Physics-Informed Machine Learning Modeling for Model Predictive Control of Nonlinear Processes

Yingzhe Zheng [a] and Zhe Wu [a, 1]

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

Abstract

This paper highlights our recent work on the incorporation of physical domain knowledge into machine learning modeling and predictive control of nonlinear processes. Specifically, we first review recurrent neural network (RNN) modeling for nonlinear dynamic systems, and discuss several methods to incorporate domain knowledge within neural network modeling. Then, we discuss physics-informed RNN (PIRNN) model that integrates physical laws with available data in the training process with a theoretical analysis of its generalization performance. When a multistage process is considered, process structural knowledge can be further utilized to improve PIRNN models, for which process-structure-based and weight-constrained-based modeling approaches are discussed. Finally, we present the formulation of RNN-based model predictive control (MPC) and conclude with discussions on some practical implementation issues with the PIRNN modeling approach.

Keywords

Machine Learning; Physical Domain Knowledge; Process Control; Model Predictive Control; Nonlinear Systems.

Introduction

Reliability, safety, and sustainability are the cardinal goals of any manufacturing enterprise in driving economic development and competitiveness, and the adoption and exploitation of advanced process control techniques such as model predictive control (MPC) thus naturally becomes the key enablers to these visions. Machine learning-based MPC (ML-MPC) presents an unprecedented opportunity for revolutionizing the manufacturing landscape, and has attracted substantial interest among researchers and practitioners. Fundamentally, ML-MPC is an optimization-based advanced control method that solves for the optimal control actions using a machine learning-based predictive model of the process by accounting for the intrinsic dynamic behaviors of the process and any physical constraints imposed on the manipulated inputs. A major challenge of ML-MPC has been associated with the generalizability of ML models, where limited or inadequately sampled training data can greatly compromise the prescriptive performance of ML-MPC in real-world applications (Karniadakis et al., 2021). In this respect, to warrant a satisfactory predictive performance of ML models, an exhaustively large dataset is typically indispensable, which, however, may not always be readily available or is prohibitively exorbitant and arduous to obtain.

To this end, the physics-informed neural network (PINN) modeling technique, as introduced by Raissi et al. (2019), is a viable potential alternative approach. Specifically, PINN effectively overcomes the small-data problem by informatively leveraging and incorporating the a priori physics-based insights (e.g., the underlying mass and energy balance equations) to provide additional penalty terms in the loss function of the learning framework. This, in turn, enhances the performance of the learning algorithm, as the ML models are now required to conform to the inherent dynamic behavior imposed not only by the empirical data but also by the mechanistic laws. The associated regularization effect engendered by embedding the underlying physics-based knowledge into the ML models thus moderates the reliance on extensively large training datasets. Currently, the physics-informed learning methodology is widely applied to feedforward neural networks Raissi et al. (2019), with its applicability to other notable neural network architectures (e.g., recurrent neural network (RNN)) remained to be investigated. Furthermore, theoretical analysis of the generalization performance of the PI-RNN model for nonlinear dynamic processes has not been reported.

[1] Corresponding author. Email: `wuzhe@nus.edu.sg`.

The financial support from NUS Start-up Grant and Pfizer Asia Manufacturing Pte. Ltd. (Grant/Award Number: R-279-000-624-592) is gratefully acknowledged.

In this work, we summarize our recent results on formalizing the generalization performance of PI-RNN, and on implementing the various physics-informed approaches to enhance the performance of RNN-based MPCs in controlling nonlinear dynamic processes.

Class of Nonlinear Process Systems

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

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

where $x \in \mathbf{R}^n$ and $u \in \mathbf{R}^k$ denote the state vector and the manipulated input vector, respectively. The input is bounded, i.e., $u \in U$, where $U := \{u_{\min} \leq u \leq u_{\max}\} \subset \mathbf{R}^k$ defines the maximum $u_{\max}$ and the minimum $u_{\min}$ allowable values of u. We assume $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$, and $n \times k$, respectively, and $f(0) = 0$ without loss of generality such that the origin is a steady-state of Eq. 1. Additionally, it is assumed that there exists a feedback controller that stabilizes the nonlinear system of Eq. 1 at the origin. Specifically, we assume that a continuously differentiable Lyapunov function $V(x)$ and a stabilizing controller $u = \Phi(x) \in U$ can be found to render the origin exponentially stable for the states in an open neighborhood D around the origin. A level set of $V(x)$ within the set D is chosen to be the stability region for the nonlinear system of Eq. 1, i.e., $\Omega_\rho := \{x \in D \mid V(x) \leq \rho\}$, $\rho > 0$. We use $|\cdot|$ to denote the Euclidean norm of a vector, and $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$ to denote set subtraction in this manuscript.

Incorporation of domain knowledge within ML modeling

The fundamental idea of physics-informed ML is essentially imposing physics-based constraints on an ML model through either designing specialized model architectures or introducing additional informative regularizing terms that reinforce the conformation to the observed patterns and the underlying governing physical laws. Specifically, carefully crafted NN architectures that are tailored to a priori domain knowledge have demonstrated superior performance (Karniadakis et al., 2021). The advent of convolutional NNs has reshaped the field of computer vision by endowing the NN with invariance properties that conform to natural representations of images. Furthermore, the design of the recurrent neural network, which has an architecture that resembles the numerical solutions of ordinary differentiation equations (ODEs), has found promising applications in modeling sequential data. The manipulation of NN weight parameters that respects the a priori known input-output relationships can also be potentially instrumental in facilitating its learning.

Another class of physics-informed approach is multi-task learning, where soft penalty constraints containing the a priori knowledge are incorporated into the loss function of the learning algorithm to guide the convergence of NN toward satisfying simultaneously the dynamic patterns embedded in the observed data and in the given set of physical constraints. This approach effectively minimizes the demand for training data and is especially useful in learning problems with limited or poorly sampled training data (Raissi et al., 2019).

Recurrent neural network (RNN) model

The RNN model with the following form is generally used to model the nonlinear system of Eq. 1 using time-series data:

$$\mathbf{h}_t = \sigma_h(U\mathbf{h}_{t-1} + W\mathbf{x}_t), \;\; \mathbf{y}_t = \sigma_y(Q\mathbf{h}_t) \tag{2}$$

where $\mathbf{x}_t \in \mathbf{R}^{r_x}$ and $\mathbf{y}_t \in \mathbf{R}^{r_y}$ denote the input and output at the t^{th} time step, respectively, and $\mathbf{h}_t$ denotes the hidden state at the t^{th} time step. W, U, and Q are the weight matrices for the input-to-hidden connections, hidden-to-hidden recurrent connections, and hidden-to-output connections, respectively. σ_h and σ_y are the activation functions for the hidden and output layer, respectively. Additionally, we assume that the RNN weights U, W, F and inputs $\mathbf{x}$ are bounded by $B_{U,F}, B_{W,F}, B_{V,F}, B_X$, respectively.

Traditional RNN modeling requires a representative training dataset that well captures the nonlinear dynamics of Eq. 1. Such datasets can be generated from industrial process operations, experimental studies, and computer simulations. For example, open-loop simulations of Eq. 1 for various initial conditions x_0 and control actions u can be conducted to generate a rich dataset, which will be partitioned into training, validation, and test datasets for the development of RNN models. It has been demonstrated in Wu et al. (2021) that the RNN models using simulation data can achieve a sufficiently small training loss and the desired generalization performance with a sufficient number of training samples. While computer simulations can provide a representative dataset that meets the minimum training sample size required by statistical learning theory, data collection and quality have been a major challenge for real-world chemical processes, as they are generally operated around steady-states without sufficient data that can be used to capture the nonlinear dynamics throughout the operating region. Therefore, harnessing domain-specific knowledge (e.g., the physical laws of Eq. 1) to improve neural network modeling is an emerging research field that presents many opportunities for further development.

Physics-informed RNN (PIRNN) model

Physics-guided learning effectively leverages mechanistic knowledge encapsulated in process data and mathematical models for building accurate 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 (i.e., the first-principles model of Eq. 1). In this respect, the output of the RNN model has to satisfy the relationship outlined by the underlying physical laws, as expressed in the ODE of Eq. 1, to warrant a good approximation performance. To put it formally, a new function $\mathcal{G}$ is first defined to be the equivalent of Eq. 1.

$$\mathcal{G}(\tilde{x}, u) := \dot{\tilde{x}} - F(\tilde{x}, u) \tag{3}$$

where $\tilde{x}$ denotes the RNN output, which takes the initial state vector x_0 of Eq. 1, and the manipulated input u as its inputs. Specifically, the RNN model is developed to predict the states of Eq. 1 for $0 \leq t \leq T$, where T is the maximum prediction horizon that depends on the time length of training data. To simplify the notation, we use $h(z;\theta)$ to represent the RNN hypothesis (i.e., Eq. 2) that approximates the ODE solutions x of the nonlinear system of Eq. 1 for time $0 \leq t \leq T$, i.e., $\tilde{x}(t) = h(z;\theta)$, $t \in [0,T]$, where $z = [x_0, u]$ denotes the RNN input vector that consists of the initial state x_0 and manipulated inputs u, and θ denotes the RNN weights. Consequently, as RNN learns to compute the solution x of the ODE governed by Eq. 1 (i.e., $h(z;\theta) \approx x$), the following expression holds.

$$\mathcal{G}(\tilde{x},u) := \dot{\tilde{x}} - F(\tilde{x},u) \approx 0,\ t \in [0,T] \tag{4}$$

The physics-informed learning approach outlined in this section can 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., the first-principles model of Eq. 1). Specifically, the physics-guided RNN $h(z;\theta)$ is trained using two groups of data, which correspond to their respective penalty terms presented in the mean squared error (MSE) loss function (Eq. 5), and an optimizer such as Adam. The first group of data, corresponding to the first loss term $MSE_{\mathcal{X}}$, consists of the dynamic process data captured from sensor measurement (i.e., contains both the input data to RNN and their corresponding output labels), which resembles the training data used in a typical regression problem. The second group of data, related to the second loss term $MSE_{\mathcal{G}}$, comprises only input data (i.e., collocation points encompassing the initial state vector and manipulated inputs) that are randomly and uniformly sampled from the designated operating regions where the ODE approximation (Eq. 4) should be valid.

$$MSE = w_{\mathcal{X}} MSE_{\mathcal{X}} + w_{\mathcal{G}} MSE_{\mathcal{G}} \tag{5}$$

$$MSE_{\mathcal{X}} = \frac{1}{N_{\mathcal{X}}} \sum_{n=1}^{N_{\mathcal{X}}} \frac{1}{N_T} \sum_{i=0}^{N_T} |x_n(t_{\mathcal{X},i}, u_{\mathcal{X},n}) - \tilde{x}_n(t_{\mathcal{X},i}, u_{\mathcal{X},n})|^2 \tag{6}$$

$$\begin{aligned} MSE_{\mathcal{G}} = & \frac{1}{N_{\mathcal{G}}} \sum_{n=1}^{N_{\mathcal{G}}} \frac{1}{N_T} \sum_{i=0}^{N_T} |\mathcal{G}(\tilde{x}_n(t_{\mathcal{G},i}, u_{\mathcal{G},n}), u_{\mathcal{G},n})|^2 \\ & + \frac{1}{N_{\mathcal{G}}} \sum_{n=1}^{N_{\mathcal{G}}} |x_n(t_{\mathcal{G},0}, u_{\mathcal{G},n}) - \tilde{x}_n(t_{\mathcal{G},0}, u_{\mathcal{G},n})|^2 \end{aligned} \tag{7}$$

where $N_{\mathcal{X}}$, $N_{\mathcal{G}}$, and N_T denote the number of training data (i.e., the number of dynamic state trajectories captured from sensors), collocation points (i.e., comprising of only initial state vector and manipulated inputs), and outputs of RNN at time $0 \leq t \leq T$, 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. $w_{\mathcal{X}}$ and $w_{\mathcal{G}}$ are coefficient weights that balance the scale of the two MSE loss terms. These weights 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 first-principles model of Eq. 1 for time $0 \leq t \leq T$ 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 $t \in [0,T]$. $\tilde{x}_n(t_{\mathcal{G},i}, u_{\mathcal{G},n})$ represents the desired output from the RNN at time $t_{\mathcal{G},i}$, and under the manipulated inputs $u_{\mathcal{G},n}$. 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 dynamic system, and the collocation points should thus be randomly sampled predominantly from operating regions that are beyond the range of training data. Consequently, the second loss term $MSE_{\mathcal{G}}$ aims to enforce the dynamic behavior imposed by the physical-law-based ODE at the finite set of randomly sampled collocation points that are beyond the range of operating conditions captured by the training data. Due to the incorporation of physical laws, the physics-informed ML modeling approach is able to overcome the problem of low data availability, satisfy the underlying conservation laws (e.g., conservation of energy, mass, momentum), and improve the approximation performance on new data points.

Generalization performance of PIRNN model

Generalization error has been widely used to characterize the ability of the machine learning model to adapt to new unseen data. Therefore, a theoretical generalization error bound is important for the implementation of machine learning models to real-world chemical processes based on the training data from past process operations, and should be taken into account in the controller design to improve the robustness. We consider an extreme scenario for the PIRNN model where only the first-principles model of Eq. 1 and the initial conditions x_0 are available while no historical data is provided. In other words, the PIRNN will be trained using the loss function of Eq. 7 only to capture the nonlinear dynamics of Eq. 1 and fit the initial condition x_0. The generalization error for the RNN 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 $u \in U$.

$$R_D(\theta) := \mathbb{E}_{\boldsymbol{z} \sim Z}[L(\mathcal{G}(h(z;\theta),u),0)] + \gamma \mathbb{E}_{\boldsymbol{z} \sim Z}[L(\tilde{x}_0, x_0)] \tag{8}$$

where $L(\cdot,\cdot)$ denotes the loss function (e.g., the mean squared error (MSE) function that is typically used for regression problems). γ is a positive real number that balances the contributions of two loss terms in Eq. 8. $\tilde{x}_0 = \tilde{x}(t=0)$ is the initial condition predicted by the PIRNN model (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.

$$R_S(\theta) := \frac{1}{N_{\mathcal{G}}} \sum_{i=1}^{N_{\mathcal{G}}} L(\mathcal{G}(h(z_i;\theta),u),0) + \gamma \frac{1}{N_{\mathcal{G}}} \sum_{i=1}^{N_{\mathcal{G}}} L(\tilde{x}_{0i}, x_{0i}) \tag{9}$$

where $N_{\mathcal{G}}$ is the number of training samples from the same distribution Z. 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 (i.e., the first term representing the system of Eq. 1) in the loss function, we design a new PIRNN model $\tilde{h}(z;\theta)$ that approximates the original PIRNN model $h(z;\theta)$ via the following equation.

$$h(z;\theta) = (t-t_0)\tilde{h}(z;\theta) + x_0 \tag{10}$$

It can be observed from Eq. 10 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 accurately predicted by $h(z;\theta)$ for any $\tilde{h}(z;\theta)$. Therefore, it remains to show that the approximation of $h(z;\theta)$ through Eq. 10 can capture the nonlinear dynamics of Eq. 3. Specifically, by substituting Eq. 10 into Eq. 3, we have the following equation.

$$\begin{aligned}\tilde{\mathcal{G}}(\tilde{h}(z;\theta),u) =& (t-t_0)\dot{\tilde{h}}(z;\theta) - F((t-t_0)\tilde{h}(z;\theta)+x_0,u)\\ &+\tilde{h}(z;\theta)\end{aligned} \tag{11}$$

Therefore, 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. 10. The optimal weight matrices of the new PIRNN model $\tilde{h}(z;\theta_D)$ can be obtained as follows.

$$\theta_D = \arg\min_{\theta} R_D(\theta) := \mathbb{E}_{z\sim Z}[L(\tilde{\mathcal{G}}(\tilde{h}(z;\theta),u),0)]. \tag{12}$$

Similarly to Eq. 9, the optimization of the RNN weights can be practically solved by minimizing the empirical error using a number of training samples. Based on the optimal weights derived from empirical loss (denoted by θ_S), the RNN model that approximates the nonlinear system of Eq. 1 is obtained as $h(z;\theta_S) = (t-t_0)\tilde{h}(z;\theta_S)+x_0$. Therefore, by designing $h(z;\theta_D)$ using Eq. 10, the training process for the original PIRNN model $h(z;\theta_D)$ using the loss function of Eq. 8 is equivalent to training the new PIRNN model $\tilde{h}(z;\theta_D)$ using the new loss function of Eq. 12. Additionally, since the new loss function of Eq. 12 only includes one loss term, and does not depend on the weight parameter γ, a generalization error bound can be developed for the new PIRNN model $\tilde{h}(z;\theta_D)$ following the method in Wu et al. (2021). Specifically, we use Rademacher complexity method in statistical learning theory to quantify the richness of an ML model class, where the definition of empirical Rademacher complexity of a hypothesis class $\mathcal{H}$ is given below (Mohri et al., 2018).

$$\mathcal{R}_S(\mathcal{H}) = \mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \varepsilon_i h(s_i)\right] \tag{13}$$

where ε_i, $i = 1,\dots,m$ are Rademacher random variables that are independent and identically distributed (i.i.d.) and satisfy $\mathbb{P}(\varepsilon_i = -1) = \mathbb{P}(\varepsilon_i = 1) = 0.5$. Consider a hypothesis class $\mathcal{H}$ of PIRNN models $\tilde{h}(z;\theta)$ that map 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). Let $\mathcal{L}_t$ be the set of loss functions that satisfy local Lipschitz continuity property (i.e., $|L(\tilde{\mathcal{G}}(\tilde{h}_1(z;\theta),u),0) - L(\tilde{\mathcal{G}}(\tilde{h}_2(z;\theta),u),0)| \le L_r\left|\tilde{h}_1(z;\theta)-\tilde{h}_2(z;\theta)\right|$) and are associated with the PIRNN class $\mathcal{H}$, i.e., $\mathcal{L}_t = \{l_t : (z,\bar{y}) \to L(\tilde{\mathcal{G}}(\tilde{h}(z;\theta),u),0), \tilde{h}\in\mathcal{H}\}$, where z and $\bar{y}$ are the PIRNN input and the ground-truth output values, respectively. The following proposition derives the generalization error bound for PIRNN models following the proof techniques in Mohri et al. (2018); Wu et al. (2021):

Proposition 1. *Given a dataset $S = (z_{i,t},\bar{y}_{i,t})_{t=1}^{T}$, $i = 1,\dots,m$ with m i.i.d. data samples, $i = 1,\dots,m$, and a loss function class $\mathcal{L}_t$ that is associated with the class of PIRNN functions $\mathcal{H}$ trained following Eq. 12. Then, the following inequality holds with probability at least $1-\delta$ over S.*

$$\begin{aligned}\mathbb{E}[l_t(z,\bar{y})] \le& \frac{1}{m}\sum_{i=1}^{m} l_t(z_i,\bar{y}_i) + 2\mathcal{R}_S(\mathcal{L}_t) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}}\\ \le& \frac{1}{m}\sum_{i=1}^{m} l_t(z_i,\bar{y}_i) + O\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right)\\ &+3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}}\end{aligned} \tag{14}$$

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. d_y is the PIRNN output dimension and L_r is the local Lipschitz constant.

The first line in Eq. 14 shows 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 of the confidence δ and the training sample size m (last term). Since the first and the last terms can be readily calculated after the training loss and the parameters m, δ are obtained, to further show that the Rademacher complexity term (second term) can be bounded, the second line in Eq. 14 is derived following the proof techniques in Wu et al. (2021), from which it is demonstrated that the Rademacher complexity depends on the bounds of PIRNN weights and inputs, the time length of data sequences, the output dimension, and the property of the loss function.

Process-structure-based RNNs for multistage processes

Neural networks are generally developed with fully-connected layers to model the nonlinear system of Eq. 1, where the weight parameters in all layers are optimized during the training process to minimize the training error. However, the underlying assumption that all the inputs affect all the neural network neurons, followed by all the outputs, in a fully-connected neural network may not be valid for all chemical processes, especially in a multistage chemical process, where upstream units affect downstream units but the impact is negligible in the opposite direction. In this case, the performance of neural network for multistage, complex processes can be improved by accounting for the fact that only a portion of inputs affects a portion of outputs. In this section, we will discuss two RNN modeling approaches that incorporate process structural knowledge into the architecture design (Wu et al., 2020).

1) Partially-connected RNN

We consider the nonlinear system of Eq. 1 under the assumption that the state vector x^1 is affected by u^1 only, and x^2 is affected by both u^1 and u^2, where $x = [x^1,\ x^2] \in \mathbf{R}^n$ and $u = [u^1 \in \mathbf{R}^{k_1},\ u^2 \in \mathbf{R}^{k_2}] \in \mathbf{R}^k$, $k_1 + k_2 = k$. For example, the two CSTRs in series in Fig. 1 is a system that meets this assumption, where the inlet stream to the second reactor does not affect the first reactor. Instead of building a fully-connected RNN model for the whole system of two CSTRs, we can decouple the fully-connected RNN model to develop a partially-connected RNN structure as shown on the right of Fig. 1 to resemble the process structure shown on the left. It is observed in Fig. 1 that in the partially-connected RNN, u^1 only affects x^1, and both u^1 and u^2 have an impact on the output u^2, which is consistent with the structural relationship in the example of two CSTRs on the left.

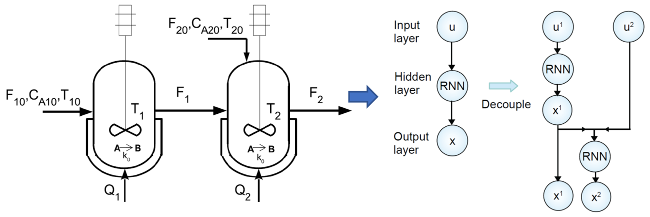

Figure 1: An example of process-structure-based RNN for two CSTRs in series.

By designing an RNN structure that explicitly removes the connection between u^2 and x^1, a better training performance can be achieved in the sense that less training time and fewer neurons can be used to achieve the desired modeling accuracy that is as good as that under a fully-connected model. While in general the training performance of RNNs can be improved with a more complex neural network structure (i.e., more neurons and layers for a fully-connected RNN), the connection between u^2 and x^1 physically does not exist, and may instead lead to a negative impact on the training process (e.g., sub-optimal solutions). Therefore, by infusing a priori process structural knowledge into RNN modeling, the model performance is improved by revealing the correct direction for RNNs to find the optimal weight parameters. A simulation study of the chemical process example of Fig. 1 was conducted in Wu et al. (2020), from which it was demonstrated that the partially-connected RNN model achieves a higher prediction accuracy than the fully-connected RNN model using the same number of neurons and layers. Additionally, when incorporating both RNN models in model predictive controllers, the closed-loop performance using a partially-connected RNN model also outperformed the one using a fully-connected model in terms of fast convergence to the steady-state.

2) Weight-constrained RNN

In addition to partially-connected architecture, weight constraints can be incorporated into RNN models to represent the input-output relationship derived from process structural knowledge in Fig. 1. Specifically, the weights connecting u^2 and x^1 (dashed gray lines in the left figure of Fig. 2) can be constrained such that the connection between u^2 and x^1 is weakened during the training process. Additionally, weight constraints can be integrated in the loss function as a regularization term as follows:

$$L = \sum_{i=1}^{N_d} (x_i - \hat{x}_i)^2 + \lambda \Pi_w \tag{15}$$

where Π_w is the product of weights that correspond to the connections between u^2 and x^1, and $\lambda > 0$ is a parameter that balances the contributions of regular MSE loss (the first term in Eq. 15) and weight penalty. It should be noted that the connection between u^2 and x^1 cannot be fully removed using the structure shown in the left figure of Fig. 2, since making any weight zero will lead to a complete disconnection between the neurons it connects. Therefore, to fully remove the connection between u^2 and x^1, another weight-constrained RNN structure is designed as shown in the right figure of Fig. 2, where additional neurons $r_{h+1}, \ldots,\ r_{2h}$ are designed in the hidden layer to build the connection between u^2 and x^2. In this way, u^2 does not affect the prediction of x^1, which is consistent with the process structure of the two CSTRs in Fig. 1. While we only discuss an example of two CSTRs in this section, the proposed process-structure-based RNN modeling methods can be extended to multi-stage processes that could be highly coupled, and of high dimension. In that case, relative gain array and feature selection methods can be used to determine the best input-output pairings for multivariate processes (Zhao et al., 2022).

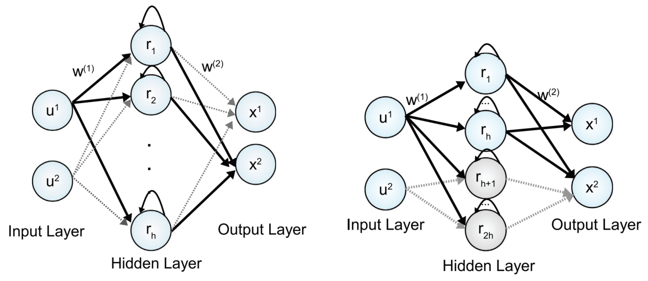

Figure 2: Weight-constrained RNNs for the example of two CSTRs in series.

ML-based model predictive control (MPC)

This section presents the formulation of a model predictive controller that incorporates a physics-informed RNN model to make predictions about future process outputs. A Lyapunov-based model predictive control (LMPC) scheme is designed to optimize process performance and ensure system stability. Specifically, the optimization problem of LMPC using RNN models is given as follows: (Wu et al. (2019))

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} (\tilde{x}^T Q \tilde{x} + u^T R u)\, dt \tag{16a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{16b}$$

$$u(t) \in U,\ \forall\, t \in [t_k, t_{k+N}) \tag{16c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{16d}$$

$$\dot{V}(x(t_k), u) \leq \dot{V}(x(t_k), \Phi_{nn}(x(t_k))), \quad \text{if } x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}} \tag{16e}$$

$$V(\tilde{x}(t)) \leq \rho_{nn},\ \forall\, t \in [t_k, t_{k+N}),\ \text{if } x(t_k) \in \Omega_{\rho_{nn}} \tag{16f}$$

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 $F_{nn}(x,u)$ represents the RNN model of Eq. 10 or the PIRNN model developed in the previous section. $\dot{V}(x,u)$ represents the time derivative of $V(x)$, i.e., $\frac{\partial V(x)}{\partial x}(F_{nn}(x,u))$. In the optimization problem of Eq. 16, the objective function of Eq. 16a is the integral of the cost function $L_{MPC}(\tilde{x},t) = (\tilde{x}^T Q\tilde{x} + u^T Ru)$ over the prediction horizon, where $L_{MPC}(0,0) = 0$ and $L_{MPC}(\tilde{x},t) > 0$, $\forall(\tilde{x},t) \neq (0,0)$. The constraint of Eq. 16b is the RNN model used to predict future states. Eq. 16c defines the input constraints over the entire prediction horizon. Eq. 16d takes the state measurement at each sampling time t_k as the initial condition $\tilde{x}(t_k)$ to solve Eq. 16b. The two Lyapunov-based constraints of Eq. 16e-16f guarantee that the closed-loop state remains inside the stability region Ω_ρ for all times and can be bounded in a terminal set around the origin ultimately. The LMPC of Eq. 16 is implemented in a sample-and-hold fashion, and the first control action from the optimal input trajectory $u^*(t), t \in [t_k, t_{k+N})$ will be applied for the following sampling period. Closed-loop stability of the nonlinear system of Eq. 1 under LMPC has been studied in our recent work (Wu et al., 2019, 2021). Specifically, Wu et al. (2019) proved closed-loop stability in a deterministic manner based on the requirement that the test error of RNN models should be bounded for all states in the stability region. More recently, in Wu et al. (2021), probabilistic closed-loop stability results were derived for the nonlinear system under LMPC, accounting for the fact that the generalization error of RNN models is bounded only in a probability manner, provided that training and test data are of the same distribution. Interested readers may refer to Wu et al. (2019, 2021) for detailed discussions of closed-loop stability.

Practical implementation issues in NN modeling and predictive control

In this section, we discuss some practical challenges such as model uncertainty and curse of dimensionality when implementing physics-informed NN modeling and predictive control methods to real-world chemical processes.

Online machine learning

Since machine learning models are generally developed offline using historical data from past normal operations that do not involve model uncertainty, the resulting machine learning models may not be able to accurately predict real-time process dynamics in the presence of model uncertainty (e.g., process disturbances, model-plant mismatch, and time-varying process dynamics). To address this issue, online learning of machine learning models provides a promising solution to improve models using real-time data for better prediction and control performance under MPC. For example, in Wu et al. (2019); Zheng et al. (2022), event-trigger and error-trigger mechanisms were proposed to update RNN models online using real-time process data.

Reduced-order ML modeling

ML modeling of large-scale, complex chemical processes may encounter the issue of curse of dimensionality, which implies that an increase in data dimension will lead to an exponential increase in computational efforts for training. To this end, model order reduction techniques such as feature extraction and A variety of feature selection methods including filter, wrapper, and embedded methods were explored in the context of RNN modeling of large-scale nonlinear processes in Zhao et al. (2022). Additionally, in Zhao et al. (2022), a feature extraction method termed autoencoder was integrated with RNN models to develop reduced-order RNN models that can capture the process dynamics in latent space.

Conclusions

In this work, we presented an overview of recent research results on physics-informed ML modeling of nonlinear processes. The formulation of physics-informed RNN, its generalization performance, and process-structure-based RNN architectures for multistage processes were discussed, followed by the design of a Lyapunov-based MPC scheme using RNN models. Finally, the potential solutions to some practical challenges in NN modeling such as model uncertainty and curse of dimensionality were discussed.

References

Karniadakis, G. E., I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang (2021). Physics-informed machine learning. *Nature Reviews Physics 3*(6), 422–440.

Mohri, M., A. Rostamizadeh, and A. Talwalkar (2018). *Foundations of machine learning*. MIT press.

Raissi, M., P. Perdikaris, and G. E. Karniadakis (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics 378*, 686–707.

Wu, Z., D. Rincon, and P. D. Christofides (2019). Real-time adaptive machine-learning-based predictive control of nonlinear processes. *Industrial & Engineering Chemistry Research 59*, 2275–2290.

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

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

Wu, Z., A. Tran, D. Rincon, and P. D. Christofides (2019). Machine learning-based predictive control of nonlinear processes. part i: theory. *AIChE Journal 65*, e16729.

Zhao, T., Y. Zheng, J. Gong, and Z. Wu (2022). Machine learning-based reduced-order modeling and predictive control of nonlinear processes. *Chemical Engineering Research and Design 179*, 435–451.

Zhao, T., Y. Zheng, and Z. Wu (2022). Feature selection-based machine learning modeling for distributed model predictive control of nonlinear processes. *Computers & Chemical Engineering*, in press.

Zheng, Y., T. Zhao, X. Wang, and Z. Wu (2022). Online learning-based predictive control of crystallization processes under batch-to-batch parametric drift. *AIChE Journal 68*, e17815.