Journal of Process Control 89 (2020) 74–84

Contents lists available at ScienceDirect

Journal of Process Control

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

Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes

Zhe Wu [a], David Rincon [a], Panagiotis D. Christofides [a,b,*]

[a] Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095, United States
[b] Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095, United States

ARTICLE INFO

Article history:
Received 29 November 2019
Revised 10 February 2020
Accepted 20 March 2020

Keywords:
Recurrent neural networks
Model predictive control
Structural process knowledge
Nonlinear systems
Chemical processes

ABSTRACT

In this work, physics-based recurrent neural network (RNN) modeling approaches are proposed for a general class of nonlinear dynamic process systems to improve prediction accuracy by incorporating a priori process knowledge. Specifically, a hybrid modeling method is first introduced to integrate first-principles models and RNN models. Subsequently, a partially-connected RNN modeling method that designs the RNN structure based on a priori structural process knowledge, and a weight-constrained RNN modeling method that employs weight constraints in the optimization problem of the RNN training process are developed. The proposed physics-based RNN models are utilized in model predictive controllers and applied to a chemical process network example to demonstrate their improved approximation performance compared to the fully-connected RNN model that is developed as a black box model.

1. Introduction

Recurrent neural networks (RNN), a class of artificial neural networks that can represent temporal dynamic behavior through feedback loops in neurons, have been utilized to model nonlinear dynamic systems and have been incorporated in the design of model predictive controllers (MPC) that optimize process performance based on RNN prediction results [23,24]. For example, in Wu et al. [24,25] an ensemble of RNN models were developed for a continuous stirred tank reactor and utilized in the RNN-based MPC to operate the system at its steady-state while maintaining closed-loop state in a stability region.

However, as neural network modeling is generally treated as a black-box modeling approach where no physical knowledge is utilized, interpretability and optimality of neural network modeling remain questionable. On the other hand, chemical processes have been studied for a long time by researchers and engineers, where first-principles knowledge has been obtained based on their predefined and well-known structure. For example, a chemical plant is designed in a sequence of intricate operation units that perform reactions, separations, among many others operations in which raw materials are fed in the first unit and products are obtained in the last unit in its simplest structure. Additionally, it is also very common that some processes are highly coupled among units through reflux of unreacted material that is recycled to upstream units to maximize the production [12,20]. However, at this stage, the incorporation of first-principles or physical knowledge of chemical processes into RNN modeling has not been thoroughly studied.

Fully-connected neural networks are developed based on the assumption that all the inputs affect all the neural network neurons, followed by all the outputs. However, it is noted that in realistic chemical processes, it is common that only a portion of inputs affect a portion of outputs, for example, in a multiple unit process in which upstream units affect downstream units but not in the opposite direction. In order to make better use of such a priori process knowledge, many researchers have started to incorporate physical knowledge of systems in the neural network formulation (e.g., [1,6,7,10,11,15]). For example, hybrid models and gray-box models have been developed to introduce chemical process knowledge into data-driven modeling in early works [2–5,14,18,19,26,29]. Recently, a neural network has been specialized by including partial physical knowledge in its structure in Lu et al. [11]. In this paper, the nodes of the first layer represent the variables with physical meaning and the connection with the inputs are based on the impact between them. It was demonstrated that the resulting neural networks were able to improve the performance when compared with a fully-connected network.

Motivated by the above considerations, in this work, we propose a hybrid model, a partially-connected RNN model, and a weight-constrained RNN model to incorporate process physical

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

https://doi.org/10.1016/j.jprocont.2020.03.013
0959-1524/

knowledge into RNN modeling and training. Subsequently, the proposed partially-connected RNN model and the weight-constrained RNN model are incorporated in the design of MPC and of economic MPC (EMPC) to provide predictions of future states for the optimization problem of MPC and EMPC that optimize process performance in terms of closed-loop stability and economic optimality. Finally, the RNN-MPC and RNN-EMPC are applied to a chemical process example to demonstrate their improved closed-loop performances in terms of faster convergence to the steady-state under RNN-MPC and enhanced process economic profits under RNN-EMPC than the controllers using a fully-connected RNN model.

2. Preliminaries

2.1. Notation

The Euclidean norm of a vector is denoted by the operator $|\cdot|$ and the weighted Euclidean norm of a vector is denoted by the operator $|\cdot|_Q$ where Q is a positive definite matrix. x^T denotes the transpose of x. The notation $L_f V(x)$ denotes the standard Lie derivative $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$. Set subtraction is denoted by "\", i.e., $A \backslash B := \{x \in \mathbf{R}^n \,|\, x \in A,\ x \notin B\}$.

2.2. Class of systems

The class of continuous-time nonlinear systems considered is described by the following state-space form:

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

where $x \in \mathbf{R}^n$ is the state vector, $u \in \mathbf{R}^m$ is the manipulated input vector, and $w \in \mathbf{W}$ is the disturbance vector, where $\mathbf{W} := \{w \in \mathbf{R}^l \,|\, |w| \le \theta, \theta \ge 0\}$. The control action constraint is defined by $u \in U := \{u_{\min} \le u \le u_{\max}\} \subset \mathbf{R}^m$, where $u_{\min}$ and $u_{\max}$ represent the minimum and the maximum value vectors of inputs allowed, respectively. $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$, $n \times m$, and $n \times l$, respectively. Without loss of generality, the initial time t_0 is taken to be zero ($t_0 = 0$), and it is assumed that $f(0) = 0$, and thus, the origin is a steady-state of the system of Eq. (1) with $u(t) = w(t) \equiv 0$.

2.3. Stabilizability assumptions expressed via lyapunov-based control

We assume that there exists a positive definite and proper Control Lyapunov function (CLF) V for the nominal system of Eq. (1) with $w(t) \equiv 0$ that satisfies the small control property (i.e., for every $\varepsilon > 0$, $\exists\, \delta > 0$, s.t. $\forall\, x \in \mathcal{B}_\delta(0)$, there exists u that satisfies $|u| < \varepsilon$ and $L_f V(x) + L_g V(x) u < 0$, Sontag [17]) and the following condition:

$$L_f V(x) < 0, \forall x \in \{z \in \mathbf{R}^n \backslash \{0\} \mid L_g V(z) = 0\} \tag{2}$$

The CLF assumption implies that there exists a stabilizing feedback control law $\Phi(x) \in U$ for the nominal system of Eq. (1) (i.e., $w(t) \equiv 0$) that renders the origin of the closed-loop system exponentially stable for all x in a neighborhood of the origin in the sense that $L_f V(x) + L_g V(x) u < 0$ holds for $u = \Phi(x) \in U$. An example of a feedback control law can be found in Lin and Sontag [9]. Based on the CLF assumption, we can first characterize a region where the time-derivative of V is rendered negative definite under the controller $\Phi(x) \in U$ as $\phi_u = \{x \in \mathbf{R}^n \mid \dot{V}(x) = L_f V + L_g V u < -kV(x), u = \Phi(x) \in U\} \cup \{0\}$, where k is a positive real number. Then, the closed-loop stability region Ω_ρ for the nonlinear system of Eq. (1) is defined as a level set of the Lyapunov function embedded in ϕ_u: $\Omega_\rho := \{x \in \phi_u \,|\, V(x) \le \rho\} \subset \phi_u$, where $\rho > 0$.

Remark 1. We consider the nonlinear system with the form of Eq. (1) since control-affine nonlinear systems are very common in the modeling of chemical processes. Additionally, with the form of Eq. (1), we can simplify the discussion on the design of a stabilizing controller $u = \Phi(x)$ by using the Sontag control law [9]. However, it should be noted that the proposed RNN modeling approaches that account for a priori process knowledge in this manuscript are not restricted to control-affine nonlinear systems, and can be generalized to nonlinear systems in a more general form: $\dot{x} = f(x,u,w)$.

2.4. Recurrent neural network model

A recurrent neural network (RNN) model that approximates the nonlinear dynamics of the system of Eq. (1) is developed with the following form:

$$\dot{\hat{x}} = F_{nn}(\hat{x},u) := A\hat{x} + \Theta^T y \tag{3}$$

where $\hat{x} \in \mathbf{R}^n$ is the RNN state vector and $u \in \mathbf{R}^m$ is the manipulated input vector. $y = [y_1, \ldots, y_n, y_{n+1}, \ldots, y_{m+n}] = [\sigma(\hat{x}_1), \ldots, \sigma(\hat{x}_n), u_1, \ldots, u_m] \in \mathbf{R}^{n+m}$ is a vector of both the network state $\hat{x}$ and the input u, where $\sigma(\cdot)$ is the nonlinear activation function (e.g., a sigmoid function $\sigma(x) = 1/(1+e^{-x})$). A is a diagonal coefficient matrix, i.e., $A = diag\{-a_1, \ldots, -a_n\} \in \mathbf{R}^{n \times n}$, and $\Theta = [\theta_1, \ldots, \theta_n] \in \mathbf{R}^{(m+n) \times n}$ with $\theta_i = b_i[w_{i1}, \ldots, w_{i(m+n)}]$, $i = 1, \ldots, n$. a_i and b_i are constants. w_{ij} is the weight connecting the jth input to the ith neuron where $i = 1, \ldots, n$ and $j = 1, \ldots, (m+n)$. a_i is assumed to be positive such that each state $\hat{x}_i$ is bounded-input bounded-state stable. It is noted that to simplify the mathematical expressions of the input vector, and of the weight matrix in this manuscript, we do not include the bias term in the notation since it can always be considered as an additional constant input (i.e., $u \in \mathbf{R}^{m+1}$), and therefore, does not affect the formulation of the continuous RNN models.

The development of RNN models for the nonlinear system of Eq. (1) follows a three-step procedure: (1) a large dataset consisting of state trajectories from various initial conditions and control actions in an operating region considered is developed from extensive open-loop simulations of the nonlinear system of Eq. (1), (2) a source platform for machine learning (e.g., Tensorflow, Keras, Caffe) is utilized to solve the optimization problem of neural networks to obtain the optimal weights (i.e., the coefficient matrices A and Θ), under which the loss function that indicates the difference between predicted states and actual states achieves its minimum value, and (3) to validate the quality of RNN models, open-loop predictions will be performed on a test dataset that has not been utilized in the training process and will be compared with the actual state trajectories from the nonlinear system of Eq. (1). Following the above procedure, a well-conditioned RNN model is obtained to represent the nonlinear dynamics of the system of Eq. (1) in the operating region, and thus, can be used in closed-loop simulations by incorporating it in the model-based predictive controllers, e.g., [24,25].

Although the universal approximation theorem [8,16] states that a neural network with a single hidden layer with sufficient number of neurons can approximate any continuous-time nonlinear function on compact subsets of $\mathbf{R}^n$, algorithmic learnability of the optimal neural network weights is not guaranteed. In fact, due to the complexity of neural network structure, availability of computing power, and feasibility of optimization algorithms, it is challenging to find such an optimal weight for the regression problems of a large-scale, complex system. Therefore, how to improve the performance of neural networks has been a major long-standing challenge for researchers in machine learning community over the past few decades, where a lot of efforts have been made to optimize neural network structure, develop advanced optimization algorithms, improve data-processing systems and so on.

In this work, we will improve the performance of RNN models in terms of enhanced prediction accuracy by incorporating structural domain knowledge of the nonlinear system of Eq. (1) (i.e., knowledge of the dependence of the state variables) into the development of the RNN structure. Specifically, instead of treating the RNN system of Eq. (3) like a black box and training it using all the inputs and outputs available (termed the fully-connected model throughout the manuscript), we modify the RNN structure according to the structural process knowledge of the nonlinear system of Eq. (1). The details of the proposed new structure are discussed in the following section.

Remark 2. It is noted that in Eq. (3), we use a one-hidden-layer RNN model with n states in order to simplify the discussion of the approximation of the nonlinear system of Eq. (1) using an RNN model. However, the RNN modeling method in this section is not restricted to a one-hidden layer RNN structure with n states only. The RNN states $\hat{x} \in \mathbf{R}^n$ in Eq. (3) can be considered to be the last hidden layer (if the output of the nonlinear system of Eq. (1) is a function of states), or the output layer of an RNN (if the state x is also the output of the nonlinear system of Eq. (1)). Therefore, before the last hidden layer/output layer, we can add another hidden layer or multiple hidden layers with a sufficient number of neurons to approximate the nonlinear system of Eq. (1).

3. Physics-based RNNs

In this section, we introduce three different methods to integrate domain knowledge into neural network modeling and training. The first method is to develop a hybrid model that integrates first-principles models with RNN models. The second method is to develop a partially-connected RNN structure using a priori knowledge of process input-output relationship. Lastly, a weight-constrained RNN model is developed by imposing constraints on the neural network weights based on the input-output relationship of the nonlinear system of Eq. (1).

3.1. Hybrid model

While first-principles modeling has been studied and applied to chemical processes for over a century and has achieved good performances, it becomes difficult to obtain a 100% accurate first-principles model for large-scale systems due to inherent complexity. Therefore, in this work, we first propose a hybrid modeling method that introduces physical knowledge (e.g., first-principles knowledge based on physical laws such as mass and energy balances) into neural network modeling by combining a first-principles model and an RNN model together. Specifically, the hybrid model is developed using an RNN function $\tilde{f}_{nn}(x,u)$ to approximate the gap between the first-principles model and the actual nonlinear process as follows:

$$\dot{x} = \tilde{f}(x) + \tilde{g}(x)u + \tilde{f}_{nn}(x,u) \tag{4}$$

where $\dot{x} = \tilde{f}(x) + \tilde{g}(x)u$ is the first-principles model that is developed based on general physical laws and assumptions, and therefore, may not be able to fully capture the dynamics of the actual nonlinear processes of Eq. (1) due to mismatch between $\dot{x} = \tilde{f}(x) + \tilde{g}(x)u$ and $\dot{x} = f(x) + g(x)u$. The RNN function $\tilde{f}_{nn}(x,u)$ in Eq. (4) is utilized to bridge the gap between the first-principles knowledge and the real process data. It is demonstrated that the hybrid model of Eq. (4) has the following advantages compared with a fully-connected RNN model. First, the RNN in the hybrid model is only used to approximate the residual between first-principles models and real process data, and therefore, may take less computing power and training time to learn. Additionally, when it comes to the operating region with no data available, the hybrid model can still be considered a reliable model due to its intrinsically physical knowledge, while the pure RNN model may be completely dysfunctional. For example, in Ba and Kadambi [1], a hybrid model that combines first-principles free-falling equations and a neural network model was developed to improve the estimation of future trajectories of a paper ball being tossed, in which the neural network was developed to learn the model mismatch between the ground truth and the first-principles model-based solution. Additionally, in Zhang et al. [28], a hybrid neural network model was developed for a chemical process where the linear part of the hybrid model is developed based on first-principles knowledge and the nonlinear term of reaction rate is provided by a neural network model using experiment/simulation data. It was demonstrated in Zhang et al. [28] that the hybrid model achieved desired approximation performance and the neural network well approximated the nonlinear term of reaction rate that depends on multiple variables with an unknown reaction mechanism.

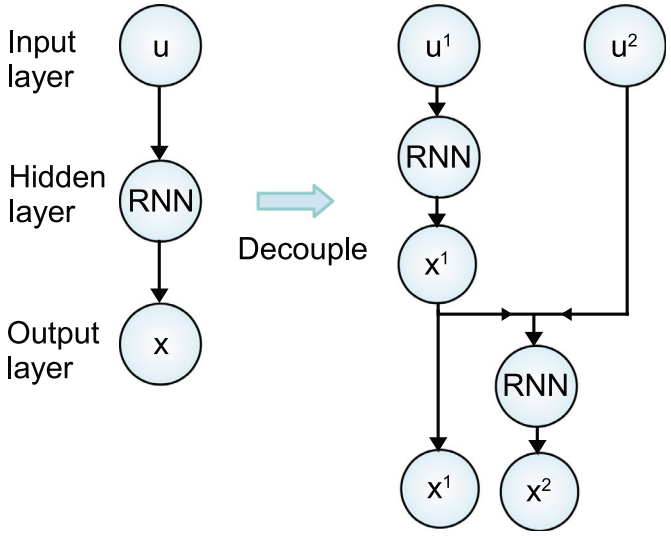

Fig. 1. A partially-connected recurrent neural network structure based on process structural knowledge, where $u = [u^1,\ u^2]$ and $x = [x^1,\ x^2]$.

3.2. Partially-connected RNN

In industrial chemical processes, the unit operations in the upstream stage of the production process affect those in the downstream stage, while the impact is ignorable in the opposite direction. This connection between upstream and downstream stages is often reflected in the first-principles model (if there is any), and is barely incorporated in the development of a data-driven model for the entire process due to the difficulty of designing model structures. In particular, since it is not clear how to derive optimal architectures for process data without any a priori knowledge, fully-connected RNN networks are often state-of-the-art for large-scale, complex systems. Specifically, a black box NN model that takes all available inputs to predict the outputs of interest is preferred in developing a dynamic process model for the integrated upstream and downstream processes as it is easy to implement using open-source machine learning software and is able to account for all possible input-output relationships. As shown in Fig. 1, the RNN model on the left represents a general structure of a fully-connected RNN model with an input layer, a hidden layer consisting of recurrent neurons, and an output layer, for which the training process follows the discussion of the three-step procedure in the previous section.

To account for the structural process knowledge into RNN modeling of the nonlinear system of Eq. (1), we develop a partially-

connected RNN structure as shown on the right of Fig. 1. Specifically, we consider the nonlinear system of Eq. (1) under the assumption that the state vector x^1 is affected by u^1 only, and x^2 is affected by both u^1 and u^2, where $x = [x^1,\ x^2] \in \mathbf{R}^n$ and $u = [u^1 \in \mathbf{R}^{m_1},\ u^2 \in \mathbf{R}^{m_2}] \in \mathbf{R}^m$, $m_1 + m_2 = m$. It is shown in Fig. 1 that in the partially-connected RNN, u^1 only affects x^1, and both u^1 and u^2 have an impact on the output u^2. By partitioning the RNN modeling problem into two blocks (i.e., the data flows from u^1 to x^1, and from u^1, u^2 to x^1) that are corresponding to the structural a priori knowledge of the nonlinear system of Eq. (1), it is demonstrated that the hidden layers (i.e., the RNN layers) in the partially-connected RNN model are analogous to the unknown nonlinear functions of the system of Eq. (1) under the above assumption on input-output relationship.

By modifying the RNN structure to explicitly exclude the connection between u^2 and x^1, a priori knowledge on process structure is infused into RNN modeling of the nonlinear system of Eq. (1), and therefore, an improved approximation performance can be derived. For example, due to the superiority of encoding priors into structure designs, in Remark 5, it is demonstrated that the number of hidden neurons and weight parameters could be significantly reduced to achieve the desired performance as good as the fully-connected model. Additionally, the partially-connected RNN model may need less training data to obtain a well-conditioned model since priors are acting to reveal the correct direction for RNN to converge to an optimal solution. Moreover, the partially-connected model may outperform the fully-connected model in the regime with no training data available since the model structure is consistent with the actual process state variable relationship of the nonlinear system of Eq. (1).

Remark 3. It is noted that the partially-connected RNN model can achieve better approximation performance not only in regimes where data are not available, but also in the regime in which training and validation data are available. While in general the approximation performance of NN model on training dataset will be improved by increasing the number of neurons and parameters, excessive number of neurons may lead to over-fitting, which means that the NN model can capture the input-output relationship well for training dataset but not for the validation/testing datasets. Additionally, as in this particular example of Fig. 1, the second input u^2 does not affect the relationship between u^1 and x^1 (i.e., the weights between u^2 and x^1 should be zero), the connection between u^2 and x^1 in a fully-connected RNN model will instead result in a negative impact on the training process in terms of longer time to converge to an optimal solution.

3.3. Weight-constrained RNN

Under the assumption that a portion of the input vector u^2 in the nonlinear system of Eq. (1) does not affect the output vector x^1, we develop an RNN model structure with constrained weight parameters representing the dynamic effects of process inputs u on the outputs x as shown in Fig. 2. Specifically, the weights connecting u^2 and x^1 (dashed gray lines in Fig. 2) are constrained in the RNN model such that the effects of u^2 on x^1 will be weakened during the training process. Based on the RNN model of Eq. (3), the output vector x^1 and the hidden neuron r_i, $i = 1, \ldots, h$ are derived as follows:

$$\dot{x}^1 = \sum_{i=1}^{h} w_i^{(2)} \dot{r}_i \tag{5}$$

$$\dot{r}_i = -a_i r_i + \theta_i y \tag{6}$$

where $\theta_i = b_i[w_{1i}^{(1)}, \ldots, w_{hi}^{(1)}, \ldots, w_{(h+m)i}^{(1)}]$ and $y = [\sigma(r_1), \ldots, \sigma(r_h), u^1, u^2]^T$. a_i, b_i are constants, $w_{ji}^{(1)}$ is the weight connecting the jth input, $j = 1, \ldots, h+m$ to ith neuron, $i = 1, \ldots, h$, and y is the input vector consisting of the hidden states r and the manipulated inputs u. $w^{(1)}$, $w^{(2)}$ represent the weight vectors before and after the hidden layer. Similarly, the bias term is not included in the notation since it can be considered as an additional constant input. Based on Eqs. (5) and (6), the following equation is derived to demonstrate the contribution of u^2 to $\dot{x}^1$:

$$\begin{aligned}\dot{x}^1 &= \sum_{i=1}^{h} w_i^{(2)}(-a_i r_i + \theta_i y) \\ &= \sum_{i=1}^{h} n_i(r, w) + w_i^{(2)} b_i([w_{(h+1)i}^{(1)}, \ldots, w_{(h+m_1)i}^{(1)}]u^1 \\ &\quad + [w_{(h+m_1+1)i}^{(1)}, \ldots, w_{(h+m)i}^{(1)}]u^2)\end{aligned} \tag{7}$$

where $n_i(\cdot, \cdot)$ is a nonlinear function of the neuron states r and weights w. Therefore, to reduce the impact of u^2 to $\dot{x}^1$, the weight product $\Pi_w = |w_i^{(2)} b_i[w_{(h+m_1+1)i}^{(1)}, \ldots, w_{(h+m)i}^{(1)}]|$ should be constrained by a sufficiently small bound, and this constraint will also be incorporated in the training of the RNN model with the above input-output relationship. It is noted that since the constraint is applied on the weight product Π_w for the weight-constrained RNN model in Fig. 2, a zero bound for the weight constraint could lead to disconnection of hidden neurons from inputs and outputs, and therefore, should be avoided in any case.

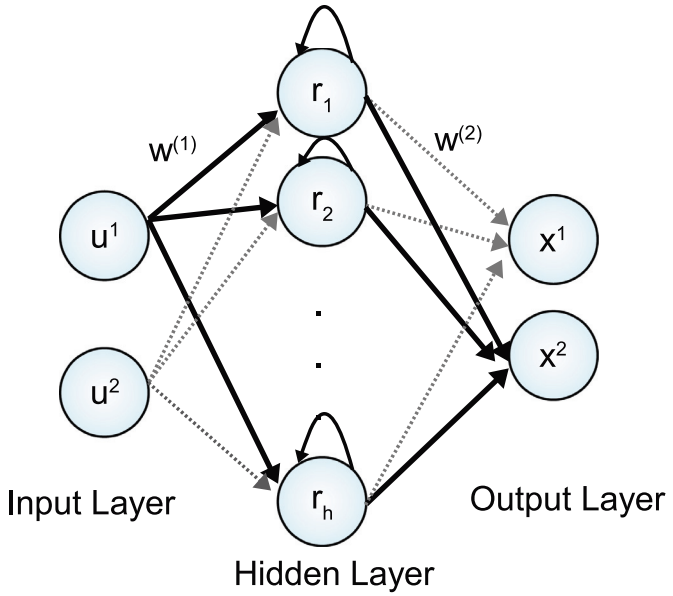

Fig. 2. A weight-constrained recurrent neural network structure, where $w^{(1)}$ and $w^{(2)}$ are the weights before and after the hidden layer, r_i, $i = 1, \ldots, h$ is the RNN hidden neuron, and the dashed gray lines denote the diminished connections between u^2 and x^1.

In addition to the weight constraints, penalty components on weight parameters can be employed in the loss function of the RNN optimization problem to introduce a priori weight knowledge into the training process. In general, regularization techniques (e.g., L1 and L2 regularization) are utilized in the training process of a neural network model to obtain a less complex model and avoid over-fitting in the presence of a large number of features in datasets. Therefore, to constrain the weight product Π_w in Eq. (7), the following loss function is developed:

$$L = \sum_{i=1}^{N_d} (x_i - \hat{x}_i)^2 + \lambda \Pi_w \tag{8}$$

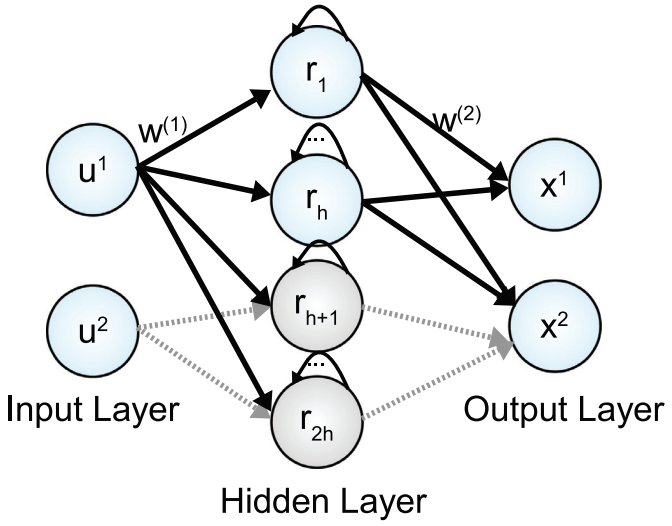

Fig. 3. A recurrent neural network structure, where the connection between u^2 and x^1 is fully removed from the blue neurons, and the connection between u^2 and x^2 is rebuilt using the gray neurons in the hidden layer. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where x_i and $\hat{x}_i$ are the actual and predicted outputs, respectively, N_d is the number of data samples in the dataset, and $\lambda > 0$ is the weight for the regularization term. It is noted that λ needs to be carefully chosen such that the regularization term can effectively decrease the values of the weights, but does not dominate the optimization problem of training a neural network model. Specifically, while a nonzero λ is required to penalize the regularization term, a large λ may render the optimization problem under-fitting due to the dominance of the regularization term in the loss function of Eq. (8). Therefore, we evaluate λ against any metric (e.g., mean-squared error, among many other criteria) and select the value of λ that achieves the desired approximation performance on training and validation datasets.

Alternatively, to fully remove the connection between u^2 and x^1, we can design another set of neurons $r_{h+1}, \ldots, r_{2h}$ in the hidden layer as shown in Fig. 3. It is demonstrated that u^2 is disconnected from the neurons $r_1, \ldots, r_h$ that contribute to the output vector x^1 to eliminate the impact of u^2 on x^1. As a result, to maintain the impact of inputs on the other output vector x^2, the new set of neurons $r_{h+1}, \ldots, r_{2h}$ are utilized in the hidden layer to connect both the inputs u^1 and u^2 to the output x^2. It is noted that compared to a fully-connected RNN model, the number of neurons and the number of weights in the weight-constrained RNN shown in Fig. 3 are increased to separate the connections to multiple output vectors.

Based on the RNN model of Eq. (3), the output vector x and the hidden neuron r_i, $i = 1, \ldots, 2h$ in Fig. 3 are derived as follows:

$$\dot{x}^1 = \sum_{i=1}^{h} w_i^{(2)} \dot{r}_i, \quad \dot{x}^2 = \sum_{i=1}^{2h} w_i^{(2)} \dot{r}_i \tag{9}$$

$$\dot{r}_i = -a_i r_i + \theta_i y, \ \ i = 1, \ldots, 2h \tag{10}$$

where $\theta_i = b_i[w_{1i}^{(1)}, \ldots, w_{(2h)i}^{(1)}, \ldots, w_{(2h+m)i}^{(1)}]$ and $y = [\sigma(r_1), \ldots, \sigma(r_{2h}), u^1, u^2]^T$. a_i and b_i are constants, $w_{ji}^{(1)}$ is the weight connecting the jth input, $j = 1, \ldots, 2h + m$ to ith neuron, $i = 1, \ldots, 2h$, and y is the input vector consisting of the hidden states r and the manipulated inputs u. $w^{(1)}$, $w^{(2)}$ represent the weight vectors before and after the hidden layer. Specifically, to train the weight-constrained RNN model with the structure of Fig. 3, we first develop a fully-connected RNN model and then let the weights between u^2 and r_i, $i = 1, \ldots, h$, and the weights between r_i, $i = h + 1, \ldots, 2h$ and x^1 (denoted by $\tilde{w}$) be zero or be constrained by a sufficiently small bound. Unlike the weight-constrained RNN model in Fig. 2, the weight constraint for the RNN model with the structure of Fig. 3 can be equipped with a zero bound such that the connections can be fully removed for the network. Additionally, the above weight constraints on the RNN weights need to be well-defined before training. It should be noted that since there exist three types of weight matrices in an RNN model: (1) the weight matrix connecting the input layer and the hidden layer, (2) the weight matrix feeding the past neuron information into the current network (i.e., the feedback loop in r_i, $i = 1, \ldots, 2h$), and (3) the weight matrix connecting the hidden layer to the output layer, the constraints need to be implemented in all the three weight matrices such that u^2 and x^1 are fully disconnected.

In this work, we train the above weight-constrained RNN model in Keras, and implement weight constraints in the *constraints.py* source file. Specifically, the constraints on the weight matrices that connect inputs to hidden neurons and hidden neurons to outputs are activated through the argument `kernal_constraint`. The weight matrix feeding the past neuron information to the current network is implemented by invoking `recurrent_constraint`. Additionally, to develop an RNN model that obtains the optimal weights subject to the weight constraints, the RNN optimizer (e.g., *adaptive learning rate optimization algorithm*) needs to be modified to minimize the loss function while accounting for the weight constraints in the optimization problem. Alternatively, the weight constraints can be implemented at the end of each training epoch such that the weights that meet the constraints remain unchanged and those exceeding the constraints will be bounded to the saturation value. The saturated weights will then be utilized as the initial condition for the optimization problem for the next training epoch, and the above process is repeated until the stopping criteria of the training process are satisfied.

Remark 4. In this section, we proposed two approaches for weight-constrained RNN models. Specifically, the first approach (i.e., Fig. 2) is to develop a weight-constrained RNN model with a regularization term on constrained weights in the training process of RNN models to reduce the connection between u^2 and x^1. This is typically used for the systems where we know a priori that the connections between certain inputs and outputs are weakly connected (but not fully unconnected). However, the second approach (i.e., Fig. 3) is to develop a weight-constrained RNN model by adding another set of neurons such that the connections between u^2 and x^1 are fully removed. Therefore, it will be applied to the systems where some of the inputs do not affect the outputs at all.

3.4. RNN training process

All the physics-based RNN models are developed using Keras library, an open-source neural-network library written in Python. Specifically, the hybrid model is developed following the construction method for a fully-connected RNN model, where the training dataset is preprocessed to represent the gap between the first-principles model and real process data, and then separated into training, validation and testing datasets. To develop a partially-connected RNN model in Fig. 1, an RNN layer is first developed to connect u^1 and x^1. Subsequently, x^1 and u^2 are concatenated and followed by a second RNN layer to ultimately obtain x^2. It is noted that instead of using the full input and output vectors u and x, the input vectors u^1, u^2 and the output vectors x^1, x^2 need to be specified and fed into the partially-connected RNN model separately. Therefore, the inputs and the outputs to the

partially-connected RNN model in the training process are defined as $[u^1, u^2]$ and $[x^1, x^2]$, respectively. The development of a weight-constrained RNN model in Fig. 2 follows that for a fully-connected RNN model except that the weight constraints need to be added in the optimization problem of RNN training process beforehand by updating Keras optimizer source files. However, to develop a weight-constrained RNN model with the structure of Fig. 3, we can either implement the weight constraints within each epoch of the optimization process in Keras optimizer source files, or saturate the corresponding elements in the weight matrices at the end of each epoch to update the initial guess of weights for the next training epoch in Keras constraints source files. The training processes for both weight-constrained RNN models follow that for a fully-connected RNN model, where the training and validation datasets are used to obtain the optimal weight matrices for RNN models, and the testing dataset is used to evaluate their prediction performances.

To prevent the weights from drifting to infinity during the RNN training process, the weight vector θ_i of the RNN model of Eq. (3) is also bounded by $|\theta_i| \leq \theta_m$, with $\theta_m > 0$. It is noted that while it is possible to obtain a theoretical value for θ_m, this value will usually be conservative. Therefore, in the implementation of the RNN training process, we give θ_m a reasonable value and see if the RNN can approximate the nonlinear system of Eq. (1) with the satisfaction of the modeling error constraint. The interested reader may refer to [8], where a σ-modification is utilized in the RNN learing algorithm to ensure that the weights are bounded during the training process. Additionally, the hybrid model, the partially-connected RNN model, and the weight-constrained RNN model are all trained with a constraint on the modeling error, i.e., $|\nu| = |F(x,u,0) - F_{nn}(x,u)| \leq \gamma|x|$, where $\gamma > 0$, such that the obtained RNN models can well represent the actual nonlinear process of Eq. (1) and can be utilized in a model-based predictive controller that stabilizes the system at its steady-state with guaranteed stability. The detailed RNN learning algorithm and the proof of the boundedness of RNN modeling error can be found in Wu et al. [24].

Remark 5. Consider the nonlinear system of Eq. (1) with $x = [x^1,\ x^2] \in \mathbf{R}^n$ and $u = [u^1,\ u^2] \in \mathbf{R}^n$, where x^1 and x^2, u^1 and u^2 are of the same dimension, respectively (i.e., $x^1, x^2 \in \mathbf{R}^{\frac{n}{2}}$, $u^1, u^2 \in \mathbf{R}^{\frac{m}{2}}$). Under the assumption of the input-output relationship in this section, the total number of weights for a partially-connected RNN model with two hidden layers, where each hidden layer has h neurons, is calculated to be $\frac{3}{2}nh + mh + 2h^2$, while the total number of weights for a fully-connected RNN model with the same two hidden layers is $mh + 3h^2 + nh$ (the bias term is ignored in the comparison as it can be considered a constant input node). Since in most cases, the number of neurons is much greater than the number of inputs and states to achieve a desired approximation performance, the number of weights for a decoupled RNN model is significantly reduced due to the incorporation of process structural knowledge ($\frac{3}{2}nh + mh + 2h^2 << nh + mh + 3h^2$ when $h >\geq m, n$). However, it is noted that the number of weights in a weight-constrained model with the structure of Fig. 3 is increased compared to the fully-connected RNN model due to the new set of hidden neurons that are used to rebuild the connection between u^2 and x^2.

Remark 6. In addition to the weight constraints as discussed above, regularization can be utilized to penalize the weights that need to be constrained in the loss function of Eq. (8). The implementation of regularization using Keras is as follows. First, the `kernel_regularizer` command is invoked to activate the regularization term in the loss function in each layer with the desired regularization technique (i.e., L1 or L2 regularization). With this, all the elements in the weight matrices that connect the inputs to the hidden neurons and connect the hidden neurons to the outputs are penalized in the loss function with a regularization parameter λ as introduced in Eq. (8). Subsequently, to ensure that only the weights that need to be constrained are penalized in the loss function, the *regularizers.py* source file is adapted in which the corresponding weights for the undesired connections are included. Finally, the weight matrix feeding the past neuron information into the current network is penalized following the same strategy as discussed above for the implementation of weight constraints.

Remark 7. It is noted that all the RNN models in this section are developed for the nominal system of Eq. (1) without disturbances. However, in the presence of time-varying disturbances, the RNN model that is trained for the nominal system may be dysfunctional in a model-based predictive controller due to a considerable model mismatch. To that end, online update of RNN models can be employed to capture the nonlinear dynamics subject to disturbances using the most recent process measurement data. The interested readers may refer to [23] for the details of implementation of online RNN update.

Remark 8. In the case that a single RNN model is not able to well represent the dynamics of the nonlinear process of Eq. (1) in the entire operating region, multiple RNN models can be developed to improve the overall prediction accuracy in the context of ensemble learning [13,27]. The development of multiple RNN models via ensemble learning can be found in [25], where k different RNN models were developed for the same nonlinear process based on a k-fold cross-validation and were utilized to derive a final prediction result that was significantly improved compared to a single RNN model. Additionally, in Wu et al. [25], to improve computational efficiency of ensemble learning and multiple RNN predictions in the real-time implementation of machine-learning-based predictive controllers, parallel computing can be employed to speed up the computation of RNN predictions using multiple compute cores in a distributed computing cluster. It was also demonstrated in Wu et al. [25] that the computation time of calculating multiple RNN prediction results in an RNN-based predictive controller was significantly reduced under parallel computation of the ensemble of RNN models.

Remark 9. To extend the proposed RNN modeling methods to high-dimensional systems, it is necessary to find the relationships between the inputs and outputs. For example, we can introduce sparsity regularization that is similar to Eq. (8) but with relatively large penalty, or we can also apply well-established methods such as relative gain array to determine the best input-output pairings for multivariable processes.

4. RNN-based predictive control

In this section, we incorporate the RNN model developed in the previous section into the design of model-based predictive controllers to optimize process performance while guaranteeing closed-loop stability. Specifically, for any initial condition $x_0 \in \Omega_\rho$, a Lyapunov-based model predictive controller (LMPC) using RNN models is developed to drive the closed-loop state of the nonlinear system of Eq. (1) to the steady-state while maintaining the state in the stability region Ω_ρ for all times. Subsequently, Lyapunov-based economic model predictive controller (LEMPC) is developed using the RNN model to optimize process economic performance with guaranteed boundedness of the state in Ω_ρ for all times.

4.1. Lyapunov-based MPC using RNN models

The Lyapunov-based model predictive control (LMPC) using the RNN model of Eq. (3) is utilized to stabilize the nonlinear system

of Eq. (1) in the stability region. The formulation of LMPC optimization problem is given as follows: [24,25]

$$\mathcal{J} = \min_{u\in S(\Delta)} \int_{t_k}^{t_{k+N}} (\tilde{x}^T Q\tilde{x} + u^T Ru)dt \tag{11a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{11b}$$

$$u(t) \in U, \ \forall\ t \in [t_k, t_{k+N}) \tag{11c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{11d}$$

$$\dot{V}(x(t_k), u) \le \dot{V}(x(t_k), \Phi_{nn}(x(t_k))), \quad \text{if } x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}} \tag{11e}$$

$$V(\tilde{x}(t)) \le \rho_{nn}, \ \forall\ t \in [t_k, t_{k+N}), \ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{11f}$$

where $\tilde{x}$ is the predicted state trajectory, $S(\Delta)$ is the set of piecewise constant functions with period Δ, N is the number of sampling periods in the prediction horizon, and $\dot{V}(x,u)$represents $\frac{\partial V(x)}{\partial x}(F_{nn}(x,u))$. In the optimization problem of Eq. (11), the objective function of Eq. (11a) is the integral of the cost function $l(\tilde{x},t) = (\tilde{x}^T Q\tilde{x} + u^T Ru)$ over the prediction horizon, where $l(0,0) = 0$ and $l(\tilde{x},t) > 0, \ \forall(\tilde{x},t) \neq (0,0)$. The constraint of Eq. (11b) is the RNN model of Eq. (3) that is used to predict the states of the closed-loop system. Eq. (11c) defines the input constraints applied over the entire prediction horizon. Eq. (11d) defines the initial condition $\tilde{x}(t_k)$ of Eq. (11b), which is the state measurement at $t = t_k$. The constraint of Eq. (11e) forces the closed-loop state to move towards the origin if $x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}}$. However, if $x(t_k)$ enters $\Omega_{\rho_{nn}}$, the states predicted by the RNN model of Eq. (11b) will be maintained in $\Omega_{\rho_{nn}}$ for the entire prediction horizon. The LMPC of Eq. (11) is implemented in a sample-and-hold fashion, i.e., an optimal input trajectory $u^*(t)$, $t \in [t_k, t_{k+N})$ is obtained by solving the LMPC optimization problem of Eq. (11) at each sampling time, from which only the control action for the first sampling period of the prediction horizon will be applied. In [24], it is demonstrated that under the LMPC of Eq. (11), the state of the nonlinear system of Eq. (1) is bounded in the stability region Ω_ρ for all times, and can ultimately converge to the origin provided that the modeling error between the nonlinear system of Eq. (1) and the RNN model of Eq. (3) is sufficiently small. Detailed proof for closed-loop stability can be found in [24] and is omitted here due to space limitations.

4.2. Lyapunov-based EMPC using RNN models

The Lyapunov-based economic model predictive control (LEMPC) using the RNN model of Eq. (3) is utilized to optimize process economic performance while maintaining the closed-loop state of the nonlinear system of Eq. (1) in the stability region Ω_ρ. TheLEMPC is formulated by the following optimization problem: [22]

$$\mathcal{J} = \max_{u\in S(\Delta)} \int_{t_k}^{t_{k+N}} l_e(\tilde{x}(t), u(t))dt \tag{12a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{12b}$$

$$u(t) \in U, \ \forall\ t \in [t_k, t_{k+N}) \tag{12c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{12d}$$

$$V(\tilde{x}(t)) \le \rho_e, \ \forall\ t \in [t_k, t_{k+N}), \ \text{if } x(t_k) \in \Omega_{\rho_e} \tag{12e}$$

$$\dot{V}(x(t_k), u) \le \dot{V}(x(t_k), \Phi_{nn}(x(t_k))), \quad \text{if } x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_e} \tag{12f}$$

where the notations follow those in Eq. (11). The optimization problem of Eq. (12) maximizes the objective function of Eq. (12a) that integrates $l_e(\tilde{x}(t), u(t))$ over the prediction horizon subject to the constraints of Eqs. (12b)–(12f). Specifically, the constraint of Eqs. (12b)–(12d) are the same as Eqs. (11b)–(11d) for LMPC. The constraint of Eq. (12e) maintains the predicted closed-loop states in Ω_{ρ_e} if $x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_e}$, where Ω_{ρ_e}, $0 < \rho_e < \rho$, is a level set of Lyapunov function that guarantees the boundedness of state in the closed-loop stability region Ω_ρ accounting for the model mismatch between the RNN model of Eq. (12b) and the nonlinear process of Eq. (1). On the other hand, if $x(t_k)$ leaves Ω_{ρ_e}, the contractive constraint of Eq. (12f) will be activated to drive the state towards the origin within the next sampling period. It is demonstrated that the closed-loop state of the nonlinear system of Eq. (1) is bounded in the stability region Ω_ρ for all times under the LEMPC of Eq. (12). The detailed proof of closed-loop stability under LEMPC is given in Wu et al. [24].

Remark 10. It is demonstrated in [24] that closed-loop stability is guaranteed for the nonlinear system of Eq. (1) under the RNN-based MPC of Eq. (11) provided that the modeling error between the RNN model and the actual nonlinear system is sufficiently small. Specifically, the constraints of Eqs. (11e) and (11f) are developed to guarantee that the closed-loop state will move towards the origin and can be ultimately bounded in a small neighborhood around the origin regardless of the length of prediction horizon. However, it is noted that the use of a longer prediction horizon in MPC can generally improve closed-loop performance by obtaining better solutions that lead to less control energy consumption and smoother state trajectories. Though we did not show the detailed proof for closed-loop stability due to space limitation in this manuscript, the proposed partially-connected RNN and weight-constrained RNN models that account for process structural knowledge by modifying the structure of RNNs, and adding certain constraints in the training process of RNNs, respectively, are developed satisfying the modeling error constraint. Therefore, closed-loop stability can be established for the closed-loop MPC using the proposed RNN models.

5. Application to a chemical process example

A chemical process example is utilized to demonstrate the application of the proposed RNN modeling with the incorporation of structural process knowledge. Specifically, two well-mixed, non-isothermal continuous stirred tank reactors (CSTR) in series are considered where an irreversible second-order exothermic reaction takes place in each reactor as shown in Fig. 4. The reaction transforms a reactant A to a product B ($A \rightarrow B$). Each of the two reactors are fed with reactant material A with the inlet concentration C_{Aj0}, the inlet temperature T_{j0} and feed volumetric flow rate of the reactor F_{j0}, $j = 1, 2$, where $j = 1$ denotes the first CSTR and $j = 2$ denotes the second CSTR. Each CSTR is equipped with a heating jacket that supplies/removes heat at a rate Q_j, $j = 1, 2$. The CSTR dynamic models are described by the following material and energy balance equations:

$$\frac{dC_{A1}}{dt} = \frac{F_{10}}{V_1}(C_{A10} - C_{A1}) - k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 \tag{13a}$$

$$\frac{dT_1}{dt} = \frac{F_{10}}{V_1}(T_{10} - T_1) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 + \frac{Q_1}{\rho_L C_p V_1} \tag{13b}$$

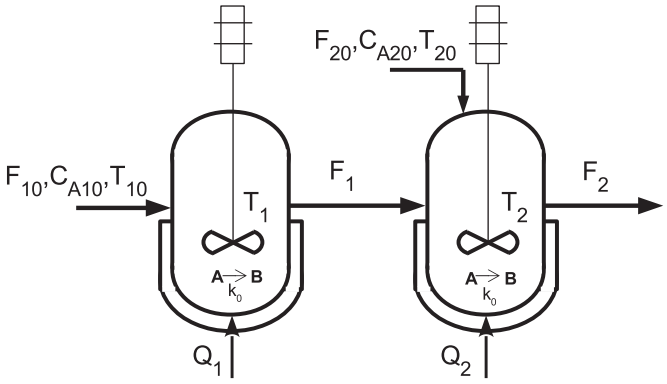

Fig. 4. Process flow diagram of two CSTRs in series.

Table 1
Parameter values of the CSTRs.

$T_{10} = 300$ K	$T_{20} = 300$ K
$F_{10} = 5$ m^3/h	$F_{20} = 5$ m^3/h
$V_1 = 1$ m^3	$V_2 = 1$ m^3
$T_{1_s} = 402$ K	$T_{2_s} = 402$ K
$C_{A1_s} = 1.95$ kmol/m^3	$C_{A2_s} = 1.95$ kmol/m^3
$C_{A10_s} = 4$ kmol/m^3	$C_{A20_s} = 4$ kmol/m^3
$Q_{1_s} = 0.0$ kJ/hr	$Q_{2_s} = 0.0$ kJ/h
$k_0 = 8.46 \times 10^6$ m^3/kmol h	$\Delta H = -1.15 \times 10^4$ kJ/kmol
$C_p = 0.231$ kJ/kg K	$R = 8.314$ kJ/kmol K
$\rho_L = 1000$ kg/m^3	$E = 5 \times 10^4$ kJ/kmol

$$\frac{dC_{B1}}{dt} = -\frac{F_{10}}{V_1}C_{B1} + k_0 e^{\frac{-E}{RT_1}} C_{A_1}^2 \tag{13c}$$

$$\frac{dC_{A2}}{dt} = \frac{F_{20}}{V_2}C_{A20} + \frac{F_{10}}{V_2}C_{A1} - \frac{F_{10}+F_{20}}{V_2}C_{A2} - k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 \tag{13d}$$

$$\frac{dT_2}{dt} = \frac{F_{20}}{V_2}T_{20} + \frac{F_{10}}{V_2}T_1 - \frac{F_{10}+F_{20}}{V_2}T_2 + \frac{-\Delta H}{\rho_L C_p}k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 + \frac{Q_2}{\rho_L C_p V_2} \tag{13e}$$

$$\frac{dC_{B2}}{dt} = \frac{F_{10}}{V_2}C_{B1} - \frac{F_{10}+F_{20}}{V_2}C_{B2} + k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 \tag{13f}$$

where C_{Aj}, V_j, T_j and Q_j, $j = 1, 2$ are the concentration of reactant A, the volume of the reacting liquid, the temperature, and the heat input rate in the first and the second reactor, respectively. The reacting liquid has a constant density of ρ_L and a heat capacity of C_p for both reactors. ΔH, k_0, E, and R represent the enthalpy of the reaction, pre-exponential constant, activation energy, and ideal gas constant, respectively, and are the same for both reactors. Process parameter values are listed in Table 1.

The manipulated inputs for both CSTRs are the inlet concentration of species A and the heat input rate, which are represented by the deviation variables $\Delta C_{Aj0} = C_{Aj0} - C_{Aj0_s}$, $\Delta Q_j = Q_j - Q_{j_s}$, $j = 1, 2$, respectively. The manipulated inputs are bounded as follows: $|\Delta C_{Aj0}| \leq 3.5$ kmol/m^3 and $|\Delta Q_j| \leq 5 \times 10^5$ kJ/h, $j = 1, 2$. Therefore, the states and the inputs of the closed-loop system are $x^T = [C_{A1} - C_{A1_s} \quad T_1 - T_{1_s} \quad C_{A2} - C_{A2_s} \quad T_2 - T_{2_s}]$ and $u^T = [\Delta C_{A10} \quad \Delta Q_1 \quad \Delta C_{A20} \quad \Delta Q_2]$, respectively, where C_{A1_s}, C_{A2_s}, T_{1_s} and T_{2_s} are the steady-state values of concentration of A and temperature in the first and second reactors, such that the equilibrium point of the system is at the origin of the state-space.

The explicit Euler method with an integration time step of $h_c = 10^{-4}$ h is used to numerically simulate the dynamic model of Eq. (13). The nonlinear optimization problems of the LMPC of Eq. (11) and of the LEMPC of Eq. (12) are solved using the python module of the IPOPT software package [21], named PyIpopt with the sampling period $\Delta = 10^{-2}$ h. Two control Lyapunov functions $V_1(x) = x^T P_1 x$, and $V_2(x) = x^T P_2 x$ are designed for two CSTRs, respectively, with the following positive definite P matrices:

$$P_1 = P_2 = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{14}$$

The closed-loop stability regions for the two CSTRs are characterized with $\rho = 380$, where $\rho_e = 260$ is chosen for the LEMPC of Eq. (12).

Table 2
RMSE comparison of open-loop prediction results with the first-principles model results.

	P-RNN	W-RNN	F-RNN
C_{A1} (kmol/m^3)	1.0×10^{-4}	5.6×10^{-6}	0.9×10^{-4}
T_1 (K)	0.14	0.018	0.15
C_{A2} (kmol/m^3)	8.2×10^{-7}	2.0×10^{-6}	2.6×10^{-6}
T_2 (K)	5.4×10^{-4}	0.0076	0.049

5.1. Open-loop simulation on testing dataset

Open-loop simulations are first carried out to demonstrate the open-loop prediction performances of the fully-connected RNN model, the partially-connected RNN model, and the weight-constrained RNN model, respectively. The development of an RNN model for the CSTR process of Eq. (13) follows that in Wu et al. [25]. It should be noted that all the RNN models are developed using the same dataset with the same neural network parameters as follows: 2 hidden layers with 30 neurons in each layer, *tanh* as the activation function, and *Adam* as the optimizer. The root mean square errors (RMSE) between the first-principles state trajectories (i.e., the state trajectories using the first-principles model of Eq. (13)) and the above three models, respectively, are reported in Table 2, where P-RNN, W-RNN and F-RNN represent the partially-connected RNN model, the weight-constrained RNN model, and the fully-connected RNN model, respectively.

From Table 2, it is demonstrated that the partially-connected RNN model and the weight-constrained RNN model outperform the fully-connected model in that the open-loop approximations of C_{A1}, C_{A2}, T_1 and T_2 are significantly improved.

Remark 11. It is noted that the comparison results in Table 2 were generated using extensive open-loop simulations with various initial conditions and control actions, under which the superiority of the proposed modeling approaches is clearly demonstrated by showing that the partially-connected RNN model and the weight-constrained RNN model outperform the fully-connected RNN model in terms of better approximation performance in the entire operating region.

5.2. Closed-loop simulation under LMPC

After demonstrating the open-loop prediction performances of the fully-connected RNN, the partially-connected RNN and the weight-constrained RNN for the CSTR process of Eq. (13) in the stability region, we perform the closed-loop simulation under the LMPC of Eq. (11) using the above three models, respectively. Additionally, the closed-loop simulation under the LMPC of Eq. (11) using the first-principles model of Eq. (13) is added as a baseline for comparison. The control objective of RNN-based LMPC is to operate the CSTR process of Eq. (13) at its steady-state while maintaining the closed-loop state in the stability region Ω_ρ for all times.

In Fig. 5, it is demonstrated that all the states (i.e., C_{A1}, T_1, C_{A2} and T_2) converge to the origin within 0.05 h under the LMPC using the partially-connected RNN model and the weight-constrained RNN model. However, under the LMPC using a fully-connected

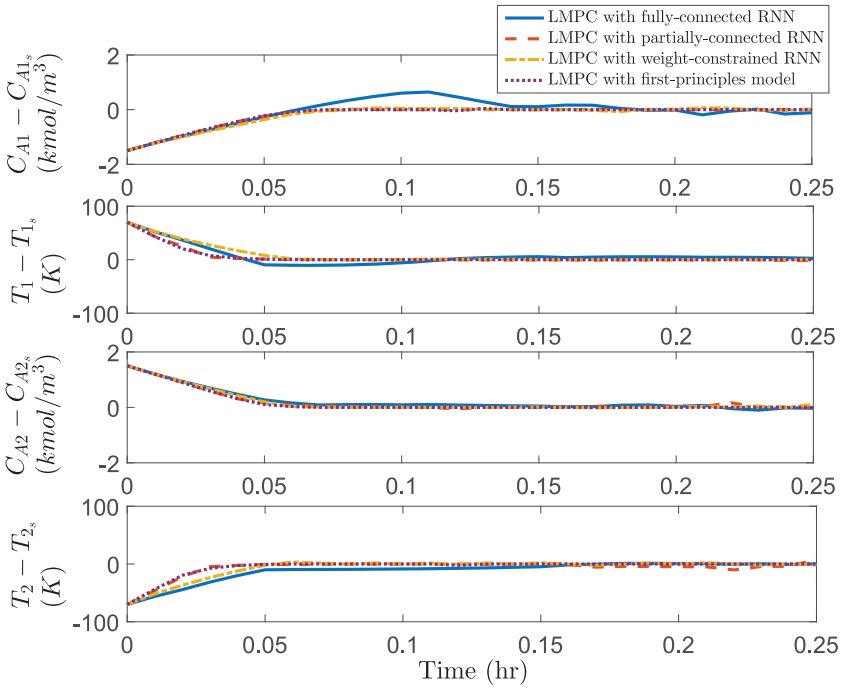

Fig. 5. The state profiles ($C_{A1} - C_{A1_s}$, $T_1 - T_{1_s}$, $C_{A2} - C_{A2_s}$ and $T_2 - T_{2_s}$) for the closed-loop simulationof two CSTRs in series under the LMPCusing the fully-connected RNN, the partially-connected RNN, the weight-constrained RNN, and the first-principles model of Eq. (13), respectively, for an initial condition (−1.5, 70, 1.5, −70).

RNN model, the concentration in the first CSTR (i.e., C_{A1}) shows undesirable oscillations around the origin due to its considerable model mismatch as reported in Table 2. Therefore, through open-loop and closed-loop simulations, the partially-connected RNN model and the weight-constrained RNN model that incorporate structural process knowledge of the CSTR process of Eq. (13) are demonstrated to achieve better approximation performance than the fully-connected RNN model.

5.3. Closed-loop simulation under LEMPC

The control objective of LEMPC is to maximize the profit of both CSTR systems described in Eq. (13) by manipulating the inlet concentration ΔC_{A10} and C_{A20} and the heat inputs rate ΔQ_1 and ΔQ_2, and meanwhile maintain the closed-loop state trajectories in the stability region Ω_ρ for all times. The objective function of the LEMPC optimizes the production rate of B as follows:

$$l_e(\tilde{x}, u) = k_0 e^{-E/RT_1} C_{A1}^2 + k_0 e^{-E/RT_2} C_{A2}^2 \tag{15}$$

Closed-loop simulations are performed under the LEMPC of Eq. (12) using the first-principles model of Eq. (13) and the three RNN models, respectively. In Fig. 6, it is demonstrated that the state trajectories for both CSTRs are bounded in the stability region Ω_ρ for all times under LEMPC. Fig. 7 shows the evolution the Lyapunov function values of V_1 and V_2 under LEMPC using the first-principles model of Eq. (13) and three different RNN models, respectively. Specifically, due to a relatively large model mismatch for the fully-connected RNN model as reported in Table 2, the contractive constraint of Eq. (12f) is activated frequently under the LEMPC using a fully-connected RNN model because the actual process state does not stay in Ω_{ρ_e} under the constraint of Eq. (12e). As a result, it is observed in Fig. 7 that the V profiles under the fully-connected model show larger oscillation compared to those under the other two RNN models and under the first-principles model.

Additionally, we compare the accumulated economic profits $L_E = \int_0^{t_p} L_e(x, u)dt$ within the operation period $t_p = 0.32$ h for the closed-loop CSTRs under the steady-state operation (i.e., the CSTRs

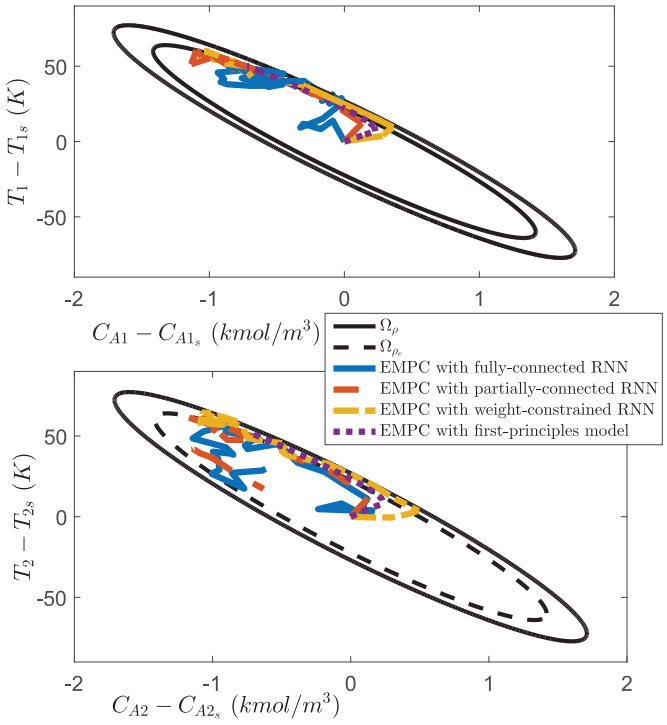

Fig. 6. The state-space profiles for the closed-loop simulationfor CSTR 1 (top plot) and CSTR 2 (bottom plot) under the EMPCusing the fully-connected RNN model, the partially-connected RNN model, the weight-constrained RNN model, and the first-principles model of Eq. (13), respectively, for an initial condition (0, 0, 0, 0).

are operated at their steady-states for all times), and the LEMPC using the first-principles model of Eq. (13) and the three RNN models, respectively. The result is shown in Fig. 8, from which it is demonstrated that the closed-loop operation under LEMPC achieves higher economic profits than the steady-state operation.

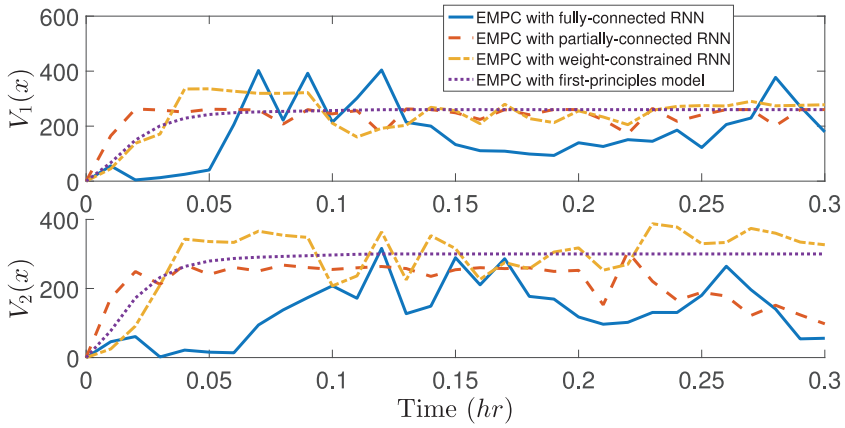

Fig. 7. The Lyapunov function value evaluation with respect to time for the closed-loop CSTR 1 (top plot) and CSTR 2 (bottom plot) under the EMPC using the fully-connected RNN model, the partially-connected RNN model, the weight-constrained RNN model, and the first-principles model of Eq. (13), respectively, for an initial condition (0, 0, 0, 0).

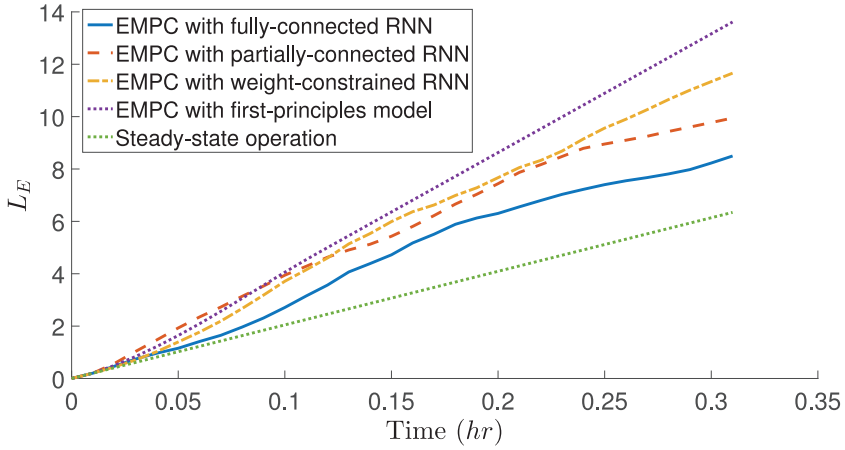

Fig. 8. Accumulated economic profits for the closed-loop CSTRs under the steady-state operation and under the EMPC using the first-principles model of Eq. (13), the fully-connected RNN model, the partially-connected RNN model, and the weight-constrained RNN model, respectively, for an initial condition (0, 0, 0, 0).

Specifically, the LEMPC using the first-principles model achieves the highest economic benefits since the closed-loop state trajectory reaches and stays at the boundary of Ω_{ρ_e} smoothly based on accurate predictions. Moreover, it is demonstrated that the LEMPC using the partially-connected RNN model and the weight-constrained RNN model economically outperform that under the fully-connected RNN model due to better prediction accuracy in the stability region. Therefore, through both open-loop and closed-loop simulations, we demonstrate that the physics-based RNN models achieve desired approximation performance for the CSTR process of Eq. (13) and provide reliable state predictions for model-based predictive controllers.

6. Conclusion

In this work, we developed three modeling approaches that incorporates a priori process knowledge into RNN models. Specifically, a hybrid model that combines a first-principles model and an RNN model was first developed. Then, a partially-connected RNN model and a weight-constrained RNN model were developed based on an assumption on process input-output relationship. The partially-connected and the weight-constrained RNN models were then utilized in RNN-MPC and RNN-EMPC, and applied to a chemical process example, from which it was demonstrated that the open-loop and closed-loop prediction performances under the LMPC and the LEMPC using the above two RNN models outperformed those under the LMPC and LEMPC using a fully-connected RNN model in terms of higher prediction accuracy, smoother state trajectories, and better economic benefits.

Declaration of Competing Interest

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

CRediT authorship contribution statement

Zhe Wu: Conceptualization, Methodology, Software, Writing - original draft. **David Rincon:** Methodology, Software, Writing - original draft. **Panagiotis D. Christofides:** Writing - review & editing.

Acknowledgments

Financial support from the National Science Foundation and the Department of Energy is gratefully acknowledged.

References

[1] G. Ba, Y. Zhao, A. Kadambi, Blending diverse physical priors with neural networks, arXiv:1910.00201 (2019).
[2] S. Feyo de Azevedo, B. Dahm, F. Oliveira, Hybrid modelling of biochemical processes: acomparison with the conventional approach, Comput. Chem. Eng. 21 (1997) S751–S756.

[3] P. Georgieva, M. Meireles, S.F. de Azevedo, Knowledge-based hybrid modelling of a batch crystallisation when accounting for nucleation, growth and agglomeration phenomena, Chem. Eng. Sci. 58 (2003) 3699–3713.
[4] O. Kahrs, W. Marquardt, The validity domain of hybrid models and its application in process optimization, Chem. Eng. Process. 46 (2007) 1054–1066.
[5] O. Kahrs, W. Marquardt, Incremental identification of hybrid process models, Comput. Chem. Eng. 32 (2008) 694–705.
[6] A. Karpatne, W. Watkins, J. Read, V. Kumar, Physics-guided neural networks (PGNN): an application in lake temperature modeling, arXiv:1710.11431 (2017).
[7] M. Kellman, E. Bostan, N. Repina, L. Waller, Physics-based learned design: optimized coded-illumination for quantitative phase imaging, arXiv:1808.03571 (2019).
[8] E.B. Kosmatopoulos, M.M. Polycarpou, M.A. Christodoulou, P.A. Ioannou, High-order neural network structures for identification of dynamical systems, IEEE Trans. Neural Netw. 6 (1995) 422–431.
[9] Y. Lin, E.D. Sontag, A universal formula for stabilization with bounded controls, Syst. Control Lett. 16 (1991) 393–397.
[10] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-net: Learning PDEs from data, arXiv:1710.09668 (2017).
[11] Y. Lu, M. Rajora, P. Zou, S. Liang, Physics-embedded machine learning: case study with electrochemical micro-machining, Machines 5 (2017) 4.
[12] M.L. Luyben, B.D. Tyreus, W.L. Luyben, Plantwide control design procedure, AIChE J. 43 (1997) 3161–3174.
[13] J. Mendes-Moreira, C. Soares, A.M. Jorge, J.F.D. Sousa, Ensemble approaches for regression: a survey, ACM Comput. Surv. 45 (2012) 10.
[14] D. Psichogios, L. Ungar, A hybrid neural network-first principles approach to process modeling, AIChE J. 38 (1992) 1499–1511.
[15] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, S. Chung, Neural lander: Stable drone landing control using learned dynamics, in: Proceedings of the International Conference on Robotics and Automation, Montreal, Canada, 2019, pp. 9784–9790.
[16] E.D. Sontag, Neural nets as systems models and controllers, in: Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems, Yale University, 1992, pp. 73–79.
[17] E.D. Sontag, A 'universal' construction of Artstein's theorem on nonlinear stabilization, Syst. Control Lett. 13 (1989) 117–123.
[18] G. Stephanopoulos, C. Han, Intelligent systems in process engineering: areview, Comput. Chem. Eng. 20 (1996) 743–791.
[19] M. Thompson, M. Kramer, Modeling chemical processes using prior knowledge and neural networks, AIChE J. 40 (1994) 1328–1340.
[20] R. Turton, R.C. Bailie, W.B. Whiting, J.A. Shaeiwitz, Analysis, Synthesis and Design of Chemical Processes, Pearson Education, 2008.
[21] A. Wächter, L.T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Math. Program. 106 (2006) 25–57.
[22] Z. Wu, P.D. Christofides, Economic machine-learning-based predictive control of nonlinear systems, Mathematics 7 (6) (2019) 494.
[23] Z. Wu, D. Rincon, P.D. Christofides, Real-time adaptive machine-learning-based predictive control of nonlinear processes, Ind. Eng. Chem. Res. 59 (2020) 2275–2290.
[24] Z. Wu, A. Tran, D. Rincon, P.D. Christofides, Machine learning-based predictive control of nonlinear processes. Part I: theory, AIChE J. 65 (2019) e16729.
[25] Z. Wu, A. Tran, D. Rincon, P.D. Christofides, Machine learning-based predictive control of nonlinear processes. Part II: computational implementation, AIChE J. 65 (2019) e16734.
[26] Q. Xiong, A. Jutan, Grey-box modelling and control of chemical processes, Chem. Eng. Sci. 57 (2002) 1027–1039.
[27] C. Zhang, Y. Ma, Ensemble Machine Learning: Methods and Applications, Springer, 2012.
[28] Z. Zhang, Z. Wu, D. Rincon, P.D. Christofides, Real-time optimization and control of nonlinear processes using machine learning, Mathematics 7 (2019) 890.
[29] L. Zorzetto, R. Maciel Filho, M. Wolf-Maciel, Processing modelling development through artificial neural networks and hybrid models, Comput. Chem. Eng. 24 (2000) 1355–1360.