Received: 24 April 2019 | Revised: 26 June 2019 | Accepted: 16 July 2019

DOI: 10.1002/aic.16734

PROCESS SYSTEMS ENGINEERING

AIChE JOURNAL

Machine-learning-based predictive control of nonlinear processes. Part II: Computational implementation

Zhe Wu[1] | Anh Tran[1] | David Rincon[1] | Panagiotis D. Christofides[1,2]

[1]Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, California

[2]Department of Electrical and Computer Engineering, University of California, Los Angeles, California

Correspondence
Panagiotis D. Christofides, Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095.
Email: pdc@seas.ucla.edu

Funding information
National Science Foundation and the Department of Energy

Abstract

Machine learning is receiving more attention in classical engineering fields, and in particular, recurrent neural networks (RNNs) coupled with ensemble regression tools have demonstrated the capability of modeling nonlinear dynamic processes. In Part I of this two-article series, the Lyapunov-based model predictive control (LMPC) method using a single RNN model and an ensemble of RNN models, respectively, was rigorously developed for a general class of nonlinear systems. In the present article, computational implementation issues of this new control method ranging from training of the RNN models, ensemble regression of the RNN models, and parallel computation for accelerating the real-time model calculations are addressed. Furthermore, a chemical reactor example is used to demonstrate the implementation and effectiveness of these machine-learning tools in LMPC as well as compare them with standard state-space model identification tools.

KEYWORDS

ensemble learning, model predictive control, nonlinear systems, parallel computing, recurrent neural networks

1 | INTRODUCTION

The last 20 years have witnessed the growth of machine learning and deep learning with the development of advanced machine-learning algorithms, innovative neural network structures, powerful computers, and user-friendly, open-source software libraries. All of the above led to a variety of successful applications of machine learning involving social media service, pattern recognition, and virtual assistants via natural language processing. As one of the most popular machine-learning techniques, neural networks have been attractive to many researchers and companies, which have contributed to a lot of research work on feedforward neural networks, convolutional neural networks, and recurrent neural networks (RNNs). Recently, RNNs have been applied to solve model predictive control problems for nonlinear systems in several research works.[1-3] In the first article of this series, an RNN-based model predictive control (MPC) method has been developed with both the approximation theory of RNN and closed-loop stability analysis. However, applications of RNNs to problems in which real-time calculations are required may be limited by the high degree of complexity present in neural networks, which makes them computational expensive to train and calculate.[4]

To address the computational issue in neural networks, both hardware acceleration and advanced machine-learning algorithms that utilize parallel computing should be developed to reduce computational time in training and implementation of a neural network. Specifically, artificial intelligence accelerator that incorporates graphics processing units or field-programmable gate arrays has been utilized in the design of a computer system that facilitates deep learning. Additionally, many works haven been done to improve computation efficiency of training neural networks. For example, in Reference 5 the training process of a neural network is distributed to multiple agents, where each agent solves a local non-convex problem with a portion of training data while communicating with each other. In References 4,6, the training process of artificial neural networks was implemented in parallel by two different approaches: data parallelization and node parallelization. Specifically, in data parallelization, the training data set is divided into

disjoint subsets for multiple threads and the weights are synchronized and distributed periodically, while in node parallelization, the network layers are divided into disjoint sets of neurons and the results are combined via synchronization before moving to the next layer. Besides the parallelization of training algorithms, in Reference [7], a parallel computing platform was developed to manage big data involving large-scale data set storage and parallel loading.

On the other hand, parallel computing is also utilized in training and implementing multiple neural networks in the context of ensemble learning, which has been demonstrated to achieve better predictive performance than a single neural network. Specifically, ensemble learning takes advantage of multiple machine-learning algorithms to solve a particular regression or classification problem, in which bagging, boosting, and stacking are the three most common methods.[8] The idea of ensemble learning has been applied in several chemical engineering problems. For instance, the ensemble Kalman filter has been proposed as a variation of the well-known Kalman filter in References 9,10. The ensemble Kalman filter has shown benefits in reducing the error of approximation of the covariance matrix for high-dimensional models, and in obtaining better error statistics for the nonlinear case.[10] In chemical process control, the constrained ensemble Kalman filter has been proposed for dealing with bounds and has been tested with a gas-phase reactor and an isothermal batch reactor.[11] Similarly, ensemble Kalman filter has been extended for systems described by differential algebraic equations.[12] Despite the great efforts that have been done in state estimation using ensemble methods, ensemble learning has not been fully investigated in model predictive control, in which parallel computing can also be utilized to reduce computation time.

Motivated by the above considerations, in this work, an ensemble of RNN models is initially developed via parallel computation, and then, the Lyapunov-based MPC (LMPC) based on the ensemble of RNN models is applied to a chemical process example in which prediction is implemented in parallel. The rest of this paper is organized as follows: in the "Computational implementation issues of RNN models" section, the implementation of RNN models for longer prediction horizon is introduced. In the "Parallel computing" section, a brief recap of the formulation of LMPC using RNN models developed in Part I is first provided. Then, the necessary approximation via numerical methods and parallel computing is introduced for the real-time implementation of LMPC. In the last section, closed-loop stability and enhanced computational efficiency of the proposed LMPC using an RNN model, and an ensemble of RNN models, respectively, are demonstrated through a continuous stirred tank reactor (CSTR) example. Specifically, the generation of data set based on extensive open-loop simulations is first discussed. Then, a linear state-space model obtained from data-driven modeling is introduced for comparison purposes. Closed-loop simulations for the CSTR are conducted under the LMPC using the first-principles nonlinear model, the linear state-space model, a single RNN model, and an ensemble of RNN models, respectively. Lastly, computational efficiency of the closed-loop simulation under LMPC is found to be improved under the parallel operation of an ensemble of RNN models.

2 | COMPUTATIONAL IMPLEMENTATION ISSUES OF RNN MODELS

In this section, we address computational implementation issues for the RNN models obtained following the training algorithm in Part I. Specifically, the implementation of RNN models for long prediction horizon is first discussed. Then, numerical methods are employed to evaluate modeling error and approximate the Lyapunov-based constraints in LMPC, respectively.

2.1 | Long prediction horizon

Although the ensemble of RNN models developed in Part I is to predict future states over $t \in [t_k, t_k + P_{nn}]$ given the states and inputs at $t = t_k$, where P_{nn} is an integer multiple of the sampling period Δ, it is noted that ensemble regression models can be applied to predict states for longer period of time (i.e., $t \in [t_k, t_k + NP_{nn}]$, $N > 1$) in practical applications, for example, model predictive control. Specifically, the obtained RNN models will be utilized successively at every prediction step $t = t_k + iP_{nn}$, $i = 0, 1, \ldots, N-1$, to predict all the states within the entire prediction horizon $t \in [t_k, t_k + NP_{nn}]$, in which the prediction results (i.e., the output vector $x(t_k + iP_{nn})$) from the previous RNN models will be used as the initial states for the current prediction to predict states over $[t_k + iP_{nn},\ t_k + (i+1)P_{nn}]$, $i = 0, 1, \ldots, N-1$. Additionally, since the means and the SD for normalizing inputs and rescaling outputs could be slightly different, intermediate rescaling and normalizing steps should be performed between two successive ensemble prediction steps during the entire prediction horizon.

Before we apply the obtained RNN models within LMPC, the testing data set that has not been used in the training process can be utilized to test the prediction performance of RNNs. In this case, the normalizing and rescaling functions before and after the ensemble of RNN models should be updated with the statistics of the testing data set. Specifically, the normalizing and rescaling functions during the training process are constructed based on the statistics of the training data set only instead of the entire data set due to the following reasons. First, the training and testing data sets may not be equally representative of the operating region considered, and thus, the training and testing data sets should be normalized separately. Second, data leakage that introduces information from outside, for example, testing data set, into RNN model should be prevented during the training process to avoid creating an overly optimistic but potentially invalid predictive model. Therefore, based on the normalizing and rescaling functions designed for the testing data set, the prediction performance of RNN models is evaluated by the mean square error between the predicted states of the RNN models and the actual states derived from the nominal nonlinear system $\dot{x} = f(x) + g(x)u$.

Remark 1. When ensemble regression models are utilized to improve prediction accuracy, the final prediction results are obtained via the stacking method in this work. Specifically, the final predicted states are obtained by averaging k RNN models that are developed from a k-fold

cross validation. However, it is noted that averaging through the stacking method is not the only approach that can be applied here. For example, the bagging method that trains multiple models based on different subsets of the training data set and calculates final predictive results through averaging or majority voting can be utilized to reduce the variance error.[13] The boosting method can also improve final predictive accuracy by adding more weights to incorrect prediction during the iterative training process. Additionally, further improvements may be achieved by combining results of multiple models that are derived using different machine-learning methods through Bayesian model averaging.[14,15]

2.2 | Approximation via numerical methods

Since we mainly discuss the continuous RNN model in Part I, while in practice, the data sets for training RNN models are mostly generated by a sample-data collection from industrial processes, lab experiments, or numerical simulation, necessary approximations should be performed to incorporate the RNN model trained on sample data within LMPC. Specifically, numerical methods are utilized to compute modeling error, characterize the closed-loop stability region $\Omega_{\hat{\rho}}$ for the RNN model, and calculate $\dot{\hat{V}}(x(t_k),u)$ in the constraint of Equation (1f) in the LMPC below, respectively.

Before we present the details of numerical approximation methods, the formulation of the LMPC using an ensemble of RNN models developed in Part I is given as follows:

$$\mathcal{J} = \min_{u\in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t),u(t))dt \tag{1a}$$

$$\text{s.t. } \dot{\tilde{x}}(t) = \frac{1}{N_e}\sum_{j=1}^{N_e} F_{nn}^j(\tilde{x}(t),u(t)) \tag{1b}$$

$$u(t)\in U, \quad \forall t\in[t_k,t_{k+N}) \tag{1c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{1d}$$

$$\dot{\hat{V}}(x(t_k),u) \le \dot{\hat{V}}(x(t_k),\Phi_{nn}(x(t_k))), \quad \text{if } x(t_k)\in\Omega_{\hat{\rho}}\backslash\Omega_{\rho_{nn}} \tag{1e}$$

$$\hat{V}(\tilde{x}(t)) \le \rho_{nn}, \quad \forall t\in[t_k,t_{k+N}), \quad \text{if } x(t_k)\in\Omega_{\rho_{nn}} \tag{1f}$$

Unlike the LMPC using a single RNN model, the LMPC of Equation (1) predicts future states by averaging prediction results of N_e RNN models to improve prediction accuracy. Additionally, Theorem 5 in Part I establishes closed-loop stability for the nonlinear system in the presence of sufficiently small bounded disturbances (i.e., $\dot{x} = f(x)+g(x)u+h(x)w, |w|\le w_m$) in the sense that the closed-loop state is bounded in the closed-loop stability region $\Omega_{\hat{\rho}}$ for all times and ultimately converges to a small neighborhood around the origin $\Omega_{\rho\min}$. In the following subsections, necessary approximation in training an RNN model, characterization of the closed-loop stability region for the RNN model, and the incorporation of RNN model for predicting future states in LMPC are presented.

2.2.1 | Approximation of modeling error

Since the RNN is trained to predict future states over $t\in[t_k, t_k+P_{nn})$, in which the RNN output is the state at t_k+P_{nn} and the time interval between internal states is chosen as the integration time step h_c, the modeling error $\nu = \dot{x}(t_k)-\dot{\hat{x}}(t_k)$ at the state $x(t_k)=\hat{x}(t_k)$ is approximated using a forward finite difference method during the training process as follows:

$$\begin{aligned}|\nu| &= \left|\frac{x(t_k+h_c)-x(t_k)}{h_c} - \frac{\hat{x}(t_k+h_c)-\hat{x}(t_k)}{h_c}\right| \\ &= \left|\frac{x(t_k+h_c)-\hat{x}(t_k+h_c)}{h_c}\right|\end{aligned} \tag{2}$$

where h_c is a sufficiently small time interval, $x(t_k+h_c)$ is obtained via explicit Euler method with an integration time step h_c, and $\hat{x}(t_k+h_c)$ is the first internal state of the RNN model. Then, the constraint $|\nu|\le\gamma|x|$ is satisfied if the following equation holds:

$$\left|\frac{x(t_k+h_c)-\hat{x}(t_k+h_c)}{x(t_k+h_c)}\right| \le \gamma h_c \tag{3}$$

According to Equation (3), the mean absolute percentage error between predicted states $\hat{x}$ and targeted states x in training data can be utilized as a metric to indicate the modeling error of RNNs.

2.2.2 | Characterization of closed-loop stability region

The stabilizing controller $u=\Phi_{nn}(x)\in U$ (e.g., the universal Sontag control law[16]) is initially utilized to characterize the set $\hat{\phi}_u$ and the closed-loop stability region $\Omega_{\hat{\rho}}$ based on the RNN model written in the form of $\dot{\hat{x}} = \hat{f}(\hat{x})+\hat{g}(\hat{x})u$. However, since it is difficult to derive the explicit forms of $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ for an RNN with a complex structure, numerical methods are utilized to approximate $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$. For example, $\hat{f}(\cdot)$ can be approximated by the predicted $\dot{\hat{x}}$ with $u=0$, where $\dot{\hat{x}}$ is obtained using the forward finite difference method as shown in the previous section. Then, $\hat{g}(\cdot)$ is approximated by $\hat{g}(\hat{x}) = \left(\dot{\hat{x}}-\hat{f}(\hat{x})\right)/u$ with a nonzero u. Since the minimum prediction step in RNNs is the sufficiently small integration time step h_c, the approximation results via numerical methods can be regarded as a good representation of the actual $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ of an RNN model.

After $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ are obtained, a simulation with a full sweep over the entire state-space based on the stabilizing controller $u=\Phi_{nn}(x)\in U$ is performed to characterize the region $\hat{\phi}_u = \left\{x\in\mathbb{R}^n \mid \dot{\hat{V}}(x) < -k\hat{V}(x), u=\Phi_{nn}(x)\in U\right\}\cup\{0\}$ with $k>0$, in which $\dot{\hat{V}}(x) = \frac{\partial\hat{V}(x)}{\partial x}(F_{nn}(x,u))$ is approximated via forward finite difference method. Subsequently, the closed-loop stability region $\Omega_{\hat{\rho}} := \left\{x\in\hat{\phi}_u \mid \hat{V}(x)\le\hat{\rho}\right\}$ is characterized as the largest level set of $\hat{V}(x)$ in $\hat{\phi}_u$.

2.2.3 | Approximation of Lyapunov-based constraints

Additionally, $\dot{\hat{V}}(x,u)$ in the Lyapunov-based constraint of Equation (1e) is approximated via the same numerical method (i.e., forward finite difference method). It is noted that the approximation of $\dot{\hat{V}}(x,u)$ does not affect closed-loop stability of the actual nonlinear system (i.e., $\dot{x} = F(x,u,w) := f(x) + g(x)u + h(x)w$) under the constraint of Equation (1e) since the same numerical method is used to approximate both $\dot{\hat{V}}(x,u)$ and $\dot{\hat{V}}(x,\Phi_{nn}(x))$. Specifically, it has been shown above that the controller $u = \Phi_{nn}(x) \in U$ is able to stabilize the actual nonlinear system at the origin for all x in the closed-loop stability region $\Omega_{\hat{\rho}}$ since $\dot{\hat{V}}(x,\Phi_{nn}(x)) \leq -k\hat{V}(x)$ is satisfied in $\Omega_{\hat{\rho}} \subset \hat{\phi}_u$ that is characterized via the numerical computation of $\dot{\hat{V}}(x,\Phi_{nn}(x))$. Therefore, the inequality $\dot{\hat{V}}(x,u) \leq \dot{\hat{V}}(x,\Phi_{nn}(x)) \leq -k\hat{V}(x)$ holds under the constraint of Equation (1e) if the same numerical method is utilized, which ensures closed-loop stability for the nonlinear system under LMPC.

3 | PARALLEL COMPUTING

It was proposed in Part I that an ensemble of RNN models is utilized in LMPC to provide more accurate predicted states through the average of the predicted results from multiple RNN models. As a result, the closed-loop performance could be improved in the sense that the closed-loop state is closer to that under the LMPC using the first-principles model, and thus, it converges to the origin quickly. Considering the significant increase of computation time from multiple RNN models, parallel computing is employed to reduce real-time computation time.

Parallel computing is a type of computation in which the execution of multiple processes is carried out simultaneously.[17] Generally, it takes advantage of multiple compute resources (e.g., a single computer with multiple processors/cores or many computers connected by a network) to solve a computationally heavy task, in which a complex problem can be broken into discrete parts that can be solved concurrently. Additionally, parallel computing can be categorized into two types based on whether there exists communication between processors/networked computers: (a) In parallel computing without communication, multiple processors execute multiple tasks simultaneously and generate the results independently. (b) In parallel computing with communication (sometimes it is also called distributed computing), networked computers or multiple processors communicate and coordinate the work through message passing interface (MPI) to obtain final results. Based on the computation tasks for training multiple RNN models and calculating the average of multiple RNN prediction results in LMPC, the first type and the second type of parallel computing are applied to these two tasks, respectively, to enhance computational efficiency in both cases.

3.1 | Training multiple RNNs in parallel

Multiple RNN models are constructed via a k-fold cross-validation method discussed in Part I. Specifically, if k RNN models are utilized in the LMPC of Equation (1), the computation time for training all RNN models in series is approximately k times larger than that for a single RNN model. It is noted that the resulting increase of computation time is unnecessary since the training processes for k RNN models are independent from each other. Therefore, parallel computing is utilized to distribute the training processes to multiple processors such that k RNN models can be trained simultaneously. The training processes of k RNN models are implemented in parallel with the following steps: (a) k processors are first reserved with sufficient memory. (b) Based on k-fold cross validation, the entire data set is partitioned into k folds with the same size, which are then distributed to all reserved processors. (c) For the kth processor, the RNN model is trained with $k - 1$ subsets (i.e., the kth subset is excluded) as the training data set and the remaining kth subset as the validation data set. (d) A bash script is created to run all k processors together such that the training processes for the k RNN models can be executed concurrently. Since the stopping criteria might not be satisfied by the k training processes simultaneously due to different training data sets, the total computation time is determined by the slowest training process.

Remark 2. It should be mentioned that the aim of parallel computing in this subsection is to train multiple RNN models simultaneously, not to speed up the RNN training process itself. Although training phase parallelism and data parallelization for accelerating RNN training[4,6] are also efficient approaches to reduce overall computation time, they are not investigated in this work.

3.2 | Parallel operation of LMPC using an ensemble of RNNs

An ensemble of RNN models is utilized in the LMPC of Equation (1), under which prediction accuracy is improved and closed-loop stability of the nonlinear system remains valid. Since the optimal solution $u^*(t)$ is now computed based on the states predicted by multiple RNN models, the computation time for an ensemble of N_e RNN models increases rapidly (at least N_e times the original computation time for the LMPC based on a single RNN model) under serial computation of Equation (1b), which greatly limits the application of ensemble regression model-based LMPC in industry. Therefore, in this subsection, parallel computing is utilized to reduce the computation time of calculating multiple RNN models of Equation (1b).

Specifically, in the LMPC optimization problem of Equation (1), the state prediction given by Equation (1b) can be broken apart into N_e similar sub-tasks that can be processed independently and simultaneously. Consider using N_e ($N_e \leq k$) RNN models for prediction of Equation (1b). The calculation of Equation (1b) through parallel computing consists of the following steps: (a) We first reserve $N_e + 1$ nodes, in which node 0 is the host node and the rest are worker

nodes. The host node is used to receive and send information while the worker nodes are mainly used for computation. (b) The optimization problem is running on the host node while the computation of multiple RNN models is assigned to worker nodes. Specifically, when it comes to state prediction using Equation (1b), the host node is executed first to broadcast $x(t_k)$ and $u(t)$ to all nodes since ensemble regression models in Equation (1b) share the same initial condition $x(t_k)$ and the same guess of control actions $u(t)$ at $t = t_k$. (c) Each worker node is assigned with an RNN model for prediction and the host node gathers the results from worker nodes and compute the average as the final result. (d) The optimal control action $u^*(t_k)$ is sent to the real system to be applied for the next sampling period by the host node. The above process is repeated every sampling step (i.e., at the next sampling time t_{k+1}, the LMPC of Equation (1) receives the state measurement $x(t_{k+1})$ and sends it to the host node. Then, steps 1–4 are repeated to parallelize the computation of Equation 1b).

Remark 3. Computational efficiency of the LMPC optimization problem of Equation (1) is significantly improved through the parallel operation of N_e independent ensemble regression models. However, it is noted that the computation time may not be reduced exactly by N_e times under parallel operation due to the communication and waiting time between the host node and the worker nodes. It is also important to mention that the communication between the LMPC and the process model, and the main program of the optimization problem itself are running on the host node only. Additionally, synchronization operation should be employed when the host node combines all the results from worker nodes to ensure that each task in worker node blocks until all tasks in the computing group reach the host node.

4 | APPLICATION TO A CHEMICAL PROCESS EXAMPLE

A chemical process example is used to illustrate the application of LMPC using RNN models to maintain the closed-loop state within the stability region. Specifically, a well-mixed, nonisothermal CSTR where an irreversible second-order exothermic reaction takes place is considered. The reaction transforms a reactant A to a product B ($A \rightarrow B$). The inlet concentration of A, the inlet temperature and feed volumetric flow rate of the reactor are C_{A0}, T_0, and F, respectively. The CSTR is equipped with a heating jacket that supplies/removes heat at a rate Q. The CSTR dynamic model is described by the following material and energy balance equations:

$$\frac{dC_A}{dt} = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT}} C_A^2 \tag{4a}$$

$$\frac{dT}{dt} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \tag{4b}$$

where C_A is the concentration of reactant A in the reactor, V is the volume of the reacting liquid in the reactor, T is the temperature of the reactor, and Q denotes the heat input rate. The concentration of reactant A in the feed is C_{A0}. The feed temperature and volumetric flow rate are T_0 and F, respectively. The reacting liquid has a constant density of ρ_L and a heat capacity of C_p. ΔH, k_0, E, and R represent the enthalpy of reaction, pre-exponential constant, activation energy, and ideal gas constant, respectively. Process parameter values are listed in Table 1.

TABLE 1 Parameter values of the continuous stirred tank reactor (CSTR)

$T_0 = 300$ K	$F = 5\ \text{m}^3/\text{hr}$
$V = 1\ \text{m}^3$	$E = 5 \times 104$ kJ/kmol
$k_0 = 8.46 \times 10^6\ \text{m}^3/\text{kmol hr}$	$\Delta H = -1.15 \times 10^4$ kJ/kmol
$C_p = 0.231$ kJ/kg K	$R = 8.314$ kJ/kmol K
$\rho_L = 1000\ \text{kg/m}^3$	$C_{A0_s} = 4\ \text{kmol/m}^3$
$Q_s = 0.0$ kJ/hr	$C_{A_s} = 1.22\ \text{kmol/m}^3$
$T_s = 438$ K	

The CSTR is initially operated at the unstable steady-state (C_{As}, T_s) = (1.95 kmol/m^3, 402 K), and $(C_{A0_s}\ Q_s) = (4\ \text{kmol/m}^3, 0\ \text{kJ/hr})$. The manipulated inputs are the inlet concentration of species A and the heat input rate, which are represented by the deviation variables $\Delta C_{A0} = C_{A0} - C_{A0s}$, $\Delta Q = Q - Q_s$, respectively. The manipulated inputs are bounded as follows: $|\Delta C_{A0}| \le 3.5\ \text{kmol/m}^3$ and $|\Delta Q| \le 5 \times 10^5$ kJ/hr. Therefore, the states and the inputs of the closed-loop system are $x^T = [C_A - C_{As}\ \ T - T_s]$ and $u^T = [\Delta C_{A0}\ \ \Delta Q]$, respectively, such that the equilibrium point of the system is at the origin of the state-space, (i.e., $(x_s^*, u_s^*) = (0,0)$).

The control objective is to operate the CSTR at the unstable equilibrium point (C_{As}, T_s) by manipulating the heat input rate ΔQ and the inlet concentration ΔC_{A0} under the LMPC using RNN models. The explicit Euler method with an integration time step of $h_c = 10^{-4}$ hr is applied to numerically simulate the dynamic model of Equation (4). The nonlinear optimization problem of the LMPC of Equation (1) is solved using the python module of the IPOPT software package,[18] named PyIpopt with the sampling period $\Delta = 10^{-2}$ hr.

4.1 | Data Generation

To apply the LMPC of Equation (1) to the CSTR of Equation (4), extensive open-loop simulations are first conducted in the closed-loop stability region Ω_ρ for the CSTR of Equation (4) to generate the data set for RNN models, and subsequently, RNN models are developed to capture the system dynamics in Ω_ρ with a desired degree of accuracy. The control Lyapunov function $V(x) = x^T P x$ is designed with the following positive definite P matrix:

$$P = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{5}$$

Then, the closed-loop stability region Ω_ρ for the CSTR with $\rho = 372$ is characterized as a level set of Lyapunov function inside the region ϕ_u, from which the origin can be rendered exponentially stable under the controller $u = \Phi(x) \in U$. Open-loop simulations are performed with a full sweep through all of the feasible initial conditions $x_0 \in \Omega_\rho$ and inputs $u \in U$ for finite sampling steps, from which the state trajectories represented by sampled data points are collected

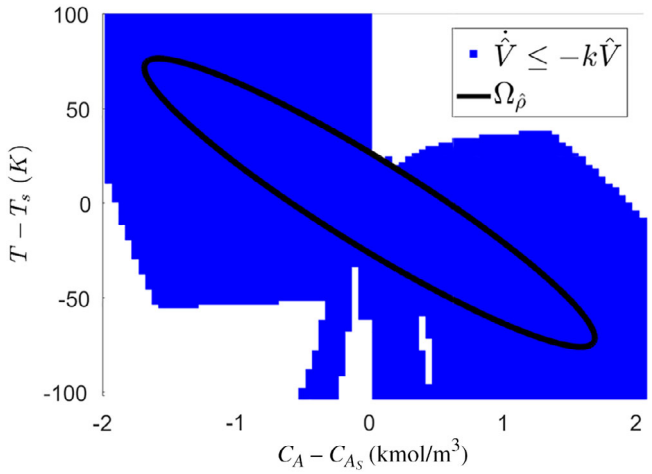

FIGURE 1 The set $\hat{\phi}_u$ represented by the blue region and the stability region $\Omega_{\hat{\rho}}$ (black ellipse) for the closed-loop CSTR under the controller $u = \Phi_{nn}(x) \in U$. CSTR, continuous stirred tank reactor [Color figure can be viewed at wileyonlinelibrary.com]

with a minimum time step as the integration time step h_c. Subsequently, the RNN is developed based on the data set generated from open-loop simulations in Ω_ρ to predict future states over one sampling period Δ with the minimum prediction period h_c using the state-of-the-art API, Keras. Specifically, the RNN model is designed to have two hidden recurrent layers consisting of 96 and 64 recurrent units, respectively, and use the sigmoid function as the activation function. The stopping criteria for the training process include the modeling error less than a threshold and early stopping being triggered. Additionally, a 10-fold cross validation is used to construct homogeneous ensemble regression models for the LMPC of Equation (1) using multiple RNN models. After the RNN model is obtained, the Lyapunov function $\hat{V}(x)$ for the RNN model is chosen to be the same as $V(x)$, and the set $\hat{\phi}_u$, in which $\dot{\hat{V}} \leq -k\hat{V}$ holds, is characterized in Figure 1 using the controller $u = \Phi_{nn}(x) \in U$ with the approximation approach discussed before. The closed-loop stability region $\Omega_{\hat{\rho}}$ for the CSTR system described by the RNN model is characterized as the largest level set of $\hat{V}$ in $\hat{\phi}_u$ and also a subset of Ω_ρ (i.e., $\Omega_{\hat{\rho}} \subset \Omega_\rho$) with $\hat{\rho} = 368$. Additionally, $\rho_{nn} = 1.6$ and $\rho_{\min} = 2$ are determined through extensive simulations for $u \in U$. The LMPC cost function of Equation (1a) is designed to be $L(x,u) = |x|^2_{Q_1} + |u|^2_{Q_2}$, where Q_1 = [500 0; 0 0.5] and Q_2 = [1 0; 0 8×10^{-11}], such that the minimum value of L is achieved at the origin. It is noted that since the steady-state (C_{As}, T_s) = (1.95 kmol/m^3, 402 K) is an unstable equilibrium point of the system of Equation (4), open-loop simulations are performed for a few sampling periods only to guarantee that state trajectories starting from Ω_ρ do not diverge quickly and can be bounded in a slightly larger region.

4.2 | Linear state-space model

To illustrate the effectiveness of the proposed LMPC of Equation (1) using RNN models, we also compare it with the LMPC using a linear state-space model and the first-principles model of Equation (4), respectively. The linear state-space model for the CSTR system of Equation (4) is identified with the following form:

$$\dot{x} = A_s x + B_s u \tag{6}$$

where x and u are the state vector and the manipulated input vector, A_s and B_s are coefficient matrices for the state-space model. Following the system identification method in Reference 19, the numerical algorithms for subspace state-space system identification is utilized to obtain A_s and B_s as follows:

$$A_s = 100 \times \begin{bmatrix} -0.154 & -0.003 \\ 5.19 & 0.138 \end{bmatrix} \tag{7}$$

$$B_s = \begin{bmatrix} 4.03 & 0 \\ 1.23 & 0.004 \end{bmatrix} \tag{8}$$

The eigenvalues of matrix A_s are calculated to be $\lambda_1 = -5$ and $\lambda_2 = 3.14$, which is consistent with the fact that the steady-state (C_{As}, T_s) = (1.95 kmol/m^3, 402 K) is an unstable equilibrium point of CSTR.

4.3 | Simulation results

We first carry out simulation results under the LMPC using the RNN model and the first-principles model of Equation (4), respectively. It should be noted that the machine-learning approach is used when only data are available. The first-principles model in the following simulations substitutes for the role of the experimental/industrial process. In other words, the MPC using first-principles model only serves as a benchmark to determine the best performance that any data-driven modeling method can achieve. In Figure 2, it is demonstrated that starting from the same initial condition $x_0 \in \Omega_{\hat{\rho}}$ with the same input sequences, the state trajectories for a fixed finite interval of time under the RNN model are close to those under the first-principles model of the nonlinear CSTR of Equation (4). This implies that the well-trained RNN model can be regarded as a good representation for the CSTR first-principles model of Equation (4). Next, the RNN model is incorporated in the LMPC of Equation (1) using a single RNN model, under which the closed-loop state trajectories, state and manipulated input profiles of the system of Equation (4) are shown in Figures 3–6.

Figure 3 demonstrates that for initial conditions $x_0 \in \Omega_{\hat{\rho}}$, the closed-loop state is bounded in the closed-loop stability region $\Omega_{\hat{\rho}}$ for all times and ultimately converges to a small neighborhood around the origin ($\Omega_{\rho\min}$) under the LMPC of Equation (1) using a single RNN model. Additionally, Figure 4 shows the comparison of state trajectories for the closed-loop system under the LMPC using a single RNN model, the state-space model of Equation (6) and the first-principles model of Equation (4), respectively. It is demonstrated that in all cases, the state of the closed-loop system of Equation (4) is maintained within $\Omega_{\hat{\rho}}$ for all times and driven to $\Omega_{\rho\min}$ under LMPC for an initial condition x_0 = (−1, 63.6). However, through the comparison of state profiles under the LMPC using three different models in Figures 5, it is shown that the state trajectory under the RNN model stays closer to the one under the actual nonlinear model of Equation (4), and thus,

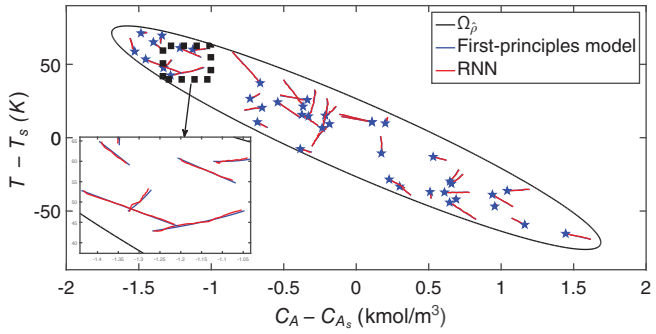

FIGURE 2 The state-space profiles for the open-loop simulation using the first-principles model of Equation (4) and the RNN model, respectively, for various sets of inputs and initial conditions (marked as blue stars) x_0 in the closed-loop stability region $\Omega_{\hat{\rho}}$. RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

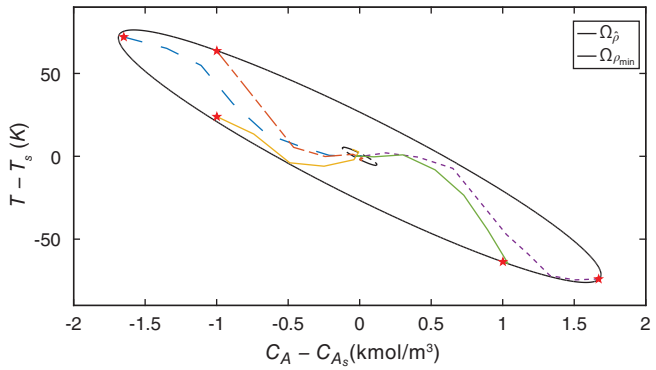

FIGURE 3 The state-space profiles for the closed-loop CSTR under the LMPC of Equation (1) using RNN models for various initial conditions (marked as red stars) in the closed-loop stability region $\Omega_{\hat{\rho}}$. CSTR, continuous stirred tank reactor; LMPC, Lyapunov-based model predictive control; RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

takes less time to settle to the steady-state compared to the LMPC using the state-space model. It is also noted that although the LMPC using the state-space model performs well for some initial conditions close to the origin, it shows oscillation for initial conditions near the boundary of the closed-loop stability region $\Omega_{\hat{\rho}}$ because the linear state-space model of Equation (6) is not able to capture the nonlinearities of the CSTR in this region. Therefore, the LMPC using RNN model outperforms the one using state-space model in terms of faster convergence speed and improved closed-loop stability. Figure 6 depicts the manipulated input profiles in deviation from the steady-state values, where the dashed horizontal lines are the upper and lower bounds for the manipulated inputs. It is shown that the input constraints are met for all times under the LMPC of Equation (1) using all three models.

4.4 | LMPC with RNNs using a lower amount of data

The above simulation results demonstrate that the RNN model based on the large data set that covers the entire operating region

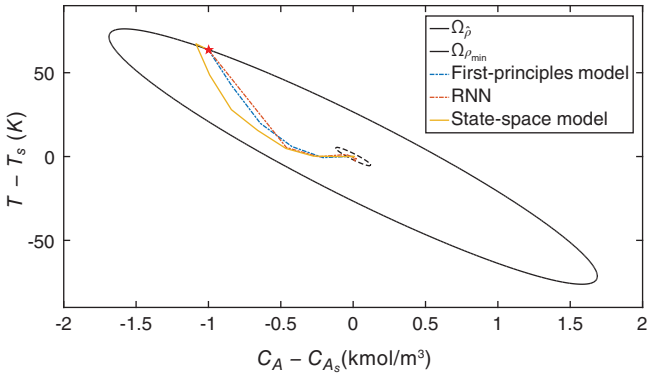

FIGURE 4 The state-space profiles for the closed-loop CSTR under the LMPC using the following models: the first-principles model (blue trajectory), the RNN model (red trajectory), and the linear state-space model (yellow trajectory) for an initial condition (−1, 63.6). CSTR, continuous stirred tank reactor; LMPC, Lyapunov-based model predictive control; RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

achieves the desired approximation performance. However, considering the case of limited data, the following simulations are performed to demonstrate that the RNN model still gives a good approximation of the best model that can be obtained from the available data. Specifically, we first investigate the case where data are only available for a portion of operating region (Figure 7). It is shown in Figure 8 that starting from initial conditions in the missing data area (the white region), the closed-loop state is still driven to the origin under the LMPC with the RNN model that is trained using the data only in the blue region. This implies that the RNN model also captures the process dynamics in the missing data area and is representative within the entire operating region. Additionally, we demonstrate that the RNN model works in the case of a small data set with much less data points than the large data set used in the previous simulations (i.e., the data set that covers the entire operating region $\Omega_{\hat{\rho}}$). Specifically, we uniformly pick 100 initial conditions within the operating region (Figure 9) and run open-loop simulations to generate the data set. The simulation result in Figure 10 shows that the closed-loop state can be driven to the origin under LMPC. However, compared to the simulation results derived from large data set-based RNNs, the state trajectories show larger oscillation since the RNN model is not able to fully capture process dynamics using a small data set. Additionally, by calculating the integral of LMPC cost function $\int_{t=0}^{t_p} L(x(\tau), u(\tau))d\tau$ over the simulation period t_p = 0.03 hr, it is obtained that for the initial condition (1.1, −65) in Figure 10, L = 18.56 under a small data set-based RNN, and L = 4.21 under a large data set-based RNN, respectively; for the other initial condition (−1.1, 60), L = 13.5 and L = 5.7 for the RNN under a small data set, and a large data set, respectively. Therefore, the closed-loop performance under a large data set-based RNN outperforms the one under a small data set-based RNN in terms of less energy and faster convergence to the origin for both initial conditions.

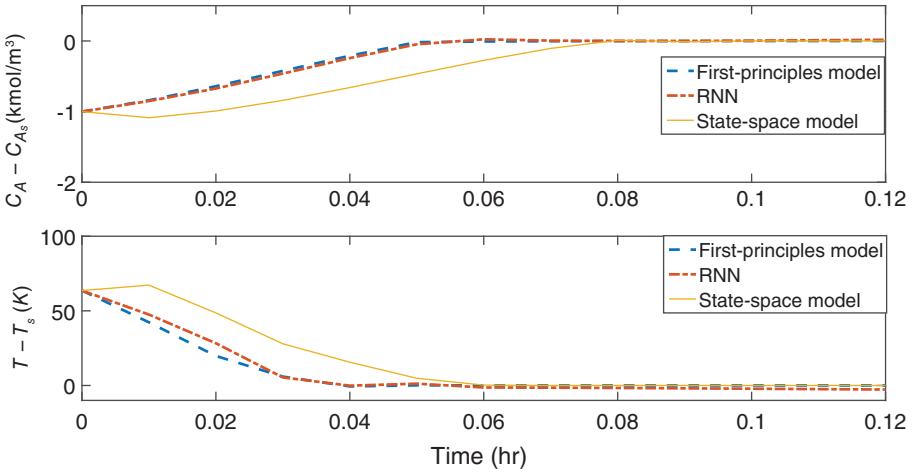

FIGURE 5 The state profiles ($x_1 = C_A - C_{As}$ and $x_2 = T - T_s$) for the initial condition (−1, 63.6) under the LMPC using the following models: the first-principles model (blue trajectory), the RNN model (red trajectory), and the linear state-space model (yellow trajectory). LMPC, Lyapunov-based model predictive control; RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

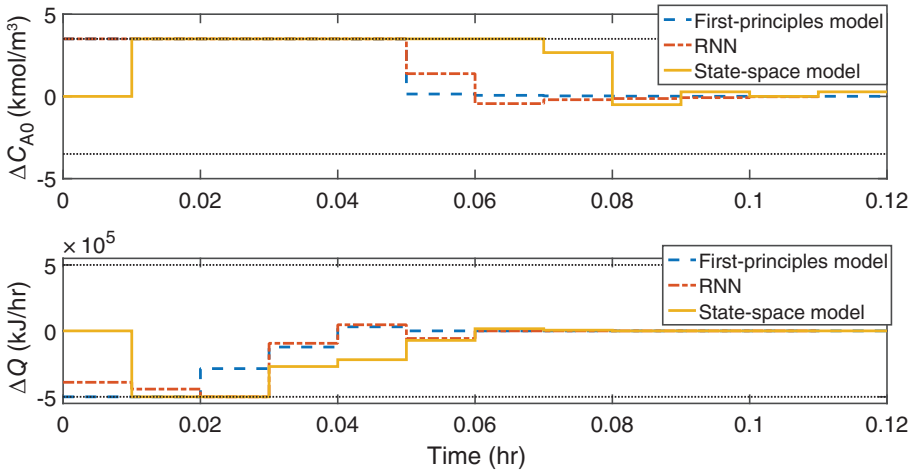

FIGURE 6 Manipulated input profiles ($u_1 = \Delta C_{A0}$ and $u_2 = \Delta Q$) for the initial condition (−1, 63.6) under the LMPC using the following models: the first-principles model (blue trajectory), the RNN model (red trajectory), and the linear state-space model (yellow trajectory), where the black dotted lines represent the upper and lower bound for u_1 and u_2, respectively. LMPC, Lyapunov-based model predictive control; RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

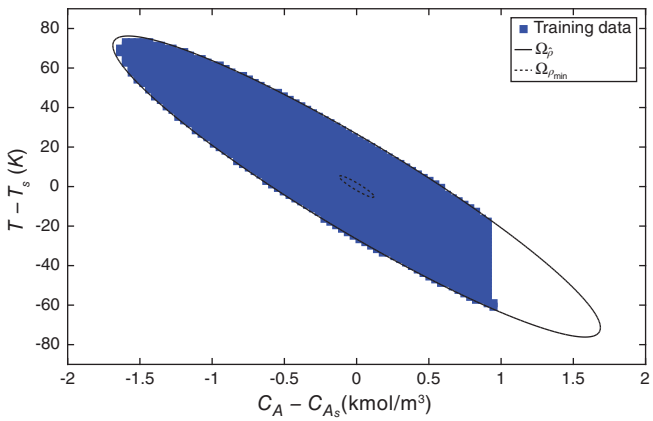

FIGURE 7 The set of initial conditions $x_0 \in \Omega_{\hat{\rho}}$ (marked as blue points) in which training data are collected for RNNs. RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

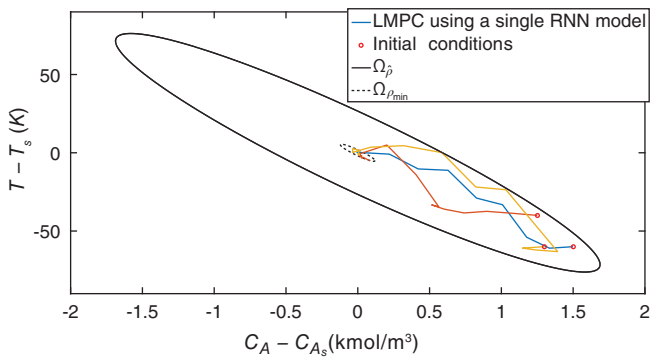

FIGURE 8 State trajectories for the closed-loop simulation using the RNN model that is obtained using a portion of data (i.e., the blue area in Figure 7) in the closed-loop stability region $\Omega_{\hat{\rho}}$. LMPC, Lyapunov-based model predictive control; RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

4.5 | Parallel computation of ensemble regression models

So far, we have demonstrated that the LMPC with a single RNN model is able to drive the closed-loop state to $\Omega_{\rho\text{min}}$, and compared the closed-loop performance of the system of Equation (4) under the LMPC with the RNN models based on a large data set, and a data set with a lower amount of data, respectively. In this section, we apply the LMPC using ensemble regression models to the CSTR of Equation (4) and perform parallel computing to improve computational efficiency. Since it is common that the RNN model may not perform perfectly for some initial conditions due to insufficient data, the

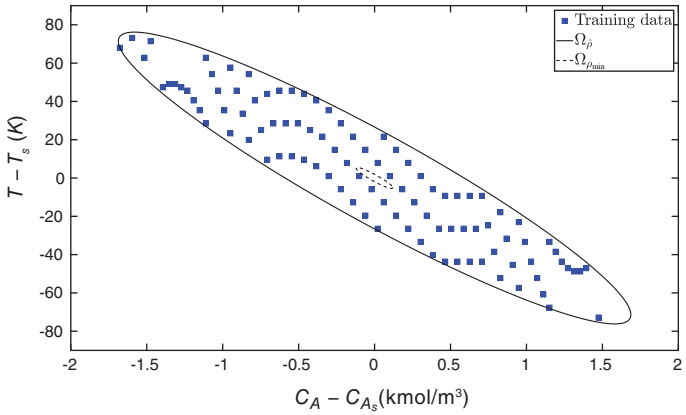

FIGURE 9 The set of initial conditions $x_0 \in \Omega_{\hat{\rho}}$ (marked as blue points) in which training data are collected for RNNs. RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

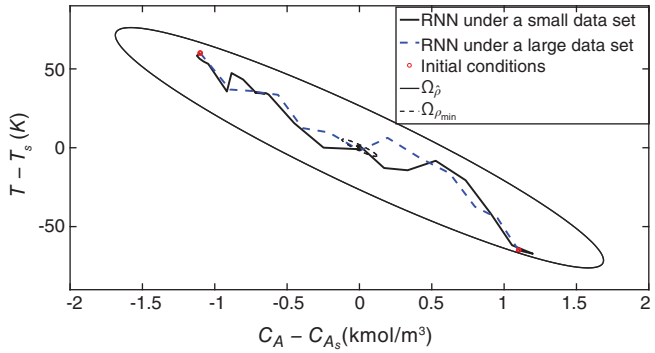

FIGURE 10 Closed-loop state trajectories using the RNN model that is obtained using a small data set (i.e., the blue points in Figure 9), and a large data set that covers the entire operating region $\Omega_{\hat{\rho}}$, respectively, for the initial conditions (1.1, −65) and (−1.1, 60). RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

utilization of ensemble regression models may improve the overall performance of RNN models within the entire operating region.

Figure 11 shows the state trajectories under the LMPC using various numbers of regression models (i.e., N_e = 1, …, 8). It is observed that starting from the initial condition (−1, 63.6), the closed-loop system of Equation (4) does not converge to the origin smoothly using a single RNN model that is trained poorly around the origin. Additionally, it is shown that as the number of regression models used in LMPC increases, the closed-loop performance is improved in terms of less oscillation and faster convergence. Therefore, in this case, the optimal number of regression models is determined to be five as no further improvement is noticed for the increase of regression models being used.

However, as more regression models are utilized in the LMPC of Equation (1), the computation time under serial operation increases significantly, which makes it challenging for the controller to be implemented in practice. Therefore, to address the computational efficiency issue, we run the LMPC of Equation (1) in the parallel mode. Specifically, a MPI for the Python programming language, named *MPI4Py*,[20] is incorporated in the program of the LMPC optimization problem to break the prediction models of Equation (1b) into five

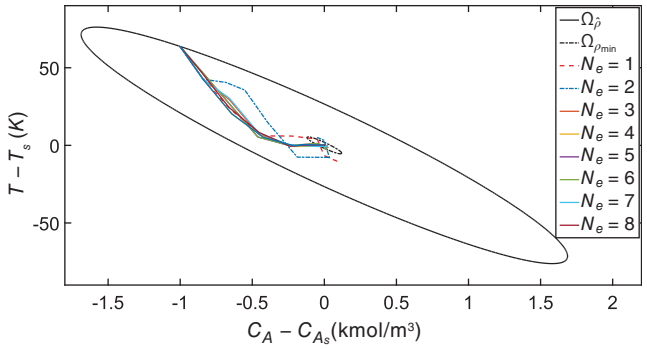

FIGURE 11 The state-space profiles for the closed-loop CSTR under the LMPC using the following models: the first-principles model (blue trajectory), the RNN model (red trajectory), and the linear state-space model (yellow trajectory) for an initial condition (−1, 63.6). CSTR, continuous stirred tank reactor; LMPC, Lyapunov-based model predictive control; RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 2 Computation time for solving the LMPC using different models

Models	Computation time (s)
First-principles model	<1
Linear state-space model	<0.1
Single RNN model	~8
Ensemble regression models in serial mode	>50
Ensemble regression models in parallel mode	~11

Abbreviations: LMPC, Lyapunov-based model predictive control; RNN, recurrent neural network.

independent computing processes. Additionally, since the main program of the LMPC optimization problem is executed on the host node only, we use a *while-loop* and synchronization mechanism to ensure that all worker nodes work with the host node simultaneously throughout the optimization process. The parallel computing of the LMPC optimization problem of Equation (1) is solved on the UCLA Hoffman2 Distributed Cluster.

The averaged computation time for solving the LMPC optimization problem per sampling step using the first-principles model of Equation (4), the linear state-space model of Equation (6), a single RNN model, five ensemble regression models in serial mode, and five ensemble regression models in parallel mode are reported in Table 2. In Table 2, it is shown that the LMPC optimization problem using state-space model is solved with the shortest computation time. The optimization problem of LMPC using RNN models is time consuming compared to the state-space model or the first-principles model due to the large number of internal states, the essential normalization and data reshaping steps, and the communication between host and worker nodes. However, it is shown that under parallel operation, the computation time for solving the LMPC optimization problem using five ensemble regression models at each sampling step is around 11 s, which is significantly reduced (approximately five times less than the serial computing), and becomes less than the sampling period

(i.e., Δ = 0.01 hr = 36 s). This implies that the LMPC using an ensemble of RNN models can be implemented in real time if parallel computing is employed. Additionally, the computation time for solving LMPC under RNN models may be further reduced if TensorFlow is employed, which is more computationally efficient than Keras.

4.6 | RNN model performance evaluation

To illustrate the advantages of the ensemble of RNN models, in this section, we characterize the region of initial conditions $x_0 \in \Omega_{\hat{\rho}}$ for which the performance of the LMPC using the data-driven model (i.e., the state-space model of Equation 6 and the ensemble of RNN models, respectively) is close to that of the LMPC using the first-principles model of Equation (4). Specifically, extensive closed-loop simulations that sweep over all the initial conditions x_0 in the closed-loop stability region $\Omega_{\hat{\rho}}$ are conducted under the LMPC of Equation (1) using the following models: the first-principles model of Equation (4), the ensemble of RNN models and the state-space model of Equation (6). It should be mentioned that methods that may improve the performance of linear state-space model, for example, the ensemble of linear state-space models and multiple linear state-space models for different portions of the closed-loop stability region $\Omega_{\hat{\rho}}$,[21] are not investigated in this work since the aim of this study is to develop a computationally efficient LMPC scheme using an ensemble of RNN models.

All the closed-loop simulations are run with a fixed time duration that is sufficiently long for the closed-loop state to converge to $\Omega_{\rho\min}$ for any initial condition $x_0 \in \Omega_{\hat{\rho}}$. Extensive closed-loop simulations demonstrate that the LMPC using the ensemble of RNN models and the LMPC using the state-space model of Equation (6) both drive the closed-loop state to $\Omega_{\rho\min}$ for any initial condition $x_0 \in \Omega_{\hat{\rho}}$. Therefore, to compare the performance of closed-loop system under different data-driven models, a performance index S is introduced to calculate the relative error between the closed-loop states under the data-driven model and the first-principles model as follows:

$$S = \frac{\sum_{i=1}^{L} |\hat{V}(x_i^d) - \hat{V}(x_i^f)|}{\sum_{i=1}^{L} \hat{V}(x_i^f)} \tag{9}$$

where L is number of sampling steps in simulation, x_i^f represents the ith closed-loop state for the first-principles model of Equation (4), and x_i^d represents the ith closed-loop state for the data-driven models, which are the ensemble of RNN models and the linear state-space model of Equation (6), respectively. Since the value of $\hat{V}(x)$ decreases as the state moves towards the origin under LMPC, the performance index S of Equation (9) indicates the closeness of the convergence speed of closed-loop states between the LMPC using the data-driven model and the LMPC using the first-principles model.

By setting the threshold S_{TH} of the performance index to be 0.65, the region of initial conditions for which the performance of the ensemble of RNN models is close to that of the first-principles model (i.e., $S \leq S_{TH}$) covers the entire closed-loop stability region, while the corresponding region for the linear state-space model is characterized as the blue region in Figure 12. It is shown in Figure 12 that the closed-loop performance of the CSTR of Equation (4) under the LMPC using the state-space model of Equation (6) is undesired in the top and bottom of the closed-loop stability region due to poor approximation of nonlinearities in these regions. Therefore, based on the performance index of Equation (9), the overall closed-loop performance of the ensemble of RNN models within the closed-loop stability region $\Omega_{\hat{\rho}}$ outperforms that of the state-space model in terms of the rate of convergence to the origin and the closeness to the closed-loop performance under the LMPC using the first-principles model.

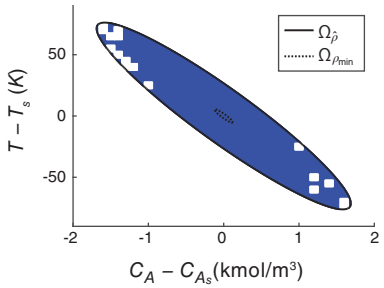

FIGURE 12 The set of initial conditions $x_0 \in \Omega_{\hat{\rho}}$ (marked as blue points) in which the closed-loop CSTR under the LMPC using the linear state-space of Equation (6) behaves similarly to the LMPC using the first-principles model of Equation (4) (i.e., $S \leq S_{TH}$ in the blue region and $S > S_{TH}$ in the white regions). CSTR, continuous stirred tank reactor; LMPC, Lyapunov-based model predictive control [Color figure can be viewed at wileyonlinelibrary.com]

5 | CONCLUSION

This work presented the computational implementation of the LMPC using an ensemble of RNN models to a chemical process. Specifically, computational implementation issues of RNN models involving long prediction horizon and approximation of continuous RNN models via numerical methods were first discussed. Subsequently, parallel computing was employed to reduce the computation time in both training multiple RNN models and predicting future results via an ensemble of RNN models in LMPC. Simulation results of the application of the proposed LMPC using an ensemble of RNN models to a chemical process example demonstrated that closed-loop stability was achieved for the nonlinear system, and the overall closed-loop performance of RNN models outperformed that of a linear state-space model. Additionally, computational efficiency was significantly enhanced under parallel computation of the ensemble of RNN models in LMPC.

ACKNOWLEDGMENT

The authors gratefully acknowledged the financial support from the National Science Foundation and the Department of Energy.

ORCID

Panagiotis D. Christofides https://orcid.org/0000-0002-8772-4348

REFERENCES

1. Pan Y, Wang J. Nonlinear model predictive control using a recurrent neural network. Paper presented at: Proceedings of the IEEE International Joint Conference on Neural Networks; Hong Kong, China; 2008:2296-2301.
2. Pan Y, Wang J. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Trans Ind Electr*. 2011;59:3089-3101.
3. Xu J, Li C, He X, Huang T. Recurrent neural network for solving model predictive control problem in application of four-tank benchmark. *Neurocomputing*. 2016;190:172-178.
4. Pethick M, Liddle M, Werstein P, Huang Z. Parallelization of a backpropagation neural network on a cluster computer. Paper presented at: Proceedings of the International Conference on Parallel and Distributed Computing and Systems; Marina del Rey, California; 2003:574-582.
5. Scardapane S, Di Lorenzo P. A framework for parallel and distributed training of neural networks. *Neural Netw*. 2017;91:42-54.
6. Vesel K, Burget L, Grézl F. Parallel training of neural networks for speech recognition. Paper presented at: Proceedings of the International Conference on Text, Speech and Dialogue; Brno, Czech Republic; 2010:439-446.
7. Gu R, Shen F, Huang Y. A parallel computing platform for training large scale neural networks. Paper presented at: Proceedings of the IEEE International Conference on Big Data; Santa Clara, California; 2013:376-384.
8. Sewell M. Ensemble learning. Technical report, UCL, London; 2010.
9. Evensen G. The ensemble Kalman filter: theoretical formulation and practical implementation. *Ocean Dynam*. 2003;53:343-367.
10. Evensen G. The ensemble Kalman filter for combined state and parameter estimation. *IEEE Control Syst Mag*. 2009;29:83-104.
11. Prakash J, Patwardhan SC, Shah SL. Constrained nonlinear state estimation using ensemble Kalman filters. *Ind Eng Chem Res*. 2010;49:2242-2253.
12. Puranik Y, Bavdekar VA, Patwardhan SC, Shah SL. An ensemble Kalman filter for systems governed by differential algebraic equations (DAEs). *IFAC Proc*. 2012;45:531-536.
13. Zhang C, Ma Y. *Ensemble Machine Learning: Methods and Applications*. New York: Springer; 2012.
14. Bishop CM. *Pattern Recognition and Machine Learning*. New York: Springer; 2006.
15. Hoeting JA, Madigan D, Raftery AE, Volinsky CT. Bayesian model averaging: a tutorial. *Stat Sci*. 1999;14:382-401.
16. Lin Y, Sontag ED. A universal formula for stabilization with bounded controls. *Syst Control Lett*. 1991;16:393-397.
17. Almasi GS, Gottlieb A. *Highly Parallel Computing*. New York: Benjamin/Cummings; 1988.
18. Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math Program*. 2006;106:25-57.
19. Kheradmandi M, Mhaskar P. Data driven economic model predictive control. *Mathematics*. 2018;6:51.
20. Dalcin LD, Paz RR, Kler PA, Cosimo A. Parallel distributed computing using python. *Adv Water Resour*. 2011;34:1124-1139.
21. Alanqar A, Ellis M, Christofides PD. Economic model predictive control of nonlinear process systems using empirical models. *AIChE J*. 2015;61:816-830.

How to cite this article: Wu Z, Tran A, Rincon D, Christofides PD. Machine-learning-based predictive control of nonlinear processes. Part II: Computational implementation. *AIChE J*. 2019;65:e16734. https://doi.org/10.1002/aic.16734