Available online at www.sciencedirect.com

Chemical Engineering Research and Design

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

On generalization error of neural network models and its application to predictive control of nonlinear processes

Mohammed S. Alhajeri[a,c,1], *Aisha Alnajdi*[b,d,1], *Fahim Abdullah*[a], *Panagiotis D. Christofides*[a,b,*]

[a] *Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095-1592, USA*
[b] *Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095-1592, USA*
[c] *Department of Chemical Engineering, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait*
[d] *Department of Electrical Engineering, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait*

ARTICLE INFO

Article history:
Received 12 November 2022
Received in revised form
30 November 2022
Accepted 1 December 2022
Available online 5 December 2022

Keywords:
Generalization error
Recurrent neural networks
Long short-term memory
Machine learning
Partially-connected RNN
Model predictive control
Nonlinear systems

ABSTRACT

In order to approximate nonlinear dynamic systems utilizing time-series data, recurrent neural networks (RNNs) and long short-term memory (LSTM) networks have frequently been used. The training error of neural networks may often be made suitably modest; however, the accuracy can be further improved by incorporating prior knowledge in the construction of machine learning-based models. Specifically, physics-based RNN modeling has yielded more reliable RNN models than traditional RNNs. Yet, a framework for constructing and assessing the generalization ability of such RNN models as well as LSTM models to be utilized in model predictive control (MPC) systems is lacking. In this work, we develop a methodological framework to quantify the generalization error bounds for partially-connected RNNs and LSTM models. The partially-connected RNN model is then utilized to predict the state evolution in a MPC scheme. We illustrate through open-loop and closed-loop simulations of a nonlinear chemical process of two reactors-in-series that the proposed approach provides a flexible framework for leveraging both prior knowledge and data, thereby improving the performance significantly when compared to a fully-connected modeling approach under Lyapunov-based MPC.

1. Introduction

An ongoing research issue in process systems engineering is the modeling of large-scale, complicated nonlinear processes. Traditional methods for modeling nonlinear systems include first-principles modeling, which is based on a fundamental comprehension of the core physico-chemical phenomena, and data-driven modeling, which identifies parameters from simulation or industry data (e.g., Cozad et al., 2015; Wilson and Sahinidis, 2017). Although the classic first-principles modeling approach has been widely utilized in the control, monitoring, and optimization of different chemical processes, applying first-principles computational methods to represent complicated nonlinear systems can be time-consuming and inaccurate. Given their ability to

[*] Corresponding author at: Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095-1592, USA.
E-mail address: pdc@seas.ucla.edu (P.D. Christofides).
[1] Contributed equally

https://doi.org/10.1016/j.cherd.2022.12.001
0263-8762/© 2022 Institution of Chemical Engineers. Published by Elsevier Ltd. All rights reserved.

successfully handle large data sets from processes and to model a diverse range of nonlinear functions, machine learning techniques are being used more and more to approximate complicated nonlinear systems (e.g., Han et al., 2013; Ali et al., 2015; Wong et al., 2018; Shahnazari et al., 2019). Among various machine learning modeling tools, when modeling nonlinear dynamic systems utilizing time-series data, recurrent neural networks (RNN) are frequently employed (Fan and Han, 2012; Xu et al., 2016). Although machine learning techniques have been used in chemical process control since the nighties (Vepa, 1993), they have recently gained popularity again due to a variety of factors, including more affordable computation (thanks to mature and effective libraries and hardware), the accessibility of large data sets, and sophisticated learning algorithms. The upcoming generation of industrial control systems will certainly be impacted by the developments of model predictive controllers (MPC) that make use of machine learning models with well-characterized fidelity.

The traditional choice to analyze time-series data in a black-box fashion is a fully-connected RNN model, which densely relates all the available inputs to all the outputs. However, this method may not always be the finest, particularly for intricate chemical processes. For instance, in an integrated chemical plant, the downstream units do not have an impact on the upstream ones. Therefore, in order to further increase a RNN model's accuracy, numerous studies (e.g., Stephanopoulos and Han, 1996; De Azevedo et al., 1997; Kahrs and Marquardt, 2007) have studied gray-box modeling, also referred to in the literature as hybrid modeling, which involves incorporating prior knowledge into the design of neural network models of various chemical processes. A strategy for combining data-driven modeling with first-principles knowledge was recently developed by Patel et al. (2020), and it explicitly permits the inclusion of data on known gains among particular inputs and outcomes. With prior knowledge of relations, this suggested strategy can be used in large-scale processes. Also, other approaches to improve the RNN model's prediction accuracy were proposed, for example a weight-constrained RNN modeling was investigated in Wu et al. (2020) with chemical process example and yield improvements in both open-loop and closed-loop simulations under ML based MPC.

Another modeling strategy to follow is the recently proposed partially-connected RNN which, as the name indicates, partially connects layers based on pre-existing knowledge in terms of physical relations among the underlying system inputs and outputs (Lu et al., 2017; Wu et al., 2020; Alhajeri et al., 2022b). Specifically, on a large and complex chemical process modeled in Aspen Plus Dynamics simulator, Alhajeri et al. (2022b) investigated this approach by carrying out open-loop and closed-loop simulations using a fully-connected RNN model against a partially-connected RNN model. A partially-connected RNN model was shown to outperform the fully-connected RNN model when incorporated into a MPC, with smoother state trajectories and less computational burden. Furthermore, in Alhajeri et al. (2022a), they considered the case of industrial noise (i.e., non-Gaussian noise), where the Monte Carlo dropout and co-teaching strategies were used to train partially-connected RNN models to overcome the over-fitting issue. Subsequently, open-loop and closed-loop simulations were performed on an Aspen Plus Dynamics process model to illustrate the superiority of partially-connected RNN based MPC over fully-connected RNN models trained with dropout/co-teaching and standard partially-connected RNN models with regular training. Additionally, one can consider the Long Short-Term Memory (LSTM), a variant of the RNN that was introduced three decades ago, to model nonlinear systems. LSTMs have a unique structure, which enables them to enhance the model's performance when dealing with data that requires long time dependencies. Such data may occur when modeling nonlinear time-delay systems or even nonlinear systems with disturbances and noise. For instance, in Alhajeri et al. (2022a), a nonlinear system was modeled as an LSTM network using noisy data, and closed loop stability was achieved. Moreover, LSTMs have been shown to overcome the vanishing gradient phenomenon that usually occurs when using RNNs (see Chen et al., 2020 for further details). Hence, LSTMs are widely used in many recent chemical engineering applications, and have proven to be an efficient and powerful machine learning tool.

The adaptation of machine-learning-based MPC to actual chemical processes is still met with several challenges, despite the effectiveness of machine learning approaches in approximating nonlinear process dynamics within the context of MPC. Furthermore, characterizing the generalization capability of machine learning models learned using finite training samples on new data is a significant challenge. The work of Wu et al. (2021) has filled in this gap by computing an explicit expression for the theoretical upper bound of fully-connected RNN models' generalization error. However, the fundamental question regarding the generalization accuracy of partially-connected RNN models in MPC has not been addressed—specifically, how the structure of an RNN model affects its generalization accuracy.

Due to the aforementioned considerations, in this work, we develop, from machine learning theory, a conceptual framework to quantify generalization error bounds for partially-connected RNN models. Also, we integrate these models into model predictive control systems to be implemented in nonlinear chemical processes. This manuscript is divided into 5 sections. Section 2 presents the class of nonlinear systems considered and assumptions regarding system stability. In Section 3, the representation and the construction of RNNs both fully-connected and partially-connected, and LSTMs is presented. Section 4 starts with key definitions and lemmas, and then develop probabilistic generalization error upper bounds for partially-connected RNN models and LSTM networks. The integrating of a partially-connected RNN model and LSTM model into a MPC while accounting for Lyapunov stability considerations is proposed and discussed in Section 5. Lastly, the improvements associated with incorporating prior physical knowledge into RNN modeling is illustrated in Section 6 via both open-loop and closed-loop simulations using a two reactors in series chemical process under Lyapunov-based MPC.

2. Preliminaries

2.1. Notation

Given a vector $b \in \mathbb{R}^n$, its Euclidean norm is denoted by the operator $\|b\|$, and the weighted Euclidean norm of a vector is denoted by the operator $\|b\|_Q$ where Q is a positive definite matrix. Moreover, the infinity norm of b is given by $\|b\|_\infty$. Generally, for $b \in \mathbb{R}^n$ and $\gamma \geq 1$, $\|b\|_\gamma = (\sum_{i=1}^{n} |b_i|^\gamma)^{\frac{1}{\gamma}}$. Given a matrix $W \in \mathbb{R}^{m\times n}$, its Frobenius and spectral norms are denoted by $\|W\|_F$ and $\|W\|_\infty$, respectively. Given real numbers γ and κ, the κ-norm of the γ-norms of the columns of W is denoted by $\|W\|_{\gamma,\kappa} = (\sum_i^m (\sum_j^n |W_{j,i}|^\gamma)^{\frac{\kappa}{\gamma}})^{\frac{1}{\kappa}}$. $\mathbb{R}_+$ denotes non-negative real numbers. x^T denotes the transpose of x. The notation $L_f V(x)$ denotes the standard Lie derivative $L_f V(x) := \frac{\partial V(x)}{x} f(x)$. Set subtraction is denoted by '−', i.e., $A - B := \{x \in \mathbb{R}^n | x \in A, x \notin B\}$. A function $f(\cdot)$ is of class $\mathscr{C}^1$ if it is continuously differentiable. A continuous function α: $[0, a) \to [0, \infty)$ belongs to class $\mathscr{K}$ if it is strictly increasing and is zero only when evaluated at zero. A function $f: \mathbb{R}^n \to \mathbb{R}^m$ is said to be L_r-Lipschitz, $L_r \geqslant 0$, if $|f(a) - f(b)| \le L_r |a - b|$ for all $a, b \in \mathbb{R}^n$. $\mathbb{P}(A)$ denotes the probability that the event A will occur. $\mathbb{E}[X]$ denotes the expected value of a random variable X. We note that the infinity norm of a vector and the spectral norm of a matrix both have the notation $\|.\|_\infty$, by mathematical convention. Therefore, in order to differentiate between them, it is important to identify the argument of the norm.

2.2. *Class of systems*

We consider the class of multi-input multi-output (MIMO) nonlinear continuous-time systems represented by the following state-space form:

$$\dot{x} = \mathbf{F}(x, u) := F(x) + G(x)u \tag{1}$$

where the state vector of the system is $x = [x_1, \ldots, x_{n_x}]^T \in \mathbb{R}^{n_x}$, $y = [y_1, \ldots, y_{n_y}]^T \in \mathbb{R}^{n_y}$ is the output vector, and the manipulated input vector is $u = [u_1, \ldots, u_{n_u}]^T \in \mathbb{R}^{n_u}$. $\mathbf{F}(x, u)$ represents a nonlinear vector function of x and u. The constraints on control inputs are given by $u \in U := \{u_i^{\min} \le u_i \le u_i^{\max}\}$. $F(\cdot)$ and $G(\cdot)$ denote nonlinear vector and matrix functions of $n_x \times 1$ and $n_x \times n_u$ dimensions, respectively, and both are assumed to be sufficiently smooth.

2.3. *Stabilizability assumption*

We assume that there exists a control law $u = \Phi(x) \in U$ based on state feedback that can make the origin of the system of Eq. (1) exponentially stable. This stabilizability assumption implies the existence of a $\mathscr{C}^1$ control Lyapunov function denoted as $V(x)$, such that the following inequalities hold for all x in an open neighborhood D around the origin:

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

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

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

where c_i, $i = 1, 2, 3, 4$, are positive constants. A candidate controller $\Phi(x)$ may be constructed via Sontag's control law (Lin and Sontag, 1991). Then, following Wu et al. (2019), we characterize the closed-loop stability region Ω_ρ to be a level set of the Lyapunov function in the region D in which the time-derivative $\dot{V}(x)$ is negative under the controller $u = \Phi(x) \in U$ such that $\Omega_\rho := \{x \in D | V(x) \le \rho\}$, where $\rho > 0$. Furthermore, based on the Lipschitz property of $\mathbf{F}(x, u)$ and the boundedness of u, there exists positive constants M, L_x, L' such that the following inequalities hold for all $x, x' \in D$ and $u \in U$:

$$\|\mathbf{F}(x, u)\| \le M \tag{3a}$$

$$\|\mathbf{F}(x, u) - \mathbf{F}(x', u)\| \le L_x\|x - x'\| \tag{3b}$$

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

3. Recurrent neural networks (RNNs)

The prospect of utilizing artificial intelligence (AI) techniques in chemical engineering has been investigated at length in the recent literature. AI technology has led to the rise of classical and powerful modeling tools such as fuzzy logic in the 1960 s (Zadeh, 1968), expert systems in the 1980 s (Liao, 2005; Lee, 1990), and machine learning (ML) in the 1990 s (Vepa, 1993). Moreover, the implementation of ML techniques in the modeling of complex systems comes with a successful history in different chemical processes applications (Banerjee et al., 2017; Singh et al., 2017; Wong et al., 2018; Dias et al., 2017). For example, in Banerjee et al. (2017), an artificial neural network (ANN) model is developed for a bio-diesel production process. The ANN model provided an approximation of the percentage of fatty acid methyl ester yield within ± 8% deviation from the experimental data. Similarly, recurrent neural networks (RNN) have been broadly employed for modeling a general class of dynamical systems for control and state estimation purposes (Pan and Wang, 2011). In Singh et al. (2017), a RNN model of a continuous binary distillation column (BDC) was trained and validated using experimental data, and the study demonstrated that the predictive ability of the RNN model could outperform the first-principles model for the large-scale, complex, nonlinear process studied. This was attributed to the RNN possessing a high degree of freedom to solve the complex non-linear regression problem with the process data set.

RNN models are a powerful tool for modeling dynamic systems. Consider an RNN model that resembles the nonlinear dynamics of the system of Eq. (1) using m sequences of T-time-length data points $(\mathbf{x}_{i,t}, \mathbf{y}_{i,t})$, with $\mathbf{x}_{i,t} \in \mathbb{R}^{d_x}$ serving as the RNN input and $\mathbf{y}_{i,t} \in \mathbb{R}^{d_y}$ serving as the RNN output with $i = 1, \ldots, m$ and $t = 1, \ldots, T$. It is important to emphasize that the nonlinear system's inputs, states, and outputs in Eq. (1) are not always represented by the RNN inputs and outputs. Hence, all the vectors for RNN models are represented in boldface to distinguish them from those of the nonlinear system of Eq. (1).

Moreover, to make the discussion simpler, the RNN model of Eq. (4) and (5) is created to forecast states over a single sampling period with overall time steps $T = \Delta/h_c$ (i.e., within one sampling period Δ, the RNN model aims to predict states evolution for each integration time step h_c). Thus, the present manipulated inputs and state measurements that will be employed over $t = 1 \to T$ make up the RNN input $\mathbf{x}_{i,t}$, while the predicted states over $t = 1 \to T$ make up the RNN output $\mathbf{y}_{i,t}$. Owing to the sample-and-hold execution of manipulated inputs, $\mathbf{x}_{i,t}$ does not change over $t = 1 \to T$. The data set is created of m data sequences that were individually selected from an underlying distribution over $\mathbb{R}^{d_x\times T} \times \mathbb{R}^{d_y\times T}$. To simplify the theoretical development, we consider a single-hidden-layer RNN model with the following form to approximate the nonlinear dynamics of Eq. (1):

$$\mathbf{h}_t = \sigma_h(U\mathbf{h}_{t-1} + W\mathbf{x}_t) \tag{4}$$

$$\mathbf{y}_t = \sigma_y(V\mathbf{h}_t) \tag{5}$$

where $\mathbf{h}_t$ denotes the hidden state, and W, U, and V are the weight matrices connecting different layers. The possibly nonlinear activation functions used are denoted by σ_h and σ_y. Specifically, σ_h is often chosen to be a nonlinear activation function that may take different forms (e.g, *tanh* or *ReLU*), while σ_y typically uses a linear element-wise activation function for regression problems. Without loss of generality, we have the following assumptions for the development of RNN models:

Assumption 1. The input data is bounded, i.e., $\|x_{i,t}\|\le B_x$ for all $i = 1, \ldots, m$ and $t = 1, \ldots, T$.

Assumption 2. The Frobenius norms of the weight matrices are bounded, i.e., $\|W\|_F\le B_{W,F}$, $\|Q\|_F\le B_{V,F}$, $\|U\|_F\le B_{U,F}$.

Assumption 3. The training, validation, and testing data sets are drawn from the same distribution.

Assumption 4. σ_h is a 1-Lipschitz continuous activation function, where, a function $f: \mathbb{R}^n \to \mathbb{R}^m$ is said to be L-Lipschitz, $L \geqslant 0$, if $|f(a) - f(b)|\le L|a - b|$ for all $a, b \in \mathbb{R}^n$. Additionally, σ_h is a positive-homogeneous function in the sense that $\sigma_h(\alpha z) = \alpha\sigma_h(z)$ holds for all $\alpha \ge 0$ and $z \in \mathbb{R}$.

Furthermore, consider a hypothesis class $\mathscr{H}$ of RNN models $h(\cdot)$ that map a d_x-dimensional input $\mathbf{x} \in \mathbb{R}^{d_x}$ to a d_y-dimensional output $\mathbf{y} \in \mathbb{R}^{d_y}$. The predicted output of the RNN model and the loss function are denoted by $\mathbf{y}_t = h(\mathbf{x}_t)$ and $L(\mathbf{y}_t, \tilde{\mathbf{y}}_t)$, respectively, where $L(\mathbf{y}, \tilde{\mathbf{y}})$ calculates the squared difference between the predicted output $\mathbf{y}$ and the true output $\tilde{\mathbf{y}}$.

3.1. *Physics-informed RNNs*

From a modeling point of view, even cutting-edge black-box ML models (e.g., dense fully-connected RNN models) have had only limited success when applied in scientific domains (Karpatne et al., 2017) due to such models' large data size needs, failure to yield physically consistent outputs, and lack of generalizability to unseen samples. Researchers have begun to investigate the continuum between mechanistic and ML models, in which both scientific knowledge and data are integrated in a synergistic way. This is because neither a pure ML algorithm nor solely scientific theory may be sufficient for complex scientific and engineering applications (e.g., Alber et al., 2019; Baker et al., 2019; Rai and Sahu, 2020). A physics-based machine learning paradigm uses domain-specific knowledge, but in supporting roles such as feature engineering or post-processing, in a way fundamentally different than dominant approaches in the ML field. On the other hand, the concept of combining scientific principles and ML in developing models has only recently gained popularity (Karpatne et al., 2017), when there has already been a substantial amount of research done on the subject. This research direction is being conducted in various disciplines including earth systems (Reichstein et al., 2019), climatology (Krasnopolsky and Fox-Rabinovitz, 2006; O'Gorman and Dwyer, 2018), material exploration (Cang et al., 2018; Schleder et al., 2019), quantum chemistry (Schütt et al., 2017; Chakraborty et al., 2014), biological sciences (Yazdani et al., 2020), and hydrology (Xu and Valocchi, 2015). Early findings in isolated and straightforward scenarios have been encouraging, and expectations are growing that this paradigm will speed up scientific advancement and aid in resolving some of the global challenges with regards to the environment (Faghmous and Kumar, 2014), healthcare (Wang et al., 2020a), and food and nutrition security (Jia et al., 2019).

Similarly, in process systems engineering, the traditional paradigm of developing numerical approaches to approximate solutions is based solely on physics—numerical differentiation and integration algorithms are used to solve systems of differential equations that reflect established physical principles through space and time (Butcher, 1996; Sagaut et al., 2013; Houska et al., 2012). A different approach is to look for simplified models that can roughly characterize the dynamics of the underlying systems, such as the Euler equations for gas dynamics and the Reynolds-averaged Navier-Stokes equations for turbulent flows (Chaouat, 2017; Tompson et al., 2017). However, creating a simplified model that accurately captures a complex phenomenon is quite difficult. More importantly, only a portion of the dynamics of many complicated real-world processes may be captured by a simple model. The equations may not accurately reflect the original system's states. On the other hand, numerous recent studies, from turbulence to reaction modeling, have demonstrated that ML-based models can produce realistic predictions and greatly speed up the simulation of complex dynamics compared to numerical solvers (Wang et al., 2020b; Kochkov et al., 2021; Luo et al., 2022). However, ML-based models are dense and purely data-driven by nature, which has many limitations. Without strict boundaries, ML-based models are likely to provide predictions that defy the fundamental principles governing physical systems. Furthermore, machine learning models frequently experience difficulties with generalization, i.e., models trained on a single data set cannot adequately adapt to unseen scenarios. Hence, approximating complicated dynamical systems in scientific applications cannot be considered to be a problem that can be easily solved by either machine-learning-based models or physics-based theory alone. There is, therefore, a significant benefit in integrating machine learning models with conventional physics-based methodologies, through which we can maximize the benefits of both techniques.

The investigation of more structured system modeling is driven by a variety of factors. In contrast to a system with structured local connectivity, fully-connected systems necessitate long-range connections, and have slower communication times between neurons. Real-world problems may have local correlations as well. It would be considerably easier and take up less memory to construct networks with organized neighborhoods than a fully-connected network (Canning and Gardner, 1988). Typically, when creating a dynamic model for a general nonlinear process, a neural network model that uses all available process inputs to predict the desired output is preferred. Creating a fully-connected, black-box dynamic model for these processes is relatively simple with open-source machine learning tools, and such a model would attempt to capture any connections that might exist between each input and each output of the underlying process. As depicted in Fig. 1, at least three layers (i.e., an input layer, hidden layers, and an output layer) make up the general structure of a fully-connected RNN. For such reasons,

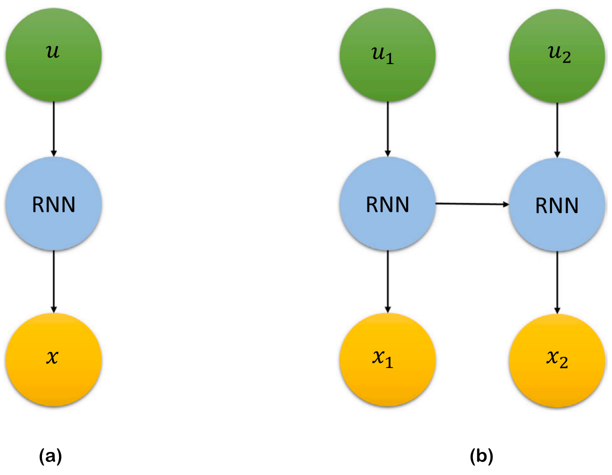

Fig. 1 – Structure of (a) standard fully-connected and (b) partially-connected RNN.

fully-connected RNN models are frequently the best option for modeling processes where no prior knowledge is available.

Although standard RNNs do not consider any domain-specific knowledge in the model development phase and generally use fully-connected layers to capture input-output relationship using the given training dataset, it has been demonstrated in Wu et al. (2020) that a priori process structural knowledge can be utilized to improve an RNN's performance by using a partially-connected architecture. Fig. 1 shows the difference between fully-connected and partially-connected RNNs, from which it can be observed that the connection between some neurons is removed in a partially-connected structure to resemble the underlying input-output relationship based on a priori process structural knowledge. Partially-connected RNNs can be used to model a multiple-unit process in which upstream units affect downstream units, but not in the opposite direction. For example, consider the nonlinear system of Eq. (1) for which the input vector u_1 affects only the state x_1, but both u_1 and u_2 affect the state x_2, where $x = [x_1 \in \mathbb{R}^{n_{x_1}}, x_2 \in \mathbb{R}^{n_{x_2}}]$ and $u = [u_1 \in \mathbb{R}^{n_{u_1}}, u_2 \in \mathbb{R}^{n_{u_2}}] \in \mathbb{R}^{n_u}$ with $n_{u_1} + n_{u_2} = n_u$ and $n_{x_1} + n_{x_2} = n_x$. Wu et al. (2020) demonstrates that, by using a partially-connected architecture, the number of weight parameters can be significantly reduced to achieve the desired model accuracy compared to a fully-connected RNN model. Additionally, in Alhajeri et al. (2022b), an Aspen simulation study of two CSTRs in series was carried out to demonstrate that the MPC using partially-connected RNN models achieved better closed-loop performances with a reduced computational time. To better understand the benefits of partially-connected RNNs in terms of higher modeling accuracy, a theoretical analysis of generalization error needs to be carried out.

3.2. *Long short-term memory RNN*

In this subsection, we present the long short-term memory (LSTM) network. LSTM is a variant of RNNs that has been widely used to make predictions based on time series data. Although standard RNNs have proven to be efficient in many engineering applications, RNNs struggle to deal with long time dependencies. Based on the structure of RNNs, as their hidden states are only propagated forward in time, they cannot receive future input data to predict the current state. As a result, the vanishing gradient phenomena is often encountered when training standard RNNs. As we proceed backwards through the layers of the network, the vanishing gradient problem occurs due to having exponentially decaying gradients in the loss function, which makes it harder to train the network and more challenging to retain information over longer time periods. Given these limitations, LSTM networks, with a well-known and unique structure, were introduced in 1997 (Hochreiter and Schmidhuber, 1997). Furthermore, in LSTMs, the concept of gates were introduced to keep track of how much useful history should be passed between the LSTM units. More specifically, the input gate, forget gate, and output gate control how much memory is to be stored in the cell state and retained throughout the network (Schmidhuber, 2015).

The LSTM is composed of several gates, and is formulated by the following equations:

$$f_t = \sigma_l(W_f \mathbf{x}_t + U_f \bar{\mathbf{h}}_{t-1}) \tag{6a}$$

$$r_t = \sigma_l(W_r \mathbf{x}_t + U_r \bar{\mathbf{h}}_{t-1}) \tag{6b}$$

$$o_t = \sigma(W_o \mathbf{x}_t + U_o \bar{\mathbf{h}}_{t-1}) \tag{6c}$$

$$\tilde{c}_t = \tanh(W_c \mathbf{x}_t + U_c \bar{\mathbf{h}}_{t-1}) \tag{6d}$$

$$c_t = f_t \odot c_{t-1} + r_t \odot \tilde{c}_t \tag{6e}$$

$$\bar{\mathbf{h}}_t = o_t \odot \tanh(c_t) \tag{6f}$$

where W_f, W_r, W_o, $W_c \in \mathbb{R}^{d_h \times d_x}$ are the weights associated with the inputs, and U_f, U_r, U_o, $U_c \in \mathbb{R}^{d_h \times d_h}$ are the weights associated with the hidden states. r_t, f_t, $o_t \in \mathbb{R}^{d_h}$ represent the input, forget, and output gates, respectively. $c_t \in \mathbb{R}^{d_h}$ is the cell state, and $\tilde{c}_t$ is the cell integrated with the input gate. σ_l is the nonlinear activation function *sigmoid*, and *tanh* is the hyperbolic tangent function. The output at time t is $\bar{\mathbf{y}}_t = \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_t)$. We note that the memory cell state c_t is one of the most important parts in LSTMs as it is the part that stores and carries the essential information for long-term and short-term dependencies. This information is then passed to the subsequent LSTM units, and is recursively updated through the other remaining gates (i.e., r_t, f_t, o_t). Specifically, the memory cell state, presented in Eq. (6e), consists of two terms: the first term expresses the quantity of old information to be discarded from the previous c_t, while the second term expresses the essential new information that is introduced to the memory cell state c_t (Ren et al., 2022). Fig. 2 shows the schematic of an LSTM cell with all its gates. In addition to the previous assumption, we also consider the following for LSTM models:

Assumption 5. The norms of weight matrices are bounded as follows:

$$\|W_f\|_F \le B_{W_f},\ \|W_r\|_F \le B_{W_r},\ \|W_o\|_F \le B_{W_o},\ \|W_c\|_F \le B_{W_c},\ \|U_g\|_F \le B_{U_g},$$

$$\|U_r\|_F \le B_{U_r},\ \|U_o\|_F \le B_{U_o},\ \|U_{\bar{h}}\|_F \le B_{U_{\bar{h}}},\ \|Z\|_{1,\infty} \le B_Z$$

Assumption 6. The nonlinear activation functions σ, $\sigma_{\bar{y}}$ are 1-Lipschitz continuous, and $\sigma(0) = \sigma_{\bar{y}}(0) = 0$.

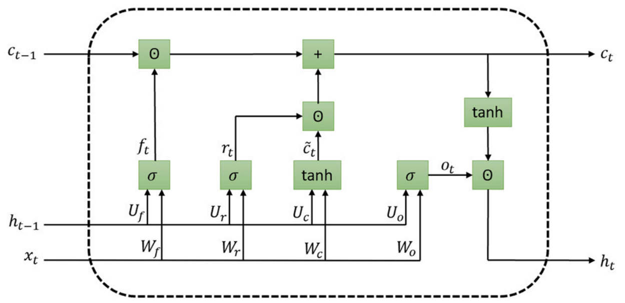

Fig. 2 – Schematic of an LSTM cell structure.

4. Generalization error

4.1. *General considerations*

Generalizability or generalization error is a metric that measures a machine learning model's ability to adapt to new, previously unseen data that is drawn from the same distribution as the one used to train the model. Several investigations were conducted for the interpretation and improvement of generalization of different machine learning-based models (e.g., Roelofs, 2019; Emmert-Streib and Dehmer, 2019). Furthermore, a theoretical analysis of the generalization error is of significant importance as it provides a fundamental understanding on how well the model performs on unseen data that will be collected in real-world systems. This section will provide derivations of the generalization error for RNNs using statistical learning theory. Before we present the results on generalization error bounds, we first introduce the necessary definitions of generalization error.

Definition 1. A centered random variable $x \in \mathbb{R}$ is said to be sub-Gaussian with variance proxy σ^2 if $\mathbb{E}[x] = 0$ and the moment generating function satisfies:

$$\mathbb{E}[\exp(aX)] \leq \exp\left(\frac{a^2\sigma^2}{2}\right), \quad \forall\, a \in \mathbb{R} \tag{7}$$

Definition 2. Given a data distribution, D, and a function, h, that predicts y (output) based on x (input), the generalization error is given by

$$\mathbb{E}[L(h(x), y)] = \int_{X\times Y} L(h(x), y)\,\rho(x, y)\, dxdy. \tag{8}$$

where $\rho(x, y)$ denotes the joint probability distribution for x and y, and Y and X represent the vector space for all possible outputs and inputs, respectively.

Definition 3. $L(\,\cdot\,,\,\cdot\,)$ is the loss function (e.g., mean squared error (MSE) for regression problems). Since the distribution may be unknown, the following empirical error is often used as an approximation measure for the generalization error:

$$\hat{\mathbb{E}}_S[L(h(x), y)] = \frac{1}{m}\sum_{i=1}^{m} L(h(x_i), y_i) \tag{9}$$

where $S = (s_1, \ldots, s_m)$, $s_i = (x_i, y_i)$ includes m data samples drawn from the data distribution D.

Definition 4. Given a set of data samples $S = \{s_1, \ldots, s_m\}$, and a hypothesis class $\mathscr{F}$ of real-valued functions, the definition of the empirical Rademacher complexity of $\mathscr{F}$ is

$$\mathscr{R}_S(\mathscr{F}) = \mathbb{E}_\epsilon\left[\sup_{f\in\mathscr{F}} \frac{1}{m}\sum_{i=1}^{m} \epsilon_i f(s_i)\right] \tag{10}$$

where $\epsilon = (\epsilon_1, \ldots, \epsilon_m)^T$, and ϵ_i are Rademacher random variables that are independent and identically distributed (i.i.d.) and satisfy $\mathbb{P}(\epsilon_i = -1) = \mathbb{P}(\epsilon_i = 1) = 0.5$.

The following lemma gives the generalization error bound for a general class of RNN models.

Lemma 1. Consider a hypothesis class $\mathscr{H}$ of vector-valued functions $h \in \mathbb{R}^{d_y}$ and a set of data samples $S = \{s_1, \ldots, s_m\}$. Let $L(.)$ be a L_r -Lipschitz function mapping $h \in \mathbb{R}^{d_y}$ to R. Then,

$$\mathbb{E}_\epsilon\left[\sup_{f\in\mathscr{F}} \sum_{i=1}^{m} \epsilon_i L(h(\mathbf{x}_i), \mathbf{y}_i)\right] \leq \sqrt{2}L_r\mathbb{E}_\epsilon\left[\sup_{h\in\mathscr{H}} \sum_{i=1}^{m}\sum_{k=1}^{d_y} \epsilon_{ik} h(\mathbf{x}_i)\right] \tag{11}$$

where $h_k(.)$ is the kth component in the vector-valued function $h(.)$, and ϵ_{ik} is an $m \times d_y$ matrix of independent Rademacher variables. In the following text, we will omit the subscript ϵ of the expectation operator for simplicity.

Since the right-hand side of the previous inequality is generally difficult to compute, we can reduce it to scalar classes and derive the following bound:

$$\mathbb{E}\left[\sup_{h\in\mathscr{H}} \sum_{i=1}^{m}\sum_{k=1}^{d_y} \epsilon_{ik} h(\mathbf{x}_i)\right] \leq \sum_{k=1}^{d_y} \mathbb{E}\left[\sup_{h\in\mathscr{H}} \sum_{i=1}^{m} \epsilon_i h(\mathbf{x}_i)\right] \tag{12}$$

where $\mathscr{H}_k$, $k = 1, \ldots, d_y$ are classes of scalar-valued functions that correspond to the components of vector-valued functions in $\mathscr{H}$. The previous inequality will later be used in the derivation of the generalization error bound for LSTM models.

Lemma 2. (c.f. Theorem 3.3 in Mohri et al. (2018)) Let $\mathscr{H}$ be the hypothesis class of ML models that map $\{\mathbf{x}_1, \ldots, \mathbf{x}_t\} \in \mathbb{R}^{d_x \times t}$ (i.e., the first t time step inputs) to $\mathbf{y}_t \in \mathbb{R}^{d_y}$ (i.e., the tth output) and $\mathscr{G}_t$ be the loss function set with $\mathscr{H}$,

$$\mathscr{G}_t = \{g_t: (\mathbf{x}, \tilde{\mathbf{y}}) \to L(h(\mathbf{x}), \tilde{\mathbf{y}}),\, h \in \mathscr{H}\} \tag{13}$$

where $\tilde{\mathbf{y}}$ and $\mathbf{x}$ are the true output vector and the input vector of the ML model, respectively. Then, given a data set consisting of m i.i.d. data samples, the inequality below holds in probability for all $g_t \in \mathscr{G}_t$ over the data samples $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, \ldots, m$:

$$\mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i) + 2\mathscr{R}_S(\mathscr{G}_t) + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2m}} \tag{14}$$

Eq. (14) demonstrates that the upper bound for the generalization error depends on the training error (first term), the Rademacher complexity of $\mathscr{G}_t$ (second term), and a function of the samples size m and the confidence δ. Therefore, to derive a generalization error bound for RNN models, an upper bound for the Rademacher complexity of RNN hypotheses needs to be developed.

Lemma 3. Given a hypothesis class $\mathscr{H}_k$ of real-valued functions corresponding to the kth component of the vector-valued function class $\mathscr{H}$ and a set of m i.i.d. data samples $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, \ldots, m$, the following inequality holds for the scaled empirical Rademacher complexity:

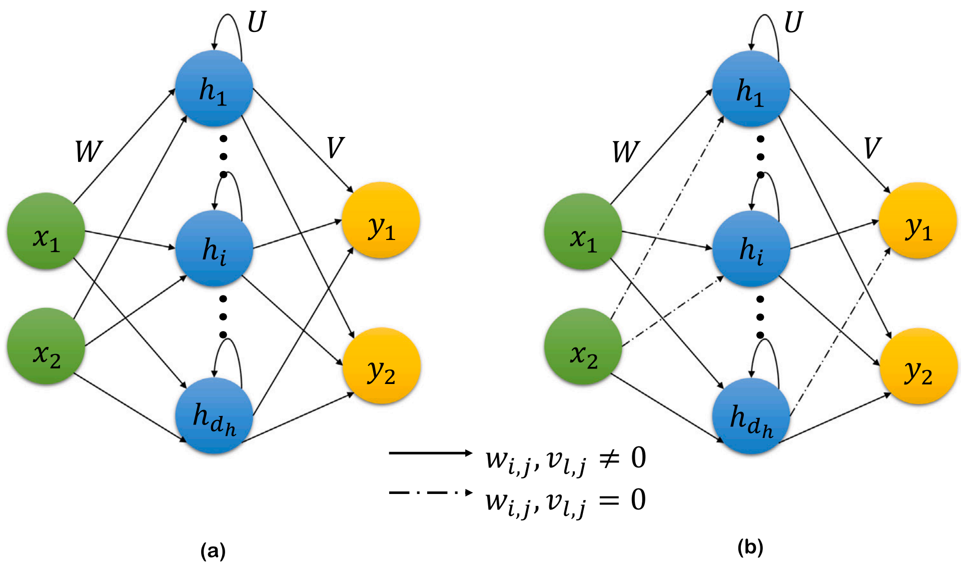

Fig. 3 – Weights and connections in (a) standard fully-connected and (b) partially-connected RNN structures, where zeroed weights for links between units are represented by dashed lines.

$$\begin{aligned} m\mathcal{R}_S(\mathcal{H}_k) &= \mathbb{E}\left[\sup_{h\in\mathcal{H}_k}\sum_{i=1}^{m}\epsilon_i h(\mathbf{x}_i)\right]. \\ &= \frac{1}{\lambda}\log\exp\left(\lambda\mathbb{E}\left[\sup_{h\in\mathcal{H}_k}\sum_{i=1}^{m}\epsilon_i h(\mathbf{x}_i)\right]\right) \\ &\le \frac{1}{\lambda}\log\mathbb{E}\left[\sup_{h\in\mathcal{H}_k}\exp\left(\lambda\sum_{i=1}^{m}\epsilon_i h(\mathbf{x}_i)\right)\right] \end{aligned} \tag{15}$$

where $\lambda > 0$ is an arbitrary parameter.

Lemma 4. Let $\mathcal{H}_{k,t}$, $k = 1, \ldots, d_y$ be the class of real-valued functions that corresponds to the k^{th} component of the RNN output at t^{th} time step, with weight matrices and activation functions satisfying Assumptions 1–4. Given a set of m i.i.d. data samples $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, \ldots, m$, the following equation holds for the Rademacher complexity:

$$\mathcal{R}_S(\mathcal{H}_{k,t}) \le \frac{M(\sqrt{2\log(2)t}+1)B_X}{\sqrt{m}} \tag{16}$$

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

Lemma 5. (c.f. Theorem 1 in Wu et al. (2021)) Given a dataset $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$ with i.i.d. data samples, $i = 1, \ldots, m$, and the L_r-Lipschitz loss function class $\mathcal{G}_t$ associated with the RNN function class $\mathcal{H}_t$ that predicts outputs at the t^{th} time step, with probability at least $1-\delta$ over S, the following inequality holds for the RNN models:

$$\begin{aligned} \mathbb{E}[g_t(\mathbf{x}, \mathbf{y})] &\le \frac{1}{m}\textstyle\sum_{i=1}^{m} g_t(\mathbf{x}_i, \mathbf{y}_i) + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2m}} \\ &+ \mathcal{O}\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \end{aligned} \tag{17}$$

The above equation represents the theoretical generalization error's upper bound for RNN models. This theory will be utilized to find a relationship between a RNN model and its structure in Section 4.2.

4.2. *Physics-based RNNs generalization bound*

In a partially-connected structure, the connections between inputs and outputs should be carefully designed to reflect a *priori* physical knowledge. In particular, as illustrated in Fig. 3, x_2 does not affect y_1, so the weights corresponding to the linkages between x_2 and y_1 (dashed lines in Fig. 3) are assigned a value of zero (i.e., $w_{i,j} = v_{l,j} = 0$). This structure superiority in accuracy and model identification to dense fully-connected RNNs has been demonstrated through several works. Hence, we develop the following theory to interpreter this observation.

Theorem 1. Consider the following inequalities: $\mathbb{E}_{\nu}[g_t(\mathbf{x}, \mathbf{y})] \le \nu$ and $\mathbb{E}_{\hat{\nu}}[g_t(\mathbf{x}, \mathbf{y})] \le \hat{\nu}$, where ν and $\hat{\nu}$ represent the generalization error bound for a fully-connected RNN model and a partially-connected RNN model, respectively. Given that both models are constructed with the same hyperparameters and trained over the same i.i.d data set with m samples, the following inequality holds:

$$\hat{\nu} < \nu \tag{18}$$

Proof. If we let ν denote the right-hand side of Eq. (17), ν can be represented as the sum of the three terms in the right-hand side of Eq. (17) i.e., $\nu = \nu_I + \nu_{II} + \nu_{III}$, where the subscripts I, II, and III are the term indices, and the same applies for $\hat{\nu}$ i.e., $\mathbf{E}_{\hat{\nu}}[g_t(\mathbf{x}, \mathbf{y})] \le \hat{\nu} = \hat{\nu}_I + \hat{\nu}_{II} + \hat{\nu}_{III}$. Then, with respect to the first terms, $\hat{\nu}_I$ and ν_I, they depend on the sizes of both the training data set and the hypothesis class $\mathcal{H}$. Due to the dense structure of FCRNN models, the size of the hypothesis class $\mathcal{H}$ will be larger, which leads to a higher probability of convergence to the optimal hypothesis h^*. On the contrary, by incorporating physical knowledge into the RNN modeling by assigning some weight entries to be zero, the size of the hypothesis class $\mathcal{H}$ is reduced, yet the model can be closer to the optimal hypothesis h^* for the data distribution D. Thus, both models will have close values for the first term (i.e., $\nu_I \approx \hat{\nu}_I$). Additionally, since we are developing the two models using the same data set with m samples, the second terms for both the PCRNN model and the FCRNN model are approximately equal (i.e., $\nu_{II} \approx \hat{\nu}_{II}$). Therefore, we are left to investigate the third term, which is given by:

$$\nu_{III} = \mathcal{O}\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{19a}$$

$$\hat{\nu}_{III} = \mathcal{O}\left(L_r d_y \frac{\hat{M}B_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right) \tag{19b}$$

where $M = B_{V,F}B_{W,F}$, and $\hat{M} = B_{\hat{V},F}B_{\hat{W},F}$.

Note that M and $\hat{M}$ are products of the RNN weight matrix bounds in Eq. (17), where the PCRNN model weight matrices are denoted by the symbols $\hat{V}$ and $\hat{W}$, and their Frobenius norm bounds are $B_{\hat{V},F}$ and $B_{\hat{W},F}$, respectively. After training both FCRNN and PCRNN models with the same random initialization and optimization algorithm, the weight matrices in the PCRNN model will have some zero entries, while the other entries (i.e., the nonzero ones) would be numerically close for both models. Since the Frobenius norm of matrix A is expressed as the square root of the matrix trace of $AA^{(H)}$, where $A^{(H)}$ is the conjugate transpose, more zero entries in the weight matrices will yield lower bounds on their Frobenius norms i.e.,

$$B_{\hat{W},F} < B_{W,F} \tag{20a}$$

$$B_{\hat{V},F} < B_{V,F} \tag{20b}$$

which yields

$$\hat{\nu}_{III} < \nu_{III} \tag{20c}$$

Hence, this proves that the partially-connected RNN modeling approach provides a lower generalization error bound than the dense fully-connected RNN architecture. □.

Remark 1. By incorporating process structural knowledge into the development of partially-connected RNN models, the complexity of RNN hypothesis class is reduced compared to fully-connected RNNs, which leads to a tighter bound on the Rademacher complexity. Additionally, by revealing the correct direction for RNNs to find the optimal weight parameters, the training error (the first term in Eq. (17)) is more likely to be minimized using the same hyperparameters (i.e., the number of layers and neurons) and the same training set of m i.i.d. data samples.

4.3. *LSTM generalization error*

LSTM networks structure and complexity are different from standard fully-connected RNN models as well as partially-connected RNNs. Hence, a generalizability theoretical framework of LSTMs is discussed in this subsection. We recall the fundamental definitions and lemmas from the Section 4.1. In addition we present the following theoretical bases (i.e., remarks and lemmas) needed for the development of a LSTM network's generalization accuracy bound.

Remark 2. From the definitions of norms, if given a vector $b \in \mathbb{R}^n$, the following inequalities hold:

$$\|b\|_\infty = \max\{|b_i|\} \le \sqrt{\textstyle\sum_i^n |b_i|^2} = \|b\| \tag{21}$$

$$\|b\|_\infty = \max\{|b_i|\} \le \textstyle\sum_i^n |b_i| = \|b\|_1 \tag{22}$$

Lemma 6. Given a hypothesis class $\mathscr{H}$ of vector-valued functions that map the LSTM inputs $\mathbf{x} \in \mathbb{R}^{d_x}$ to the hidden states $\bar{\mathbf{h}} \in \mathbb{R}^{d_h}$, and any convex and monotonically increasing function p: $R \to [0, \infty)$, the following inequality holds for the LSTM model of Eq. (6) with a 1-Lipschitz activation function $\sigma_y(0) = 0$, applied element-wise:

$$\mathbb{E}\left[\sup_{\bar{h}\in\mathscr{H}, \|Z\|_{1,\infty}\le B_Z} p\left(\lambda \left\|\textstyle\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_i)\right\|_\infty\right)\right] \le 2 \cdot \mathbb{E}\left[\sup_{\bar{h}\in\mathscr{H}} p\left(B_Z \left\|\textstyle\sum_{i=1}^m \epsilon_i \bar{\mathbf{h}}_i\right\|_\infty\right)\right] \tag{23}$$

where $\|Z\|_{1,\infty}$ is the maximal 1-norm of its rows.

Proof. Based on the previous works of Golowich et al. (2018), Wu et al. (2021), we proceed with this proof as follows:

$$\begin{aligned} &\mathbb{E}\sup_{\|Z\|_{1,\infty}\le B_Z} p\left(\left\|\textstyle\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_i)\right\|_\infty\right) \\ =\ &\mathbb{E}\sup_{\|z_k\|_1\le B_Z} \max_j p\left(\left|\textstyle\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(z_j^T\bar{\mathbf{h}}_i)\right|\right) \\ =\ &\mathbb{E}\sup_{\|z\|_1\le B_Z} p\left(\left|\textstyle\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(z^T\bar{\mathbf{h}}_i)\right|\right) \end{aligned} \tag{24}$$

where z_k is the k-th row of the matrix Z. Since p is a convex monotonically increasing function, $p(|x|) \le p(x) + p(-x)$, and hence, Eq. (24) can be bounded by:

$$\begin{aligned} &\mathbb{E}\sup_{\|Z\|_{1,\infty}\le B_Z} p\left(\left\|\textstyle\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_i)\right\|_\infty\right) \\ \le\ &\mathbb{E}\sup_{\|z\|_1\le B_Z} p\left(\textstyle\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(z^T\bar{\mathbf{h}}_i)\right) + \mathbb{E}\sup_{\|z\|_1\le B_Z} p\left(-\textstyle\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(z^T\bar{\mathbf{h}}_i)\right) \end{aligned} \tag{25}$$

Note that ϵ_i follows a symmetric distribution, i.e. $\mathbb{P}(\epsilon_i = 1) = \mathbb{P}(\epsilon_i = -1) = 0.5$. Hence, Eq. (25) becomes:

$$\begin{aligned} &\mathbb{E}\sup_{\|Z\|_{1,\infty}\le B_Z} p\left(\left\|\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_i)\right\|_\infty\right) \\ =\ &2\mathbb{E}\sup_{\|z\|_1=B_Z} p\left(\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(z^T\bar{\mathbf{h}}_i)\right) \\ \le\ &2\mathbb{E}\sup_{\|z\|_1=B_Z} p\left(\sum_{i=1}^m (\epsilon_i z^T\bar{\mathbf{h}}_i)\right) \\ \le\ &2\mathbb{E}\sup_{\|z\|_1=B_Z} p\left(\|z\|_\infty \left\|\sum_{i=1}^m (\epsilon_i\bar{\mathbf{h}}_i)\right\|_\infty\right) \end{aligned} \tag{26}$$

Using Eq. (22) in Remark 2, the inequality in Eq. (26) is bounded as follows:

$$\begin{aligned} &\mathbb{E}\sup_{\|Z\|_{1,\infty}\le B_Z} p\left(\left\|\sum_{i=1}^m \epsilon_i \sigma_{\bar{y}}(Z\bar{\mathbf{h}}_i)\right\|_\infty\right) \\ \le\ &2\mathbb{E}\sup_{\|z\|_1=B_Z} p\left(\|z\|_1 \left\|\sum_{i=1}^m (\epsilon_i\bar{\mathbf{h}}_i)\right\|_\infty\right) \\ \le\ &2\mathbb{E}\sup p\left(B_Z \left\|\sum_{i=1}^m (\epsilon_i\bar{\mathbf{h}}_i)\right\|_\infty\right) \end{aligned} \tag{27}$$

This completes the proof of Lemma 6, where it "peels off" the matrix Z between the LSTM hidden layer and the output layer. □.

Remark 3. It is well established that the activation functions *sigmoid* and tanh are essential for the development of different types of neural networks, including LSTMs. This is due to the LSTM network's special structure and the importance of the gating functionality performed by *sigmoid* functions. Therefore, the purpose of utilizing the infinity-norm, which is based on the peeling strategy as in Lemma 6, is to eliminate the requirement of the positive homogeneity property in activation functions. Hence, this lemma suits neural networks requiring non-positive and non-homogeneous activation functions.

Remark 4. In LSTMs, signals are categorized into two main types. The first type are propagating signals, which are $\bar{\mathbf{h}}_t$ and c_t. These signals connect the LSTM cells in sequential time steps with each other. In other words, the LSTM unit at a certain time step t, receives as an input the hidden state $\bar{\mathbf{h}}_{t-1}$ and the cell state c_{t-1}, both from the previous time step $t-1$, in addition to $\mathbf{x}_t$, which is the input vector at time t. The second type of signals are gating signals, represented by o_t, f_t, and r_t, which are responsible for the flow of information inside the LSTM unit. Their outputs range from 0 to 1, which means they determine how much of the information is passed. In the extreme cases, if the output is 0, this means no information is passed, and if the output is 1, this means all the information is passed. Taking this into consideration, these gating signals can be upper-bounded by 1, for simplicity in the generalization error bound proof.

Lemma 7. Let $\mathscr{H}_{k,t}$, $k = 1, \ldots, d_y$ be the class of real-valued functions that corresponds to the k^{th} component of the LSTM output at the t^{th} time step, with weight matrices and activation functions satisfying Assumptions 1–6. Given a set of m i.i.d. data samples $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^T$, $i = 1, \ldots, m$, the following equation holds for the Rademacher complexity:

$$m\mathscr{R}_s(\mathscr{H}_{k,t}) \le \left(\frac{\sqrt{2}+1}{\sqrt{2}}\right)\bar{M}\sqrt{m} \tag{28}$$

where $\bar{M} = B_V B_{W_c} B_x \frac{1-\beta^t}{1-\beta}$ and $\beta = 1 + B_{U_c}$.

Proof. Recalling that z_k is the k^{th} row of the weight matrix Z and by using Eq. (16) in lemma 3, we obtain the following bound for the Rademacher complexity $m\mathscr{R}_s(\mathscr{H}_{k,t})$:

$$\begin{aligned} m\mathscr{R}_s(\mathscr{H}_{k,t}) &= \mathbb{E}\left[\sup \sum_{i=1}^{m} \epsilon_i \sigma_{\bar{y}}(z_k \bar{\mathbf{h}}_{i,t})\right] \\ &\le \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(\lambda\sum_{i=1}^{m}\epsilon_i\sigma_{\bar{y}}(z_k\bar{\mathbf{h}}_{i,t})\right)\right] \end{aligned} \tag{29}$$

Now, by applying the peeling strategy of Eq. (23), we can further bound the previous inequality as follows:

$$\begin{aligned} m\mathscr{R}_s(\mathscr{H}_{k,t}) &\le \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\left\|\sum_{i=1}^{m}\epsilon_i\bar{\mathbf{h}}_{i,t}\right\|_\infty\right)\right] \\ &\le \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\bar{\mathbf{h}}_{i,t}\|_\infty\right)\right] \\ &= \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|_\infty\|\bar{\mathbf{h}}_{i,t}\|_\infty\right)\right] \end{aligned} \tag{30}$$

Using Eq. (6f) from the LSTM equations and applying Eq. (21) from Remark 2, we expand the propagation signal $\bar{\mathbf{h}}_{i,t}$ and get the following bound:

$$\begin{aligned} m\mathscr{R}_s(\mathscr{H}_{k,t}) &= \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|\|o_{i,t}\|_\infty\|\tanh(c_{i,t})\|\right)\right] \\ &\le \frac{1}{\lambda}\log\mathbb{E}\left[\sup\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|\|o_{i,t}\|_\infty\|c_{i,t}\|\right)\right] \end{aligned} \tag{31}$$

Following Remark 4, we can bound the gating signal $o_{i,t}$ as follows: $\|o_{i,t}\|_\infty \le 1$. Also, we apply Eq. (21) from Remark 2 and further expand the propagation signal $c_{i,t}$ using Eq. (6e) based on the LSTM structure, thereby obtaining the following inequality:

$$\begin{aligned} m\mathscr{R}_s(\mathscr{H}_{k,t}) \le \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|(\|f_{i,t}\|_\infty\|c_{i,t-1}\| \\ + \|r_{i,t}\|_\infty\|\tilde{c}_{i,t}\|)\Bigg)\Bigg] \end{aligned} \tag{32}$$

We further expand $\tilde{c}_{i,t}$ using Eq. (6d) and Remark 2. Subsequently, we expand $\bar{\mathbf{h}}_{i,t-1}$ using Eq. (6f). Moreover, using the upper bound of the input data and the upper bounds of the weight matrices mentioned in assumption 1 and assumption 5, respectively, the following bound is obtained:

$$\begin{aligned} m\mathscr{R}_s(\mathscr{H}_{k,t}) &\le \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|(\|f_{i,t}\|_\infty\|c_{i,t-1}\| \\ &\quad + \|r_{i,t}\|_\infty\|\tanh(W_c x_{i,t} + U_c\bar{\mathbf{h}}_{i,t-1})\|)\Bigg)\Bigg] \\ &\le \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|(\|f_{i,t}\|_\infty\|c_{i,t-1}\| \\ &\quad + \|r_{i,t}\|_\infty\|W_c x_{i,t} + U_c\bar{\mathbf{h}}_{i,t-1}\|)\Bigg)\Bigg] \\ &\le \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|\Big(\|f_{i,t}\|_\infty\|c_{i,t-1}\| \\ &\quad + \|r_{i,t}\|_\infty(B_{W_c}B_x + B_{U_c}\|\bar{\mathbf{h}}_{i,t-1}\|)\Big)\Bigg)\Bigg] \\ &\le \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|\Big(\|f_{i,t}\|_\infty\|c_{i,t-1}\| \\ &\quad + \|r_{i,t}\|_\infty(B_{W_c}B_x + B_{U_c}\|o_{i,t-1}\|_\infty\|c_{i,t-1}\|)\Big)\Bigg)\Bigg] \\ &= \frac{1}{\lambda}\log\mathbb{E}\Bigg[\sup\exp\Bigg(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|\Big((\|f_{i,t}\|_\infty \\ &\quad + \|r_{i,t}\|_\infty\|o_{i,t-1}\|_\infty B_{U_c})\|c_{i,t-1}\| + B_{W_c}B_x\|r_{i,t}\|_\infty\Big)\Bigg)\Bigg] \end{aligned} \tag{33}$$

From the LSTM formulation, Eq. (6), and the fact that the LSTM's gating signals can be bounded by 1 (i.e., $\|f_{i,t}\| \le 1$, $\|r_{i,t}\| \le 1$, and $\|o_{i,t-1}\| \le 1$), Eq. (33) can be further bounded as follows:

$$m\mathscr{R}_s(\mathscr{H}_{k,t}) \le \frac{1}{\lambda}\log\mathbb{E}\left[\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|\left(\left(1 + B_{U_c}\right)\|c_{i,t-1}\| + B_{W_c}B_x\right)\right)\right] \tag{34}$$

By expanding the term $\|c_{i,t-1}\|$ recursively, the above inequality reaches:

$$\begin{aligned} m\mathscr{R}_s(\mathscr{H}_{k,t}) &\le \frac{1}{\lambda}\log\mathbb{E}\left[\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|B_{W_c}B_x(1+\beta+\beta^2+\cdots)\right)\right] \\ &= \frac{1}{\lambda}\log\mathbb{E}\left[\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|B_{W_c}B_x\sum_{p=0}^{t-1}\beta^p\right)\right] \end{aligned} \tag{35}$$

We apply the formula for the sum of a geometric series, $\sum_{p=0}^{t-1}\beta^p = \frac{1-\beta^t}{1-\beta}$ to obtain the following:

$$m\mathscr{R}_s(\mathscr{H}_{k,t}) = \frac{1}{\lambda}\log\mathbb{E}\left[\exp\left(B_Z\lambda\sum_{i=1}^{m}\|\epsilon_i\|_2 B_{W_c}B_x\frac{1-\beta^t}{1-\beta}\right)\right] \tag{36}$$

where $\beta = 1 + B_{U_c}$.

Let $q = \bar{M}\sum_{i=1}^{m}\|\epsilon_i\|$, where $\bar{M}$ is the product of some weight matrices and contains a fraction involving $\bar{\beta}$, specifically

$\bar{M} = B_V B_{W_c} B_x \frac{1-\beta^t}{1-\beta}$. Notice that the Rademacher complexity variables ϵ_i are the elements giving rise to randomness in the bound. Then, the Rademacher complexity bound in inequality (36) becomes:

$$\begin{aligned} m\mathscr{R}_s(\mathscr{H}_{k,t}) &\leq \frac{1}{\lambda}\log\mathbb{E}[\exp(\lambda q)] \\ &= \frac{1}{\lambda}\log\mathbb{E}[\exp(\lambda(q-\mathbb{E}[q]))] + \mathbb{E}[q] \end{aligned} \tag{37}$$

Using Jensen's inequality, $\mathbb{E}[q]$ is bounded as follows:

$$\begin{aligned} \mathbb{E}[q] &= \mathbb{E}\left[\bar{M}\sum_{i=1}^{m}\|\epsilon_i\|\right] \leq \bar{M}\sqrt{\mathbb{E}\left[\sum_{i=1}^{m}\|\epsilon_i\|^2\right]} = \bar{M}\sqrt{\mathbb{E}\left[\sum_{i=1,\bar{i}=1}^{m}\epsilon_i^T\epsilon_{\bar{i}}\right]} \\ &= \bar{M}\sqrt{\mathbb{E}\left[\sum_{i=1,\bar{i}=1}^{m}(1)\right]} = \bar{M}\sqrt{m} \end{aligned} \tag{38}$$

We can show q is sub-Gaussian with the following variance factor υ, since q satisfies a bounded-difference condition with respect to its random variables ϵ_i, i.e., $q(\epsilon_1, \ldots, \epsilon_i, \ldots, \epsilon_m) - q(\epsilon_1, \ldots, -\epsilon_i, \ldots, \epsilon_m) \leq$ 2\bar{M}$\|\mathbf{x}_{i,t}\|$:

$$\upsilon = \frac{1}{4}\sum_{i=1}^{m}(2\bar{M}\|\mathbf{x}_{i,t}\|)^2 = \bar{M}^2\sum_{i=1}^{m}\|\mathbf{x}_{i,t}\| \tag{39}$$

According to the property of sub-Gaussian random variables in Definition 1, the following inequality holds for q:

$$\frac{1}{\lambda}\log\mathbb{E}[\exp(\lambda(q-\mathbb{E}[q]))] \leq \frac{\lambda\bar{M}^2\sum_{i=1}^{m}\|\mathbf{x}_{i,t}\|}{2} \tag{40}$$

Assuming $\lambda = \frac{\sqrt{2t}}{\bar{M}\sqrt{\sum_{i=1}^{m}\|\mathbf{x}_{i,t}\|^2}}$, the Rademacher complexity can be bounded as the following:

$$\begin{aligned} m\mathscr{R}_s(\mathscr{H}_{k,t}) &\leq \frac{\lambda M^2\sum_{i=1}^{m}\|\mathbf{x}_{i,t}\|}{2} + \bar{M}\sqrt{m} \\ &= \frac{\bar{M}\sqrt{mt}}{\sqrt{2}} + \bar{M}\sqrt{m} \\ &= \frac{\sqrt{2}+\sqrt{t}}{\sqrt{2}}\bar{M}\sqrt{m} \end{aligned} \tag{41}$$

□

Theorem 2. Let $\mathscr{G}_t$ be the family of loss function associated with the hypothesis class $\mathscr{H}_t$ of vector-valued functions that map the LSTM inputs to the LSTM output at tth time step, with weight matrices and activation functions satisfying Assumptions 1–6. Given a set of m i.i.d. data samples $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, \ldots, m$, with probability at least $1-\delta$ over S, we have:

$$\begin{aligned} \mathbb{E}[g_t(\mathbf{x},\mathbf{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i,\mathbf{y}_i) + \mathscr{O}\left(L_r d_y\frac{(\sqrt{2}+\sqrt{t})\bar{M}}{\sqrt{m}}\right) \\ + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2m}} \end{aligned} \tag{42}$$

where $\bar{M} = B_Z B_{W_c} B_x \frac{1-\beta^t}{1-\beta}$ and $\beta = 1 + B_{U_c}$.

Proof. Using the inequalities obtained in lemma 1, we can derive the following upper bound for the loss function $L(\bar{\mathbf{h}}(\mathbf{x}_i), \mathbf{y}_i)$, with $\bar{\mathbf{h}}(\mathbf{x}_i)$ being vector-valued functions, as follows:

$$\begin{aligned} R_S(G_t) &= \mathbb{E}\left[\sup_{h\in H}\frac{1}{m}\sum_{i=1}^{m}\epsilon_i L(\bar{\mathbf{h}}(\mathbf{x}_i),\mathbf{y}_i)\right] \leq \sqrt{2}L_r\mathbb{E}\left[\sup_{h\in H}\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{d_y}\epsilon_{ik}\bar{\mathbf{h}}_k(\mathbf{x}_i)\right] \\ &\leq \sqrt{2}L_r d_y\frac{(\sqrt{2}+\sqrt{t})\bar{M}\sqrt{m}}{\sqrt{2}m} \\ &= L_r d_y\frac{(\sqrt{2}+\sqrt{t})\bar{M}}{\sqrt{m}} \end{aligned} \tag{43}$$

Plugging the Rademacher complexity bound that we just obtained, i.e., $\mathscr{R}_S(\mathscr{G}_t) \leq L_r d_y\frac{(\sqrt{2}+\sqrt{t})\bar{M}}{\sqrt{m}}$, into inequality (14) mentioned in lemma 2, we get the following inequality of Theorem 2:

$$\begin{aligned} \mathbb{E}[g_t(\mathbf{x},\mathbf{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i,\mathbf{y}_i) + \mathscr{O}\left(L_r d_y\frac{(\sqrt{2}+\sqrt{t})\bar{M}}{\sqrt{m}}\right) \\ + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2m}} \end{aligned} \tag{44}$$

□

Remark 5. By observing the three different terms in the generalization error bound of Eq. (42), it appears that several ways to reduce the generalization error may be considered. The first approach is to design a LSTM network such that the value of the empirical loss $\frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i,\mathbf{y}_i)$ over the training data samples is minimized. Additionally, noticing that the number of training samples m occurs in the denominator of both the second and third terms of Eq. (42), one can consider increasing the number of training samples m and, as a result, the value of the generalization error should decrease. Also, it is important to note that the complexity hypothesis class may affect the value of the generalization error, in the sense that increasing the complexity hypothesis class results in an increase in the values of the weight matrices bounds M and, subsequently, an increase in the second term $\mathscr{O}$ in Eq. (42). Hence, when designing the LSTM network, we consider starting with a simple design and, based on the training and testing performance, we can increase the complexity such that it improves the model performance for a given application without causing overfitting to the data.

5. RNN/LSTM based model predictive control

In this section, we integrate an RNN/LSTM model into a Lyapunov-based model predictive controller (LMPC) formulation. In particular, the partially-connected modeling of RNN/LSTM is executed as discussed in Alhajeri et al. (2022b) and then employed as a predictive model to provide state estimation to solve the optimization problem of the LMPC, which is expressed in the following form:

$$\mathscr{J} = \min_{u\in S(\Delta)}\int_{t_k}^{t_k+P} L_c(\tilde{x}(t), u(t))\,dt \tag{45a}$$

$$\text{s.t. } \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{45b}$$

$$u(t) \in U, \quad \forall\, t \in [t_k, t_k + P) \tag{45c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{45d}$$

$$\dot{V}(x(t_k), u) \leq \dot{V}(x(t_k), \Phi_{nn}(x(t_k))),$$

$$\text{if } x(t_k) \in \Omega_\rho - \Omega_{\rho_{nn}} \tag{45e}$$

$$V(\tilde{x}(t)) \leq \rho_{nn}, \quad \forall\, t \in [t_k, t_k + P), \quad \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{45f}$$

where $S(\Delta)$ denotes a set of piecewise constant functions with period Δ, $\tilde{x}$ is the state trajectory predicted by the RNN/LSTM model, and P is the prediction horizon expressed as a multiple of the sampling period (i.e., $P = N \times \Delta$, $N > 0$). The time-derivative of the Lyapunov function V in Eq. (45e) is given as $\dot{V}(x, u)$, i.e., $\frac{\partial V(x)}{\partial x} F_{nn}(x, u)$. During the prediction horizon $t \in [t_k, t_k + P)$, the LMPC computes the optimum input sequence $u^*(t)$ and delivers the first control signal $u^*(t_k)$ to the system to be implemented for the following sampling period. After that, at the following sampling interval, the LMPC receives new data and is resolved with updated state estimations. Furthermore, the MPC optimization problem's goal is to minimize the integral of $L_c(\tilde{x}(t), u(t))$, given in Eq. (45a), which represents the cost function over the prediction horizon while satisfying the constraints of Eqs. (45b) to (45f). The RNN/LSTM model from Eq. (45b) is used to forecast the evolution of the closed-loop state trajectory $\tilde{x}(t_k)$ under the MPC, and its initial conditions are updated according to Eq. (45d), where $x(t_k)$ is the last state measurement. The input constraints are expressed in Eq. (45c), and they are imposed across the prediction horizon.

To ensure the stability of the closed-loop system, when $x(t_k) \in \Omega_\rho - \Omega_{\rho_{nn}}$, where $\Omega_{\rho_{nn}}$ is the target region, the condition of Eq. (45e) is triggered. As a result of this constraint, the Lyapunov function of the closed-loop state declines, and the state approaches the steady-state within a finite period of time. Eventually, when the state $x(t_k)$ arrives to $\Omega_{\rho_{nn}}$, the predicted closed-loop state will be kept within this region for the duration of the prediction horizon via the constraint of Eq. (45f). Following section 2.3, the controller $\Phi_{nn}(x)$ was developed with the intent of ensuring that the origin of the RNN/LSTM system is exponentially stable.

A well-conditioned RNN or LSTM model with sufficient model accuracy can be readily produced when utilizing noise-free data for training. Therefore, the closed-loop state is guaranteed to be bound inside the predefined stability region Ω_ρ during the simulation time and will finally converge to a small region around the origin via application of the RNN/LSTM-based LMPC of Eq. (45) for the regulation of the nonlinear system of Eq. (1). This is true provided that the modeling error, which is given by $|\nu| = |F(x, u) - F_{nn}(x, u)|$, is sufficiently small (Wu et al., 2019; Alhajeri et al., 2021).

6. Application

A chemical process example is used for demonstrating the anticipated improvements associated with physics-informed modeling of RNNs. Particularly, two non-isothermal continuous-stirred tank reactors (CSTR) in sequence with ideal mixing are taken into consideration, as shown in Fig. 4, with each reactor containing an irreversible second-order exothermic reaction, where a raw material A is transformed to a product B (i.e., A → B). The feed flow rate to each reactor F_{i_0} contains only chemical A with initial concentration and temperature C_{A,i_0} and T_{i_0}, respectively, where $i = 1, 2$ is the reactor index. Each reactor has a heating jacket that delivers or removes heat at a rate of Q_i. The dynamical model describing the two CSTRs is derived from material and energy balance equations in the form of the following ODEs:

$$\frac{dC_{A,1}}{dt} = \frac{F_{1_0}}{V_1}\left(C_{A1_0} - C_{A,1}\right) - k_o e^{\frac{-E}{RT_1}} C_{A,1}^2 \tag{46a}$$

$$\frac{dT_1}{dt} = \frac{F_{1_0}}{V_1}\left(T_{1_0} - T_1\right) + \frac{-\Delta H}{\rho C_p} k_o e^{\frac{-E}{RT_1}} C_{A,1}^2 + \frac{Q_1}{\rho C_p V_1} \tag{46b}$$

$$\frac{dC_{A,2}}{dt} = \frac{F_1}{V_2} C_{A,1} + \frac{F_{2_0}}{V_2} C_{A,2_0} - \frac{F_1 + F_{2_0}}{V_2} C_{A,2} - k_o e^{\frac{-E}{RT_2}} C_{A,2}^2 \tag{46c}$$

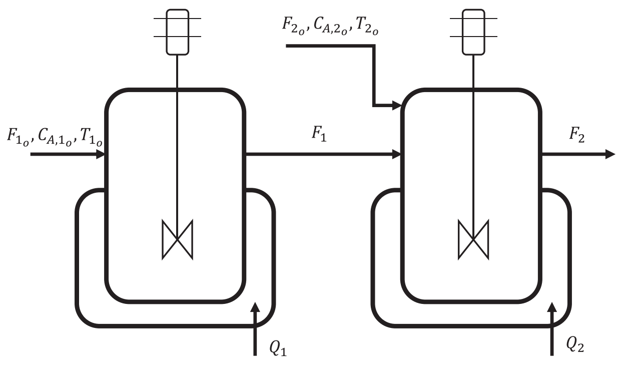

Fig. 4 – Two continuous-stirred tank reactors in series.

$$\frac{dT_2}{dt} = \frac{F_{2_0}}{V_2} T_{2_0} + \frac{F_1}{V_2} T_1 - \frac{F_1 + F_{2_0}}{V_2} T_2 + \frac{-\Delta H}{\rho C_p} k_o e^{\frac{-E}{RT_2}} C_{A,2}^2 + \frac{Q_2}{\rho C_p V_2} \tag{46d}$$

The notations $C_{A,i}$, T_i, and Q_i represent the reactant A concentration, reactor temperature, and the heat supply rate, respectively. V_i is the volume of the reacting liquid, which has a density of ρ and a heat capacity of C_p that are constants for both reactors. ΔH, k_o, R, and E denote the reaction's enthalpy, pre-exponential constant, ideal gas constant, and activation energy, in the same order, and these parameters are unchanged for both reactors. Process parameter values are listed in Table 1.

The manipulated inputs for this process are the heat supply rate to both reactors (i.e., Q_1 and Q_2), which are represented in deviation form from their steady-state values as $u_1 = Q_1 - Q_{1s}$ and $u_2 = Q_2 - Q_{2s}$. The upper and lower physical bounds on the inputs are $[U^{\max}, U^{\min}] = [5, -5] \times 10^5$ *kJ/h*, respectively. The states are also represented in deviation fashion from their steady-state values as $[x_1, x_2, x_3, x_4] = [C_{A,1} - C_{A,1_s}, T_1 - T_{1s}, C_{A,2} - C_{A,2_s}, T_2 - T_{2s}]$, such that the origin is the equilibrium point of the state-space representation of the underlying system.

6.1. *Data generation and RNN models construction*

Large data sets are necessary for the development of machine-learning-based models, and, generally speaking, the larger the data set size, the more accurate the model can be (Wu et al., 2022), provided that the data is independent and identically distributed. Large data sets are also accessible from a variety of sources, including industries, pilot plants and laboratories, and computer-based simulations. Industrial data is typically not accessible to the general public, and collecting data from pilot plants and laboratory studies is both expensive and time-consuming. Hence, we use

Table 1 – Parameter and steady-state values for the CSTR.

$C_{A,1_s} = 1.95$ *kmol/m*^{3}	$T_{1s} = 402$ K
$C_{A,1_0} = 4$ *kmol/m*^{3}	$T_{2s} = 402$ K
$C_{A,2_s} = 1.95$ *kmol/m*^{3}	$Q_{1s} = 0.0$ *kJ/h*
$C_{A,2_0} = 4$ *kmol/m*^{3}	$Q_{2s} = 0.0$ *kJ/h*
$T_{1_0} = 300$ K	$T_{2_0} = 300$ K
$F_{1_0} = 5$ *m*^{3}*/h*	$F_{2_0} = 5$ *m*^{3}*/h*
$V_1 = 1$ *m*^{3}	$V_2 = 1$ *m*^{3}
$k_o = 8.46 \times 10^6$ *m*^{3}*/kmolh*	$E_A = 5 \times 10^4$ *kJ/kmol*
$R = 8.314$ *kJ/(kmol K)*	$\Delta H = -\ 1.15 \times 10^4$ *kJ/kmol*
$\rho = 1000$ *kg/m*^{3}	$C_p = 0.231$ *kJ/(kg K)*

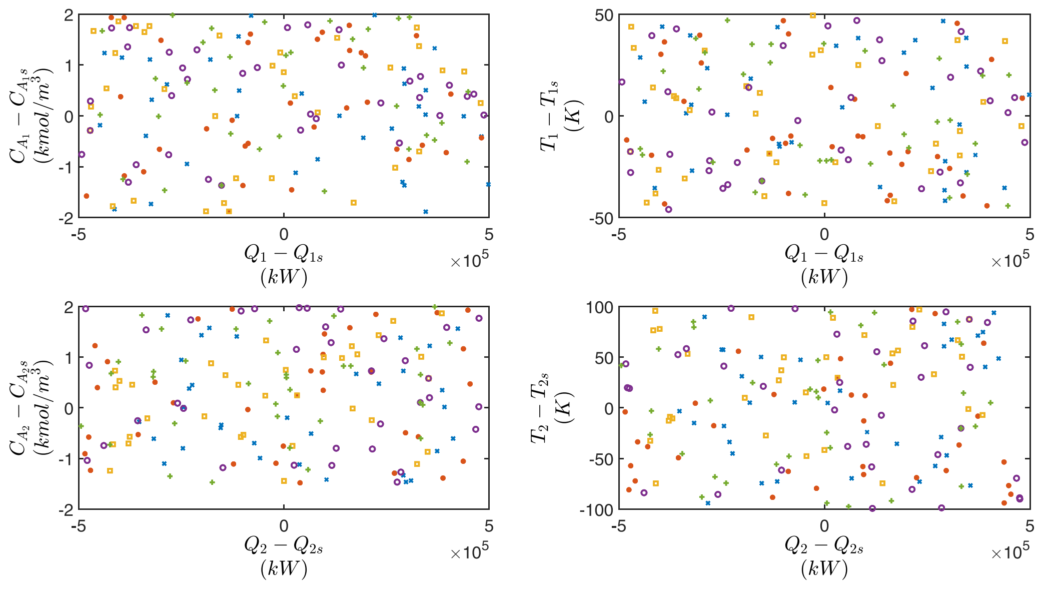

Fig. 5 – Five different testing data sets, where each marker indicates a single set.

extensive open-loop simulations in our work to create our data set.

For the development of the RNN model, the procedures for data generation, neural network training, and validation are described. The explicit Euler method with an integration time step of $h_c = 5 \times 10^{-4}\, h$ is used to numerically simulate the dynamic model of Eq. (46) for one sampling period Δ under various initial conditions (a total of 3000 different combinations of initial conditions). Particularly, MATLAB is used to create a data set of size m_{data}. The data set is then split into two matrices: an output matrix with x_1, x_2, x_3, and x_4 as outputs at $t = t_k + \Delta$ and an input matrix with u_1, u_2, x_1, x_2, x_3, and x_4 at $t = t_k$.

Subsequently, by using the generated data and the Keras library, two RNN models are constructed, where each of the models has two hidden layers with 30 neurons in each, and hyperbolic tangent (i.e., $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$) as the activation function for all layers except for the input and output layers. The activation function in the output layer is set to be linear. The links between the layers is untouched in the fully-connected RNN modeling, while, on the other hand, the inputs are fed to different layers in the partially connected RNN modeling in a manner that reflects the physical structure of the underlying process. Specifically, the partially-connected RNN model is developed following the algorithm discussed in Alhajeri et al. (2022b).

Using input information from the previous sampling interval, we forecast the evolution of the states for the subsequent 0.01 hr (the equivalent of one sampling time Δ). We use the Adam optimizer, a combination of RMSprop and gradient descent with momentum optimization techniques, as opposed to the conventional gradient descent optimization process. Additionally, we perform five-fold cross-validation on the RNN models in order to produce more reliable models, and select the models with the lowest validation MSE. In addition, we use five, different (i.e., no repetition) testing data sets, as shown in Fig. 5, to test the developed models. The generalization error for each testing data set is illustrated in Fig. 6, where the partially-connected RNN model yielded higher generalization accuracy (i.e., less error). These results aligned with Theorem 1.

6.2. *Open-loop simulation*

Before incorporating the generated models into closed-loop tests, open-loop simulations are essential to check that the predictive models can estimate the future trajectory adequately. Hence, we carried out open-loop simulations, illustrated in Fig. 7, where the time-varying inputs are randomly chosen. Furthermore, from the figure, it can be noticed that the state trajectories predicted by the partially-connected RNN model are all closer to the true state trajectories (denoted by FP) than the states predicted by the fully-connected RNN model.

Table 2 presents the open loop simulation MSEs between the predicted states from each RNN model architecture and the corresponding first-principles model outputs as the ground-truth process output value. From Table 2, the ratios of fully-connected RNN MSE to the partially-connected RNN MSE for x_1, x_2, x_3, and x_4 are 6.05, 2.3991, 4.3018, and 10.3013, receptively. All the ratios are greater than one, which implies that the partially-connected RNN architecture yields a higher

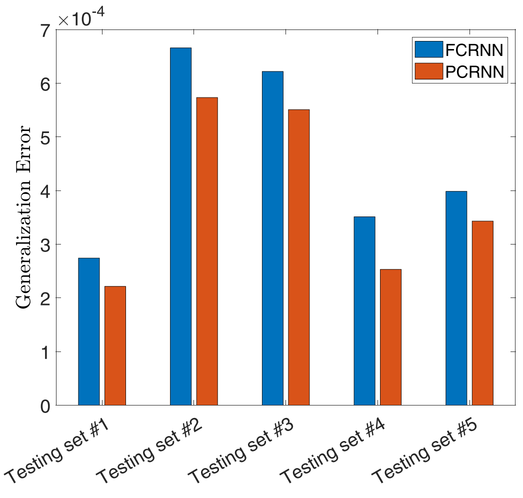

Fig. 6 – Generalization error for five different testing data sets, where PCRNN and FCRNN stand for partially-connected RNNs (orange bars) and fully-connected RNNs (blue bars), respectively.

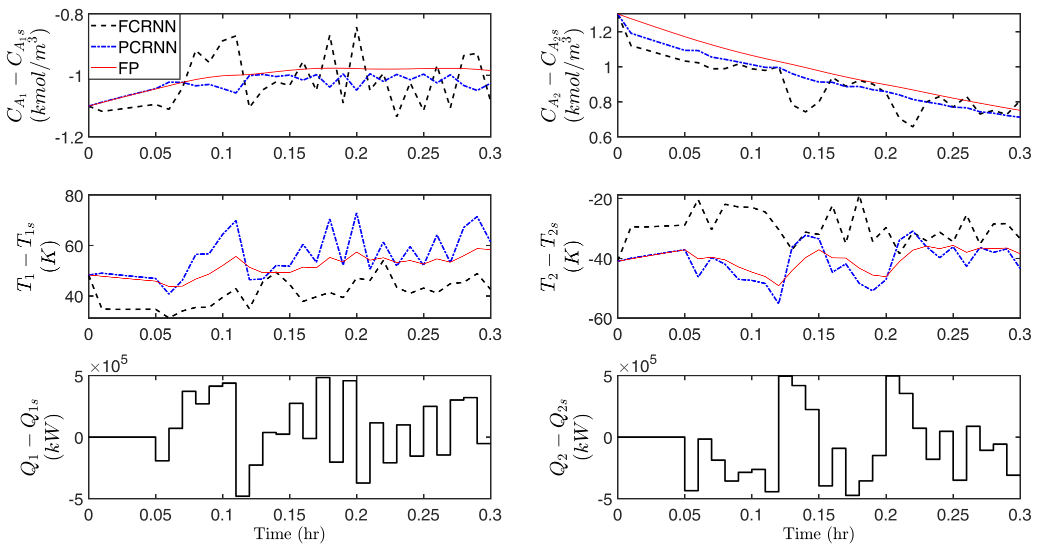

Fig. 7 – Time-varying profiles of the states and inputs for the second open-loop simulation under random time-varying inputs using the first-principles process model (red line), the partially-connected RNN model (blue line), and the fully-connected RNN model (black line).

Table 2 – Open-loop prediction results (MSE).

State	Modeling architecture	
	FCRNN	PCRNN
x_1	0.0065	0.0011
x_2	125.4551	52.2929
x_3	0.0134	0.0031
x_4	156.3076	15.1736

accuracy for state estimation. Furthermore, the open-loop responses initiated close to the steady state and predicted by the partially-connected and the fully-connected RNN models under a step change only in u_2 are depicted in Fig. 8. The figure demonstrates that the partially-connected RNN model captures the process dynamics better, as the trajectory of the first reactor's temperature was not altered by this change. All these results indicate that both RNN models provide reasonable prediction, yet, the partially-connected RNN model approximates the underlying process model more accurately.

6.3. Closed-loop simulation

Next, in order to run closed-loop simulations with the certainty that both RNN models give high accuracy approximation for the process outputs, we design LMPCs based on the fully-connected RNN model and the partially-connected RNN model, respectively. For each sampling period, the nonlinear minimization problem of the LMPC is solved using the Python front-end of the interior point optimizer (IPOPT) software (Biegler and Zavala, 2009). For the purpose of solving complex nonlinear optimization problems, this optimizer is an open source program. It uses an interior point line search filter technique to try to locate a local solution to a nonlinear mathematical program. The LMPC objective function is defined as $L_c(x, u) = x^T \mathbf{Q} x + u^T \mathbf{R} u$, where $\mathbf{Q}$ and $\mathbf{R}$ are diagonal penalty matrices for the set-point error and control

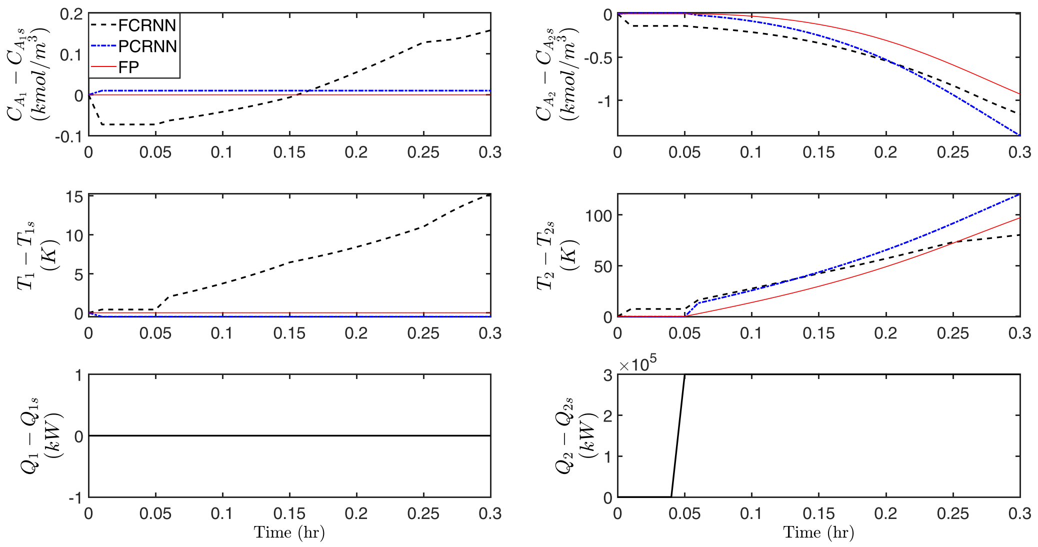

Fig. 8 – Time-varying profiles of the states and inputs for the open-loop simulation under a step change in u_2 using the first-principles process model (red line), the partially-connected RNN model (blue line), and the fully-connected RNN model (black line).

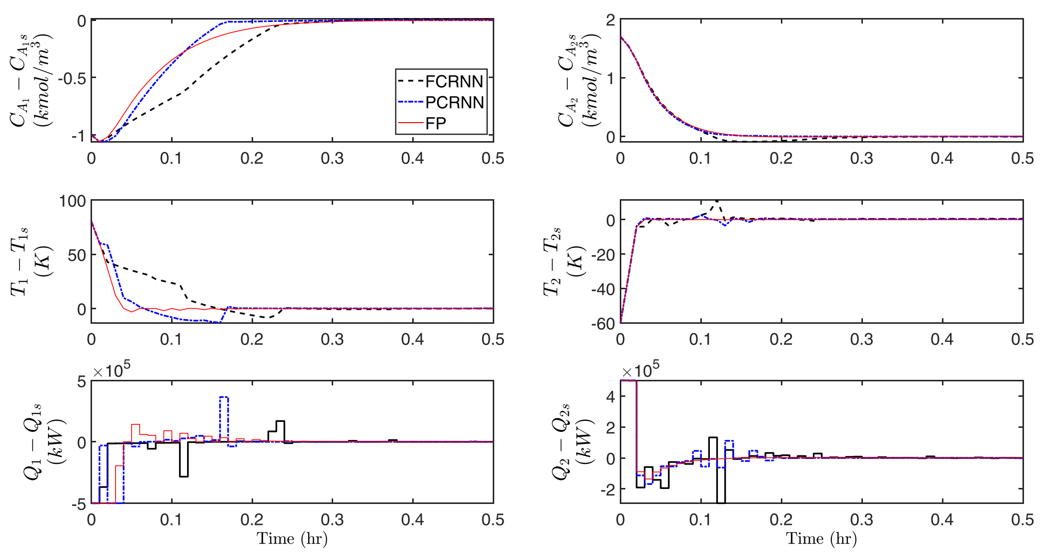

Fig. 9 – State and input profiles of the first closed-loop simulation under the LMPC using three models: first-principles (red line), partially-connected RNN (blue line), and fully-connected RNN (black line).

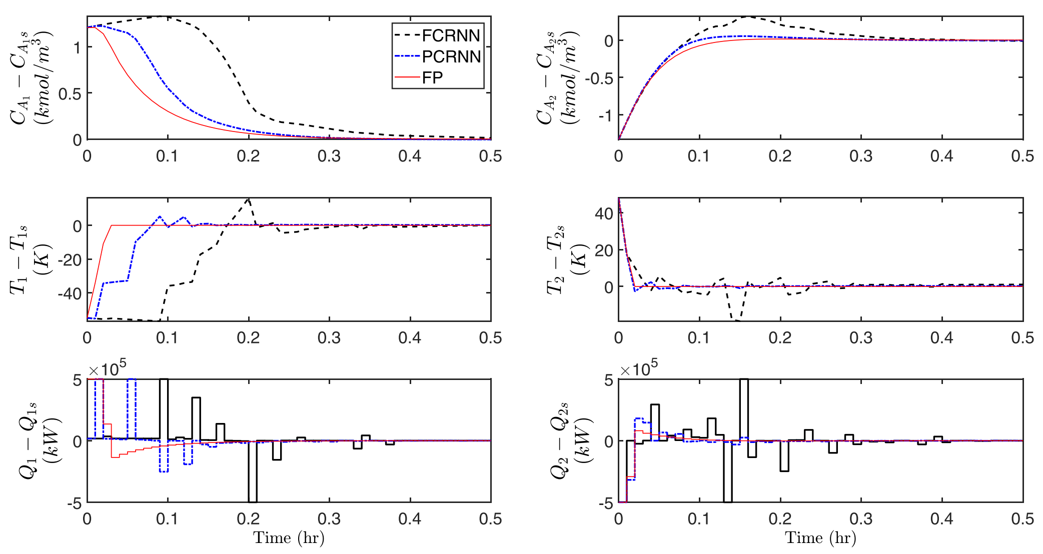

Fig. 10 – State and input profiles of the second closed-loop simulation under the LMPC using three models: first-principles (red line), partially-connected RNN (blue line), and fully-connected RNN (black line).

actions, respectively. The two matrices critically impact the performance of the LMPC and require proper tuning. MPC tuning guidelines discussed in Alhajeri and Soroush (2020) are followed. Lastly, we choose $V(x) = x^T P x$ as the Lyapunov function, where P is a positive definite matrix obtained by applying a grid search.

Under the LMPCs, we perform closed-loop simulations initiating from two different initial conditions, and the results are shown in Figs. 9 and 10. From the figures, both LMPCs, each based on its predictive RNN model, were able to drive the states to the steady-state values and to stabilize the system within a small neighborhood around the origin. However, the partially-connected RNN-based LMPC yielded better performance in terms of state trajectories being smoother and not exhibiting fluctuation around the steady state. Moreover, the MSE of each state corresponding to the

Table 3 – Closed-loop prediction results (MSE).

State	1st Closed-loop Simulation		2nd Closed-loop Simulation	
	FCRNN	PCRNN	FCRNN	PCRNN
x_1	0.020705316	0.002051591	0.2411215725	0.025175541
x_2	170.280344	39.13188574	596.3087136	88.72292971
x_3	0.001812249	0.000111788	0.015817003	0.001061295
x_4	3.788903947	0.511854559	20.3473683	0.400205264

two RNN architectures are calculated for the two closed-loop simulations and presented in Table 3. As it can be noted from the table, the partially-connected RNN model yielded a more reliable controller with smaller MSE values by an order of magnitude compared to the fully-connected RNN model. Due to the fully-connected RNN's deterioration in the prediction accuracy caused by the assumption that every input influences every potential output, these results are expected.

7. Conclusion

In this study, we used the Rademacher complexity approach for vector-valued functions to create an upper bound for the generalization error of two classes of RNN models—partially-connected and fully-connected—and LSTM networks. For the partially-connected RNN, theoretical results connecting a RNN model's accuracy to its architecture were established and proved, as for the latter a generalizability bound for this specific structure of RNNs. Open-loop simulations utilizing a complex, nonlinear chemical process was performed to demonstrate the superior model accuracy of the partially-connected RNN when compared to the fully-connected RNN across various testing data sets. Additionally, the developed partially-connected RNN model was then utilized in the design of a Lyapunov-based MPC. Through several closed-loop simulations, it was shown that adopting the partially-connected RNN model yielded smoother state trajectories with lower loss function values (i.e., smaller mean squared errors), and the applied inputs suffered less from oscillatory behavior.

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.

Acknowledgement

M. S. Alhajeri would like to express his genuine appreciation for the funding from KFAS. The first and the second authors acknowledge Kuwait University support via the KU-scholarship program. Financial support from the National Science Foundation is also gratefully acknowledged. We would like to thank Dr. Minshuo Chen for pointing out an important remark used in the LSTM generalization error bound proof. The authors would also like to thank Professor Zhe Wu for his support in the development of the theory for generalization error bounds used in this work.

References

Alber, M., BuganzaTepole, A., Cannon, W.R., De, S., Dura-Bernal, S., Garikipati, K., Karniadakis, G., Lytton, W.W., Perdikaris, P., Petzold, L., et al., 2019. Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. NPJ Digit. Med. 2, 1–11.

Alhajeri, M.S., Abdullah, F., Wu, Z., Christofides, P.D., 2022a. Physics-informed machine learning modeling for predictive control using noisy data. Chem. Eng. Res. Des. 186, 34–49.

Alhajeri, M.S., Luo, J., Wu, Z., Albalawi, F., Christofides, P.D., 2022b. Process structure-based recurrent neural network modeling for predictive control: a comparative study. Chem. Eng. Res. Des. 179, 77–89.

Alhajeri, M.S., Soroush, M., 2020. Tuning guidelines for model-predictive control. Ind. Eng. Chem. Res. 59, 4177–4191.

Alhajeri, M.S., Wu, Z., Rincon, D., Albalawi, F., Christofides, P.D., 2021. Machine-learning-based state estimation and predictive control of nonlinear processes. Chem. Eng. Res. Des. 167, 268–280.

Ali, J.M., Hussain, M.A., Tade, M.O., Zhang, J., 2015. Artificial intelligence techniques applied as estimator in chemical process systems–a literature survey. Expert Syst. Appl. 42 (14), 5915–5931.

Baker, N., Alexander, F., Bremer, T., Hagberg, A., Kevrekidis, Y., Najm, H., Parashar, M., Patra, A., Sethian, J., Wild, S., et al. (2019). Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States).

Banerjee, A., Varshney, D., Kumar, S., Chaudhary, P., Gupta, V.K., 2017. Biodiesel production from castor oil: ANN modeling and kinetic parameter estimation. Int. J. Ind. Chem. 8, 253–262.

Biegler, L.T., Zavala, V.M., 2009. Large-scale nonlinear programming using ipopt: an integrating framework for enterprise-wide dynamic optimization. Comput. Chem. Eng. 33 (3), 575–582.

Butcher, J.C., 1996. A history of runge-kutta methods. Appl. Numer. Math. 20 (3), 247–260.

Cang, R., Li, H., Yao, H., Jiao, Y., Ren, Y., 2018. Improving direct physical properties prediction of heterogeneous materials from imaging data via convolutional neural network and a morphology-aware generative model. Comput. Mater. Sci. 150, 212–221.

Canning, A., Gardner, E., 1988. Partially connected models of neural networks. J. Phys. A: Math. Gen. 21 (15), 3275.

Chakraborty, I., Bodurtha, K.J., Heeder, N.J., Godfrin, M.P., Tripathi, A., Hurt, R.H., Shukla, A., Bose, A., 2014. Massive electrical conductivity enhancement of multilayer graphene/polystyrene composites using a nonconductive filler. ACS Appl. Mater. Interfaces 6 (19), 16472–16475.

Chaouat, B., 2017. The state of the art of hybrid rans/les modeling for the simulation of turbulent flows. Flow Turbul. Combust. 99 (2), 279–327.

Chen, S., Wu, Z., Christofides, P.D., 2020. Decentralized machine-learning-based predictive control of nonlinear processes. Chem. Eng. Res. Des. 162, 45–60.

Cozad, A., Sahinidis, N.V., Miller, D.C., 2015. A combined first-principles and data-driven approach to model building. Comput. Chem. Eng. 73, 116–127.

De Azevedo, S.F., Dahm, B., Oliveira, F., 1997. Hybrid modelling of biochemical processes: a comparison with the conventional approach. Comput. Chem. Eng. 21, S751–S756.

Dias, A.C.S.R., daSilva, W.B., Dutra, J.C.S., 2017. Propylene polymerization reactor control and estimation using a particle filter and neural network. Macromol. React. Eng. 11, 1700010.

Emmert-Streib, F., Dehmer, M., 2019. Evaluation of regression models: Model assessment, model selection and generalization error. Mach. Learn. Knowl. Extr. 1, 521–551.

Faghmous, J.H., Kumar, V., 2014. A big data guide to understanding climate change: the case for theory-guided data science. Big Data 2 (3), 155–163.

Fan, J. and Han, M. (2012). Nonliear model predictive control of ball-plate system based on gaussian particle swarm optimization.In 2012 IEEE Congress on Evolutionary Computation, 1–6.Birsbane, Australia.

Golowich, N., Rakhlin, A., and Shamir, O. (2018). Size-independent sample complexity of neural networks.In Conference on Learning Theory, 297–299.

Han, H., Wu, X., Qiao, J., 2013. Real-time model predictive control using a self-organizing neural network. IEEE Trans. Neural Netw. Learn. Syst. 24 (9), 1425–1436.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Houska, B., Logist, F., Diehl, M., Impe, J.V., 2012. A tutorial on numerical methods for state and parameter estimation in nonlinear dynamic systems. Identif. Automot. Syst. 67–88.

Jia, X., Khandelwal, A., Mulla, D.J., Pardey, P.G., Kumar, V., 2019. Bringing automated, remote-sensed, machine learning methods to monitoring crop landscapes at scale. Agric. Econ. 50, 41–50.

Kahrs, O., Marquardt, W., 2007. The validity domain of hybrid models and its application in process optimization. Chem. Eng. Process.: Process Intensif. 46, 1054–1066.

Karpatne, A., Atluri, G., Faghmous, J.H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., Kumar, V., 2017. Theory-guided data science: a new paradigm for scientific discovery from data. IEEE Trans. Knowl. Data Eng. 29, 2318–2331.

Kochkov, D., Smith, J.A., Alieva, A., Wang, Q., Brenner, M.P., Hoyer, S., 2021. Machine learning–accelerated computational fluid dynamics. Proc. Natl. Acad. Sci. U.S.A. 118, e2101784118.

Krasnopolsky, V.M., Fox-Rabinovitz, M.S., 2006. Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction. Neural Netw. 19, 122–134.

Lee, C., 1990. Fuzzy logic in control systems: fuzzy logic controller. I. IEEE Trans. Syst. Man Cybern. 20, 404–418.

Liao, S., 2005. Expert system methodologies and applications—a decade review from 1995 to 2004. Expert Syst. Appl. 28, 93–103.

Lin, Y., Sontag, E.D., 1991. A universal formula for stabilization with bounded controls. Syst. Control Lett. 16, 393–397.

Lu, Y., Rajora, M., Zou, P., Liang, S., 2017. Physics-embedded machine learning: case study with electrochemical micro-machining. Machines 5, 4.

Luo, J., Canuso, V., Jang, J.B., Wu, Z., Morales-Guio, C.G., Christofides, P.D., 2022. Machine learning-based operational modeling of an electrochemical reactor: handling data variability and improving empirical models. Ind. Eng. Chem. Res. 61, 8399–8410.

Mohri, M., Rostamizadeh, A., Talwalkar, A., 2018. Foundations of Machine Learning. MIT Press.

O'Gorman, P.A., Dwyer, J.G., 2018. Using machine learning to parameterize moist convection: Potential for modeling of climate, climate change, and extreme events. J. Adv. Model. Earth Syst. 10 (10), 2548–2563.

Pan, Y., Wang, J., 2011. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. IEEE Trans. Ind. Electron. 59, 3089–3101.

Patel, N., Nease, J., Aumi, S., Ewaschuk, C., Luo, J., Mhaskar, P., 2020. Integrating data-driven modeling with first-principles knowledge. Ind. Eng. Chem. Res. 59, 5103–5113.

Rai, R., Sahu, C.K., 2020. Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus. IEEE Access 8, 71050–71073.

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., et al., 2019. Deep learning and process understanding for data-driven earth system science. Nature 566 (7743), 195–204.

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

Roelofs, R. (2019). Measuring Generalization and Overfitting in Machine learning.Doctoral Dissertation, University of California, Berkeley.

Sagaut, P., Terracol, M., Deck, S., 2013. Multiscale and Multiresolution Approaches in Turbulence-LES, DES and Hybrid RANS/LES Methods: Applications and Guidelines. World Scientific.

Schleder, G.R., Padilha, A.C., Acosta, C.M., Costa, M., Fazzio, A., 2019. From dft to machine learning: recent approaches to materials science–a review. J. Phys.: Mater. 2 (3), 032001.

Schmidhuber, J., 2015. Deep learning in neural networks: an overview. Neural Netw. 61, 85–117.

Schütt, K., Kindermans, P.-J., SaucedaFelix, H.E., Chmiela, S., Tkatchenko, A., Müller, K.-R., 2017. Schnet: a continuous-filter convolutional neural network for modeling quantum interactions. Adv. Neural Inf. Process. Syst. 30.

Shahnazari, H., Mhaskar, P., House, J.M., Salsbury, T.I., 2019. Modeling and fault diagnosis design for hvac systems using recurrent. Neural Netw. Comput. Chem. Eng. 126, 189–203.

Singh, A., Singh, H.P., and Mishra, S. (2017). Validation of ann-based model for binary distillation column.In Proceeding of International Conference on Intelligent Communication, Control and Devices, 235–242.Springer, Singapore.

Stephanopoulos, G., Han, C., 1996. Intelligent systems in process engineering: a review. Comput. Chem. Eng. 20, 743–791.

Tompson, J., Schlachter, K., Sprechmann, P., and Perlin, K. (2017). Accelerating eulerian fluid simulation with convolutional networks.In International Conference on Machine Learning, 3424–3433.

Vepa, R., 1993. A review of techniques for machine learning of real-time control strategies. Intell. Syst. Eng. 2, 77–90.

Wang, L., Chen, J., Marathe, M., 2020a. Tdefsi: theory-guided deep learning-based epidemic forecasting with synthetic information. ACM Trans. Spat. Algorithms Syst. 6 (3), 1–39.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. (2020b). Towards physics-informed deep learning for turbulent flow prediction.In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 1457–1466.

Wilson, Z.T., Sahinidis, N.V., 2017. The alamo approach to machine learning. Comput. Chem. Eng. 106, 785–795.

Wong, W.C., Chee, E., Li, J., Wang, X., 2018. Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. Mathematics 6 (11), 242.

Wu, Z., Alnajdi, A., Gu, Q., Christofides, P.D., 2022. Statistical machine-learning–based predictive control of uncertain nonlinear processes. AIChE J. 68, e17642.

Wu, Z., Rincon, D., Christofides, P.D., 2020. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. J. Process Control 89, 74–84.

Wu, Z., Rincon, D., Gu, Q., Christofides, P.D., 2021. Statistical machine learning in model predictive control of nonlinear processes. Mathematics 9, 1912.

Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019. Machine learning-based predictive control of nonlinear processes. part I: Theory. AIChE J. 65, e16729.

Xu, J., Li, C., He, X., Huang, T., 2016. Recurrent neural network for solving model predictive control problem in application of four-tank benchmark. Neurocomputing 190, 172–178.

Xu, T., Valocchi, A.J., 2015. Data-driven methods to improve baseflow prediction of a regional groundwater model. Comput. Geosci. 85, 124–136.

Yazdani, A., Lu, L., Raissi, M., Karniadakis, G.E., 2020. Systems biology informed deep learning for inferring parameters and hidden dynamics. PLoS Comput. Biol. 16 (11), e1007575.

Zadeh, L.A., 1968. Probability measures of fuzzy events. J. Math. Anal. Appl. 23, 421–427.