



Physics-informed recurrent neural network modeling for predictive control of nonlinear processes[☆]

Yingzhe Zheng^a, Cheng Hu^a, Xiaonan Wang^{a,b}, Zhe Wu^{a,*}

^a Department of Chemical and Biomolecular Engineering, National University of Singapore, 117585, Singapore

^b Department of Chemical Engineering, Tsinghua University, Beijing 100084, China



ARTICLE INFO

Article history:

Received 8 July 2022

Received in revised form 29 March 2023

Accepted 24 May 2023

Available online xxxx

Keywords:

Physics-informed neural networks

Recurrent neural networks

Generalization error

Model predictive control

Nonlinear systems

Chemical processes

ABSTRACT

In this work, we present a physics-informed recurrent neural network (PIRNN) modeling approach, and a PIRNN-based predictive control scheme for a general class of nonlinear dynamic systems. Specifically, we first develop a hybrid data-driven and physics-guided modeling framework that integrates measurement data and mechanistic mathematical models to construct high-fidelity RNN models. Then, we derive a generalization error bound of the PIRNN model based on a nominal system model via the Rademacher complexity technique from statistical machine learning theory. Subsequently, the PIRNN model is utilized in Lyapunov-based model predictive controllers and applied to a chemical reactor example with Gaussian measurement noise to demonstrate its improved noise rejection and generalization performance in comparison to the purely data-driven and the purely physics-guided RNN-based predictive control schemes.

© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

Model predictive control (MPC) is an optimization-based advanced control method that solves for the optimal control actions based on a process model while accounting for the intrinsic dynamic behavior of the process. It is a highly flexible and powerful control scheme that permits the inclusion of practical constraints (e.g., temperature limits of a heating jacket) on the manipulated variables, which facilitates the search of feasible solutions to optimization-based control problems by respecting any physical limits imposed. However, a major challenge that prohibits the real-time implementation of MPC to nonlinear processes has been largely associated with model representation and identification. The development of accurate and computationally efficient process models thus remains the bottleneck. In attempts to address this limitation, one of the promising pursuits encompasses the synergy between machine learning (ML) and MPC, which promotes the conversion of process data into manufacturing intelligence via the integration between data-driven modeling and model-based control methods. Specifically, ML-MPC has achieved astounding computational and control performances in previous works (e.g., in controlling a diffusion-reaction process [1], a batch crystallization process [2], and a plasma etch process [3]).

However, although ML-MPC seems to be the hallmark among the state-of-the-art industrial process control techniques, a major concern associated with this approach is the generalization capability of the ML models, which is contingent on the quality of training data [4]. Limited or inadequately sampled training data can compromise the predictive prowess of ML models, and this greatly hinders the prescriptive performance and hence the applicability of ML-MPC to real-world manufacturing systems. To this end, as standard ML modeling techniques generally require an extensive set of training data, which may not always be readily available or is cumbersome to obtain, to warrant satisfactory predictive performance, more practicable alternatives are thus of compelling need. To address this issue, many works have been developed to embed prior physical knowledge into the formulation of neural networks (e.g., Refs. [5–9]). For example, in Ref. [9], a priori process structural knowledge was incorporated into the design of RNN architecture, and demonstrated its superiority in improving approximation performance compared to a conventional fully-connected RNN model. Additionally, the physics-informed neural network (PINN) modeling approach, which was initially proposed in Ref. [10], has elicited a substantial interest in the modeling industry [11]. Fundamentally, PINN effectively overcomes the small data problem by informatively exploiting a priori knowledge of the nonlinear system for constructing robust feedforward neural networks. In the physics-informed modeling framework, training of the neural network is facilitated by directly embedding the underlying governing physical laws (e.g., physical phenomena expressed in terms of

[☆] Financial support from A*STAR MTC YIRG 2022, Singapore Grant (0959-1524/© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

* Corresponding author.

E-mail address: wuzhe@nus.edu.sg (Z. Wu).

ordinary differential equations (ODE) or partial differential equations (PDE)) as regularization terms (e.g., nonlinear differential operators on the outputs of the model) in the cost function of the learning problem [10,11]. Intuitively and functionally, the addition of the penalty terms augments the learning process by enforcing the required behavior imposed not only by empirical data, but also by the mechanistic laws. The regularization effect is thus instrumental to the neural network in conforming to the a priori structure dictated by the physical laws, and this moderates the reliance on exhaustively large process datasets [4,12].

Subsequently, a number of extensions and variants to the original PINN modeling approach have since been proposed: parareal PINN (PPINN) for parallelized learning of very long-time integration of a time-dependent PDE via decomposing the long-time problem into several distinct short-time problems [13], multi-fidelity PINN (MPINN) that learns from training data of disparate precisions [14,15], Bayesian PINN (BPINN) for solving both the forward and inverse nonlinear problems involving noisy data [16], and DeepXDE that incorporates a residual-based adaptive refinement (RAR) technique for boosting the training efficiency of PINN [17]. In addition, the applications of PINN are expansive and span across numerous engineering disciplines. Previous works have explored PINN for various applications, such as in modeling of exothermic heat transfer in a composite-tool system [18], identification of unknown geometric and material parameters for characterizing internal structures and defects in materials [19], and predicting fluid flow in the gas drainage of a water-filled porous medium [20]. However, while the physics-informed learning methodology has been widely applied to feed-forward neural networks [10], its applicability to other notable neural network architectures (e.g., recurrent neural network (RNN) that has been widely used to model nonlinear dynamic systems) remains to be investigated. Furthermore, theoretical analysis of the generalization performance that characterizes the model's ability to sufficiently adapt to previously unseen data of the same distribution has not been reported, and the application of physics-based machine learning modeling in advanced control methods (e.g., MPC) is still in its infancy [12].

Motivated by the above considerations, in this manuscript, we develop a general physics-based modeling framework for constructing RNN models and formulating a physics-informed RNN (PIRNN)-based predictive control scheme for nonlinear systems. Specifically, we first present a hybrid modeling approach (i.e., PIRNN) that integrates data-driven and physics-informed techniques, following the framework outlined in Ref. [10]. Next, we develop the analysis of the generalization performance of PIRNN on the nominal nonlinear process model, and incorporate the PIRNN model into the MPC scheme. Finally, the effectiveness of the PIRNN-MPC scheme is demonstrated through a continuous stirred tank reactor (CSTR) example with Gaussian measurement noise, where the closed-loop performance of the PIRNN-MPC is compared with that of the first-principles model-, a purely data-driven RNN model-, and a purely physics-based RNN model-MPC control schemes.

2. Preliminaries

2.1. Notation

The operator $|\cdot|$ represents the Euclidean norm of a vector. x^T is the transpose of a vector x . $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$ represents the standard Lie derivative. The notation “ $\setminus$ ” denotes set subtraction, i.e., $A \setminus B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$. The function $f(\cdot)$ is said to belong to the differentiability class C^1 if its derivative $f'(\cdot)$ exists and is continuous in its domain. A continuous function $\alpha : [0, a) \rightarrow [0, \infty)$ is of class $\mathcal{K}$ if it is strictly increasing and is zero only when evaluated at zero. $\mathbb{E}[X]$ is the expected value of a random variable X , and $\mathbb{P}(A)$ is the probability of the event A occurring.

2.2. Class of nonlinear systems

In this work, the class of continuous-time nonlinear dynamic systems, described by the first-order ODE of the following general state-space form, is considered.

$$\dot{x} - F(x, u) := \dot{x} - f(x) - g(x)u = 0, \quad x(t_0) = x_0 \quad (1a)$$

$$y = x + \omega \quad (1b)$$

where $x \in \mathbf{R}^n$ is the state vector of the nonlinear system. $x(t)$ denotes the solution of the ODE, $u \in \mathbf{R}^m$ is the manipulated input vector, $y \in \mathbf{R}^n$ represents the real-time sensor measurements of the state vector, and $\omega \in \mathbf{R}^n$ denotes the random Gaussian noise vector bounded by $|\omega| \leq \omega_m$, $\omega_m > 0$. The constraint on the manipulated variable u is defined by $u \in U := \{u_{\min} \leq u \leq u_{\max}\} \subset \mathbf{R}^m$, where $u_{\min}$ and $u_{\max}$ represent the lower and upper bounds (i.e., physical limits) of the manipulated input, respectively. $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth vector and matrix function of dimensions $n \times 1$, and $n \times m$, respectively. Without loss of generality, it is assumed that the initial time $t_0 = 0$ and $f(0) = 0$, and the origin is therefore a steady-state of the nominal system (i.e., $\omega(t) \equiv 0$) of Eq. (1), with $(x_s^*, u_s^*) = (0, 0)$ denoting the steady-state system state, and manipulated input vectors, respectively. In addition, it is also assumed that the full system states are measurable and are readily available in real time.

2.3. Stabilization via Lyapunov-based controller

Consider the nonlinear system of Eq. (1a) for which noise-free state measurements are attainable (i.e., $\omega(t) \equiv 0$ and $y(t) = x(t)$). In order to guarantee the stabilizability of the closed-loop system, it is assumed that there exists a stabilizing control law $u = \Phi(x) \in U$ (e.g., the universal Sontag control law [21]) that is capable of rendering the origin of the noise-free nominal system of Eq. (1a) exponentially stable. This stabilizability assumption then stipulates the existence of a C^1 Control Lyapunov function $V(x)$ such that the following inequalities hold for all x in an open neighborhood D around the origin:

$$c_1|x|^2 \leq V(x) \leq c_2|x|^2, \quad (2a)$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \leq -c_3|x|^2, \quad (2b)$$

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

where c_1, c_2, c_3 and c_4 are positive constants. $F(x, u)$ denotes the nonlinear dynamic system of Eq. (1a). $\phi_u \in \mathbf{R}^n$ is characterized as a set of states where the inequalities of Eq. (2) are satisfied under the controller $u = \Phi(x) \in U$. The closed-loop stability region Ω_ρ for the nonlinear dynamic system of Eq. (1a) is then defined as a level set of the Lyapunov function embedded in ϕ_u , i.e., $\Omega_\rho := \{x \in \phi_u \mid V(x) \leq \rho\} \subset \phi_u$, where $\rho > 0$.

3. Physics-informed (PI) RNN

In this section, a synergistic way of integrating and embedding mechanistic knowledge into the neural network's learning (i.e., physics-informed (PI) learning), as proposed in Ref. [10], is introduced. The PI learning strategy integrates the information from both actual measurements and physical laws (e.g., process dynamics governed by Eq. (1a)) by incorporating the governing ODEs as additional penalty terms into the loss function of an RNN. Consequently, the introduction of physics-based penalty terms imposes a priori structure dictated by the physical laws on the RNN, which improves the extrapolating capability of the model (e.g., predicting system dynamics beyond the range of training data) and the model generalization performance.

3.1. Recurrent neural network (RNN) model

In contrast to the one-way connectivity between neurons in a conventional feedforward NN, signals in recurrent neural networks (RNN) travel in bi-directions (i.e., forward and backward) due to the introduction of recurrent connections (i.e., loops) in the network architecture. This permits the feedback of signals derived from previous inputs back into the network, thereby exhibiting a temporal dynamic behavior which inherently resembles the numerical procedures (e.g., the explicit Euler method) for solving an ODE. Therefore, the architecture of the RNN is especially apposite for modeling nonlinear continuous systems governed by ODEs. Furthermore, as a powerful dynamic modeling approach, the application of RNN has achieved remarkable results across a wide range of disciplines (e.g., natural language processing [22], speech recognition [23], computational biology [24], and general time-series forecasting involving sequential data [25]).

In this work, a recurrent neural network (RNN) model that approximates the nonlinear dynamics of the system of Eq. (1) is developed, which can be represented by the following form:

$$\dot{\tilde{x}} = F_{nn}(\tilde{x}, u) := A\tilde{x} + \Psi^T z \quad (3)$$

where $\tilde{x} \in \mathbf{R}^{d_x}$ is the RNN state vector and $u \in \mathbf{R}^{d_u}$ is the manipulated input vector. $z = [z_1, \dots, z_{d_x}, z_{d_x+1}, \dots, z_{d_x+d_u}] = [\sigma(\tilde{x}_1), \dots, \sigma(\tilde{x}_{d_x}), u_1, \dots, u_{d_u}] \in \mathbf{R}^{d_x+d_u}$ is a vector of both the network state $\tilde{x}$ and the input u , and $\sigma(\cdot)$ is the nonlinear activation function (e.g., *ReLU* function $\sigma(x) = \max\{0, x\}$ and *Tanh* function $\sigma(x) = (e^x - e^{-x})/(e^x + e^{-x})$). A and Ψ are the coefficient matrices comprising RNN weights, which will be optimized during training. Throughout the manuscript, x will be used to represent the state vector of the actual nonlinear system of Eq. (1), and $\tilde{x}$ will be used to denote the state vector of the RNN of Eq. (3). To simplify the notation, we will use θ to represent a matrix of all the RNN weights to be optimized.

Traditionally, the development of an RNN model for the continuous-time nonlinear system employs a three-step approach:

1. A comprehensively large dataset consisting of state trajectories from diverse initial conditions in the stable operating region (i.e., $x_0 \in \Omega_p$) and control actions spanning their physical operating limits (i.e., $u \in U$) is collected through conducting extensive experimental runs of the nonlinear system of Eq. (1) in a sample-and-hold fashion (i.e., the manipulated inputs are maintained at constant values within each sampling period Δ , where $u(t) = u(t_k)$, $\forall t \in [t_k, t_{k+1})$, and $t_{k+1} := t_k + \Delta$).
2. A machine learning framework (e.g., Tensorflow and PyTorch) is utilized to develop and train an RNN to obtain the optimal weights (i.e., the coefficient matrices A and Ψ) by minimizing the loss function that encompasses the mean squared error (MSE) between the predicted and actual system states from the RNN and the sensor measurement, respectively.
3. The predictive performance of the trained RNN model will be evaluated using a test dataset that has been intentionally excluded from the training process, where the model outputs will be compared with the actual state trajectories acquired.

However, although the universal approximation theorem states that an RNN with a suitable architecture is capable of approximating any nonlinear dynamic system to an arbitrary accuracy on compact subsets of the state space for finite time [26], algorithmic learnability of the optimal neural network weights is not guaranteed. In reality, pinpointing the optimal weights and model hyperparameters (e.g., number of layers and recurrent

units) for the regression tasks of a large-scale and complex nonlinear system is a multifaceted problem, which is often demanding due to the involvement of various factors such as the complexity of neural network structure, availability of computing resources, and feasibility of optimization algorithms. Therefore, improving the performance of neural networks has been a long-standing challenge for researchers in machine learning community over the past few decades, where significant efforts have been made to optimize neural network architecture, develop advanced optimization algorithms, and improve data quality (e.g., via design of experiments).

In this work, we will improve the performance of RNN models by incorporating domain knowledge of the nonlinear system into the RNN development, as proposed in Ref. [10]. The formulation of physics-informed (PI) RNN will be presented in the next subsection.

Remark 1. Although the formulation of RNN shown in Eq. (3) represents an RNN model with only one hidden layer (i.e., Eq. (3) illustrates an RNN architecture comprising three layers: an input layer, a hidden layer, and an output layer) for the purpose of simplifying the discussion of approximating the nonlinear system of Eq. (1) with RNN, its actual development is not confined, and can be extended to multiple hidden layers for better approximation performance. It should be noted that the introduction of additional hidden layers solely increases the number of RNN parameters (or weights), and hence the model complexity, and has no impact on the number of outputs of the model. Furthermore, in this work, since the RNN model is developed to predict future system states given the current state measurements and manipulated inputs, its internal states would correspond directly to the process states output by the nonlinear dynamic system of Eq. (1a) (when given the same initial states and manipulated inputs) at various time steps.

3.2. Regularization through physics-guided loss function

Physics-guided learning effectively harnesses mechanistic knowledge encapsulated in process data and mathematical models for building accurate and robust surrogate RNN models. The incorporation of the physical-law-based regularization terms as soft constraints in the loss function furnishes an RNN with a priori knowledge by reasonably and informatively restricting the weights of the neural network, which in turn reinforces the learning and identification of solutions that are consistent with the physics-based model. The physics-informed learning can often be viewed as a subset of multi-task learning where the RNN model is concurrently enforced to fit the observed data (e.g., data from sensors), and to generate outputs that conform to the physics-based constraints (e.g., governing equation of Eq. (1a)). To simplify the discussion, we assume that the full process model of Eq. (1a) is known for the design of physics-informed RNNs in this section. However, it should be clarified that the proposed PIRNN modeling approach does not necessarily require the knowledge of a full process model. In other words, the proposed PIRNN modeling approach can be applied when the first-principles model is partially known. For example, consider a real-world system where data is only available for a certain number of system states that can be easily measured. We can take advantage of the first-principles model that describes the unmeasured states (i.e., partial physical knowledge), and incorporate it into physics-informed RNNs, together with training data for measured states. In this way, the resulting RNN model with partial knowledge of the first-principles model and limited training data will fit well on the provided training data points, and is also able to predict the dynamics of the unmeasured states based on prior physical knowledge. The integration of (partial) physical knowledge

with the process data available improves the applicability of ML modeling approaches and the generalization performance over a wider range of operating conditions.

To put it formally, a new function $\mathcal{G}$ is first defined to be the equivalent of the left-hand side of Eq. (1a).

$$\mathcal{G}(\tilde{x}, u) := \dot{\tilde{x}} - F(\tilde{x}, u) \quad (4)$$

where $\tilde{x}$ represents the output of an RNN, which accepts the initial state vector x_0 of the dynamic system, and the manipulated input u as its inputs. $\tilde{x} = h(z; \theta)$, $z = [x_0, u]$ denotes the RNN input vector, and $h(z; \theta)$ is the approximation of the solution x of the nonlinear system of Eq. (1a) for time $0 \leq t \leq \Delta$ subject to the same x_0 and u . Consequently, as the RNN learns to compute the solution x of the ODE governed by Eq. (1a), the following expression holds.

$$\mathcal{G}(\tilde{x}, u) := \dot{\tilde{x}} - F(\tilde{x}, u) \approx 0, \quad t \in [0, \Delta] \quad (5)$$

Specifically, the PIRNN $h(z; \theta)$ is trained using an optimizer such as Adam and two groups of data, which correspond to their respective penalty terms presented in the mean squared error (MSE) loss function (Eq. (6)). The first group of data is involved in the computation of $MSE_{\mathcal{X}}$ of Eq. (6), and consists of the process data captured by sensors from experiments or past operations that resemble the training data used in a typical regression problem. The second group of data pertains to $MSE_{\mathcal{G}}$, and comprises only the collocation points of the initial state vector and manipulated inputs that are randomly and uniformly sampled from the designated operating regions (Ω_{ρ}) where the RNN approximation (Eq. (5)) should be valid.

$$MSE = w_{\mathcal{X}} MSE_{\mathcal{X}} + w_{\mathcal{G}} MSE_{\mathcal{G}} \quad (6)$$

where

$$MSE_{\mathcal{X}} = \frac{1}{N_{\mathcal{X}}} \sum_{n=1}^{N_{\mathcal{X}}} \frac{1}{N_T} \sum_{i=0}^{N_T} |x_n(t_{\mathcal{X}}^{(i)}, u_{\mathcal{X},n}) - \tilde{x}_n(t_{\mathcal{X}}^{(i)}, u_{\mathcal{X},n})|^2 \quad (7)$$

$$MSE_{\mathcal{G}} = \frac{1}{N_{\mathcal{G}}} \sum_{n=1}^{N_{\mathcal{G}}} \frac{1}{N_T} \sum_{i=0}^{N_T} |\mathcal{G}(\tilde{x}_n(t_{\mathcal{G}}^{(i)}, u_{\mathcal{G},n}), u_{\mathcal{G},n})|^2 + \gamma \frac{1}{N_{\mathcal{G}}} \sum_{n=1}^{N_{\mathcal{G}}} |x_n(t_{\mathcal{G}}^{(0)}, u_{\mathcal{G},n}) - \tilde{x}_n(t_{\mathcal{G}}^{(0)}, u_{\mathcal{G},n})|^2 \quad (8)$$

where $N_{\mathcal{X}}$, $N_{\mathcal{G}}$, and N_T denote the number of training data, collocation points, and outputs of RNN at time $0 \leq t \leq \Delta$, respectively. $MSE_{\mathcal{X}}$ and $MSE_{\mathcal{G}}$ represent the MSE losses with respect to the collected process data (i.e., sensor measurements), and the randomly sampled collocation points, respectively. γ denotes the coefficient that serves to balance the magnitude of the two loss terms in $MSE_{\mathcal{G}}$. $w_{\mathcal{X}}$ and $w_{\mathcal{G}}$ are coefficient weights that balance the scale of the two MSE loss terms. These coefficients are usually user-defined to adjust the interplay between the two contributing MSE loss terms and to facilitate the training of RNN. Subscripts $\mathcal{X}$ and $\mathcal{G}$ denote the loss terms with respect to the conventional MSE loss, and the physics-based loss, respectively. With a slight abuse of notation, we use $x_n(t_{\mathcal{G}}^{(i)}, u_{\mathcal{G},n})$ to represent the solution of the ODE for time $0 \leq t \leq \Delta$ under the initial condition $x_n(t_{\mathcal{G}}^{(0)})$ at $t = t_0$ and the manipulated inputs $u = u_{\mathcal{G},n}$, which are kept constant for all $t \in [0, \Delta]$. $\tilde{x}_n(t_{\mathcal{G}}^{(i)}, u_{\mathcal{G},n})$ represents the desired output from the RNN at time $t_{\mathcal{G}}^{(i)}$, and under the manipulated inputs $u_{\mathcal{G},n}$. The superscript (i) in $t^{(i)}$ is the time index. It should be noted that the training data garnered for calculating the first loss term $MSE_{\mathcal{X}}$ are directly captured from the actual nonlinear system near a set of desired setpoints, and the collocation points are thus sampled predominantly from operating regions that are beyond the range of training data.

Consequently, the two loss terms $MSE_{\mathcal{X}}$ and $MSE_{\mathcal{G}}$ collectively aim to enforce the dynamic behavior that conforms to the physical laws (e.g., mass and energy balances) while contemporaneously forcing the best possible fit to plain training data points, and this improves the interpretability of PIRNNs in the operating region with no data available. It should also be noted that, as denoted in Eq. (7)–(8), the output of the PIRNN model consists of the system states from $t = t_0$ onward (instead of starting from $t = t_1$), and this was found to facilitate the learning process especially for minimizing the physics-loss $MSE_{\mathcal{G}}$. In addition, when the PIRNN is trained solely by minimizing the loss function of Eq. (7), the training process resembles the standard learning algorithm that seeks minimize the MSE difference between the predicted and the labeled output data. On the other extreme end where no labeled data is available, the PIRNN model can be trained entirely based on Eq. (8) only, for which the model will eventually conform to the dynamic behavior dictated by the governing ODE at the given initial conditions.

Remark 2. In case the process model $F(x, u)$ is not fully known or involves unknown parameters, the inverse problem of physics-informed machine learning has been recently developed to learn unknown parameters from data (e.g., Ref. [10,16,27]). Specifically, a new function $\mathcal{G}(\tilde{x}, u, \lambda) := \dot{\tilde{x}} - F(\tilde{x}, u, \lambda)$, similar to Eq. (4), can be developed in the inverse problem to learn unknown parameters λ in the incomplete physics model. The inverse problem of physics-informed machine learning is essentially a multi-task learning where the neural network model is trained to not only achieve the best possible fit to the observed data, but also estimate the values of λ that conform to the physical laws. After λ is identified, we can then carry out the forward problem proposed in this manuscript to generate sufficient collocation points from $F(\tilde{x}, u, \lambda)$ to learn the dynamics of the given system. Additionally, in case the state functions are unknown for some states in $F(x, u)$, physics-informed machine learning method can include the state functions that are fully known only, since these are the physical knowledge we have.

Remark 3. The proposed physics-informed machine learning method provides a solution to improving model performance in the presence of insufficient data with the help of physical knowledge. In addition to improved generalization performance with the use of physical knowledge, physics-informed NNs can potentially improve the computational efficiency compared to first-principles model (if known) due to the nature of the matrix operations involved in NNs. It has been demonstrated in many recent works that NNs can significantly reduce the computational cost for solving nonlinear dynamic systems (e.g., Ref. [2,28–31]). Furthermore, another benefit of conventional NNs or physics-informed NNs is their ability to model a wide variety of functions and the possibility to reduce the time and cost associated with physical model's design, tuning, and adaptation to new plant's operating conditions (see more discussions in Ref. [32]).

3.3. Physics-informed RNN training

The PIRNN model is developed using PyTorch, an open-source machine learning framework based on the Torch library [33]. The inputs to the PIRNN model consist of the system initial state vector x_0 , the manipulated input u , and the (optional) times t at which the future system states are to be predicted (i.e., as outputs of the model). In this work, the nonlinear dynamic system is assumed to be predominantly operating at a specific steady-state setpoint, and hence the process data collected only capture the system dynamics in a small neighborhood around the setpoint. Note that this assumption is made to better demonstrate the

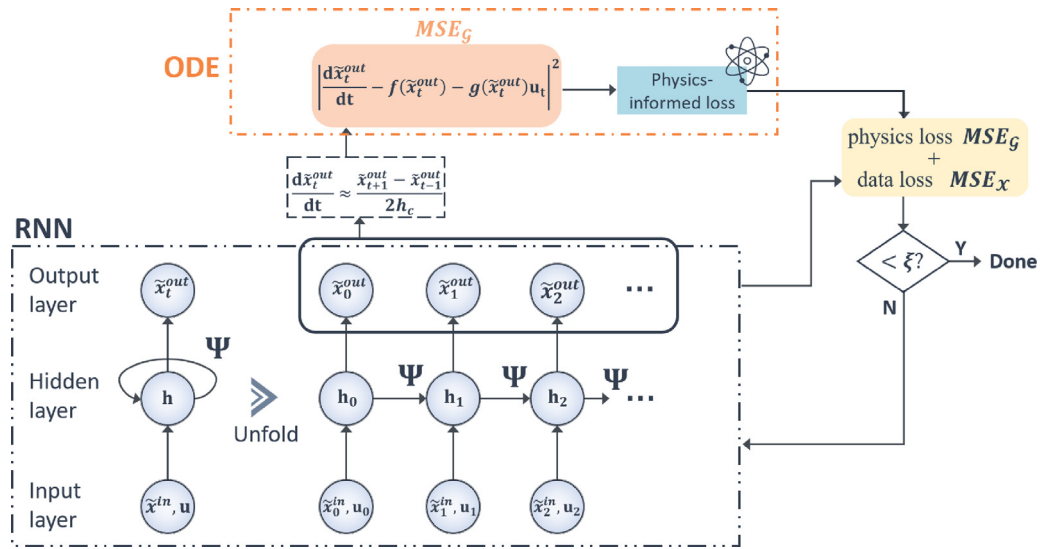


Fig. 1. Schematic of the PIRNN model with data-driven MSE_X and physics-guided MSE_G regularization terms embedded into the loss function, where the derivatives can be computed using the finite difference method based on the outputs of the RNN model. The PIRNN model will be fine-tuned repeatedly until its outputs attain a satisfactory conformance (e.g., $MSE < \xi$) to the governing ODEs and the training data by minimizing both the physics and data losses simultaneously.

improved generalization performance of PIRNNs in the entire operating region in the absence of sufficient data, compared to conventional data-driven modeling approaches. It should also be clarified that the limited data assumption made in this section is not a requirement for the implementation of the proposed PIRNN approach. It is used to showcase the benefits of PIRNN over purely data-driven modeling approaches, as the latter approaches can certainly achieve a sufficiently small modeling error when data is exhaustive (i.e., the process data is available for the entire operating region).

In this work, the process data is developed by numerically solving the nonlinear dynamic system of Eq. (1a) using the Explicit Euler method with an integration time step of h_c , and with initial states x_0 and manipulated inputs u randomly sampled in a small region around their respective steady-state setpoints. These state measurements are subject to the measurement noise of Eq. (1b) to mimic data retrieved from real-life process operations. The collocation points are developed by randomly and uniformly sampling x_0 and u beyond the setpoints and across the entire stability region Ω_ρ , and the physical limits of the manipulated inputs U , respectively. Specifically, no labeled data is required for training using the collocation points. Instead, the ODE is explicitly embedded in the loss function of the RNN in the way that the LHS of the ODE ($\dot{x} = F(x, u)$) is calculated based on the RNN predicted states, where the derivative is approximated using the finite difference method. The RHS of the ODE (i.e., the function $F(x, u)$) is computed by substituting the PIRNN predicted states and the manipulated inputs into $F(x, u)$. The second loss term MSE_G in the loss function of Eq. (8) is to minimize the error between the LHS and RHS of the ODE using the PIRNN prediction results so that the PIRNN can learn the underlying physical principles. Therefore, in principle, numerical solution to the ODE is not required for the proposed PIRNN approach, which can significantly reduce the computation time for training a PIRNN.

In addition, scaling factors w_X and w_G are incorporated into the total loss function (i.e., $MSE = w_X \cdot MSE_X + w_G \cdot MSE_G$) to ensure both MSE loss terms are of approximately equal scale. Specifically, while MSE_X is in exact analogous to the conventional MSE loss employed in a typical ML regression task, the calculation of MSE_G demands the determination of the both the derivative of $\tilde{x}$ (i.e., $\dot{\tilde{x}}$) and the standard MSE between the predicted states at $t = t_0$ and the initial states supplied to the PIRNN

(i.e., $\frac{1}{N_G} \sum_{n=1}^{N_G} |x_n(t_G^{(0)}, u_{G,n}) - \tilde{x}_n(t_G^{(0)}, u_{G,n})|^2$). As shown in Fig. 1, $\dot{\tilde{x}}$ can be approximated by the finite difference method using the predicted outputs from the PIRNN directly. h_c is the integration time step (i.e., time interval between two successive time steps).

Furthermore, we consider the PIRNN model that is developed following the same structure of the standard RNN model of Eq. (3) can be rewritten in the following discrete-time form: $\mathbf{h}_t = \sigma_h(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t)$, $\mathbf{y}_t = \sigma_y(\mathbf{V}\mathbf{h}_t)$, where $\mathbf{x}_t$, $\mathbf{h}_t$, and $\mathbf{y}_t$ represent the PIRNN inputs, hidden states, and outputs at the t_{th} time step, $t = 1, \dots, T$, respectively. $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{V}$ are the weight matrices for the input layer, the hidden layer, and the output layer, respectively. The nonlinear activation functions in the hidden and output layers are denoted by σ_h and σ_y , respectively. The following conditions are assumed to hold for the construction and training of PIRNN models. Specifically, the assumption on the boundedness of RNN inputs and weight matrices in Assumption 1 is consistent with our data generation method that only considers a bounded set around the steady-state of the nonlinear system of Eq. (1) under bounded manipulated inputs. Assumption 2 is a standard requirement in NN training to ensure all the data has the same distribution, and it is also a requirement for the analysis of the generalization error in the next section. Assumption 3 is an assumption on the activation functions of the PIRNN model. An example of an activation function satisfying Assumption 3 is *ReLU*. In addition, it is noted in Ref. [34] that the requirement on positive homogeneity is not mandatory for the derivation of generalization error bound, and can be further relaxed to allow more activation functions (e.g., *Tanh*) to be chosen for PIRNN.

Assumption 1. The PIRNN inputs and weight matrices are bounded, i.e., $|\mathbf{x}_{i,t}| \leq B_{X,i}$, for all $i = 1, \dots, m$ and $t = 1, \dots, T$, and $\|\mathbf{W}\|_F \leq B_{W,F}$, $\|\mathbf{U}\|_F \leq B_{U,F}$, $\|\mathbf{V}\|_F \leq B_{V,F}$, where $\|\cdot\|_F$ denotes the Frobenius norm.

Assumption 2. All the datasets (i.e., training, validation, and testing) are drawn from the same distribution.

Assumption 3. The activation function for the hidden layers is a 1-Lipschitz continuous activation function, and is positive-homogeneous in the sense that $\sigma_h(\alpha z) = \alpha \sigma_h(z)$ holds for all $\alpha \geq 0$ and $z \in \mathbf{R}$.

Remark 4. It should be noted that the calculation of Eq. (4) is based on the PIRNN predicted states $\tilde{x}$ instead of state measurement data. The predicted time-derivative of state $\dot{\tilde{x}}$, and the nonlinear function $F(\tilde{x}, u)$ are both calculated using the PIRNN predicted states $\tilde{x}(t)$, $\forall t \in [0, \Delta]$. Specifically, as shown in Fig. 1, based on the PIRNN model that is developed to predict the entire state vector (the PIRNN output could include the prediction of both measured and unmeasured states), $\tilde{x}$ is approximated using numerical method, and $F(\tilde{x}, u)$ is calculated by substituting the predicted $\tilde{x}$ and u into the first-principles model. Furthermore, for those states that can be measured, the conventional MSE loss term (i.e., Eq. (7)) can be added into the overall loss function to minimize the error between PIRNN predicted output and observational data, such that the prediction accuracy can be further improved. However, it should also be noted that the conventional MSE loss term and measurement data are not mandatory in the framework of physics-informed machine learning. Therefore, regardless of whether the state is measurable or not, Eq. (4) can be calculated based on PIRNN predicted states and the process model $F(\tilde{x}, u)$. If only a partially-known process model is available, then the physics-informed loss term of Eq. (4) can include the state functions that are fully known, where the calculations are still based on the full state vector predicted by the PIRNN.

Remark 5. The power of neural networks can only be fully harvested given suitable architectures and hyperparameters to yield a desired prediction performance. However, manual hyperparameter tuning through grid search has proven to be an arduous task even for expert ML practitioners, where tedious episodes of trial and error are often required. To this end, Optuna proffers a convenient and accessible tool for efficient automatic hyperparameter tuning, and is especially useful for determining the optimal hyperparameters of the PIRNN (e.g., the number of hidden recurrent layers and units, and the learning rate) in this work. In essence, Optuna is a resource-effective automatic hyperparameter optimization framework that adopts the state-of-the-art Bayesian optimization algorithms for sampling hyperparameters while pruning unpromising trials to maximize the tuning efficiency. Interested readers may refer to Refs. [35–37] for a more detailed discussion on Optuna and Bayesian optimization.

4. Generalization error analysis for physics-informed RNN

In this section, we will evaluate the generalization performance of the PIRNN model of Eqs. (3)–(5) by developing a generalization error bound using statistical learning theory. In our previous work, a generalization error bound for purely data-driven RNN models with one loss term (i.e., the standard MSE loss of the training data) in the loss function was derived in Ref. [38]. With respect to physics-informed NNs, the generalization error bounds for physics-informed feedforward NNs approximating solutions of the forward and inverse problems for PDEs were derived in Refs. [39,40], respectively. However, the generalization error bounds in Refs. [39,40] were derived for a general class of data-driven models that do not account for the architecture of NNs, and therefore, the generalization error results cannot be used to guide the construction of NNs in terms of the number of neurons and layers, etc. In this section, we derive – for the first time – a generalization error bound for physics-informed RNN models that account for the physics-based regularization term in loss function, and the architecture of RNNs, where the proof technique of Rademacher complexity for RNNs follows Ref. [38]. Specifically, we consider the extreme scenario where no historical data is available, and only the initial conditions x_0 and the first-principles model of Eq. (1a) are used to develop the PIRNN model. In other words, the PIRNN will be trained using the loss function

of Eq. (8) only to fit the initial condition x_0 and capture the nonlinear dynamics of Eq. (1a). Let $\tilde{x} = h(z; \theta)$ denote the predicted output of the PIRNN model, where $z = [x_0 \ u]$ is the PIRNN input and θ is a set of all the PIRNN weight matrices. The generalization error for the PIRNN model $h(z, \theta)$, with z drawn independently from some underlying distribution $z \in Z$, is defined as follows, where Z is the set of bounded states $x_0 \in \Omega_\rho$ and bounded manipulated inputs $u \in U$.

$$R_D(\theta) := \mathbb{E}_{z \sim Z} [L(\mathcal{G}(h(z; \theta), u), 0)] + \gamma \mathbb{E}_{z \sim Z} [L(\tilde{x}_0, x_0)] \quad (9)$$

$L(\cdot, \cdot)$ denotes the loss function, where the MSE loss is used in this study. γ is a positive real number that balances the two loss terms in Eq. (9). $\tilde{x}_0 = \tilde{x}(t = 0)$ is the predicted initial condition (i.e., the first element in the PIRNN model output vector $\tilde{x}(t) = h(z; \theta)$, $\forall t \in [0, T]$). Since the generalization error $R_D(\theta)$ is difficult to compute due to the unknown probability distribution Z , it is generally approximated by an empirical error calculated using a finite set of training samples z_i drawn from the same distributions Z as follows.

$$R_S(\theta) := \frac{1}{N_G} \sum_{i=1}^{N_G} L(\mathcal{G}(h(z_i; \theta), u), 0) + \gamma \frac{1}{N_G} \sum_{i=1}^{N_G} L(\tilde{x}_{0,i}, x_{0,i}) \quad (10)$$

where N_G is the number of training samples from the same distribution Z . Note that in Eq. (9), the first term of the generalization error represents the loss term computed based on the first-principles model of Eq. (4) and the second term penalizes the deviation between the predicted initial condition and its actual value. To achieve a desired training performance in the sense that the PIRNN model captures both the dynamics of Eq. (4) and fits the initial condition x_0 , the weight parameter γ needs to be carefully chosen to balance the two loss terms. Therefore, to mitigate the impact of γ on the overall training performance, and also to simplify the analysis of the PIRNN generalization error by maintaining only one loss term in the loss function, we design a new PIRNN model $\tilde{h}(z; \theta)$ that approximates the original PIRNN model $h(z; \theta)$.

$$h(z; \theta) = (t - t_0)\tilde{h}(z; \theta) + x_0 \quad (11)$$

It can be observed from Eq. (11) that $h(z; \theta) = x_0$ holds when $t = t_0$ regardless of the choice of $\tilde{h}(z; \theta)$. In other words, through the design of the new PIRNN model $\tilde{h}(z; \theta)$, the initial condition can be predicted accurately by $h(z; \theta)$ for any $\tilde{h}(z; \theta)$. Therefore, it remains to show that the approximation of $h(z; \theta)$ through Eq. (11) can capture the nonlinear dynamics of Eq. (4). Specifically, by substituting Eq. (11) into Eq. (4), we have the following equation:

$$\tilde{\mathcal{G}}(\tilde{h}(z; \theta), u) := (t - t_0)\dot{\tilde{h}}(z; \theta) + F((t - t_0)\tilde{h}(z; \theta) + x_0, u) + \tilde{h}(z; \theta). \quad (12)$$

As a result, the new PIRNN model $\tilde{h}(z; \theta)$ is developed by minimizing the loss function $L(\tilde{\mathcal{G}}(\tilde{h}(z; \theta), u), 0)$ only, while the initial condition can be readily satisfied using Eq. (11). The optimal weight matrices of the new PIRNN model $\tilde{h}(z; \theta_D)$ can be obtained as follows.

$$\theta_D = \arg \min_{\theta} R_D(\theta) := \mathbb{E}_{z \sim Z} [L(\tilde{\mathcal{G}}(\tilde{h}(z; \theta), u), 0)]. \quad (13)$$

Using Eq. (11), the PIRNN model $h(z; \theta_D)$ that meets both the initial condition and the physical law of Eq. (4) is given by

$$h(z; \theta_D) = (t - t_0)\tilde{h}(z; \theta_D) + x_0 \quad (14)$$

Similar to Eq. (10), the optimization of the PIRNN weights can be performed based on the minimization of the empirical error as follows.

$$\theta_S = \arg \min_{\theta} R_S(\theta) := \frac{1}{N_G} \sum_{i=1}^{N_G} L(\tilde{\mathcal{G}}(\tilde{h}(z; \theta), u), 0) \quad (15)$$

Finally, the PIRNN model that approximates the nonlinear system of Eq. (1a) based on the training dataset is obtained as $h(z; \theta_S) = (t - t_0)\tilde{h}(z; \theta_S) + x_0$. Therefore, through the design of PIRNN model $h(z; \theta_D)$ using Eq. (11), the training process for the original PIRNN model $h(z; \theta_D)$ using the loss function of Eq. (9) is equivalent to training the new PIRNN model $\tilde{h}(z; \theta_D)$ using the new loss function of Eq. (13). Additionally, since the new loss function of Eq. (13) only includes one loss term, and does not depend on the weight parameter γ , the generalization error analysis developed in Ref. [38] can be adopted to develop a generalization error bound for the new PIRNN model $\tilde{h}(z; \theta_D)$ obtained from Eq. (13). Specifically, due to the boundedness of PIRNN inputs and outputs implied by the boundedness of process states and manipulated inputs, as well as the Lipschitz property assumed for the nonlinear function $F(x, u)$, we can show the MSE loss function satisfies the following inequality from local Lipschitz continuity property, for all $t \in [0, T]$, $u \in U$, $z \in Z$, and θ satisfying **Assumptions 1–3**:

$$|L(\tilde{g}(\tilde{h}_1(z; \theta), u), 0) - L(\tilde{g}(\tilde{h}_2(z; \theta), u), 0)| \leq L_r \|\tilde{h}_1(z; \theta) - \tilde{h}_2(z; \theta)\| \quad (16)$$

where the local Lipschitz constant is denoted by L_r . To derive the generalization error bound for the PIRNN model, we will introduce a notion termed Rademacher complexity that is widely used in statistical learning theory to quantify the richness of a function class. Specifically, given a set of data samples $S = \{s_1, \dots, s_m\}$, and a hypothesis class $\mathcal{H}$ of real-valued functions $h(\cdot)$, the empirical Rademacher complexity of $\mathcal{H}$ is defined as follows [41].

$$\mathcal{R}_S(\mathcal{H}) = \mathbb{E}_\epsilon \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \epsilon_i h(s_i) \right] \quad (17)$$

where $\epsilon = (\epsilon_1, \dots, \epsilon_m)^T$, 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 subscript ϵ of expectation will be omitted in the following text for simplicity.

Let $\mathcal{H}$ denote the hypothesis class of PIRNN models $\tilde{h}(z; \theta)$ that maps the input $z \in \mathbf{R}^{d_x \times t}$ (i.e., the first t -time-step inputs) to the output $y_t \in \mathbf{R}^{d_y}$ (i.e., the t th output), and $\mathcal{L}_t$ be the loss function set associated with $\mathcal{H}$.

$$\mathcal{L}_t = \{l_t : (z, \bar{y}) \rightarrow L(\tilde{g}(\tilde{h}(z; \theta), u), 0), \tilde{h} \in \mathcal{H}\} \quad (18)$$

where $\bar{y}$ and z are the true output vector and the PIRNN input vector, respectively. Then, given a PIRNN model trained using m i.i.d. data samples, the following inequality [41] holds with probability at least $1 - \delta$ for all $l_t \in \mathcal{L}_t$ over the data samples $S = (z_{i,t}, \bar{y}_{i,t})_{i=1}^T$, $i = 1, \dots, m$:

$$\mathbb{E}[l_t(z, \bar{y})] \leq \frac{1}{m} \sum_{i=1}^m l_t(z_i, \bar{y}_i) + 2\mathcal{R}_S(\mathcal{L}_t) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \quad (19)$$

Eq. (19) demonstrates that the generalization error bound depends on the training loss (first term), the Rademacher complexity of the PIRNN hypothesis class (second term), and a function with respect to the confidence δ and the training sample size m (last term). Therefore, to compute the generalization error bound of Eq. (19), it remains to show that an upper bound can be derived for the Rademacher complexity term, since both the training loss and the last term can be readily obtained once the training process is completed and a confidence level δ is chosen. Following the proof techniques in Ref. [38], we first introduce the following contraction inequality to generalize the Rademacher complexity of real-valued functions in Eq. (17) to vector-valued functions since the PIRNN model is developed for a class of multiple-input and multiple-output nonlinear system of Eq. (1a).

Proposition 1 (c.f. Corollary 4 in [42]). Consider a hypothesis class $\mathcal{H}$ of vector-valued functions $\tilde{h} \in \mathbf{R}^{d_y}$, and an L_r -Lipschitz function $L(\cdot, \cdot)$ that maps $\tilde{h} \in \mathbf{R}^{d_y}$ to $\mathbf{R}$. Given a set of data samples $S = \{s_1, \dots, s_m\}$, the following inequality holds.

$$\mathbb{E} \left[\sup_{\tilde{h} \in \mathcal{H}} \sum_{i=1}^m \epsilon_i L(\tilde{g}(\tilde{h}(z_i; \theta), u_i), 0) \right] \leq \sqrt{2} L_r \mathbb{E} \left[\sup_{\tilde{h} \in \mathcal{H}} \sum_{i=1}^m \sum_{k=1}^{d_y} \epsilon_{ik} \tilde{h}_k(z_i; \theta) \right] \quad (20)$$

where $\tilde{h}_k(\cdot)$ is the k th component in the vector-valued function $\tilde{h}(\cdot)$, and ϵ_{ik} is an $m \times d_y$ matrix of independent Rademacher variables.

Using the contraction inequality of Eq. (20), the following proposition is developed for the class of loss function $\mathcal{L}_t$ associated with the vector-valued PIRNN models. The proof is omitted here as it closely follows the proof of Theorem 1 in Ref. [38]. The key steps are to derive an upper bound for the Rademacher complexity term in Eq. (19) through a “peeling” approach that peels off the activation functions and the weights layer by layer. Interested readers may refer to Ref. [38] for the full proof of this proposition.

Proposition 2 (c.f. Theorem 1 in [38]). Given a dataset $S = (z_{i,t}, \bar{y}_{i,t})_{i=1}^T$, $i = 1, \dots, m$ with m i.i.d. data samples, $i = 1, \dots, m$, let $\mathcal{L}_t$ denote the loss function class in Eq. (18) that is associated with the class of PIRNN functions $\mathcal{H}$ trained following Eq. (13). Then, the following inequality holds, with probability at least $1 - \delta$ over S , for the PIRNN models satisfying **Assumptions 1–3**.

$$\mathbb{E}[l_t(z, \bar{y})] \leq \frac{1}{m} \sum_{i=1}^m l_t(z_i, \bar{y}_i) + \mathcal{O} \left(L_r d_y \frac{MB_X(1 + \sqrt{2 \log(2/t)})}{\sqrt{m}} \right) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \quad (21)$$

where $M = \frac{1 - (B_{U,F})^t}{1 - B_{U,F}} B_{W,F} B_{V,F}$ is the product of PIRNN weight matrices bounds, and B_X is the PIRNN input bound defined in **Assumption 1**. d_y is the PIRNN output dimension and L_r is the local Lipschitz constant defined in Eq. (16).

5. PIRNN-based predictive control

In this section, we incorporate the PIRNN model outlined in Section 3 into the design of model predictive controllers (MPC) to optimize process performance while maintaining closed-loop stability. Specifically, for any initial condition $x_0 \in \Omega_\rho$, a Lyapunov-based model predictive controller (LMPC) is developed to drive the closed-loop state of the nonlinear system of Eq. (1) to the steady-state setpoint while maintaining the state in the stability region Ω_ρ for all times.

The LMPC using the PIRNN model of Eq. (3) is utilized to stabilize the nonlinear system of Eq. (1) in the stability region. The formulation of PIRNN-based LMPC optimization problem is given as follows: [43]

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} (\tilde{x}^T Q \tilde{x} + u^T R u) dt \quad (22a)$$

$$\text{s.t. } \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \quad (22b)$$

$$u(t) \in U, \forall t \in [t_k, t_{k+N}) \quad (22c)$$

$$\tilde{x}(t_k) = x(t_k) \quad (22d)$$

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

$$\text{if } x(t_k) \in \Omega_\rho \setminus \Omega_{\rho_{nn}} \quad (22e)$$

$$V(\tilde{x}(t)) \leq \rho_{nn}, \quad \forall t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega_{\rho_{nn}} \quad (22f)$$

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 $\frac{\partial V(\tilde{x})}{\partial x}(F_{nn}(x, u))$. In the optimization problem of Eq. (22), the objective function of Eq. (22a) is the integral of the cost function $l(\tilde{x}, t) = (\tilde{x}^T Q \tilde{x} + u^T R u)$ 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. (22b) is the PIRNN model of Eq. (3) that is used to predict the states of the closed-loop system. Eq. (22c) defines the input constraints applied over the entire prediction horizon. Eq. (22d) defines the initial condition $\tilde{x}(t_k)$ of Eq. (22b), which is the state measurement at $t = t_k$. The constraint of Eq. (22e) forces the closed-loop state to move towards the origin if $x(t_k) \in \Omega_\rho \setminus \Omega_{\rho_{nn}}$, where $\Omega_{\rho_{nn}}$ is a small neighborhood around the origin. However, if $x(t_k)$ enters $\Omega_{\rho_{nn}}$, the states predicted by the PIRNN model of Eq. (22b) will be maintained in $\Omega_{\rho_{nn}}$ for the entire prediction horizon. The LMPC of Eq. (22) 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. (22) at each sampling time, from which only the control action for the first sampling period of the prediction horizon will be applied. The objective of the LMPC of Eq. (22) is to achieve closed-loop stability for the nonlinear system of Eq. (1) in the sense that the closed-loop state is bounded in the stability region Ω_ρ for all times, and can be ultimately bounded in a terminal set $\Omega_{\rho_{min}}$ that is a superset of $\Omega_{\rho_{nn}}$ designed accounting for the impact of modeling error and sample-and-hold implementation.

Recursive feasibility and closed-loop stability of the nonlinear system of Eq. (1) under the LMPC of Eq. (22) can be proven by showing that (1) the stabilizing controller $u = \Phi_{nn}(x)$ is a feasible solution to the MPC of Eq. (22) for all times, (2) the closed-loop state can be driven towards the origin, and ultimately bounded in the terminal set under $u = \Phi_{nn}(x)$, and (3) due to the two Lyapunov-based constraints of Eqs. (22e)–(22f), the optimal solution solved by LMPC will achieve an equally good closed-loop performance, if not better, than the stabilizing controller. Interested readers may refer to Ref. [38] for the detailed proof of closed-loop stability for the nominal system of Eq. (1), and Ref. [44] for the discussion of closed-loop stability for uncertain nonlinear systems with stochastic disturbances.

6. Application to a chemical process example

In this section, a chemical process example is employed to demonstrate the application of LMPC using the PIRNN model, whose prediction performance is augmented by leveraging a priori knowledge governed by the physical laws, to drive and maintain the closed-loop system state at its setpoint and within the stability region for all times. Specifically, a nonisothermal continuous stirred tank reactor (CSTR) under homogeneous mixing, and with an irreversible second-order exothermic reaction is considered. The reaction proceeds to convert a reacting species A to a product B (i.e., $A \rightarrow B$). A heating jacket is also equipped to either remove or supply heat energy at a rate Q , which either accelerates or suppresses the rate of reaction. The following material and energy balance equations collectively describe the dynamics of the CSTR, and these governing equations also form the cornerstone for physics-informed learning algorithms.

$$\frac{dC_A}{dt} = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT}} C_A^2 \quad (23a)$$

$$\frac{dT}{dt} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \quad (23b)$$

Table 1

Process parameter values adopted in the dynamic model of CSTR.

$F = 5 \text{ m}^3/\text{h}$	$V = 1 \text{ m}^3$
$k_0 = 8.46 \times 10^6 \text{ m}^3/\text{kmol h}$	$E = 5 \times 10^4 \text{ kJ/kmol}$
$R = 8.314 \text{ kJ/kmol K}$	$T_0 = 300 \text{ K}$
$\Delta H = -1.15 \times 10^4 \text{ kJ/kmol}$	$\rho_L = 1000 \text{ kg/m}^3$
$C_p = 0.231 \text{ kJ/kg K}$	$C_{A0s} = 4 \text{ kmol/m}^3$
$Q_s = 0.0 \text{ kJ/h}$	$C_{As} = 1.95 \text{ kmol/m}^3$
$T_s = 402 \text{ K}$	

where C_{A0} and F denote the inlet concentration of species A , and feed volumetric flow rate, respectively. C_A and V represent the concentration of reactant A , and the volume of reacting fluid in the CSTR, respectively. k_0 , E , and R are the frequency factor, activation energy, and the universal gas constant in the Arrhenius equation, respectively. T_0 and T denote the temperatures of feed, and reacting liquid in the CSTR, respectively. ΔH represents the change in enthalpy of the reaction system. In addition, the reacting liquid is assumed to have a constant density of ρ_L and a constant heat capacity of C_p . The values adopted for the various parameters in Eq. (23), together with the steady-state values for controlled (i.e., C_A and T) and manipulated (i.e., Q and C_{A0}) variables, are listed in Table 1.

It is intended to operate the CSTR at an unstable steady-state $(C_{As}, T_s) = (1.95 \text{ kmol/m}^3, 402 \text{ K})$ via the manipulated inputs C_{A0} and Q , where $(C_{A0s}, Q_s) = (4 \text{ kmol/m}^3, 0 \text{ kJ/h})$. The system states and the manipulated inputs are expressed in their respective deviation forms: $\Delta C_A = C_A - C_{As}$, $\Delta T = T - T_s$, $\Delta C_{A0} = C_{A0} - C_{A0s}$, and $\Delta Q = Q - Q_s$. In addition, physical constraints are also imposed on the manipulated inputs, where $|\Delta C_{A0}| \leq 3.5 \text{ kmol/m}^3$ and $|\Delta Q| \leq 5 \times 10^5 \text{ kJ/h}$. More succinctly, the states and the manipulated inputs of the closed-loop system are represented as $x^T = [\Delta C_A, \Delta T]$, and $u^T = [\Delta C_{A0}, \Delta Q]$, respectively, and the origin of the state-space therefore denotes the equilibrium point of the system (i.e., $(x_s^*, u_s^*) = (0, 0)$).

Operationally, the control objective stipulates that the system states of the CSTR are to be maintained at the unstable steady-state (C_{As}, T_s) by adjusting the manipulated inputs ΔC_{A0} and ΔQ under the LMPC using PIRNN as the predictive model. The sampling period Δ is specified to be 10^{-2} h , and the nonlinear optimization problem of the LMPC of Eq. (22) is solved using Pylpopt, a Python module of the state-of-the-art IPOPT solver package for solving nonlinear optimization problems involving large-scale continuous systems. IPOPT stands for interior-point optimizer, which implements the interior-point methods for solving linear and nonlinear convex optimization problems. The first-order derivatives, which are required by the IPOPT solver in determining the optimal manipulated input u at every sampling period Δ , are estimated using the finite difference method (e.g., central difference) by adding and subtracting a small perturbation Δu on the control actions u , respectively.

6.1. Process data generation

In continuous flow processes, operational data collected are generally clustered within a small region around the steady-state, and this renders purely data-driven modeling approaches implausible and unattractive since extrapolation beyond the operating range captured by the process data is immensely hampered by its limited and ill-sampled nature. Furthermore, real-world process data garnered are often corrupted by random measurement noise due to instrumentation uncertainties and hardware limitations, and this poses an additional practical barrier for effective learning that is ascribed to the high capacity of neural networks in overfitting the noisy data. Nevertheless, the latter challenge can be adequately addressed by using the dropout technique, which

is a powerful and yet facile regularization method that reduces overfitting in neural networks. Functionally, the dropout method randomly masks selected neural network weights during training, thereby deliberately reducing dependencies in deep neural networks. This haphazardly dropping of connections between neurons in a network facilitates the intrinsic reduction of model complexity and the realization of several independent internal representations of the training data which are instrumental in promoting the learning of the underlying true system dynamics. Specifically, standard dropout technique has been successfully applied to various types of neural networks to reduce overfitting, and Monte Carlo (MC) dropout that was proposed by Ref. [45] has been demonstrated to achieve better performance for RNNs with some types of data noise. For example, in Ref. [46], we have integrated MC dropout with purely data-driven RNNs for modeling nonlinear systems with noisy data. It is demonstrated in Ref. [46] that an RNN model using MC dropout method is able to reduce the overfitting to non-Gaussian noisy data. The main difference between MC dropout and standard dropout techniques is that the random masking behavior is preserved after training for MC dropout, but not for standard dropout. Therefore, the predictions made by RNNs with MC dropout are seen as random samples drawn from a probabilistic distribution, which will be non-deterministic. While MC dropout may be superior to standard dropout for some types of data noise (e.g., non-Gaussian noise), we have shown in Ref. [46] that standard NN modeling without dropout, or with standard dropout may achieve a comparable modeling accuracy to that using MC dropout for noisy data of Gaussian distribution. Therefore, in this work, the standard dropout method is used in PIRNN to reduce overfitting to data with Gaussian noise, and is demonstrated to achieve the desired modeling performance in this section.

Additionally, process data are assumed to be limited and are available only in a small region around the unstable steady-state of $(C_A, T_s) = (1.95 \text{ kmol/m}^3, 402 \text{ K})$. The process data are generated via open-loop simulations by numerically solving the dynamic process model of Eq. (23) based on a moderate number of initial conditions and control actions using the SciPy implementation of the standard explicit Runge-Kutta fourth-order numerical algorithm with an integration time step of $h_c = 1 \times 10^{-3} \text{ h}$ in the Python programming language. The manipulated inputs are fed to the dynamic model of Eq. (23) as piecewise constants (i.e., $\Delta C_{A0}(t) = \Delta C_{A0}(t_k)$, and $\Delta Q(t) = \Delta Q(t_k) \forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$ with $\Delta = 1 \times 10^{-2} \text{ h}$ indicating one sampling period). Furthermore, to simulate the effect of measurement noise on process data garnered in reality, Gaussian noises $\omega_{C_A} \sim \mathcal{N}(0 \text{ kmol/m}^3, 1.0 \times 10^{-3} (\text{kmol/m}^3)^2)$, and $\omega_T \sim \mathcal{N}(0 \text{ K}, 25 (\text{K})^2)$ are added to the concentration C_A , and temperature T measurements retrieved from open-loop simulations, respectively, to yield the noisy dataset.

Specifically, the initial states $(\Delta C_A, \Delta T)$ are uniformly and randomly sampled from a small region around the unstable steady state C_{A_s} and T_s (i.e., $\Delta C_A (\text{kmol/m}^3) \sim U(-0.2, 0.2)$ and $\Delta T (\text{K}) \sim U(-20, 20)$), as shown in Table 2. Similarly, the manipulated inputs $(\Delta C_{A0}, \Delta Q)$ are randomly and uniformly selected from a small neighborhood around their respective steady states C_{A0s} and Q_s (i.e., $\Delta C_{A0} (\text{kmol/m}^3) \sim U(-1.5, 1.5)$, and $\Delta Q (\text{kJ/h}) \sim U(-5 \times 10^3, 5 \times 10^3)$, as shown in Table 2.

With the aforementioned configurations, dynamic state trajectories $(\Delta C_A$ and $\Delta T)$, subject to their respective Gaussian measurement noises ω_{C_A} and ω_T , are sampled at every integration time step $h_c = 1 \times 10^{-3} \text{ h}$ within each sampling period Δ . Specifically, as shown in Table 2, state trajectories are simulated for each of the 64 pairs of initial system states (e.g., 8 from each state) that are sampled from their respective uniform distributions, and are

Table 2

Sampling distributions of initial states and manipulated inputs for generating process data.

Variable	Distribution	No. of points
Initial ΔC_A (kmol/m ³)	$U(-0.2, 0.2)$	8
Initial ΔT (K)	$U(-20, 20)$	8
ΔC_{A0} (kmol/m ³)	$U(-1.5, 1.5)$	10
ΔQ (kJ/h)	$U(-5 \times 10^3, 5 \times 10^3)$	15
Total number of state trajectories generated		9600

subject to each of the 150 pairs of manipulated inputs (e.g., 10 from ΔC_{A0} and 15 from ΔQ) in a sample-and-hold fashion. As a result, 9600 state trajectories (comprising noisy measurements) are obtained, which are then partitioned into training (80%), and validation (20%) datasets. Fig. 2 shows a few samples of the state trajectories obtained within a small neighborhood ($\Omega_{\rho_{min}}$) around the steady-state for one sampling period Δ .

Subsequently, a PIRNN model is developed using the noisy dataset and the governing physical laws expressed in the form of the dynamic system model of Eq. (23). A conventional RNN model (i.e., without incorporating any physics knowledge into the model) is constructed solely based on the noisy process data, which serves as a baseline for comparison with the PIRNN model to demonstrate its predictive capability beyond the range of training data. Additionally, a purely physics-based RNN model (i.e., without utilizing the observed process data) is also developed, and serves as another baseline for comparison with the PIRNN to showcase its robustness against measurement noise. The dropout technique is implemented in both the PIRNN and the purely data-driven RNN models to reduce overfitting and to support the learning of the underlying noise-free process dynamics. Similarly, dropout is added to the purely physics-based RNN model for consistency.

The collocation points (i.e., consisting of the initial system states ΔC_A and ΔT , and the manipulated inputs ΔC_{A0} and ΔQ including and beyond the range captured in the process data) that are used to embed the system dynamics knowledge and to enforce the structure imposed by Eq. (23) to the PIRNN are uniformly sampled across the closed-loop stability region Ω_ρ defined for the CSTR, as shown in Fig. 2. Specifically, the control Lyapunov function $V(x) = x^T P x$ is designed with a positive definite matrix $P = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix}$. The closed-loop stability region Ω_ρ for the CSTR is then characterized as a level set of the Lyapunov function with $\rho = 368$, where the origin can be rendered exponentially stable under the controller $u = \Phi(x) \in U$. Similarly, the manipulated inputs ΔC_{A0} and ΔQ are uniformly sampled across their physical limits (i.e., $-3.5 \text{ kmol/m}^3 \leq \Delta C_{A0} \leq 3.5 \text{ kmol/m}^3$ and $-5 \times 10^5 \text{ kJ/h} \leq \Delta Q \leq 5 \times 10^5 \text{ kJ/h}$). Altogether, 110 pairs of initial states are sampled across the closed-loop stability region Ω_ρ , as shown in Fig. 2, with each subject to 500 pairs of uniformly sampled manipulated inputs in a sample-and-hold fashion, thereby yielding 55,000 collocation points (i.e., 55,000 different combinations of 110 pairs of ΔC_A and ΔT with 500 pairs of ΔC_{A0} and ΔQ) for physics-informed learning. Similar to process data, the collocation points are also split into training (80%), and validation (20%) datasets.

The noise-free process data, which correspond to the noisy dataset with the absence of measurement noise ω_{C_A} and ω_T , are also collected for training the three aforementioned RNN models and their prediction performance are compared with those trained using the noisy dataset.

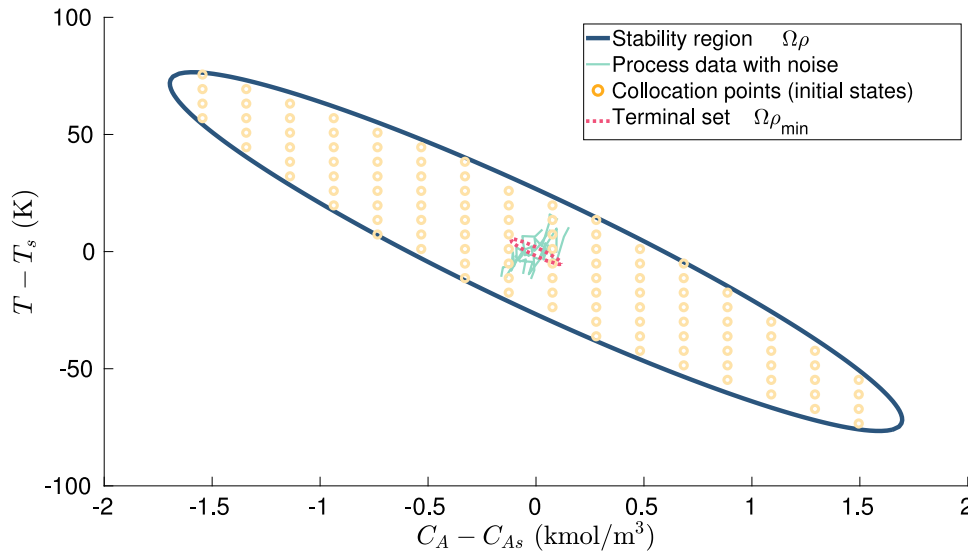


Fig. 2. Collocation points sampled uniformly across the stability region Ω_ρ , together with a few examples of the noisy process state trajectories captured within a small neighborhood around the origin (i.e., steady-state C_{A_s} and T_s).

6.2. Open-loop simulation results of RNN and PIRNN

In this section, the three RNN models described in Section 6.1 are developed and their generalization performance are compared. Specifically, the first RNN model is a standard RNN trained solely using the process data captured; the second model is the PIRNN model, for which the governing equation (i.e., Eq. (23)) of the nonlinear dynamic process is incorporated as regularization terms in addition to the loss term based on the process data in its loss function; the third model is a purely physics-driven RNN model that is trained entirely based on the physics-informed loss function using uniformly sampled collocation points across the operating regions, and without the involvement of any process data. Furthermore, the three RNN models are constructed with, and without considering the effect of measurement noise, respectively.

6.2.1. RNN prediction performance without Gaussian noise

With the noise-free process data generated using the procedures outlined in Section 6.1, the three RNN models are developed with the identical network architecture consisting of three hidden recurrent layers with 128, 256, and 64 recurrent units, respectively. It should be noted that no dropout is incorporated into these RNN models as the process data utilized for training is not subject to measurement noise. Moreover, the three RNN models are trained using the same parameter settings in terms of the number of epochs = 300, optimizer = *Adam*, learning rate = 0.001, weight initialization $\sim U(-\sqrt{k}, \sqrt{k})$ where $k = \frac{1}{\text{hidden layer size}}$, and activation function = *Tanh* in order to compare their prediction performances on the same ground. With these configurations, the three RNN models are developed with 4 input features and 2 outputs (i.e., $F_{nn}(\tilde{x}, u)$, with $\tilde{x} \in \mathbf{R}^2$ and $u \in \mathbf{R}^2$), using the state-of-the-art deep learning framework PyTorch in the Python programming language, to predict the future system states (i.e., ΔC_A and ΔT) for one sampling period $\Delta = 1 \times 10^{-2}$ h given the initial state measurements (ΔC_A and ΔT at $t = 0.0$ h), and the manipulated inputs (ΔC_{A0} and ΔQ). In addition, to facilitate a more efficient learning process, all inputs and outputs of the RNN models are standardized by subtracting the mean of noise-free process data and dividing by their respective standard deviations.

As shown in Fig. 1, since the process data are collected at an integration time step h_c during the open-loop simulations, the

time interval between two consecutive internal states h_{t-1} and h_t of the unfolded RNN is thus chosen to directly correspond to h_c , where h_t represents the RNN hidden state at the t_{th} time step, $t = 1, \dots, T$. Furthermore, all the states between $t = 0$ and $t = \Delta$ (i.e., the RNN states in one prediction horizon) are treated as the internal states of the RNN and can be used as the predicted system states by the RNN models for one sampling period forward, where the total number of the RNN outputs over one sampling period is $T = \Delta/h_c$. The resulting purely data-driven RNN, PIRNN, and purely physics-driven RNN attained MSEs of approximately 1.1×10^{-3} , 1.6×10^{-3} , 1.3×10^{-3} on their training datasets, respectively, and of approximately 2.3×10^{-3} , 2.1×10^{-3} , 1.9×10^{-3} on their validation datasets, respectively. The MSEs on the training and validation datasets are of the same magnitudes, indicating that the RNN models are not overfitting and can generalize well to the process dynamics within the operating range dictated by their respective datasets.

Fig. 3 shows the comparison of open-loop state profiles predicted by the three RNN models, with RNN, PIRNN, and PIRNN_{without_MSE_χ} denoting the purely data-driven RNN, PIRNN trained with both process data and collocation points, and purely physics-driven RNN, respectively. Specifically, the system states are initially set to be at their respective steady-states of $(\Delta C_A, \Delta T) = (0, 0)$, which are then subject to small perturbations triggered by the pseudo-randomly generated manipulated input profiles (as shown in Fig. 4) in a sample and hold fashion where $\Delta C_{A0} \sim U(-1.5, 1.5)$ and $\Delta Q \sim U(-1 \times 10^{-3}, 1 \times 10^{-3})$ are fed to the RNN models as piecewise constants (i.e., $\Delta C_{A0}(t) = \Delta C_{A0}(t_k)$, and $\Delta Q(t) = \Delta Q(t_k) \forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$) for 15 sampling periods from $t = 0$ h to $t = 0.15$ h. Beyond $t = 0.15$ h, the dynamic system is subject to more vigorous control actions with $\Delta C_{A0} \sim U(-3.5, 3.5)$ and $\Delta Q \sim U(-1 \times 10^{-5}, 1 \times 10^{-5})$ for another 15 sampling periods from $t = 0.15$ h to $t = 0.3$ h. The true process dynamics in Fig. 3, subject to the same manipulated input profiles shown in Fig. 4, is used as the benchmark to evaluate the prediction performance of the three RNN models.

Since all of the three RNN models are developed to predict only one sampling period $\Delta = 1 \times 10^{-2}$ h forward given the current state measurements and the manipulated inputs, they are recursively invoked to predict the entire trajectory from $t = 0$ h to $t = 30 \times \Delta = 0.3$ h. As shown in Fig. 3, it is apparent that the purely data-driven RNN (trained using solely noise-free process

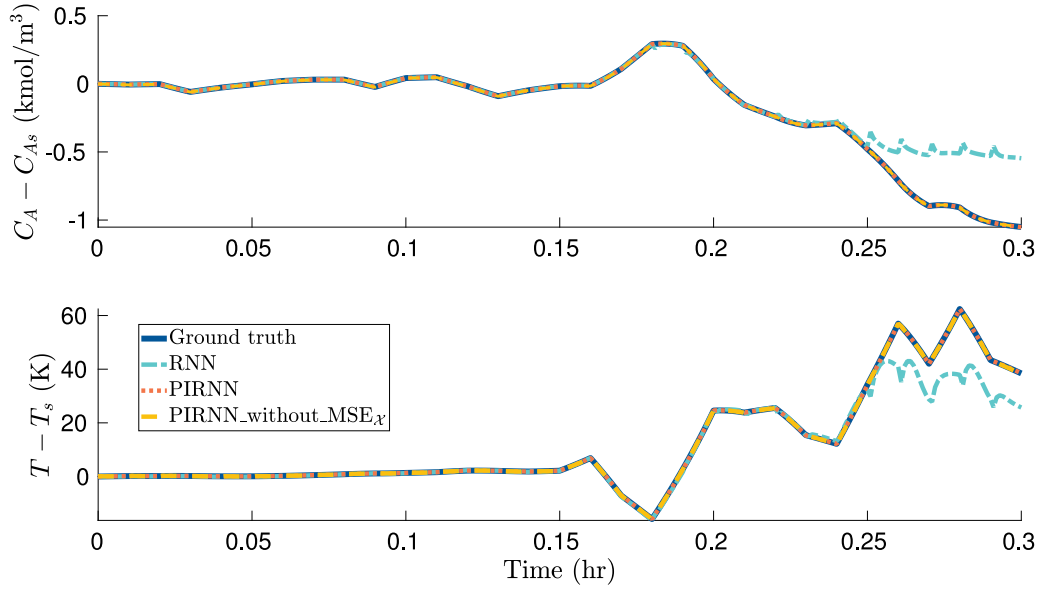


Fig. 3. Comparison of open-loop state profiles (i.e., ΔC_A (top figure), and ΔT (bottom figure)) predicted by RNN (green dotted-dashed line), PIRNN (red dotted line), and purely physics-driven RNN (orange dashed line) with the ground truth (blue solid line). Noise-free state measurements were used to train the three RNN models.

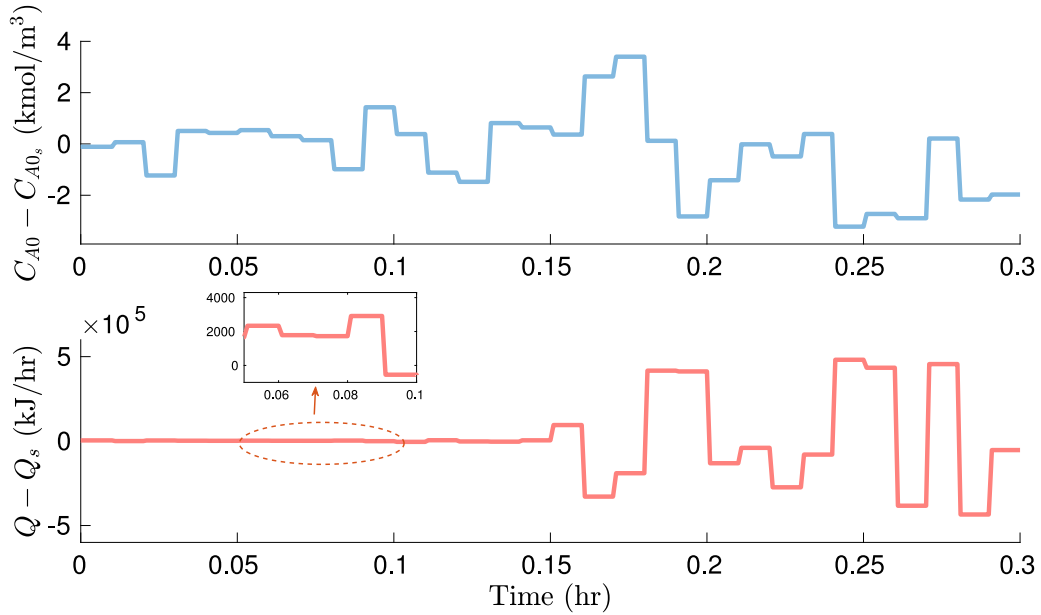


Fig. 4. Open-loop manipulated input profiles: ΔC_{A0} (top figure), and ΔQ (bottom figure) for the three RNN models.

data), PIRNN (trained using noise-free process data and collocation points), and purely physics-based RNN (trained using only collocation points) achieve excellent predictions that conform to the true process dynamics (i.e., data generated from the first-principles model of Eq. (23)) between the time interval $t = 0$ h to approximately $t = 0.25$ h. This can be ascribed to the observed dynamics of the system states (i.e., ΔC_A , ΔT profiles shown in Fig. 3) and the manipulated inputs (i.e., ΔC_{A0} , ΔQ profiles shown in Fig. 4) remaining within a small neighborhood around their respective steady-state values for t from $t = 0$ h to $t = 0.25$ h, which have been well captured by all of the RNN models during their training.

However, as the dynamic system experiences more rigorous manipulated inputs of larger magnitudes beyond $t = 0.15$ h and with system states beginning to drift considerably from their respective steady-state values, the prediction performance of the

purely data-driven RNN becomes unsatisfactory, as shown by the appreciable deviations of its predicted state profiles from the ground truth in Fig. 3 from $t = 0.25$ h onward. This highlights the limitation of the purely data-driven RNN, which suffers from poor generalizability beyond the operating range captured in the training data. In contrast, the PIRNN model, which learns the underlying process dynamics from both the noise-free process data and through the governing physical laws of Eq. (23), demonstrates remarkable extrapolation performance where its predicted state profiles are well matched with the true profiles, as shown in Fig. 3 from $t = 0.25$ h to $t = 0.3$ h. Moreover, the purely physics-driven RNN model also demonstrates indistinguishable prediction performance to the PIRNN model as it is trained using collocation points that span the entire operating regions of the system states and manipulated inputs, respectively. Therefore, the advantage of physics-informed learning approach over the

standard RNN training approach manifests in its generalization capability, and this is especially instrumental for controlling dynamic systems in the chemical industry, where the process data collected are usually ill-sampled (e.g., predominantly within a small neighborhood around the steady-state setpoints). The embedding of mechanistic knowledge (e.g., the governing ODEs of the system) into the RNN models thus tremendously enhances their predictive performance, thereby fostering a more effective control with ML-MPC.

6.2.2. RNN prediction performance with Gaussian noise

Similarly, with the noisy dataset generated using the procedures outlined in Section 6.1, three RNN models are developed: (1) a standard RNN is designed with three hidden recurrent layers comprising 128, 256, and 64 recurrent units, respectively, and with each layer regularized by a dropout probability of 0.25; (2) a PIRNN is constructed with the identical network architecture and dropout rates as the standard RNN described in (1); (3) a purely physics-driven RNN model is built with the same configurations as (1) and (2). It should be noted that, contrary to the RNN models in Section 6.2.1, dropouts are incorporated into the RNN models in this section to prevent overfitting the noisy process data which are subject to the measurement noises ω_{CA} and ω_T . The three RNN models in this section are trained using the same parameter settings with number of epochs = 150, optimizer = Adam, learning rate = 0.001, weight initialization $\sim U(-\sqrt{k}, \sqrt{k})$ where $k = \frac{1}{\text{hidden layer size}}$, and activation function = Tanh for the purpose of comparing their prediction performances on the same ground.

The resulting purely data-driven RNN, PIRNN, purely physics-driven RNN models attained MSEs of approximately 0.339, 0.409, and 3.0×10^{-2} on their respective training datasets, and of approximately 0.379, 0.510, and 2.6×10^{-2} on their respective validation datasets. For purely data-driven RNN and PIRNN models, their relatively high MSEs (as compared to the MSE values yielded using noise-free process data in Section 6.2.1) on the training and validation datasets indicate that the RNN models are not capturing the noisy process data in its entirety, but are rather attempting to learn the underlying true dynamics. The incorporation of dropout thus indeed seeks to alleviate susceptibility to overfitting of the noisy process data. Furthermore, while the identical set of collocation points to those utilized in Section 6.2.1 has been used to train the purely physics-driven RNN model in this section, the resulting MSEs generated are larger than those reported in Section 6.2.1. This can be attributed to the inclusion of dropouts in the purely physics-driven RNN model, which intrinsically reduces the model complexity and thereby diminishing its ability to learn the complex dynamic behavior entailed in the governing equation of Eq. (23).

Fig. 5 shows the comparison of open-loop state profiles predicted by the standard RNN (denoted by RNN), PIRNN (denoted by PIRNN), and the purely physics-driven RNN models (denoted by PIRNN_{without_MSE_X}), respectively, together with the noisy process data and the true process dynamics, under the same pseudo-randomly generated manipulated input profiles as shown in Fig. 6. Specifically, similar to Section 6.2.1, the system states are initially set at their respective steady-state values of $(\Delta C_A, \Delta T) = (0, 0)$, which are then subject to small perturbations triggered by the pseudo-randomly generated manipulated input profiles (as shown in Fig. 6) in a sample and hold fashion where $\Delta C_{A0} \sim U(-1.5, 1.5)$ and $\Delta Q \sim U(-1 \times 10^{-3}, 1 \times 10^{-3})$ are fed to the RNN models as piecewise constant functions for 15 sampling periods from $t = 0$ h to $t = 0.15$ h. Subsequently, the dynamic system is subject to more vigorous control actions with $\Delta C_{A0} \sim U(-3.5, 3.5)$ and $\Delta Q \sim U(-1 \times 10^{-5}, 1 \times 10^{-5})$ for another 15 sampling periods from $t = 0.15$ h to $t = 0.3$ h. Furthermore, while the true process dynamics without measurement

noise, subject to the same manipulated input profiles shown in Fig. 6, is used as the benchmark to evaluate the prediction and noise rejection performance of the three RNN models, the system states that are supplied to the RNN models are corrupted by the measurement noise ω_{CA} and ω_T , respectively.

As shown in Fig. 5, it is evident that both the purely data-driven RNN (trained using solely noisy process data) and PIRNN (trained using noisy process data and collocation points) are robust against measurement noise where their predictions conform more closely to the true underlying process dynamics between the time interval $t = 0$ h to $t = 0.15$ h. This can be ascribed to the inclusion of dropout layers that aid in noise rejection by preventing overfitting via model complexity reduction. Moreover, as the observed dynamics of the system states (i.e., ΔC_A and ΔT profiles shown in Fig. 5) and the manipulated inputs (i.e., ΔC_{A0} and ΔQ profiles shown in Fig. 6) are found to remain relatively close to their respective steady-state values from $t = 0$ h to $t = 0.15$ h, their dynamic behaviors have already been well captured by the purely data-driven RNN and PIRNN models in their training phases. It is also noticed that, in Fig. 5, the noise rejection performance of the purely physics-driven RNN model is inferior to that of the RNN and PIRNN models, and this is expected since the noisy process data has been excluded from training the purely physics-driven RNN model, and its adaptability to measurement noise is therefore unsatisfactory.

However, as the dynamic system experiences more rigorous manipulated inputs of larger magnitudes beyond $t = 0.15$ h, the system states begin to drift considerably from their respective steady-state values, and the prediction performance of the purely data-driven RNN model becomes unsatisfactory, as shown by the appreciable deviations of its predicted state profiles from the ground truth in Fig. 5 from $t = 0.2$ h onward. This highlights the limitation of the purely data-driven RNN model, which suffers from poor generalizability beyond the range of training data. In contrast, the PIRNN and the purely physics-driven RNN models, which have been incorporated with the knowledge of the underlying process dynamics through the governing physical laws of Eq. (23), demonstrate excellent extrapolation performance where their predicted state profiles are well matched with the true profiles, as shown in Fig. 5 from $t = 0.2$ h to $t = 0.3$ h. The advantage of physics-informed learning over conventional algorithms manifests in its integration of a priori knowledge with existing data in augmenting the generalization capability of the resulting RNN models, and this is especially instrumental for controlling dynamic systems in the chemical industry, where the mechanistic knowledge (e.g., the governing ODEs of the system) can be directly embedded into the RNN models to enhance their predictive performance in MPC. Note that beyond $t = 0.15$ h, inputs to all the three RNN models switched from noisy states to true system states as both the PIRNN and the purely physics-driven RNN are trained to approximate the noise-free dynamic model of Eq. (23) based on the collocation points. In addition, it is noticed that the overall prediction performance by the PIRNN and purely physics-driven RNN are inferior to those seen in Fig. 3 for $t \geq 0.15$ h. This can be attributed to the inclusion of dropout layers, which reduces the model complexity and hence their ability to model the complex nonlinear dynamics.

6.3. Closed-loop simulation results of RNN- and PIRNN-based LMPC

The CSTR dynamic system, which is described by Eq. (23) and is subject to measurement noise ω_{CA} and ω_T , is used to illustrate the application of PIRNN-based LMPC of Eq. (22) with the objective of operating the CSTR at an unstable equilibrium point $(C_A^s, T_s) = (1.95 \text{ kmol/m}^3, 402 \text{ K})$ by adjusting the manipulated inputs ΔC_{A0} and ΔQ . Three additional LMPCs using the standard

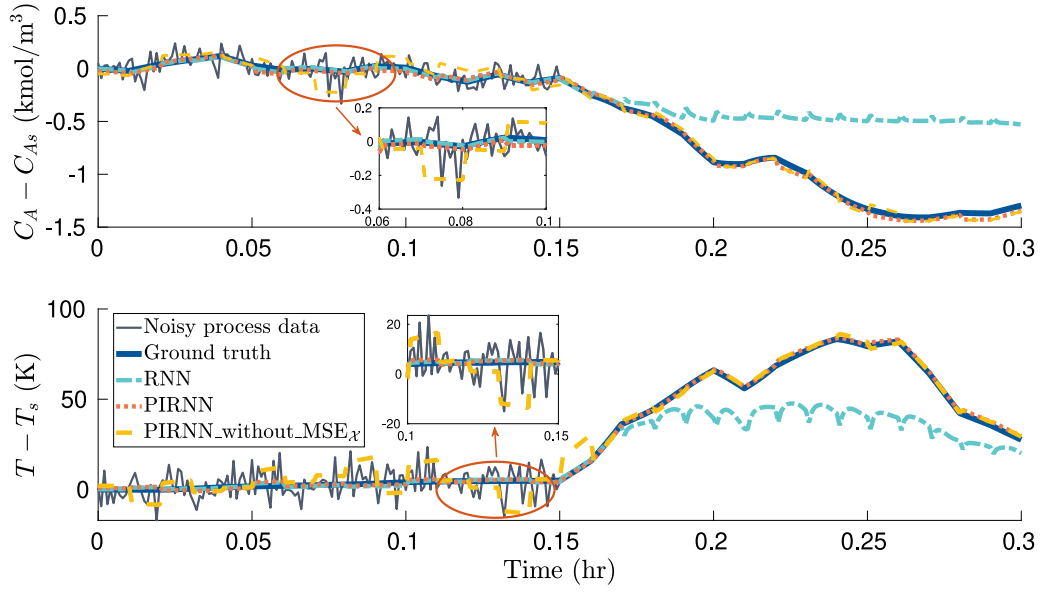


Fig. 5. Comparison of open-loop state profiles (i.e., ΔC_A (top figure), and ΔT (bottom figure)) predicted by RNN (green dotted-dashed line), PIRNN (red dotted line), and purely physics-driven RNN (orange dashed line) with the ground truth (blue solid line). The noisy state measurements are represented by the gray solid lines. Noisy state measurements were input to the RNN models from $t = 0$ h to $t = 0.15$ h, and noise-free states were used post $t = 0.15$ h.

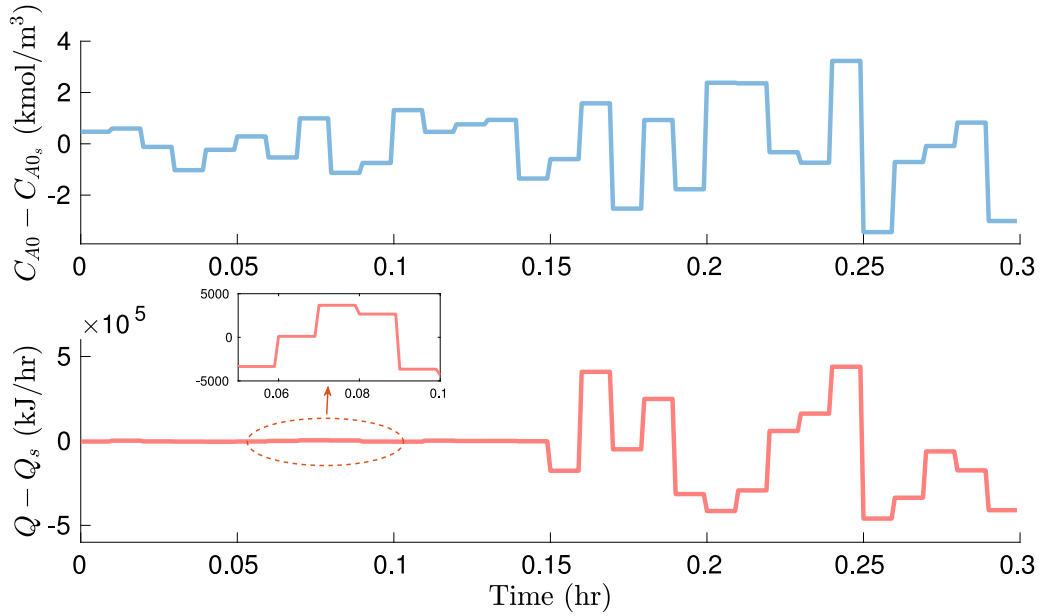


Fig. 6. Open-loop manipulated input profiles: ΔC_{A0} (top figure), and ΔQ (bottom figure) for the three RNN models.

RNN model, the purely physics-driven RNN model, and the true process dynamic model of Eq. (23), respectively, will serve as benchmark cases for evaluating the performance of the PIRNN-based LMPC. All RNN models employed in LMPCs in this section are taken from those illustrated in Section 6.2.2 (i.e., with the inclusion of dropout). Specifically, the LMPC objective function of Eq. (22a) has the following form which yields a minimum value at the steady-state values C_{As} , T_s , C_{A0s} , and Q_s :

$$L(x, u) = |x|_{Q_1}^2 + |u|_{Q_2}^2 \quad (24)$$

where Q_1 and Q_2 denote the coefficient matrices that balance the scale of each contributing term (i.e., the system states x and the manipulated inputs u in their respective deviation forms) and to indicate the relative importance of one process specification (e.g., energy and reactant expenditures) to another. In this work,

$Q_1 = (1 \times 10^3, 1)$ and $Q_2 = (1 \times 10^{-1}, 1 \times 10^{-10})$ are selected to reflect the priority of maintaining system states at their desired setpoints over resource consumption, with the scale of $|x|_{Q_1}^2$ chosen to be of approximately an order of magnitude larger than that of $|u|_{Q_2}^2$. The nonlinear optimization problem of the LMPC (i.e., Eq. (22)) is solved using the PyIpopt optimizer with a sampling period $\Delta = 1 \times 10^{-2}$ h and a total operation time of 0.2 h (i.e., 20 MPC iterations).

The state-space trajectories (ground truth) and the state profiles (corrupted by noise) for the dynamic system with an initial state $x_0 = (-1.5, 70)$ are shown in Figs. 7 and 8, respectively. While the ground truth process dynamics under the various LMPCs are presented in Fig. 7 to facilitate a more convenient assessment of their relative control performance, the states input

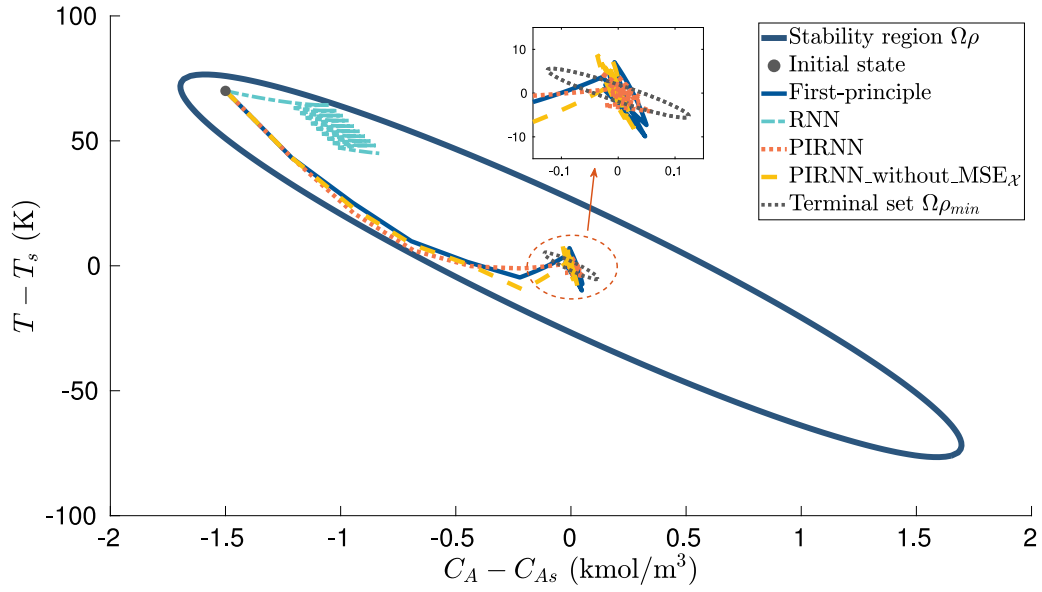


Fig. 7. Closed-loop system state trajectories under the LMPC using first-principles model (blue solid line), purely data-driven RNN (green dotted-dashed line), PIRNN (red dotted line), purely physics-driven (orange dashed line), respectively, for the initial condition of $(-1.5 \text{ kmol/m}^3, 70 \text{ K})$. The state trajectories shown are the true values without noise.

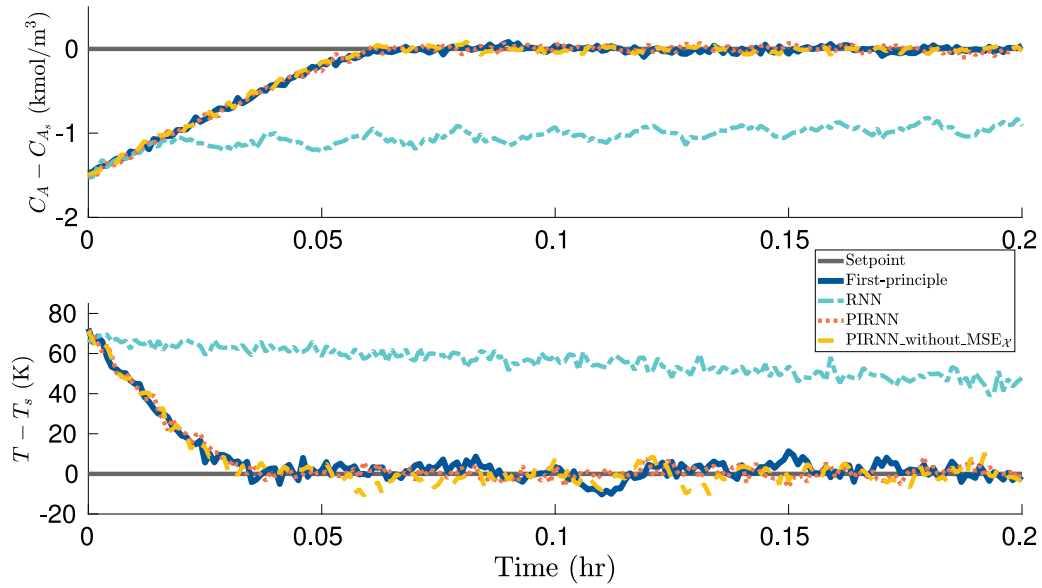


Fig. 8. Closed-loop system state profiles under the LMPC using first-principles model (blue solid line), purely data-driven RNN (green dotted-dashed line), PIRNN (red dotted line), purely physics-driven (orange dashed line), respectively, for the initial condition of $(-1.5 \text{ kmol/m}^3, 70 \text{ K})$. The state profiles shown are the system states with noise.

to the various LMPCs are corrupted by noise during the closed-loop simulations, as shown in Fig. 8. In Fig. 7 (ground truth dynamics), it is observed that the dynamic trajectories produced by the true process dynamic model of Eq. (23) (denoted by *First-principle*), the purely data-driven RNN (denoted by *RNN*), the PIRNN (denoted by *PIRNN*), and the purely physics-driven RNN (denoted by *PIRNN_without_MSE χ*) based LMPCs remain inside the stability region Ω_ρ for the entire duration of the CSTR operation, and with the dynamic trajectories of the true process model-, the PIRNN-, and the purely physics-driven RNN-based LMPCs ultimately oscillate in a small neighborhood around the state-steady that is beyond the small ball $\Omega_{\rho_{\min}}$ in the presence of Gaussian measurement noise. However, it is apparent that the state trajectory yielded by PIRNN-based LMPC results in a smaller oscillation (i.e., nearly converges to $\Omega_{\rho_{\min}}$) around the

state-steady as compared to that of true process model- and purely physics-driven RNN-based LMPCs. This can be attributed to its adaptability to measurement noise, thereby delivering a superior noise rejection performance. In this example, it is assumed that the first-principles model of Eq. (23) captures the evolution of process states in the absence of measurement noise. The LMPC using purely physics-informed RNN or the first-principles model will thus engender deteriorating control performance in the presence of measurement noise, as evident in their larger oscillations around the steady-state (Fig. 7). On the contrary, standard RNN-based LMPC provides inadequate control and completely fails to drive the CSTR to its desired setpoints. This is ascribed to the employment of ill-sampled process data, which greatly hampers the prediction performance of RNN beyond the range of training data. Fig. 8 shows the individual state profiles (corrupted

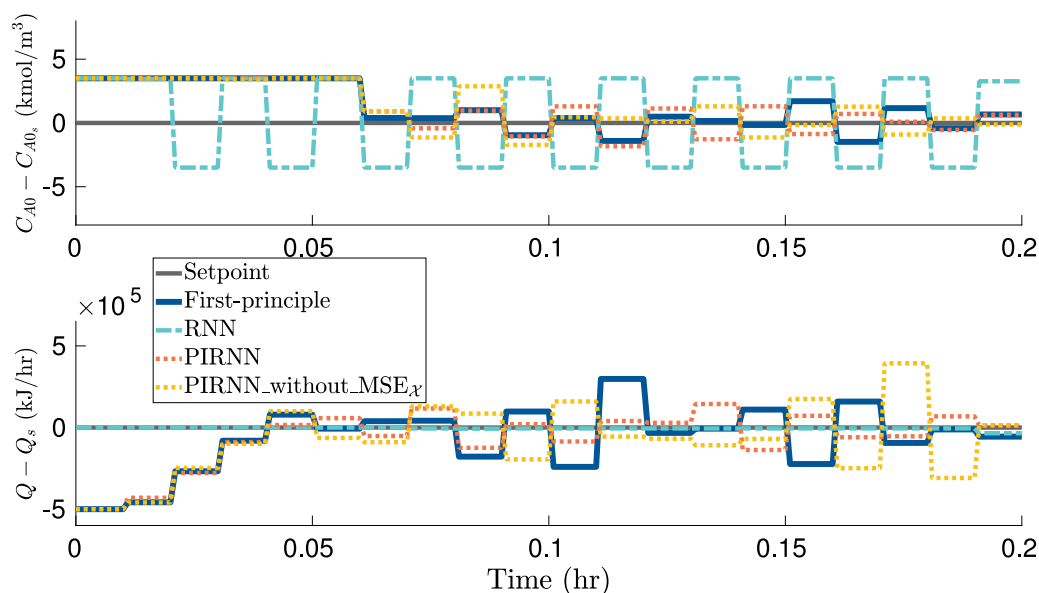


Fig. 9. Closed-loop manipulated inputs under the LMPC using first-principles model (blue solid line), purely data-driven RNN (green dotted-dashed line), PIRNN (red dotted line), purely physics-driven (orange dashed line), respectively, for the initial system state of $(-1.5 \text{ kmol/m}^3, 70 \text{ K})$.

by measurement noise) produced by the various LMPCs, with purely data-driven RNN- and PIRNN-based LMPCs yielding the most undesired and desired closed-loop control performances, respectively.

Fig. 9 shows the respective control profiles of the four LMPCs studied. Specifically, the true process model-, the PIRNN-, and the purely physics-driven RNN-based LMPCs implement aggressive control actions (i.e., utilizing ΔC_{A0} and ΔQ values that are close to their imposed physical limits) in the initial CSTR operation to drive the system states towards the setpoints rapidly from $t = 0.0 \text{ h}$ to $t = 0.06 \text{ h}$. Subsequently, the control actions of the three LMPCs oscillate around their setpoints, and their magnitudes are dependent on the values of state measurements, which are corrupted by Gaussian noise, received. Since measurement noise has the least effect on PIRNN-based LMPC, its control action outputs exhibit oscillations of the smallest magnitude from $t = 0.06 \text{ h}$ onward.

Remark 6. As demonstrated in Fig. 7, the true process dynamic model- and the purely physics-driven RNN-based LMPCs are capable of providing adequate control to successfully drive the CSTR from x_0 to a small neighborhood around its steady-state setpoints, and the effect of measurement noise on the two LMPCs is minor in the initial operation of the CSTR as the absolute difference between x_0 and the setpoint is much larger in magnitude than that of the Gaussian noise. Therefore, the prediction performances of the true process dynamic model and the purely physics-driven RNN will only become inferior to PIRNN when the predicted states required are close to the input state x supplied, and the adverse impact of measurement noise thus manifests only when the LMPCs are required to stabilize the system in a small neighborhood (e.g., $\Omega_{\rho_{\min}}$) around its steady-state setpoints.

Remark 7. While more sophisticated RNN variants (such as long short-term memory (LSTM) and gated recurrent unit (GRU) networks) exist, they tend to have minimal impacts on the model generalizability since all of them are data-driven models in nature. Additionally, the dominant benefits of LSTM and GRU over traditional RNN, which include a better capacity at handling long sequential data as they are less susceptible to the vanishing gradient problem [47], are not explicitly favored in this case as the

number of time steps required in the model prediction is only 10. In fact, the exploitation of more involved RNN architectures such as LSTM and GRU might even be detrimental to LMPC implementations as these complex networks generally require more plant data in order to learn effectively. Moreover, the training of LSTM and GRU can be computationally intensive given their inherent more complicated architecture, and this may engender a potential hindrance to their applications in real-time online learning MPC, where the predictive performance of RNN models are improved in real-time by updating them using the latest plant data retrieved once a significant discrepancy between model predictions and sensor measurements are discovered. Refs. [2,48] demonstrate the outstanding modeling capacity of RNN in describing a highly complex and nonlinear batch crystallization process, and its superior computational efficiency in an online learning-based RNN-MPC scheme. In principle, although the physics-informed modeling approach can be applied to LSTM and GRU, the number of prediction time steps required should be sufficiently large that renders the simple RNN ineffective, and a switch to the more complex variants is thus necessary to warrant adequate modeling performance for satisfactory control. In other cases, RNN should be the preferred choice of surrogate model in LMPC applications.

7. Conclusion

In this work, we developed a physics-informed recurrent neural network modeling scheme that seamlessly integrates the noisy process data and mechanistic mathematical models in augmenting the learning process. A generalization error bound was first derived for the PIRNN models based on the nominal system using the Rademacher complexity technique from statistical learning theory. Subsequently, a PIRNN-LMPC control scheme was formulated and implemented to control nonlinear systems subject to Gaussian measurement noise. A chemical reactor example was employed to demonstrate the effectiveness of the PIRNN-LMPC scheme, in which the dropout technique was utilized to facilitate the learning of the underlying process dynamics (ground truth) and to prevent overfitting the noisy data. Both the open- and closed-loop simulations demonstrated improved model accuracy and control performance using PIRNN models.

CRediT authorship contribution statement

Yingzhe Zheng: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft. **Cheng Hu:** Methodology, Formal analysis, Writing – original draft. **Xiaonan Wang:** Supervision. **Zhe Wu:** Conceptualization, Methodology, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Data availability

Data will be made available on request

References

- [1] T. Zhao, Y. Zheng, J. Gong, Z. Wu, Machine learning-based reduced-order modeling and predictive control of nonlinear processes, *Chem. Eng. Res. Des.* 179 (2022) 435–451.
- [2] Y. Zheng, X. Wang, Z. Wu, Machine learning modeling and predictive control of the batch crystallization process, *Ind. Eng. Chem. Res.* 61 (16) (2022) 5578–5592.
- [3] T. Xiao, Z. Wu, P.D. Christofides, A. Armaou, D. Ni, Recurrent neural-network-based model predictive control of a plasma etch process, *Ind. Eng. Chem. Res.* 61 (1) (2021) 638–652.
- [4] F. Arnold, R. King, State-space modeling for control based on physics-informed neural networks, *Eng. Appl. Artif. Intell.* 101 (2021) 104195.
- [5] Y. Ba, G. Zhao, A. Kadambi, Blending diverse physical priors with neural networks, 2019, arXiv preprint arXiv:1910.00201.
- [6] A. Daw, A. Karpadne, W.D. Watkins, J.S. Read, V. Kumar, Physics-guided neural networks (PGNN): An application in lake temperature modeling, in: *Knowledge-Guided Machine Learning*, Chapman and Hall/CRC, 2017, pp. 353–372.
- [7] M.R. Kellman, E. Bostan, N.A. Repina, L. Waller, Physics-based learned design: optimized coded-illumination for quantitative phase imaging, *IEEE Trans. Comput. Imaging* 5 (2019) 344–353.
- [8] Y. Lu, M. Rajora, P. Zou, S.Y. Liang, Physics-embedded machine learning: case study with electrochemical micro-machining, *Machines* 5 (2017) 4.
- [9] Z. Wu, D. Rincon, P.D. Christofides, Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes, *J. Process Control* 89 (2020) 74–84.
- [10] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* 378 (2019) 686–707.
- [11] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, *Nat. Rev. Phys.* 3 (6) (2021) 422–440.
- [12] E.A. Antonelo, E. Camponogara, L.O. Seman, E.R. de Souza, J.P. Jordanou, J.F. Hubner, Physics-informed neural nets for control of dynamical systems, 2021, arXiv preprint arXiv:2104.02556.
- [13] X. Meng, Z. Li, D. Zhang, G.E. Karniadakis, PPINN: Parareal physics-informed neural network for time-dependent PDEs, *Comput. Methods Appl. Mech. Engrg.* 370 (2020) 113250.
- [14] X. Meng, G.E. Karniadakis, A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems, *J. Comput. Phys.* 401 (2020) 109020.
- [15] M. Mahmoudabadbozchelou, M. Caggioni, S. Shahsavari, W.H. Hartt, G.E. Karniadakis, S. Jamali, Data-driven physics-informed constitutive meta-modeling of complex fluids: A multifidelity neural network (MFNN) framework, *J. Rheol.* 65 (2) (2021) 179–198.
- [16] L. Yang, X. Meng, G.E. Karniadakis, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, *J. Comput. Phys.* 425 (2021) 109913.
- [17] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, Deepxde: A deep learning library for solving differential equations, *SIAM Rev.* 63 (1) (2021) 208–228.
- [18] S.A. Niaki, E. Haghighat, T. Campbell, A. Poursartip, R. Vaziri, Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture, *Comput. Methods Appl. Mech. Engrg.* 384 (2021) 113959.
- [19] E. Zhang, M. Dao, G.E. Karniadakis, S. Suresh, Analyses of internal structures and defects in materials using physics-informed neural networks, *Sci. Adv.* 8 (7) (2022) eabk0644.
- [20] M.M. Almajid, M.O. Abu-Al-Saud, Prediction of porous media fluid flow using physics informed neural networks, *J. Pet. Sci. Eng.* 208 (2022) 109205.
- [21] Y. Lin, E.D. Sontag, A universal formula for stabilization with bounded controls, *Systems Control Lett.* 16 (6) (1991) 393–397.
- [22] K. Yao, G. Zweig, M. Hwang, Y. Shi, D. Yu, Recurrent neural networks for language understanding, in: *Proceedings of the Interspeech*, 2013, pp. 2524–2528.
- [23] A. Graves, A. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.
- [24] C. Angermueller, T. Pärnamaa, L. Parts, O. Stegle, Deep learning for computational biology, *Mol. Syst. Biol.* 12 (7) (2016) 878.
- [25] A. Sherstinsky, Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network, *Physica D* 404 (2020) 132306.
- [26] A.M. Schäfer, H.-G. Zimmermann, Recurrent neural networks are universal approximators, *Int. J. Neural Syst.* 17 (04) (2007) 253–263.
- [27] S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics-informed neural networks: where we are and what's next, *J. Sci. Comput.* 92 (3) (2022) 88.
- [28] R.K. Al Seyab, Y. Cao, Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation, *J. Process Control* 18 (6) (2008) 568–581.
- [29] D. Chaffart, L.A. Ricardez-Sandoval, Optimization and control of a thin film growth process: A hybrid first principles/artificial neural network based multiscale modelling approach, *Comput. Chem. Eng.* 119 (2018) 465–479.
- [30] Z. Wu, A. Tran, Y.M. Ren, C.S. Barnes, S. Chen, P.D. Christofides, Model predictive control of phthalic anhydride synthesis in a fixed-bed catalytic reactor via machine learning modeling, *Chem. Eng. Res. Des.* 145 (2019) 173–183.
- [31] N. Sitapure, J.S. Kwon, Neural network-based model predictive control for thin-film chemical deposition of quantum dots using data from a multiscale simulation, *Chem. Eng. Res. Des.* 183 (2022) 595–607.
- [32] F. Bonassi, M. Farina, J. Xie, R. Scattolini, On recurrent neural networks for learning-based control: Recent results and ideas for future developments, *J. Process Control* 114 (2022) 92–104.
- [33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, *Adv. Neural Inf. Process. Syst.* 32 (2019).
- [34] N. Golowich, A. Rakhlin, O. Shamir, Size-independent sample complexity of neural networks, in: *Proceedings of the Conference on Learning Theory*, PMLR, 2018, pp. 297–299.
- [35] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631.
- [36] T. Agrawal, Hyperparameter optimization in machine learning, Apress Berkeley, Berkeley, CA, USA, 2021, pp. 81–108.
- [37] P.S. Pravin, J.Z.M. Tan, K.S. Yap, Z. Wu, Hyperparameter optimization strategies for machine learning-based stochastic energy efficient scheduling in cyber-physical production systems, *Digit. Chem. Eng.* 4 (2022) 100047.
- [38] Z. Wu, D. Rincon, Q. Gu, P.D. Christofides, Statistical machine learning in model predictive control of nonlinear processes, *Mathematics* 9 (16) (2021) 1912.
- [39] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating PDEs, *IMA J. Numer. Anal.* 00 (2022) 1–43.
- [40] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs, *IMA J. Numer. Anal.* 42 (2022) 981–1022.
- [41] M. Mohri, A. Rostamizadeh, A. Talwalkar, *Foundations of Machine Learning*, MIT Press, 2018.
- [42] A. Maurer, A vector-contraction inequality for rademacher complexities, in: *Proceedings of the International Conference on Algorithmic Learning Theory*, Springer, 2016, pp. 3–17.
- [43] 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.

- [44] Z. Wu, A. Alnajdi, Q. Gu, P.D. Christofides, Statistical machine-learning-based predictive control of uncertain nonlinear processes, *AIChE J.* 68 (2022) e17642.
- [45] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: *International Conference on Machine Learning*, PMLR, 2016, pp. 1050–1059.
- [46] Z. Wu, D. Rincon, J. Luo, P.D. Christofides, Machine learning modeling and predictive control of nonlinear processes using noisy data, *AIChE J.* 67 (2021) e17164.
- [47] R. DiPietro, G.D. Hager, Deep learning: RNNs and LSTM, in: *Handbook of Medical Image Computing and Computer Assisted Intervention*, Elsevier, 2020, pp. 503–519.
- [48] Y. Zheng, T. Zhao, X. Wang, Z. Wu, Online learning-based predictive control of crystallization processes under batch-to-batch parametric drift, *AIChE J.* 68 (11) (2022) e17815.