the last updated RNN model if its testing error is less than the threshold $\gamma|x - x_{s_k}|$; otherwise, this new RNN model is discarded and the prediction model in RNN-MPC remains unchanged. Online learning will continue by repeating Step 2 until a total of $\mathcal{T}_N$ RNN models satisfying the generalization error test are collected, which will lead to Step 4. *Step 4*: The optimization problem of Eq. (16) for ensemble hypothesis is performed to find the optimal weight λ_τ for each model h_τ, and the final RNN model is given by $h = \sum_{\tau=1}^{\mathcal{T}} \lambda_\tau h_\tau$.

Algorithm 1 Implementation Strategy for Online Learning within RNN-MPC

Step 1: Use the initial RNN model in MPC and the controller Φ_{nn_k} for mode $k \in \ell$.
Step 2: Given any current state measurement $x(t_q) \in \Omega_{\hat{\rho}_k}$, solve RNN-MPC with the latest RNN model: **if** feasible: apply the optimal control action $u_k^\star(t)$ **else**: apply $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k})$.
Step 3: Update the RNN model and go back to step 2. Stop updating RNN models when a total of $\mathcal{T}_N$ RNN models are collected, and then go to step 4.
Step 4: Obtain the final RNN model by solving the optimization problem of Eq. (16).

Closed-loop stability under RNN-MPC

We next demonstrate that the RNN-MPCs of Eq. (27) and Eq. (29) using the initial offline-trained RNN model can guarantee closed-loop stability and recursive feasibility simultaneously provided that the modeling error is sufficiently small. However, when the RNN-MPCs of Eq. (27) and Eq. (29) use online learning RNN models, we will demonstrate that closed-loop stability and recursive feasibility can be guaranteed under the implementation strategy in Algorithm 1 (i.e., applying the optimal solution of RNN-MPC whenever it is feasible and applying the stabilizing controller $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k})$ when RNN-MPC is infeasible).

Theorem 2. *Consider the closed-loop nonlinear system of Eq.* (1) *switched between different modes for some $k, f \in \ell$ under the RNN-MPC of Eq.* (29) *implemented following Algorithm 1, and ultimately operated in a specific terminal mode for some $z \in \ell$ under the RNN-MPC of Eq.* (27)*. Given any initial state $x(t_k^{in}) \in \Omega_{\hat{\rho}_k}$ at $t = t_k^{in}$, if the modeling error is constrained by $|F(x,u) - F_{nn}(x,u))| \leq$*

$E_M \leq \gamma|x - x_{s_k}|$, then for each sampling time step, closed-loop stability for the nonlinear system of Eq. (1) under the RNN-MPCs of Eq. (27) and Eq. (29) is achieved with a probability at least $1-\delta$ in the sense that closed-loop state $x(t)$ is bounded in $\Omega_{\hat{\rho}_k}$ for each switching interval $t \in [t_k^{in}, t_k^{out})$ and enters the stability region $\Omega_{\hat{\rho}_f}$ of the subsequent mode f at $t = t_k^{out} = t_f^{in}$ and ultimately converges to the terminal set $\Omega_{\rho_{min_z}}$ defined by the terminal mode z.

Proof. The proof is divided into three parts. We first discuss recursive feasibility and closed-loop stability for the nonlinear system of Eq. (1) under the RNN-MPC of Eq. (29) in Part 1 and Part 2, respectively. Then, we discuss recursive feasibility and closed-loop stability for the nonlinear system of Eq. (1) under the RNN-MPC of Eq. (27) in Part 3.

Part 1: We first discuss recursive feasibility of Eq. (29) when the initial offline-trained RNN model is used for Eq. (29b). For $x(t_q) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{nn_k}}$ at $t = t_q$, the control action $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k}) \in U$, $t \in [t_q, t_{q+1})$ is a feasible solution to Eq. (29) since it satisfies the input constraint of Eq. (29c) and the constraint of Eq. (29e) by using the equal sign. When $x(t_q) \in \Omega_{\rho_{nn_k}}$, as shown in Proposition 2, the RNN predicted state $\tilde{x}(t)$ is maintained within $\Omega_{\rho_{nn_k}}$ within one sampling period. As a result, the input constraint of Eq. (29c) and the constraint of Eq. (29f) are satisfied using $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k}) \in U$, $t \in [t_q, t_k^{out})$. Additionally, it has been shown in Proposition 3 that the closed-loop state can enter the stability region $\Omega_{\hat{\rho}_f}$ under the controller $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k}) \in U$, $t \in [t_k^{in}, t_k^{out})$, which implies that the constraint of Eq. (29g) is satisfied. Therefore, the RNN-MPC of Eq. (29) using the initial RNN model is recursively feasible for any state in $\Omega_{\hat{\rho}_k}$ if the conditions in Propositions 1-3 hold. When Eq. (29b) uses online learning RNN models, it is noted that the LHS of the contractive constraint of Eq. (29e) uses the updated RNN models while the RHS of Eq. (29e) still uses the initial RNN model trained offline. Due to the model inconsistency, there may not exist a feasible solution that satisfies Eq. (29e). Additionally, the constraints of Eq. (29f) and Eq. (29g) depend on the future states $\tilde{x}$ that are predicted by online learning RNN models of Eq. (29b). However, as the results in Propositions 2-3 hold only for the controller $u_k = \Phi_{nn_k}(x(t_q) - x_{s_k}) \in U$ designed using the initial RNN model (see Remark 7), the control action $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k}) \in U$ may not be a feasible solution to Eq. (29f) and Eq. (29g) either. Therefore, recursive feasibility of the RNN-MPC of Eq. (29) using online learning RNN models is no longer guaranteed and a stabilizing controller $u = \Phi_{nn_k}(x(t_q) - x_{s_k})$ will be applied when the RNN-MPC is infeasible as discussed in the previous section.

Part 2: Next, we discuss closed-loop stability of the nonlinear system of Eq. (1) under the RNN-MPC of Eq. (29). We first consider the case when Eq. (29) has a feasible solution. For each switching interval $t \in [t_k^{in}, t_k^{out})$, if $x(t_q) \in \Omega_{\hat{\rho}_k} \backslash \Omega_{\rho_{nn_k}}$ at $t = t_q$, the constraint of Eq. (29e) is

activated such that the value of the Lyapunov function keeps decreasing for each sampling period with probability at least $1-\delta$, which implies that the state can converge to $\Omega_{\rho_{nn_k}}$ in a finite number of sampling times. After the state enters $\Omega_{\rho_{nn_k}}$, the constraint of Eq. (29f) is activated such that the predicted state is bounded in $\Omega_{\rho_{nn_k}}$, and the actual state of the nonlinear system of Eq. (1) can be maintained in $\Omega_{\rho_{min_k}}$ with a probability at least $1-\delta$ as shown in Proposition 2. Therefore, the constraints of Eq. (29e) and (29e) guarantee that the closed-loop state moves towards $\Omega_{\rho_{nn_k}}$ without leaving the stability region $\Omega_{\hat{\rho}_k}$ during the switching interval $t \in [t_k^{in}, t_k^{out})$ for the mode k. At the time of the mode transition, $t = t_k^{out} = t_f^{in}$, the constraint of Eq. (29g) ensures that the closed-loop state enters the stability region $\Omega_{\hat{\rho}_f}$ of the subsequent mode f. In the event that Eq. (29) does not yield a feasible solution, the controller $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k})$ obtained by the initial model will be applied to stabilize the nonlinear system of Eq. (1). It has been proven in Propositions 2-3 that under $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k})$, the closed-loop state $x(t)$ remains inside in $\Omega_{\hat{\rho}_k}$ for $t \in [t_k^{in}, t_k^{out})$ and enters the stability region $\Omega_{\hat{\rho}_f}$ of the subsequent mode f at $t = t_k^{out} = t_f^{in}$.

Part 3: Finally, we discuss recursive feasibility and closed-loop stability of the nonlinear system of Eq. (1) under the RNN-MPC of Eq. (27). When the nonlinear system of Eq. (1) is operated in the terminal mode z of after which the switching schedule is terminated. The constraint of of Eq. (29g) is removed and a fixed prediction horizon N is utilized due to $t_k^{out} = \infty$, and thus, the RNN-MPC of Eq. (29) reduces to Eq. (27). Therefore, recursive feasibility and closed-loop stability of the nonlinear system of Eq. (1) under the RNN-MPC of Eq. (27) follow the proof for the RNN-MPC of Eq. (29). Specifically, when Eq. (27) has a feasible solution, the constraints of Eq. (27e) and Eq. (27f) guarantee that the closed-loop state is maintained in $\Omega_{\hat{\rho}_z}$ and ultimately converges to $\Omega_{\rho_{min_z}}$ in probability. However, when Eq. (27) is infeasible, the controller $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k})$ implemented in a sample-and-hold fashion will be used to drive the state towards the steady-state. In Proposition 2, it has been shown that with a probability at least $1-\delta$, the closed-loop state under $u_k(t) = \Phi_{nn_k}(x(t_q) - x_{s_k})$ can be bounded in $\Omega_{\hat{\rho}_z}$ and ultimately maintain it inside $\Omega_{\rho_{min_z}}$. □

Remark 9. In this work, we study the closed-loop performance of the nonlinear system with multiple steady-states under RNN-MPC, where the system is required to switch between different steady-states following a prescribed switched schedule. It is noted that while in general, switched systems are defined as a class of continuous-time systems with discrete switching events that may evolve the varying system dynamics, the system dynamics of Eq. (1) in this work does not change the between mode transitions. This assumption on system dynamics allows us to develop a generalization error bound for online learning RNN models with real-time data drawn i.i.d from the same distribution. However, it should be pointed out that the stability results derived in this section can be generalized

to the switched systems with varying dynamics and similar closed-loop stability results can be developed following[16,22] provided that a generalization error bound can be derived for the online learning models using non-i.i.d samples. The non-i.i.d setting for developing RNN models is beyond the scope of this study and will be explored in future work.

Remark 10. When the nonlinear system of Eq. (1) is switched between different modes, we consider the case of a finite switching schedule only and do not require the boundedness of the closed-loop state in the terminal set $\Omega_{\rho_{min_k}}$ for each mode $k \in \ell$ except for the terminal mode z. Specifically, while the contractive constraint of Eq. (29e) in our RNN-MPC design of Eq. (29) can drive the closed-loop state towards $\Omega_{\rho_{min_k}}$, it is not possible to know how many sampling steps it needs to drive the state into $\Omega_{\rho_{min_k}}$ beforehand, and whether it can be done within the switching interval $t \in [t_k^{in}, t_k^{out})$. Instead we only require that the closed-loop states remain inside in the stability region $\Omega_{\hat{\rho}_k}$ of the current mode k and enter the stability region $\Omega_{\hat{\rho}_f}$ of the subsequent mode f at the switching time $t = t_k^{out}$. When the system enters the terminal mode z under the RNN-MPC of Eq. (27), it has been shown in Theorem 2 that the closed-loop state can ultimately converge to a small terminal set $\Omega_{\rho_{min_z}}$. Note that this is different from the MPC scheme in[22] that considers the case of an infinite switching sequence, for which multiple Lyapunov function (MLF) constraints were used to ensure that the value of the Lyapunov function is less than what it was when the system was operated in mode k for the last time. Based on the stability conditions of the MLF, the value of the Lyapunov function decreases all the time such that the closed-loop state can converge to a small terminal set $\Omega_{\rho_{min_k}}$ for each mode $k \in \ell$ of the switched systems if the number of switches is infinite.

Remark 11. Due to the lack of sufficient training samples and the limited range of data available for offline training of the initial RNN model, the stability regions $\Omega_{\hat{\rho}_k}$, $k \in \ell$ characterized using the initial RNN model may be conservative for some steady-states. Additionally, since the terminal set $\Omega_{\rho_{min_k}}$ is developed to ensure the boundedness of states by accounting for the modeling error between the RNN model and the nonlinear system of Eq. (1), the terminal set may not be sufficiently close to the steady-state if the generalization error for the initial RNN model is not small. Therefore, through online learning of RNN models, the generalization performance could be further improved using the process data around the steady-state. Although the stability region $\Omega_{\rho_{min_k}}$ remains unchanged for RNN-MPC in this work, closed-loop performance can be improved through online learning by maintaining the state in a smaller terminal set $\Omega_{\rho_{min_k}}$ with reduced modeling error, which will be demonstrated using a chemical process example in the next section.

Application to a Chemical Process Example

In this section, we present a chemical process example to illustrate the application of LMPC using online learning of RNN models. Specifically, we consider a well-mixed and non-isothermal continuous stirred tank reactor (CSTR) in which an irreversible second-order exothermic reaction $A \rightarrow B$ takes place. A and B denote a reactant and a product, respectively. The inlet stream consists of a pure reactant A at a concentration of C_{A0}, a temperature T_0, and a flow rate F. A heating jacket is used to supply/remove heat from the reactor at a rate Q. The following material and energy balance equations are used to develop a dynamic model for the CSTR:

$$\frac{dC_A}{dt} = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT}} C_A^2, \quad \frac{dT}{dt} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \tag{30}$$

where C_A, T, and V denote the concentration of the reactant A, the temperature, and volume of the reacting liquid in the reactor, respectively. ρ_L and C_p represent the constant density and the heat capacity of the reacting liquid, respectively. The enthalpy of the reaction, the pre-exponential constant, the activation energy, and the ideal gas constant are denoted by ΔH, k_0, E, and R, respectively. Q is the heat input rate. In this example, we consider two steady-states $(C_{As_1},\ T_{s_1}) = (1.22\ kmol/m^3,\ 438\ K)$ for mode 1 and $(C_{As_2},\ T_{s_2}) = (1.95\ kmol/m^3,\ 402\ K)$ for mode 2 under the steady-state input values $(C_{A0_s}\ Q_s) = (4\ kmol/m^3,\ 0\ kJ/hr)$. The values of the process parameters can be found in[34] and are omitted here. The CSTR is operated in each mode by manipulating the inlet concentration of species A and the rate of heat input Q. The manipulated inputs are subject to the constraints given by $0.5\ kmol/m^3 \leq C_{A0} \leq 7.5\ kmol/m^3$ and $|Q| \leq 5 \times 10^5\ kJ/hr$. Therefore, the dynamic model of the CSTR of Eq. (30) can be represented in the form of the nonlinear system of Eq. (1) with $x^T = [C_A\ T]$ and $u^T = [C_{A0}\ \ Q]$, such that the equilibrium points of the system are given by (x_{s_k}, u_s), where $k \in \ell = \{1, 2\}$, i.e., $x_{s_1}^T = [1.22\ kmol/m^3\ 438\ K]$, $x_{s_2}^T = [1.95\ kmol/m^3\ 402\ K]$, and $u_s^T = [4\ kmol/m^3\ \ 0\ kJ/hr]$. The control objective is to initially operate the CSTR in the stability region of mode 1 for all $t \in [0, t_1^{out})$ and to drive the closed-loop state into the stability region of mode 2 by the time of switch $t = t_1^{out} = t_2^{in}$ and ultimately stabilize the CSTR at the unstable equilibrium point $x_{s_2}^T = [1.95\ kmol/m^3\ 402\ K]$ under RNN-MPC. The dynamic model of the CSTR of Eq. (30) is numerically integrated using the explicit Euler method with a sufficiently small integration time step of $\bar{h}_c = 10^{-4}\ hr$. The nonlinear optimization problems of the RNN-MPCs of Eq. (27) and Eq. (29) are solved using PyIpopt (the Python module of the IPOPT software package[29]) with the sampling period $\Delta = 10^{-2}\ hr$.

Data generation

For both modes, we consider quadratic Lyapunov functions $V_k(x - x_{s_k}) = (x - x_{s_k})^T P_k (x - x_{s_k})$ designed with the positive definite matrix: $P_1 = P_2 = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix}$. Then, following the construction method in,[33] the closed-loop stability regions Ω_{ρ_k} for the CSTR operated in mode k, $k \in \{1, 2\}$, are characterized as level sets of the Lyapunov function V_k with $\rho_1 = 485$ and $\rho_2 = 372$, respectively. We assume that the CSTR is initially operated in a small region (denoted by Ω_0 and marked as a black rectangle in Figure 1) around the steady-state x_{s_1} for safe and stable operation, where $\Omega_0 := \left\{ x \in \mathbb{R}^2 \mid [1.05 \;\; kmol/m^3 \;\; 420 \;\; K] \leq x^T \leq [1.35 \;\; kmol/m^3 \;\; 450 \;\; K] \right\}$. Open-loop simulations are carried out under various initial conditions $x_0 \in \Omega_0$ and manipulated inputs $u \in U$ to generate the dataset for training the initial RNN model, where the control actions are implemented in a sample-and-hold fashion, i.e., $u(t) = u(t_q), \forall t \in [t_q, t_q + \Delta)$. Each simulation runs for one sampling period Δ that includes 100 integration time steps $\bar{h}_c$. Based on the dataset, an initial RNN model is developed to predict future states for one sampling period with a total of 100 integration time steps using Keras. Specifically, a hidden layer consisting of 16 recurrent units is designed for the initial RNN model and the *tanh* function is used as the activation function. After the initial RNN model is developed, the Lyapunov function $\hat{V}_k(x - x_{s_k})$ for the RNN model is set to be the same as $V_k(x - x_{s_k})$, $k \in \{1, 2\}$. Based on the method in,[33] the closed-loop stability regions $\Omega_{\hat{\rho}_k}$ for the initial RNN model operated in mode k, $k \in \{1, 2\}$, are estimated as a level set of the Lyapunov function $\hat{V}_k$ with $\hat{\rho}_1 = 480$ and $\hat{\rho}_2 = 368$, respectively. Additionally, $\rho_{nn_1} = \rho_{nn_2} = 1.6$ and $\rho_{min_1} = \rho_{min_2} = 2$ are determined by conducting extensive simulations for $u \in U$.

Since the initial RNN model is trained using the limited dataset collected from a small region Ω_0 around the steady-state x_{s_1} of mode 1, we will apply the online learning algorithm to update the RNN models after the CSTR is switched to mode 2 at $t = t_2^{in}$. In this example, we update the RNN models using the previous RNN models and the most recent process data collected at every five sampling periods. It should be noted that the updated RNN model is incorporated into LMPC to replace the previous RNN models only if the new RNN model satisfies the generalization error condition; otherwise, the new RNN model is discarded, and the predictive model in LMPC remains unchanged.

Closed-loop simulation results

We next carry out closed-loop simulations under the following switching schedule: the CSTR is operated in mode 1 from time $t = 0$ to $t = 0.2$ hr, and the CSTR is switched from mode 1 to mode 2 at $t = 0.2$ hr and to be stabilized at the steady-state x_{s_2} for the remaining time. We consider the CSTR starting from an initial state that belongs to the stability region $\Omega_{\hat{\rho}_1}$ of

mode 1, i.e., $x_0^T = [0\ kmol/m^3\ 500\ K]$, for which the closed-loop state trajectories, the state and the manipulated input profiles for the CSTR of Eq. (30) are displayed in Figures 1- 2. Since the stability region $\Omega_{\hat{\rho}_1}$ of mode 1 may include physically infeasible initial conditions (i.e., negative concentration values), the initial conditions should be chosen accounting for physical feasibility.

We first discuss the case when the RNN-MPCs of Eq. (27) and Eq. (29) use the initial offline-trained RNN model for all times. Figure 1(a) shows that the closed-loop state remains inside the stability region $\Omega_{\hat{\rho}_1}$ of mode 1 and ultimately converges to a small neighborhood (denoted by $\Omega_{\rho_{min_1}}$) around the steady-state x_{s_1} for $t \in [0,\ 0.2\ hr)$ under the RNN-MPC of Eq. (29) using the initial RNN model. Additionally, while the closed-loop state enters the stability region $\Omega_{\hat{\rho}_2}$ of mode 2 at $t = 0.2\ hr$ and remains inside $\Omega_{\hat{\rho}_2}$ after $t = 0.2\ hr$ under the RNN-MPC of Eq. (27) using the initial RNN model, it cannot converge to a small neighborhood (denoted by $\Omega_{\rho_{min_2}}$) around the steady-state x_{s_2}. It can be seen more clearly from Figure 2(a) that the concentration C_A (top figure) shows an oscillation between the interval $[1.62\ kmol/m^3\ 1.78\ kmol/m^3]$ and the temperature T (bottom figure) converges to nearly 410 K when the CSTR is operated in mode 2 after $t = 0.2\ hr$, while the CSTR is required to be stabilized within $\Omega_{\rho_{min_2}}$ around the steady-state $x_{s_2}^T = [1.95\ kmol/m^3\ 402\ K]$. Figure 2(b) shows the manipulated input profiles, where the upper and lower bounds for the constraints of the manipulated input are represented by the dashed horizontal lines. It is shown that the input constraints are not violated for all times under RNN-MPC using the initial model. However, it is observed in Figure 2(b) that C_{A0} (top figure) shows an oscillation at all times for the CSTR operated in mode 2. It is noted that the initial RNN model is trained using the dataset Ω_0 collected from a small region around x_{s_1}, and thus, the initial RNN model is able to capture the nonlinear dynamics of the CSTR around x_{s_1}. Therefore, it is not surprising to see that under RNN-MPC using the initial model, the CSTR is successfully stabilized at $\Omega_{\rho_{min_1}}$ for mode 1. To further demonstrate why closed-loop stability does not hold for mode 2, Figure 3 shows the prediction error that calculates the squared error between the outputs predicted by the RNN models and the true outputs measured by the CSTR at every sampling time when the CSTR is operated in mode 2. It is shown in Figure 3 that the prediction error of the initial RNN model is considerably large as the closed-loop state moves towards the steady-state x_{s_2} of mode 2, which implies that the true outputs substantially deviate from the predicted outputs. The poor prediction of the initial RNN model results in the undesired closed-loop performance of RNN-MPC using the initial RNN model for mode 2.

We next investigate the case when online learning of RNN models is carried out for the CSTR operated in mode 2. Specifically, it is shown in Figure 1(b) that the closed-loop state is successfully

driven into a small neighborhood $\Omega_{\rho_{min_2}}$ around the steady-state x_{s_2} under the RNN-MPC of Eq. (27) using online learning RNN models. Additionally, Figure 2 shows that the closed-loop states and manipulated inputs are stabilized at their steady-states (x_{s_2}, u_s) under the RNN-MPC of Eq. (27) using the updated RNN models, while those under the RNN-MPC of Eq. (27) using the initial RNN model fail to converge to (x_{s_2}, u_s), showing a considerable oscillation in the state and input profiles (i.e., C_A, C_{A0}). Therefore, the closed-loop performance of the CSTR operated in mode 2 under the RNN-MPC of Eq. (27) is significantly improved via online learning of RNN models. Similarly, Figure 3 shows the prediction error of the updated RNN models at every sampling time. Specifically, the online update of RNN models is activated three times at $t = 0.25\ hr$, $t = 0.3\ hr$, and $t = 0.35\ hr$, respectively, according to the model update strategy that activates an online update of RNN models every five sampling periods. It is observed in Figure 3 that the prediction error is identical for both the initial and the updated RNN models from $t = 0.2\ hr$ to $t = 0.25\ hr$. It is demonstrated that the prediction errors of the updated RNN models are always lower than those of the initial RNN model for the subsequent sampling periods. The gap in the prediction error for the initial RNN model and the updated RNN models is more apparent after the second update of the RNN models is activated. Additionally, it is demonstrated in the zoomed graph (top left figure) of Figure 3 that the prediction error of the updated RNN models gradually decreases and converges to a sufficiently small number. This declining trend implies that the prediction performance of updated RNN models is improved as real-time process data is used for online learning.

During the closed-loop simulations, a testing dataset that has not been used in the initial training stage is used to evaluate the generalization performance of all the RNN models. The obtained testing errors (MSE) for the initial RNN model, and the three updated RNN models are 4.6676, 0.6186, 0.4814, and 0.1575, respectively, which demonstrates that the generalization performance of the three online learning RNN models gradually improves and is much better than that of the initial RNN model. Furthermore, the optimization problem of Eq. (16) with $\alpha = 0.6$ is performed using the training errors of the three online learning RNN models to find the optimal weights λ_τ for each model h_τ, which yields $\lambda_1 = 0.0333$, $\lambda_2 = 0.3334$, and $\lambda_3 = 0.6333$. The final RNN model developed using an ensemble model $h = \sum_{\tau=1}^{3} \lambda_\tau h_\tau$ can be used in MPC for the remaining operation time. However, if the CSTR is switched to another steady-state at some point in the future and the final ensemble RNN model does not perform well, an online update of RNN models will be activated again using real-time process data.

Through comparisons of prediction and testing errors between the initial and three online learning RNN models, it is demonstrated that the prediction performance of the three online learning

RNN models significantly outperforms that of the initial RNN model. Additionally, through closed-loop simulations under RNN-MPC, we have demonstrated that RNN-MPC using online learning RNN models successfully stabilizes the CSTR in a small neighborhood around the steady-state of mode 2, which outperforms the RNN-MPC using the initial offline learning RNN model.

Conclusion

This work proposed a Lyapunov-based MPC design with online learning RNN models for nonlinear systems that transit between their multiple operating regions according to the prescribed switching times. Specifically, an RNN model was initially developed offline using a limited dataset that was collected in a small operating region around a certain steady-state, and then an online update of RNN models was performed using the real-time process data when the nonlinear system was switched to another operating region following a prescribed switching schedule. The generalization error bound for online learning models was derived based on the regret bound analysis. Probabilistic closed-loop stability was developed for the switched nonlinear system under RNN-MPC. A chemical process example was utilized to demonstrate that the RNN-MPC using online learning RNN models achieved superior closed-loop performance compared with that using the offline learning RNN model.

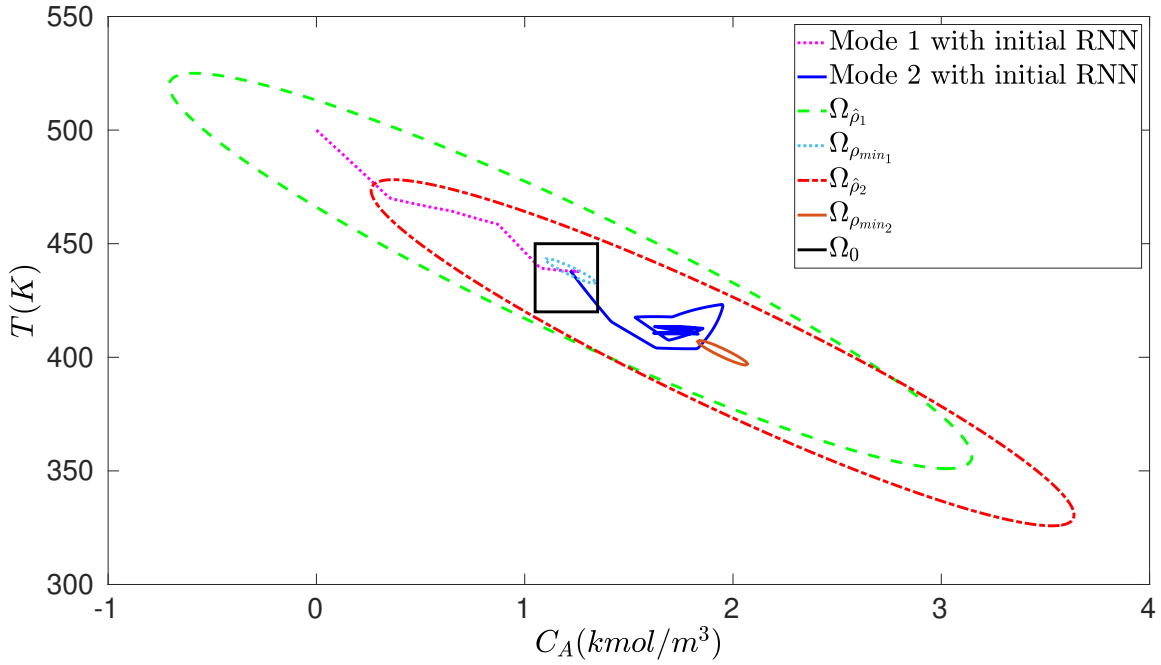
Literature Cited

1. A. Alacaoglu, Y. Malitsky, P. Mertikopoulos, and V. Cevher. A new regret analysis for Adam-type algorithms. In Proceedings of the 37th International Conference on Machine Learning, pages 202–210. Online, PMLR, 2020.
2. A. Alanqar, H. Durand, and P. D. Christofides. Error-triggered on-line model identification for model-based feedback control. AIChE Journal, 63:949–966, 2017.
3. Z. Allen-Zhu and Y. Li. Can SGD learn recurrent neural networks with provable generalization? Advances in Neural Information Processing Systems, 32, 2019.
4. Y. Cao and Q. Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In Proceedings of the 33th Conference on Neural Information Processing Systems, pages 10835–10845, Vancouver, Canada, 2019.
5. Y. Cao and Q. Gu. Generalization error bounds of gradient descent for learning over-parameterized deep ReLU networks. In Proceedings of the 34th AAAI Conference on Artificial Intelligence, pages 3349–3356, New York, USA, 2020.
6. N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. IEEE Transactions on Information Theory, 50:2050–2057, 2004.

7. N. Cesa-Bianchi and C. Gentile. Improved risk tail bounds for on-line algorithms. IEEE Transactions on Information Theory, 54:386–390, 2008.
8. D. Chaffart and L. A. Ricardez-Sandoval. Optimization and control of a thin film growth process: A hybrid first principles/artificial neural network based multiscale modelling approach. Computers & Chemical Engineering, 119:465–479, 2018.
9. M. Chen, X. Li, and T. Zhao. On generalization bounds of a family of recurrent neural networks. arXiv preprint arXiv:1910.12947, 2019.
10. X. Chen, S. Liu, R. Sun, and M. Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. arXiv preprint arXiv:1808.02941, 2018.
11. M. C. Choy, D. Srinivasan, and R. L. Cheu. Neural networks for continuous online learning and control. IEEE Transactions on Neural Networks, 17:1511–1531, 2006.
12. O. Dekel and Y. Singer. Data-driven online to batch conversions. Advances in Neural Information Processing Systems, 18, 2005.
13. J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12:2121–2159, 2011.
14. N. Golowich, A. Rakhlin, and O. Shamir. Size-independent sample complexity of neural networks. In Proceedings of the 31th Conference On Learning Theory, pages 297–299, Stockholm, Sweden, 2018.
15. J. Hanson, M. Raginsky, and E. Sontag. Learning recurrent neural net models of nonlinear systems. arXiv preprint arXiv:2011.09573, 2020.
16. M. Heidarinejad, J. Liu, and P. D. Christofides. Economic model predictive control of switched nonlinear systems. Systems & Control Letters, 62:77–84, 2013.
17. S. M. Kakade and A. Tewari. On the generalization ability of online strongly convex programming algorithms. Advances in Neural Information Processing Systems, 21, 2008.
18. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
19. V. Kuznetsov and M. Mohri. Time series prediction and online learning. In Proceedings of the 29th Annual Conference on Learning Theory, pages 1190–1213, New York, USA, 2016.
20. N. Littlestone. From on-line to batch learning. In Proceedings of the Second Annual Workshop on Computational Learning Theory, pages 269–284, Santa Cruz, USA, 1989.
21. J. Mendes-Moreira, C. Soares, A. Jorge, and J. F. D. Sousa. Ensemble approaches for regression: A survey. ACM Computing Surveys (CSUR), 45:1–40, 2012.

22. P. Mhaskar, N. H. El-Farra, and P. D. Christofides. Predictive control of switched nonlinear systems with scheduled mode transitions. IEEE Transactions on Automatic Control, 50:1670–1680, 2005.
23. M. Mohri, A. Rostamizadeh, and A. Talwalkar. Foundations of machine learning. MIT Press: Cambridge, MA, USA, 2018.
24. C. Ning and F. You. Online learning based risk-averse stochastic MPC of constrained linear uncertain systems. Automatica, 125:109402, 2021.
25. S. J. Reddi, S. Kale, and S. Kumar. On the convergence of Adam and beyond. arXiv preprint arXiv:1904.09237, 2019.
26. O. Sagi and L. Rokach. Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8:e1249, 2018.
27. L. Schwenkel, M. Gharbi, S. Trimpe, and C. Ebenbauer. Online learning with stability guarantees: A memory-based warm starting for real-time MPC. Automatica, 122:109247, 2020.
28. T. Tieleman, G. Hinton, et al. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning, 4:26–31, 2012.
29. A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming, 106:25–57, 2006.
30. Z. Wu, A. Alnajdi, Q. Gu, and P. D. Christofides. Statistical machine-learning–based predictive control of uncertain nonlinear processes. AIChE Journal, 68:e17642, 2022.
31. Z. Wu, D. Rincon, and P. D. Christofides. Real-time adaptive machine-learning-based predictive control of nonlinear processes. Industrial & Engineering Chemistry Research, 59:2275–2290, 2019.
32. Z. Wu, D. Rincon, Q. Gu, and P. D. Christofides. Statistical machine learning in model predictive control of nonlinear processes. Mathematics, 9:1912, 2021.
33. Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine learning-based predictive control of nonlinear processes. Part I: theory. AIChE Journal, 65:e16729, 2019.
34. Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine learning-based predictive control of nonlinear processes. Part II: Computational implementation. AIChE Journal, 65:e16734, 2019.
35. Y. Zheng, T. Zhao, X. Wang, and Z. Wu. Online learning-based predictive control of crystallization processes under batch-to-batch parametric drift. AIChE Journal, in press, 2022.
36. D. Zhou, J. Chen, Y. Cao, Y. Tang, Z. Yang, and Q. Gu. On the convergence of adaptive gradient methods for nonconvex optimization. arXiv preprint arXiv:1808.05671, 2018.

(a)



(b)

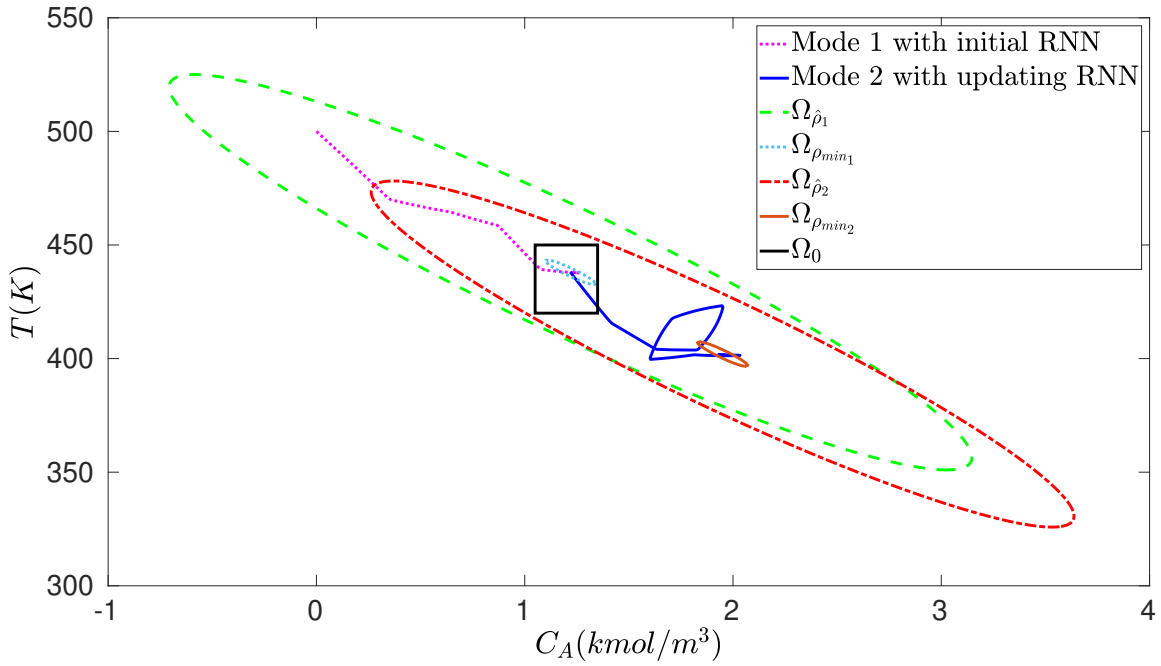
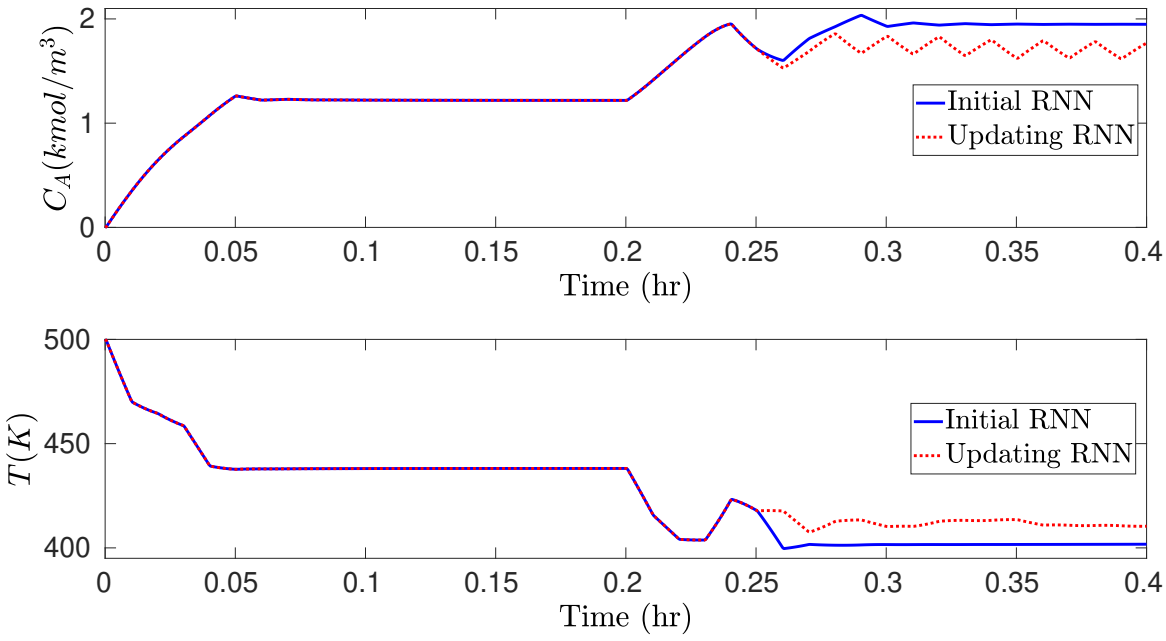


Figure 1: Closed–loop state trajectories for an initial state $x_0^T = [0\ kmol/m^3\ 500\ K]$ when the CSTR is operated in mode 1 for $t \in [0,\ 0.2\ hr)$ under the RNN-MPC of Eq. (29) (dashed pink line), and when the CSTR is switched to mode 2 at $t = 0.2\ hr$ under the RNN-MPC of Eq. (27) (solid blue line), (a) using the initial RNN model for both mode 1 and 2, and (b) using the initial RNN model for mode 1 and online learning RNN models for mode 2.

(a)



(b)

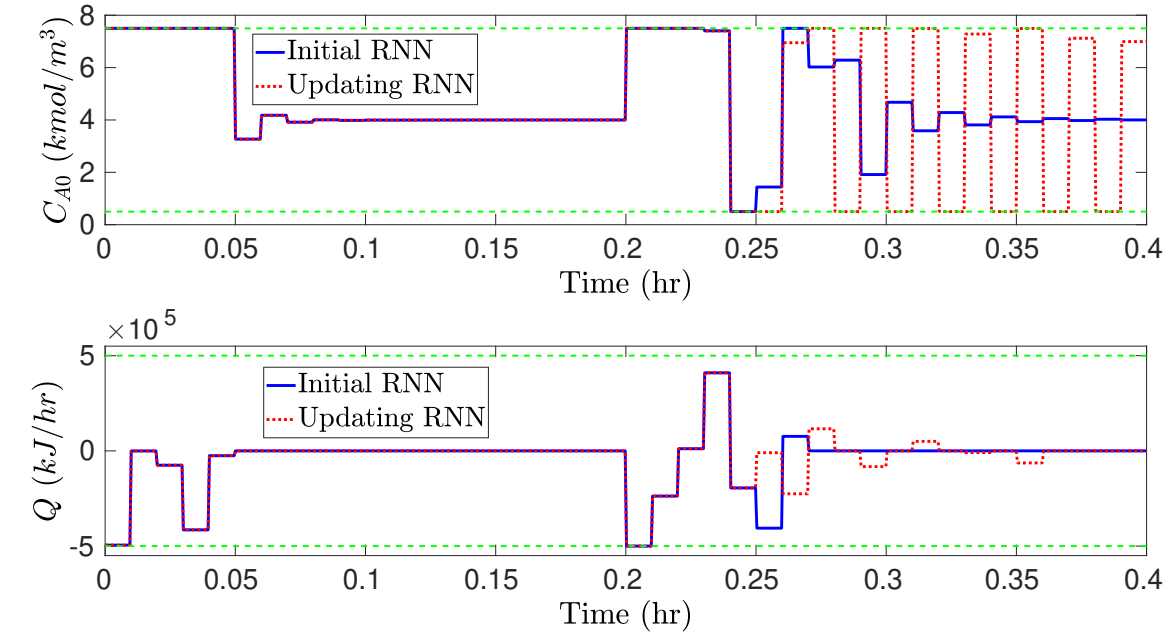


Figure 2: (a) Closed–loop state and (b) manipulated input profiles for an initial state $x_0^T = [0\ kmol/m^3\ 500\ K]$ when the CSTR is operated in mode 1 for $t \in [0,\ 0.2\ hr)$ under the RNN-MPC of Eq. (29), and when the CSTR is switched to mode 2 at $t = 0.2\ hr$ under the RNN-MPC of Eq. (27), using the initial RNN model for both mode 1 and 2 (dashed line), and using the initial RNN model for mode 1 and online learning RNN models for mode 2 (solid line).

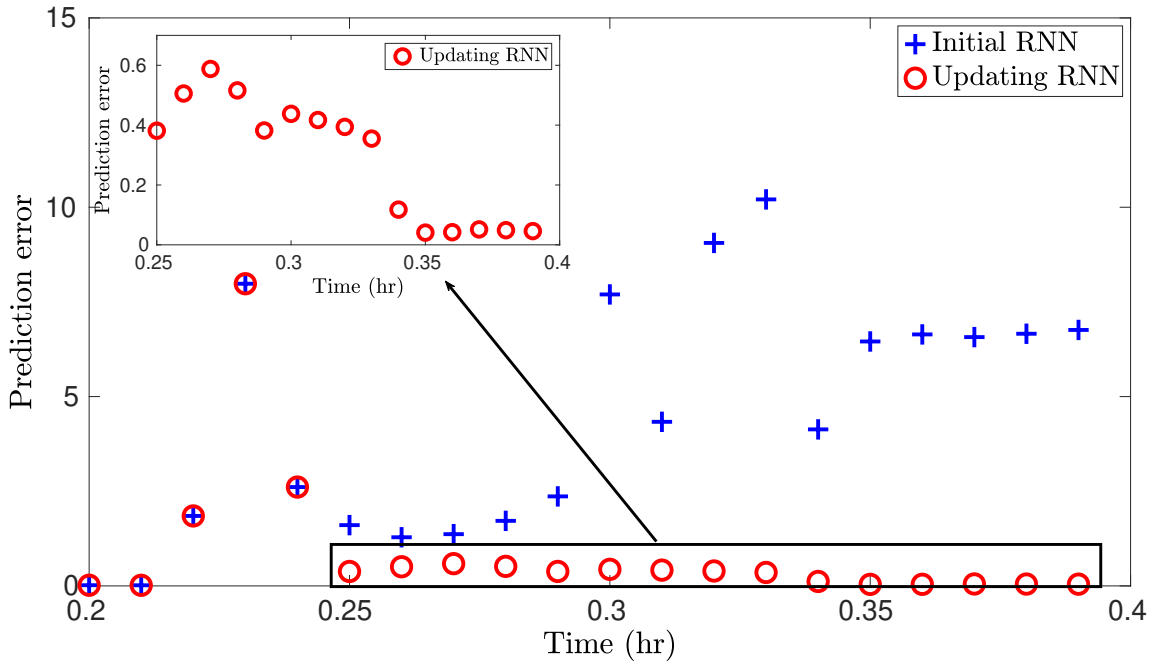


Figure 3: Prediction error for each sampling step when the CSTR is switched to mode 2 at $t = 0.2\ hr$ under the RNN-MPC of Eq. (27) using the initial RNN model (marked as blue +), and the online learning RNN models (marked as red circle).