Digital Chemical Engineering 3 (2022) 100027

Contents lists available at ScienceDirect

Digital Chemical Engineering

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

Research Protocols

Improving computational efficiency of machine learning modeling of nonlinear processes using sensitivity analysis and active learning

Tianyi Zhao[a,b], Yingzhe Zheng[b], Zhe Wu[b,*]

[a] Joint School of National University of Singapore and Tianjin University, International Campus of Tianjin University, Binhai New City, Fuzhou 350207, China
[b] Department of Chemical and Biomolecular Engineering, National University of Singapore, 117585, Singapore

ARTICLE INFO

Keywords:
Model predictive control
Machine learning
Recurrent neural networks
Feature selection
Sensitivity analysis
Active learning

ABSTRACT

In this work, we develop a model reduction method using sensitivity analysis and active learning to improve the computational efficiency of machine learning modeling of nonlinear processes. Specifically, sensitivity analysis is first used to identify important connections between model outputs and inputs. Subsequently, active learning is used to enrich the training set by iteratively identifying the training data that most efficiently improve model performance. Reduced-order recurrent neural networks (RNN) using the important input features obtained from sensitivity analysis are developed to approximate the nonlinear system, and are incorporated within model predictive control (MPC) to stabilize the nonlinear system at the steady-state. Finally, the effectiveness of the proposed machine learning modeling approach using sensitivity analysis and active learning and machine-learning-based predictive control scheme are demonstrated using a reactor-reactor-separator process example.

1. Introduction

Modeling and control of high-dimensional and nonlinear processes in chemical industry have remained as major challenges for process systems engineers. Since first-principles modeling approach is often susceptible to high computational costs and is of low predictive accuracy in practice especially in modeling large-scale and complex chemical processes, machine learning (ML) or data-driven modeling approach, in contrast, has gathered substantial attentions recently due to its ability to approximate complex nonlinear dynamics and its improved computational efficiency when dealing with large datasets obtained from process operations. ML has been widely applied in modeling chemical industry processes and has achieved remarkable results.

Among many ML algorithms, recurrent neural network (RNN) has been one of the most popular ML modeling approaches due to its ability to capture dynamic behaviors of nonlinear dynamic systems using time-series data. For modeling large-scale chemical processes with high-dimensional input and output space, the required size of dataset grows exponentially with the input/output dimensions. Additionally, the complexity of the RNN hypothesis class also increases for modeling large-scale processes such that longer training time, more neurons, and more layers are required for building RNN models with a desired accuracy, which ultimately lead to higher computational costs for practical implementation of large-scale RNN models. Therefore, to mitigate the aforementioned drawbacks of highly complex RNN models, model dimension reduction techniques are necessary when modeling large-scale nonlinear systems. Generally, dimension reduction can be divided into two categories: feature extraction and feature selection, where feature extraction (or feature transformation) generates a new reduced set of features derived from the original dataset while feature selection selects a subset of the existing features Cord and Cunningham (2008); Zheng et al. (2022).

Considering the various feature extraction methods such as partial least squares and clustering, principal component analysis (PCA) and its variants (e.g., nonlinear PCA) are likely the most popular techniques due to their effectiveness and ease of implementation (Shlens, 2014). Fundamentally, PCA aims to reduce the dimensionality of the input space by computing a new set of orthogonal variables (termed as principal components (PCs)) as linear or nonlinear combinations of the original variables with the largest variance (Maimon and Rokach, 2005). Although there are as many PCs as the number of original inputs, only the first few PCs are selected for model construction as they are capable of explaining most of the variance, and the others can thus be omitted at a minimal loss of information. However, since feature extraction transforms the original features, identification of their relative contribution (e.g., the relative importance of one feature to another in predicting the output variables) is not in a straight forward fashion as it requires additional analysis and specialized knowledge, and the transformed features are generally uninterpretable (Khalid et al., 2014).

On the contrary, feature selection reduces model dimension by selecting a subset of the original features, which can be readily interpreted and leads to an improved understanding of data based on a priori process knowledge (Ladha and Deepa, 2011). In general, tra-

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

https://doi.org/10.1016/j.dche.2022.100027
Received 17 March 2022; Received in revised form 7 April 2022; Accepted 14 April 2022

ditional feature selection approaches can be divided into three categories: wrapper-, embedded-, and filter-based methods. Wrapper methods are based on greedy search algorithms where the predictive performances of a specific machine learning algorithm on all possible combinations of subsets of features are evaluated (Kohavi and John, 1997). Typically, each subset of variables is chosen under certain search algorithm (e.g., sequential selection algorithms (Reunanen, 2003); genetic algorithm (JouanRimbaud et al., 1995)), and the subset that engenders the best model predictive performance will be selected as the important features. For high-dimensional datasets, wrapper methods are computationally expensive due to the exponential growth of the search space (El Aboudi and Benhlima, 2016) (e.g., for a process of N variables, the search space is 2^N). Embedded methods integrate the feature selection as a part of model training process, which are of less computational complexity than wrapper methods (Hsu et al., 2011; Rong et al., 2019). However, both wrapper and embedded methods are mostly adopted in classification problems only (Rong et al., 2019). Filter methods (e.g., using Pearson correlation) are data-based, which select features that have the highest scores from various statistical tests. They stand out for selection efficiency and independence of learning algorithms. However, filter methods may lead to poor performance for systems with coupled variables (Kamalov, 2018).

Moreover, due to its ease of explicability, feature selection is often used prior to the training process to improve the machine learning model performance in terms of learning efficiency, predictive accuracy, and effective data collection. It is a crucial component in ML for data dimension reduction by removing irrelevant and redundant features (Dhal and Azad, 2022). Particularly, sensitivity analysis, which determines the effect of an input variable on model output response to quantify its uncertainty, has been used for feature selection in a number of regression problems, e.g., Kowalski and Kusy (2017); Nikishova et al. (2020). Additionally, sensitivity-based feature selection method has been reported to be a powerful tool for feature evaluation and selection, and has been implemented in regression problems and systems with coupling variables (Kamalov, 2018; Naik et al., 2021). It has also been demonstrated to integrate well with machine learning models such as neural networks in Kowalski and Kusy (2017); Zurada et al. (1994).

In addition to sensitivity analysis, active learning is also effective in reducing the size of dataset and minimizing the cost of obtaining redundant data by avoiding similar samples which give little extra information Qiu et al. (2016); Sarma et al. (2019). At its core, active learning aims to maximize the generalization prediction performance of the model using as few training data as possible Cai et al. (2013); Yang et al. (2022). Although active learning has found extensive applications in classification problems, it has proved to be useful in regression problems (e.g., in elucidating relations between fabrication recipes and sensor performance for sensor design Cai et al. (2013)). Some common active learning strategies include predictive variance for Gaussian process regression Cohn et al. (1994), query by committee Burbidge et al. (2007), and maximizing expected model change Cai et al. (2013); Settles et al. (2007). The problem of active learning can be reduced to the problem of uncertainty estimation of the model (i.e., excluding training data with low uncertainty in prediction) Kowalski and Kusy (2017); Sarma et al. (2019).

Motivated by the above considerations, in this work, we improve the computational efficiency of machine learning modeling of large-scale nonlinear processes using sensitivity analysis and active learning, and develop a machine-learning-based MPC scheme using reduced-order RNN models. Specifically, in Section 2, preliminaries including the class of nonlinear systems, assumptions on system stability, and recurrent neural networks are presented. In Section 3, sensitivity analysis, active learning, and the development of reduced-order RNN models are introduced. In Section 4, the MPC using reduced-order RNN model is developed to stabilize the nonlinear system. In Section 5, a reactor-reactor-separator process example is used to demonstrate the implementation of the proposed computationally efficient ML modeling method.

2. Preliminaries

2.1. Notation

Throughout the manuscript, the notation $|\cdot|$ denotes the Euclidean norm of a vector. x^T denotes the transpose of x. The function $f(\cdot)$ is of class $\mathcal{C}^1$ if it is continuously differentiable in its domain. If a continuous function $\alpha : [0, a) \to [0, \infty)$ is strictly increasing and is zero only when evaluated at zero, it belongs to class $\mathcal{K}$ functions. "\" denotes set subtraction (i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$).

2.2. Class of systems

We consider a continuous-time nonlinear system consisting of m interconnected subsystems. Each of the subsystem can be described by the following system of nonlinear ordinary differential equations (ODE):

$$\dot{x}_i(t) = f_i(x) + g_{si}(x)u_i,\ x(t_0) = x_0,\ i = 1, \dots, m \tag{1}$$

where $x_i(t) \in \mathbf{R}^{n_{xi}}$ denotes the state vector, and $u_i(t) \in \mathbf{R}^{n_{ui}}$ denotes the manipulated input vector of the i^{th} subsystem. $x \in \mathbf{R}^{n_x}$ denotes the state vector of the entire system, where $x = [x_1^T \dots x_i^T \dots x_m^T]$. Then, the entire nonlinear system can be described by:

$$\dot{x}(t) = f(x) + \sum_{i=1}^{m} g_i(x)u_i(t) \tag{2}$$

where $f = [f_1^T \dots f_i^T \dots f_m^T]^T$, $g_i = [\mathbf{0}^T \dots g_{si}^T \dots \mathbf{0}^T]^T$, and $\mathbf{0}$ is the zero matrix of appropriate dimensions. The control action constraint is $u_i \in U := \{u^{\min} \le u_i \le u^{\max}, i = 1, \dots, m\} \subset \mathbf{R}^{n_{ui}}$, where the input constraints U are implemented in an element-wise manner. $f(\cdot)$, and $g_i(\cdot), i = 1, \dots, m$ are assumed to be locally Lipschitz vector functions. Throughout the manuscript, the initial time t_0 is taken to be zero ($t_0 = 0$). It is assumed that the origin is a steady-state of the unforced nonlinear system (i.e., $u_i(t) = 0, i = 1, \dots, m$) of Eq. (2) such that $f(0) = 0$.

2.3. Stabilization via Control Lyapunov function

We assume a stabilizing control law $h(x) = [h_1(x) \dots h_m(x)]$ with $u_i = h_i(x) \in U$ exists for the nonlinear system of Eq. (2) such that the origin of Eq. (2) is rendered exponentially stable. The stabilizability assumption implies there exist functions $\alpha_i(\cdot), i = 1, 2, 3, 4$ of class $\mathcal{K}$ and a $\mathcal{C}^1$ Control Lyapunov function $V(x)$ that satisfies the following inequalities for all x in an open neighborhood O around the origin Christofides et al. (2013); Zhao et al. (2022):

$$\alpha_1(|x|) \le V(x) \le \alpha_2(|x|), \tag{3a}$$

$$\frac{\partial V(x)}{\partial x}(f(x) + \sum_{i=1}^{m} g_i(x)h_i(x)) \le -\alpha_3(|x|), \tag{3b}$$

$$\left|\frac{\partial V(x)}{\partial x}\right| \le \alpha_4(|x|) \tag{3c}$$

A level set of Lyapunov function embedded (denoted by Ω_ρ) is then chosen as the closed-loop stability region for the nonlinear system of Eq. (2), i.e., $\Omega_\rho := \{x \in \mathbf{R}^n \mid V(x) \le \rho\}$, where $\Omega_\rho \subseteq O$.

2.4. Recurrent neural network

In this work, the recurrent neural network with the following form is developed to approximate the nonlinear system of Eq. (1) using time-series data Wu et al. (2022).

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

where $\mathbf{x}_t \in \mathbf{R}^{r_x}$ denotes the input of the RNN at the t^{th} time step, $\mathbf{y}_t \in \mathbf{R}^{r_y}$ denotes the output at the t^{th} time step, and $\mathbf{h}_t$ denotes the hidden state at t^{th} time step. W and Q denote the weight matrix for hidden layer and output layer, respectively. U denotes the weight matrix for hidden state. σ_h denotes the activation function for the hidden layer and σ_y denotes the activation function for the output layer.

3. Model order reduction via sensitivity analysis

Curse of dimensionality has remained a crucial challenge for neural network modeling of large-scale nonlinear systems due to increasing model complexity and computational efforts required to handle increasing data dimensions. In general, the nonlinear system of Eq. (1) consisting of m subsystems can be modeled either by a single RNN model that accounts for all the process state and input variables, or by multiple RNN models that approximate each subsystem of Eq. (1). However, to predict future states, both modeling approaches require the development of deep neural networks with a large number of input features to account for all the state variables since $f_i(\cdot)$, $g_{si}(\cdot)$, $f(\cdot)$ and $g(\cdot)$ in Eqs. 1–2 are the functions of the state variables of the entire system. Therefore, to address the issue of curse of dimensionality, feature selection is used in this section to develop reduced-order RNN models for the nonlinear system of Eq. (1).

3.1. Sensitivity analysis

In this section, sensitivity analysis is employed for feature selection and is integrated with the development of neural network models. Consider the RNN model described by Eq. (4) with a single hidden layer of J neurons that is used to approximate the subsystem of Eq. (1). Given a training dataset $(\bar{\mathbf{x}}_t, \bar{\mathbf{y}}_t)$ consisting of P training data sequences, we first normalize the RNN inputs and outputs such that they are in the same scale. Then the sensitivity analysis for RNN model is performed using the following equation:

$$S_{k,i}{}^{(p)} = \left.\frac{\partial \mathbf{y}_{t,k}}{\partial \mathbf{x}_{t,i}}\right|_{(p)} \tag{5}$$

where $\mathbf{y}_{t,k}$ denotes the k^{th} RNN output, where $k = 1, \dots, r_y$; $\mathbf{x}_{t,i}$ denotes the i^{th} attribute of input vector $\mathbf{x}_t$, where $i = 1, \dots, r_x$; $S_{k,i}$ denotes the sensitivity index of the k^{th} output with respect to the i^{th} input. (p) represents that the sensitivity index is calculated based on the p^{th} training data sequence, where $p = 1, 2, \dots, P$. The derivative of Eq. (5) can be further expressed as follows using Eq. (4):

$$\frac{\partial \mathbf{y}_{t,k}}{\partial \mathbf{x}_{t,i}} = \sigma_y{}' \sum_{j=1}^{J} Q_{j,k} \frac{\partial \mathbf{h}_{t,j}}{\partial \mathbf{x}_{t,i}} \tag{6}$$

where $\mathbf{h}_{t,j}$ denotes the hidden state of the j^{th} neuron of the hidden layer, and σ_y' denotes the value of derivative of the activation function. By further expanding the hidden state vector $\mathbf{h}$, it follows that:

$$\frac{\partial \mathbf{y}_{t,k}}{\partial \mathbf{x}_{t,i}} = \sigma_y{}' \sum_{j=1}^{J} Q_{j,k} \sigma_h{}' \left(\sum_{i=1}^{r_x} W_{i,j} + \sum_{j'}^{J} U_{j',j} \frac{\partial \mathbf{h}_{t-1,j}}{\partial \mathbf{x}_{t,i}}\right) \tag{7}$$

where $\sigma_h{}'$ denotes the value of derivative of the activation function. Using Eq. (5), the sensitivity index matrix for the RNN model of Eq. (4), where $\mathbf{x} \in \mathbf{R}^{r_x}$ and $\mathbf{y} \in \mathbf{R}^{r_y}$, can be presented as follows:

$$\mathbf{S}^{(p)} = \begin{bmatrix} S_{1,1}{}^{(p)} & S_{1,2}{}^{(p)} & \cdots & S_{1,r_x}{}^{(p)} \\ S_{2,1}{}^{(p)} & S_{2,2}{}^{(p)} & \cdots & S_{2,r_x}{}^{(p)} \\ \vdots & \vdots & \ddots & \vdots \\ S_{r_y,1}{}^{(p)} & S_{r_y,2}{}^{(p)} & \cdots & S_{r_y,r_x}{}^{(p)} \end{bmatrix} \tag{8}$$

The sensitivity matrix of Eq. (8) needs to be evaluated over the whole data set for feature selection. Thus an overall sensitivity matrix $\mathbf{S}$ will be determined after computing $\mathbf{S}^{(p)}$ for all P training data sequences. The most commonly used approaches for evaluating the overall sensitivity index over the entire dataset include mean-square average sensitivity, absolute value average sensitivity, and maximum sensitivity Kowalski and Kusy (2017). In this work, the absolute value average sensitivity is utilized to determine the overall sensitivity. Specifically, Eq. (9) is used to calculate the absolute value average sensitivity.

$$S_{k,i}{}^{abs} = \frac{\sum_{p=1}^{P} |S_{k,i}{}^{(p)}|}{P}, \ \forall S_{k,i}{}^{(p)} \in \mathbf{S}^{(p)} \tag{9}$$

The overall sensitivity matrix $\mathbf{S}$, consisting of all the $S_{k,i}$ calculated above, facilitates the identification of a subset of inputs that have a relatively greater impact on the outputs. For example, given the k^{th} RNN output, the inputs with relatively small sensitivity indices $S_{k,i}$ do not significantly contribute to the output $\mathbf{y}_{t,k}$, and therefore, can be dropped in the development of RNN models to reduce the dimension of input features. Additionally, the 'largest gap method' is utilized to determine which inputs can be pruned based on the sensitivity analysis results. Specifically, the significance of the i^{th} input $\mathbf{x}_{t,i}$ over the entire dataset is defined as:

$$\Psi_i = \max_{i=1,\dots,r_x} \{S_{k,i}\} \tag{10}$$

Subsequently, Ψ are sorted in descending order to demonstrate the importance of inputs on the output $\mathbf{y}_{t,k}$:

$$\Psi_{i,r} \geq \Psi_{i,r+1}, \ r = 1, \dots, r_x - 1 \tag{11}$$

where r denotes a sequence of sorted input numbers. The measure of gap in 'largest gap method' is defined as:

$$\mathcal{G}_r = \frac{\Psi_{i,r}}{\Psi_{i,r+1}}, \ r = 1, \dots, r_x - 1 \tag{12}$$

If we find a gap $\mathcal{G}$ that is sufficiently large (e.g., 1×10^1 for normalized input and output data), it implies that all the $\Psi_{i,r}$ with smaller values have little or negligible influence on the output $\mathbf{y}_{t,k}$. Therefore, all the inputs with index $r + 1, \dots, r_x$ can be pruned for reduced-order RNN modeling of high-dimensional nonlinear processes.

Remark 1. The specific number of selected features is determined based on the descending order sequence of importance and a pre-determined gap $\mathcal{G}$. If we find a gap $\mathcal{G}_r$, which is a measure of the relative importance of r^{th} and $(r+1)^{th}$ inputs as shown in Eq. (12), that is no less than $\mathcal{G}$, all the inputs with index $r + 1, \dots, r_x$ will be pruned for constructing the reduced-order RNN models of the high-dimensional nonlinear processes. Additionally, the gap $\mathcal{G}$ is determined based on the scale of data; for example, 1×10^1 can be considered a sufficiently large gap for a normalized dataset.

3.2. Active learning

As the dimensionality of system increases, the number of training samples required for constructing RNN models with a desired approximation performance increases exponentially. However, generating such a representative dataset with various combinations of features for a large-scale nonlinear process is computationally expensive. To develop an informative training dataset in a more computationally efficient way, active learning is integrated with feature selection in this subsection to iteratively choose informative data samples for training reduced-order RNN models.

The RNN model of Eq. (4) is developed to model the nonlinear subsystem of Eq. (1) with the following training set:

$$D_{train} = \{(\bar{\mathbf{x}}_{t,j}, \bar{\mathbf{y}}_{t,j}), j = 1, \dots, N_{train}\} \tag{13}$$

where $(\bar{\mathbf{x}}_{t,j}, \bar{\mathbf{y}}_{t,j})$ denotes a sample in the training set that consists of N_{train} training samples; $\bar{\mathbf{x}}_{t,j}$ and $\bar{\mathbf{y}}_{t,j}$ are the RNN inputs and outputs, respectively.

Since generating a representative dataset that includes a wide range of operating conditions for high-dimensional systems through simulation is computationally expensive, we initially develop a training set D_{train} for a small range of inputs and outputs around the steady state. Then, we build a full-order RNN that uses all the input features to predict outputs based on the training set D_{train}. Subsequently, based on D_{train}, active learning is applied to introduce additional training samples to iteratively improve the training performance of RNN models. Additionally, feature selection is applied prior to the active learning step to reduce the dimension of input feature to further reduce the amount of data required by the training process.

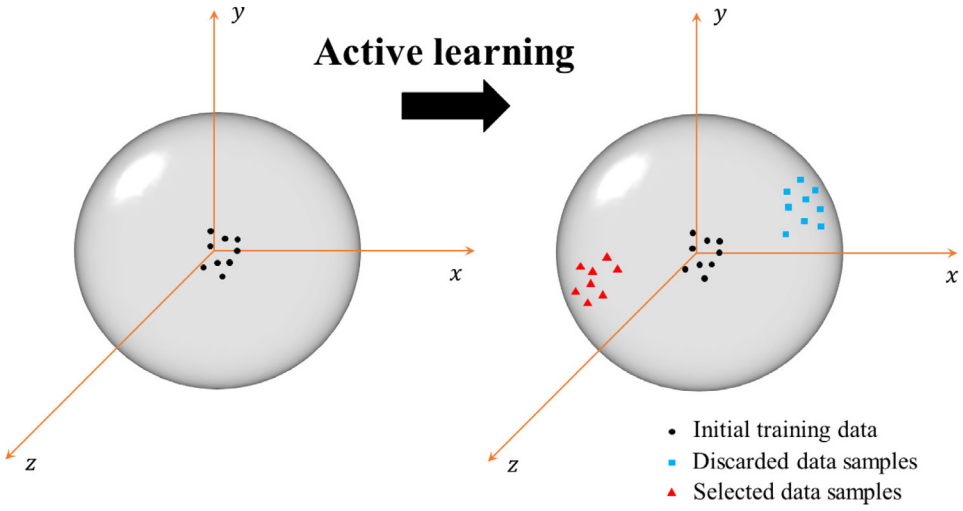

Fig. 1. A schematic for active learning, where black dots are initial data samples, red triangles are data samples with poor prediction performance, and blue rectangles are data samples with desired prediction performance. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

To implement active learning, we consider a set P (termed 'pool') that denotes all possible values $\bar{\mathbf{x}}_t$ can take:

$$P = \{\bar{\mathbf{x}}_{t,j}, j = 1, \ldots, N_{pool}\} \tag{14}$$

The set P for the nonlinear system of Eq. (1) can be developed to include all possible operating conditions under which the system stability can be achieved (e.g., the closed-loop stability region Ω_ρ for the nonlinear system of Eq. (2)). The objective of active learning is to choose as few data samples from P as possible for training while remaining a desired quality of the RNN model. The following function A is employed to select candidate data samples:

$$A(F_{nn},\ P,\ D_{train}) : P \rightarrow P_{candidate},\ P_{candidate} \subset P \tag{15}$$

where F_{nn} denotes the RNN model. Based on the initial full-order RNN model F_{nn} and the initial training set D_{train}, the function A yields a set of candidate data samples $P_{candidate}$, which is a subset of the 'pool' P with the samples of highest priority to be added to the training set D_{train}. Subsequently, the training dataset D_{train} is augmented using $P_{candidate}$, and the RNN model F_{nn} is re-trained using the updated training dataset. In this work, sensitivity analysis is used to guide the selection of $P_{candidate}$, and data augmentation is carried out for the selected important input features only to generate a larger dataset D_{train} by varying those important features under a wider range of operating conditions, while the remaining non-significant features still vary within the initial range of operating conditions.

In addition to the sensitivity-assisted active learning method, conventional active learning methods are also utilized to identify the data samples that will be added to training set D_{train} from the pool P after reducing the feature dimension. Specifically, the selection of training samples via active learning is carried out using ranking strategies such as variance reduction Cohn et al. (1994), query by committee (QBC) for regression Krogh and Vedelsby (1994), and maximizing expected model change Cai et al. (2013); Settles et al. (2007). Among many ranking methods, variance reduction ranking strategy is widely used for minimizing the generalization error of model in regression problems. Samples with a higher rank imply a worse prediction performance, and therefore, will be included in the updated training set to improve model performance Tsymbalov et al. (2018). Fig. 1 shows a schematic of data identification process via active learning. Specifically, the RNN model is initially trained using the initial dataset (represented by the black dots around the origin). Subsequently, the data samples with poor prediction performance (represented by red triangles) identified from active learning will be iteratively added into the training set D_{train} while those with desired prediction performance (represented by blue rectangles) will be abandoned.

Algorithm 1 shows the proposed algorithm for data generation and development of reduced-order RNN models, which is presented by the following steps: 1) a dense dataset D_{train} is initially generated for a small range of input and output features around the steady states, 2) D_{train} is used to train an initial full-order RNN model F_{nn} that accounts for all the states and inputs, 3) sensitivity analysis is applied using the full-order RNN model F_{nn} and D_{train} as discussed in Section 3.1 to identify important input features, 4) the dataset D_{train} is updated with data augumentation of $P_{candidate}$ for selected input features only, where selected important input features varied in a larger operating region while other non-significant features remain inside the initial small region, 5) a reduced-order RNN $\hat{F}_{nn}$ is trained using the updated training set D_{train}, where only selected important input features are used, 7) sensitivity analysis is applied again using the reduced-order RNN $\hat{F}_{nn}$ to re-identify important input features, 8) iteratively apply sensitivity analysis on up-

Algorithm 1 Data generation using active learning.

1: Generate a dense dataset for a small range of $(\bar{\mathbf{x}}_t, \bar{\mathbf{y}}_t)$ as D_{train}
2: Train a full-order RNN model F_{nn} based on D_{train}
3: Apply sensitivity analysis using the RNN model F_{nn} and training set D_{train} to identify important input features
4: Generate a candidate dataset $P_{candidate}$ based on the input features selected
5: **while** Selected features are different from the input features used in the RNN model **do**
6: Add samples from $P_{candidate}$ to training set D_{train}
7: Rebuild a reduced-order RNN model $\hat{F}_{nn}$ using selected variables based on the updated training set D_{train}
8: Apply sensitivity analysis using reduced-order RNN model $\hat{F}_{nn}$ and update training set D_{train} to re-identify important features
9: **end while**
10: **while** Predictive accuracy of the reduced-order RNN model $\hat{F}_{nn}$ is not sufficiently high **do**
11: Apply ranking strategy of variance reduction for samples in the 'pool' P
12: Update training set D_{train} based on the ranking results
13: Improve the reduced-order RNN model $\hat{F}_{nn}$ using updated training set D_{train}
14: **end while**

dated dataset until no further changes on the selection of input features are noticed for two consecutive iterations, and 9) conventional active learning algorithms such as variance reduction ranking strategy are applied to further enrich the training dataset.

Remark 2. As discussed in Section 3.1, sensitivity analysis identifies the important connections between input features and the outputs of interest. Using Eq. (15), the candidate dataset $P_{candidate}$ is generated with selected important input features varied in a larger operating region while other non-significant variables remain inside the initial small region in the data augmentation step. Additionally, once the important input features are identified, a reduced-order RNN is built based on the updated training set D_{train} for model dimension reduction such that the computational efficiency for training and data generation is significantly improved compared to the initial full-order RNN model. Therefore, by integrating sensitivity analysis with active learning, model reduction and data collection are carried out simultaneously to improve the model performance in terms of less training time and reduced model complexity while remaining a desired training accuracy.

Remark 3. Active learning improves the efficiency of neural network training procedures by adding the data points of high priority from the pool to the training dataset. Unlike the traditional approach of randomly selecting training samples that may lead to a large but highly redundant training set, active learning approaches reduce the amount of training data by selecting the most informative data to build a small but informative training set. While conventional active learning methods generally provide a guideline to characterize the range of data points in which the model performs poorly (e.g., data of high variance via variance reduction method), the proposed integration of sensitivity analysis and active learning reduces the amount of training data by reducing the dimension and range of training data simultaneously. Specifically, sensitivity analysis is utilized for model dimension reduction, and ranking strategy such as variance reduction method is utilized to select the most informative data in a well-characterized operating region.

Remark 4. In this section, sensitivity analysis is implemented based on a full-order RNN model since the initial dataset may be sparse and cannot provide an accurate calculation of sensitivity index matrix in Eq. (8). In case a dense and representative dataset is available, sensitivity analysis can be directly applied using the dataset to obtain the sensitivity index matrix. In this way, the sensitivity analysis in Algorithm 1 will be applied using data only, and reduced-order RNN models will be developed using the important input features selected from data.

3.3. Reduced-order RNN model

After identifying the important input features from sensitivity analysis, a reduced-order RNN is developed to model the i^{th} subsystem of Eq. (1). Unlike the conventional RNN modeling approach (termed full-order RNN model in this manuscript) that uses the state vector $x(t_k)$ of the entire system and manipulated input $u_i(t_k)$ (RNN inputs) to predict future states (RNN outputs) based on the training dataset $(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i)$, $\bar{\mathbf{x}}_i \in \mathbf{R}^{n_x+n_{u_i}}$ and $\bar{\mathbf{y}}_i \in \mathbf{R}^{n_x}$. We use sensitivity analysis to select the most important input features $\bar{\mathbf{x}}_{r,i} \in \mathbf{R}^l$ that play a dominant role in the prediction of the future states of the i^{th} subsystem $\bar{\mathbf{y}}_{r,i} \in \mathbf{R}^{n_{x_i}}$. Note that the selected input features may consist of the states of subsystems and the manipulated inputs. Therefore, we use $\bar{\mathbf{x}}_{r,i}$ as the inputs of the reduced-order RNN model while the output is the prediction of future states $\bar{\mathbf{y}}_{r,i}$, where $(\bar{\mathbf{x}}_{r,i}, \bar{\mathbf{y}}_{r,i}) \in D_{train}$ represents the training samples for reduced-order RNN. We can write the reduced-order RNN in the following form:

$$\mathbf{h}_{r,t} = \sigma_{h,r}(U_r \mathbf{h}_{r,t-1} + W_r \mathbf{x}_{r,t}), \quad \mathbf{y}_{r,t} = \sigma_{y,r}(Q_r \mathbf{h}_{r,t}) \tag{16}$$

where $\mathbf{x}_{r,t}$, $\mathbf{y}_{r,t}$ and $\mathbf{h}_{r,t}$ represent the inputs, outputs, and hidden states of the reduced-order RNN at the t^{th} time step, respectively. W_r, U_r and Q_r represent the weight matrices for input layer, hidden states, and output layer, respectively. $\sigma_{h,r}$ and $\sigma_{y,r}$ represent the activation functions for the hidden layer and output layer, respectively.

Remark 5. The dataset for training the reduced-order RNN model can be generated by carrying out open-loop simulations for the subsystem of Eq. (1) under various conditions $x \in \Omega_\rho$ and $u \in U$ for data generation, where u is implemented in a sample-and-hold fashion, i.e., $u(t) = u(t_k)$ holds $\forall t \in [t_k, t_k + \Delta)$. The RNN model is developed to predict future states $x(t)$, $\forall t \in [t_k, t_{k+p})$, where $t_{k+p} = t_k + p\Delta$, using $[x(t_k)\, u(t)]$ as inputs.

4. MPC using reduced-order RNN models

In this section, we develop a Lyapunov-based MPC (LMPC) using reduced-order RNN model to stabilize the nonlinear system of Eq. (1). To simplify the notation of RNN-based MPC, the reduced-order RNN model for the subsystem of Eq. (1) is written in the following continuous-time form:

$$\dot{x}_i = \hat{F}_{nn}(x_r, u_i) \tag{17}$$

where x_r denotes the state vector of important input features for the i^{th} subsystem according to sensitivity analysis. Note that in Eq. (17), the manipulated inputs u_i for the i^{th} subsystem are explicitly used as one of the inputs of the RNN model $\hat{F}_{nn}$ since u_i will be used as the optimization variables in MPC. However, in general, u_i will be identified by sensitivity analysis as one of the important input features that affect the states x_i of the i^{th} subsystem, and thus, will be included as one of the 'states' in x_r.

We assume there exists a stabilizing feedback feedback controller $u_i = \hat{\Phi}_{nn}(x_i) \in U$ that renders the origin of the reduced-order RNN model exponentially stable in a neighborhood around the origin. Similar to the stabilizability assumption in Section 2.3, a control Lyapunov function $\hat{V}$ can be found for the reduced-order RNN model that renders $\frac{\partial \hat{V}(x)}{\partial x}\hat{F}_{nn}$ negative within a stability region $\Omega_{\hat{\rho}}$ around the origin. Based on the stabilizing control law $u_i = \hat{\Phi}_{nn}(x_i) \in U$, the Lyapunov-based MPC design using neural network model is presented as follows Wu et al. (2019a,b):

$$\mathcal{J} = \min_{u_i \in S(\Delta)} \int_{t_k}^{t_{k+N}} L_{MPC}(\tilde{x}_i(t), u_i(t))dt \tag{18a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}_i(t) = \hat{F}_{nn}(\tilde{x}_r(t), u_i(t)) \tag{18b}$$

$$u_i(t) \in U, \ \forall\, t \in [t_k, t_{k+N}) \tag{18c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{18d}$$

$$\dot{\hat{V}}(x_i(t_k), u_i) \le \dot{\hat{V}}(x_i(t_k), \hat{\Phi}_{nn}(x_i(t_k))), \quad \text{if } x_i(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{18e}$$

$$\hat{V}(\tilde{x}_i(t), u_i) \le \rho_{nn}, \ \forall\, t \in [t_k, t_{k+N}), \ \text{if } x_i(t_k) \in \Omega_{\rho_{nn}} \tag{18f}$$

where L_{MPC} represents the objective function of MPC with the minimum value achieved at the origin. $\tilde{x}_i$ represents the future state predicted by the RNN model $\hat{F}_{nn}$. $S(\Delta)$ denotes the class of piece-wise constant functions of one sampling period Δ. The control actions u_i are optimized over the entire prediction horizon $[t_k, t_{k+N})$. The objective of RNN-MPC is to stabilize the subsystem by ensuring x_i is bounded in the closed-loop stability region $\Omega_{\hat{\rho}}$ over the entire operation period and ultimately converges to a small terminal set $\Omega_{\rho_{nn}}$ around the origin. Specifically, Eq. (18b) is the reduced-order RNN model. Eq. (18c) is the constraint for manipulated inputs over the entire prediction horizon. Eq. (18d) is the state measurements x of the entire system at each sampling time. Eqs. 18e–18f are the Lyapunov-based constraints that ensure closed-loop stability for the i^{th} subsystem of Eq. (1) under reduced-order RNN-MPC.

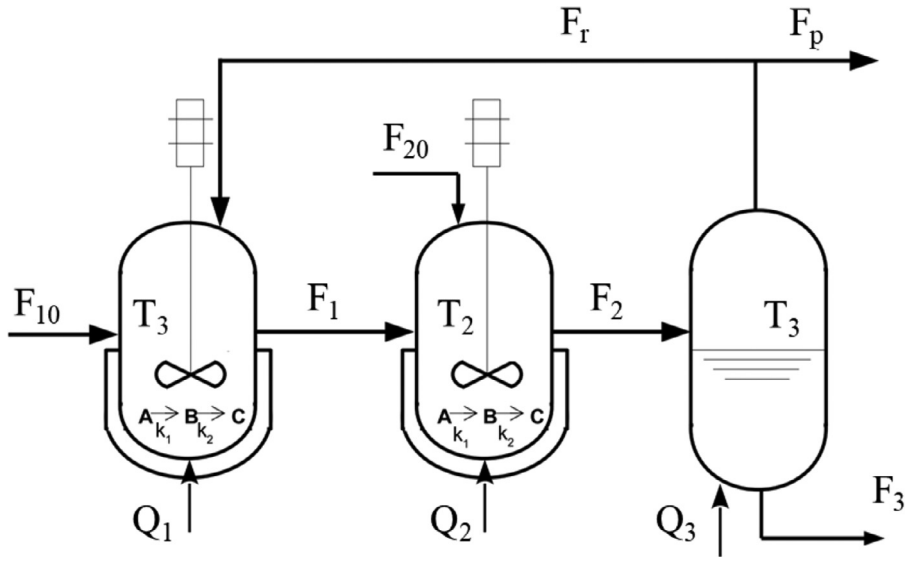

Fig. 2. Diagram for the reactor-reactor-separator process.

5. Application to a reactor-reactor-separator process

In this section, a reactor-reactor-separator process is used to demonstrate the implementation of the proposed ML modeling approach using sensitivity analysis and active learning. Open-loop and closed-loop simulations of the system are carried out under the MPC using reduced-order and full-order RNN models, respectively, to illustrate the efficacy of the proposed method.

5.1. Process description

Consider a chemical process consisting of two CSTRs and a flash tank separator in sequence, as shown in Fig. 2 Chen et al. (2020). In both reactors, chain reactions take place (A → B → C), and the overhead vapor from the flash tank is recycled to the first CSTR. Mass and energy balances are used to develop the process model, which includes 9 nonlinear differential equations as shown in Eq. (19). T_1, T_2, T_3 denote the temperatures of the three vessels, and x_{A1}, x_{A2}, x_{A3}, x_{B1}, x_{B2}, x_{B3} denote the mass fractions of species A and B, respectively in the three vessels. The state vector of the system $x(t)$ consists of the temperatures and mass fractions of the three vessels, i.e., $x(t) = [x_{A1}\ x_{B1}\ T_1\ x_{A2}\ x_{B2}\ T_2\ x_{A3}\ x_{B3}\ T_3]$. The manipulated input vector $u(t)$ consists of the heat inputs to the three vessels Q_1, Q_2, Q_3 and the feed stream flow rate to the second CSTR F_{20} (i.e., $u(t) = [Q_1\ Q_2\ Q_3\ F_{20}]$). The reaction in the separator tank is assumed to be negligible and the relative volatility of each species is assumed to be constant. The compositions of the recycle stream are shown in Eq. (20).

$$\frac{dx_{A1}}{dt} = \frac{F_{10}}{V_1}(x_{A10} - x_{A1}) + \frac{F_r}{V_1}(x_{Ar} - x_{A1}) - k_1 e^{\frac{-E_1}{RT_1}} x_{A1} \tag{19a}$$

$$\frac{dx_{B1}}{dt} = \frac{F_{10}}{V_1}(x_{B10} - x_{B1}) + \frac{F_r}{V_1}(x_{Br} - x_{B1}) \tag{19b}$$

$$+ k_1 e^{\frac{-E_1}{RT_1}} x_{A1} - k_2 e^{\frac{-E_2}{RT_1}} x_{B1} \tag{19c}$$

$$\frac{dT_1}{dt} = \frac{F_{10}}{V_1}(T_{10} - T_1) + \frac{(-\Delta H_1)}{\rho C_p} C_M k_1 e^{\frac{-E_1}{RT_1}} x_{A1} \tag{19d}$$

$$+ \frac{(-\Delta H_2)}{\rho C_p} C_M k_2 e^{\frac{-E_2}{RT_1}} x_{B1} + \frac{Q_1}{\rho C_p V_1} + \frac{F_r}{V_1}(T_3 - T_1) \tag{19e}$$

$$\frac{dx_{A2}}{dt} = \frac{F_1}{V_2}(x_{A1} - x_{A2}) + \frac{F_{20}}{V_2}(x_{A20} - x_{A2}) - k_1 e^{\frac{-E_1}{RT_2}} x_{A2} \tag{19f}$$

$$\frac{dx_{B2}}{dt} = \frac{F_1}{V_2}(x_{B1} - x_{B2}) + \frac{F_{20}}{V_2}(x_{B20} - x_{B2}) \tag{19g}$$

$$+ k_1 e^{\frac{-E_1}{RT_2}} x_{A2} - k_2 e^{\frac{-E_2}{RT_2}} x_{B2} \tag{19h}$$

$$\frac{dT_2}{dt} = \frac{F_{20}}{V_2}(T_{20} - T_2) + \frac{(-\Delta H_1)}{\rho C_p} C_M k_1 e^{\frac{-E_1}{RT_2}} x_{A2} \tag{19i}$$

$$+ \frac{(-\Delta H_2)}{\rho C_p} C_M k_2 e^{\frac{-E_2}{RT_2}} x_{B2} + \frac{Q_2}{\rho C_p V_2} + \frac{F_1}{V_2}(T_1 - T_2) \tag{19j}$$

$$\frac{dx_{A3}}{dt} = \frac{F_2}{V_3}(x_{A2} - x_{A3}) - \frac{F_r + F_p}{V_3}(x_{Ar} - x_{A3}) \tag{19k}$$

$$\frac{dx_{B3}}{dt} = \frac{F_2}{V_3}(x_{B2} - x_{B3}) - \frac{F_r + F_p}{V_3}(x_{Br} - x_{B3}) \tag{19l}$$

$$\frac{dT_3}{dt} = \frac{F_2}{V_3}(T_2 - T_3) + \frac{Q_3}{\rho C_p V_3} + \frac{(F_r + F_p)C_M}{\rho C_p V_3} \times (x_{Ar}\Delta H_{vapA} \tag{19m}$$

$$+ x_{Br}\Delta H_{vapB} + x_{Cr}\Delta H_{vapC}) \tag{19n}$$

$$x_{Ar} = \frac{\alpha_A x_{A3}}{\alpha_A x_{A3} + \alpha_B x_{B3} + \alpha_C x_{C3}} \tag{20a}$$

$$x_{Br} = \frac{\alpha_B x_{B3}}{\alpha_A x_{A3} + \alpha_B x_{B3} + \alpha_C x_{C3}} \tag{20b}$$

$$x_{Ar} = \frac{\alpha_C x_{C3}}{\alpha_A x_{A3} + \alpha_B x_{B3} + \alpha_C x_{C3}} \tag{20c}$$

where α denotes the constant relative volatility of each species. The values of other process parameters included in Eq. (19) and Eq. (20), the steady-state values, and the corresponding steady-state input values can be found in Chen et al. (2020), and are omitted here. In this study, an MPC is developed to stabilize the temperature of flash tank separator T_3 at its steady-state and optimize the closed-loop performance of

the flash tank separator, and two proportionalintegral (PI) controllers are utilized to control the two reactors. Reduced-order RNN model is thus developed for the third subsystem (i.e., flash tank separator) using the sensitivity analysis to reduce the dimension of the RNN input vector.

5.2. Data generation

Open-loop simulations are carried out under various operating conditions to generate the dataset for the modeling of the reactor-reactor-separator process. The first-principles model of Eq. (19) and Eq. (20) is simulated using explicit Euler method under different initial conditions and input values within the following ranges: $x_{Ai} \in [0, 1]$, $x_{Bi} \in [0, 1]$, $T_i \in [-50\ K, +50\ K]$ $(i = 1, 2, 3)$, $Q_i \in [-5 \times 10^9\ kJ/hr,\ 5 \times 10^9\ kJ/hr]$ $(i = 1, 2, 3)$, $F_{20} \in [1\ m^3/hr,\ 9\ m^3/hr]$, where T_i and Q_i are in deviation variable form. The inputs are implemented in a sample-and-hold fashion in the sense that for $t \in [t_k, t_k + \Delta]$ (Δ is the sampling period), the input vectors $u(t)$ are piecewise constant functions. Each simulation runs for one sampling period $\Delta = 0.01\ hr$, which includes 100 integration time steps, with each integration time step $h_c = 1.0 \times 10^{-4}\ hr$, generating a time-series state profile $[x_0, x_1, \ldots, x_{100}]$ (x_0 is the initial condition). Due to the high computational cost for data generation that involves 9 states and 4 manipulated inputs, we initially generate a small dataset for the initial conditions around the origin, and enrich the dataset via active learning. To further reduce the computation time for reduced-order RNN model development, a state profile is extracted every 20 integration time steps to develop a sparse dataset $[x_0, x_{20}, x_{40}, \ldots, x_{100}]$ for model construction.

5.3. Feature selection and reduced-order RNN models

Sensitivity analysis is employed to select important state and input features for predicting the temperature of flash tank separator T_3. First, the dataset generated above is normalized such that all the states and manipulated inputs are in the same scale. A full-order RNN model that accounts for all the states and inputs is developed to model the system of Eq. (19) and Eq. (20), where the initial state vector x_0 and the manipulated input vector $u(t)$ are used as the RNN inputs to predict the future states $[x_{20}, x_{40}, \ldots, x_{100}]$ within one sampling period. The full-order RNN model includes 1 hidden layer of 30 neurons and 1 dense layer of 9 neurons as the output layer. Linear activation function is utilized for the dense layer. The training input-output data is split into training (70%), validation (20%), and testing (10%) datasets. The development of the full-order RNN uses mean squared error as the loss function and achieves a validation loss of less than 3.5×10^{-5}. The full-order RNN model will be used to calculate the sensitivity index in sensitivity analysis, and will also be used as a benchmark for the comparison with the reduced-order RNN model in Section 5.4.

Subsequently, we apply sensitivity analysis to identify the important state and input features using the full-order RNN model. The sensitivity index is defined as follows:

$$s_i = \frac{\sum_{n=1}^{N} \left| \frac{\partial T_{3,n}}{\partial \mathbf{x}_i} \right|}{N}, \mathbf{x} = [x_0\ u] \tag{21}$$

where N denotes the number of time steps in the predicted sequence, which is 5 in this case. $\mathbf{x}$ denotes the input vector of the RNN model, which consists of the initial condition x_0 and manipulated inputs u. $\frac{\partial T_3}{\partial \mathbf{x}_i}$ is the derivative of T_3 with respect to the i^{th} input. The derivatives are computed numerically by introducing a small perturbation on states and inputs, i.e., $\frac{T_3(\mathbf{x}_i + \delta) - T_3(\mathbf{x}_i)}{\delta}$ with a sufficient small step size δ.

Figure 3 shows the sensitivity indices for each input variable of the RNN model. It is demonstrated that $Q_2, Q_3, F_{20}, T_1, T_2, T_3$ have significantly higher sensitivity indices than other variables, indicating their significant contributions to T_3.

Based on the sensitivity analysis results, $Q_2, Q_3, F_{20}, T_1, T_2, T_3$ are selected as the inputs for the reduced-order RNN model. Active learning is applied to develop a representative dataset D_{train} to improve the generalization performance of the reduced-order RNN model that predicts the 5 subsequent temperature values of the flash tank separator $[T_{3,20}, T_{3,40}, \ldots, T_{3,100}]$ within the following sampling period. Specifically, we carry out active learning for 50 iterations, and within each iteration, 1000 samples are evaluated using the current RNN model. Finally, the prediction errors of 1000 samples are organized in descending order, from which the top 500 samples are added to the training dataset D_{train} to update the reduced-order RNN model. Fig. 4 demonstrates that the prediction error for the testing set calculated by mean squared error decreases as the active learning process evolves and the error ultimately approaches zero, which implies the final RNN model achieves desired generalization performance. However, it is demonstrated that the prediction error of the initial RNN model on the testing set remains at a high level at all times. Therefore, active learning significantly improves

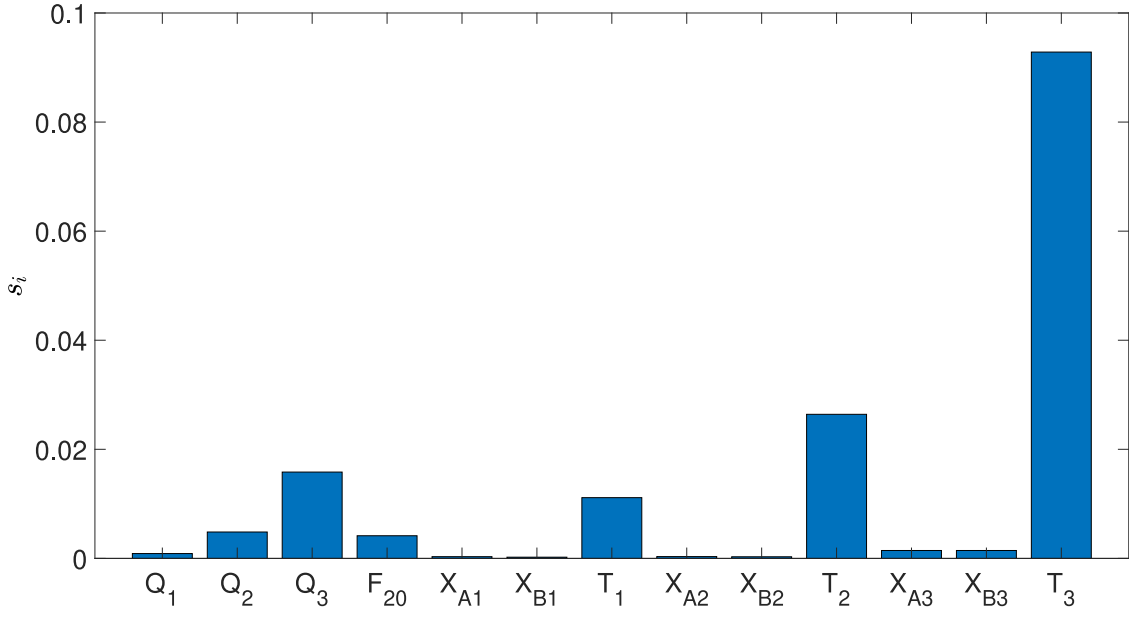

Fig. 3. Sensitivity indices for reactor-reactor-separator process states.

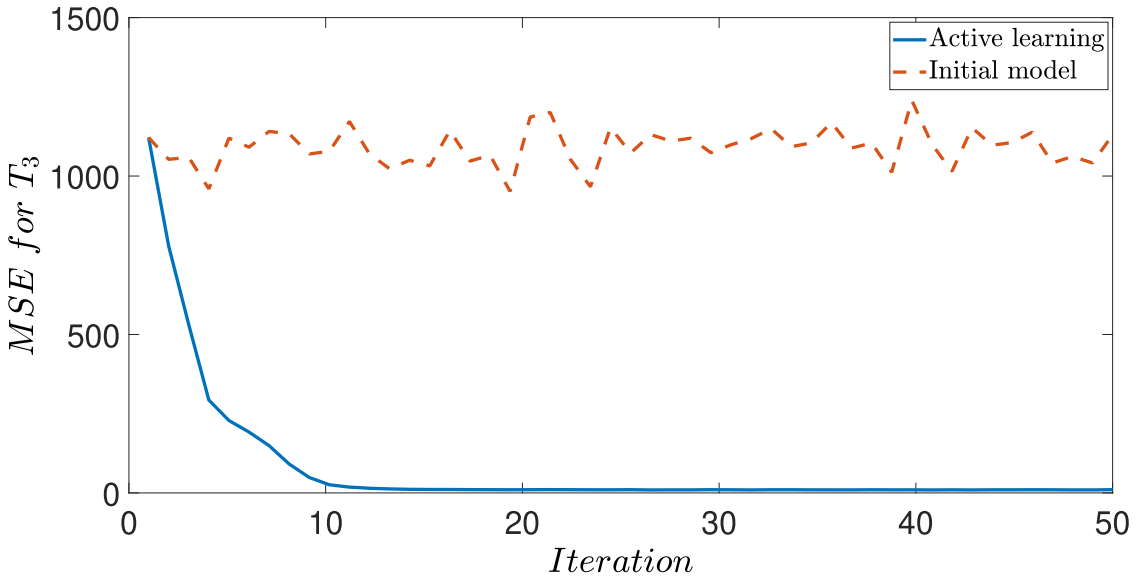

Fig. 4. Mean squared error during active learning process for reduced-order RNN model.

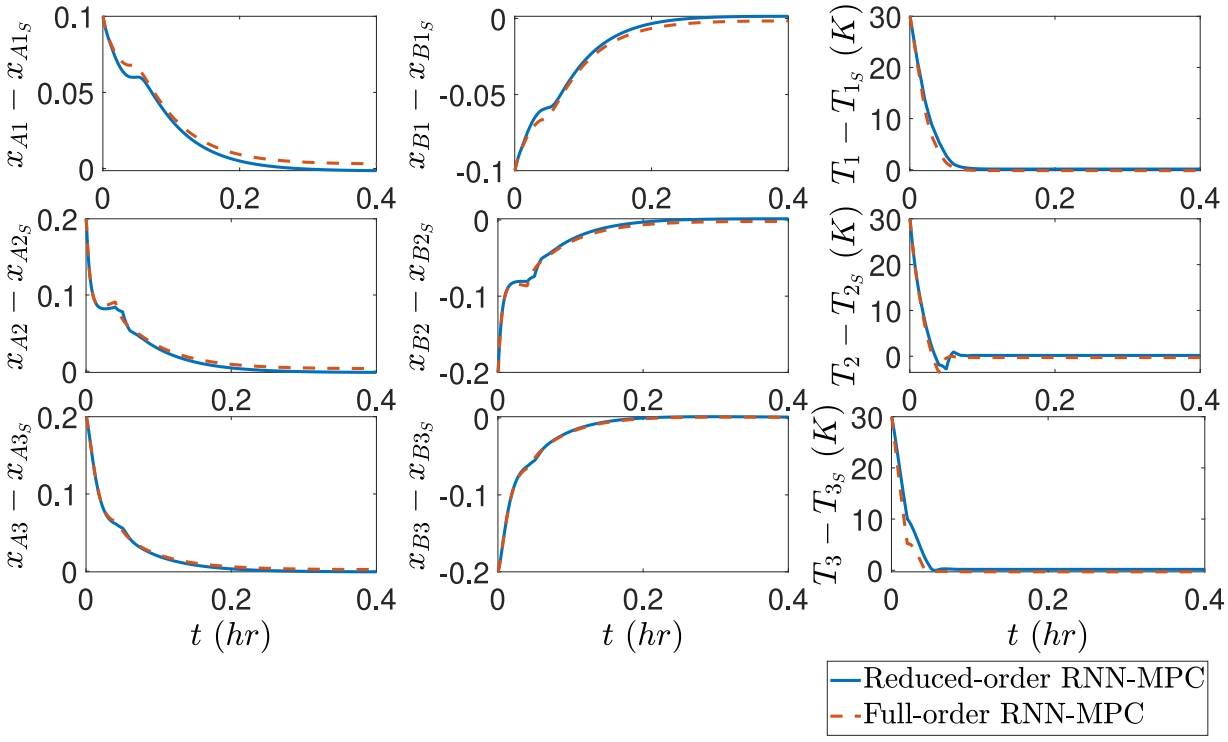

Fig. 5. Closed-loop state profiles under the MPC using full-order RNN and reduced-order RNN model.

the generalization ability of the reduced-order RNN model by selecting the most informative training samples into training dataset.

The final reduced-order RNN consists of 1 hidden layer of 5 neurons and 1 dense layer, and achieves a validation loss (measured by mean squared error) of around 3.5×10^{-5}. The full-order RNN model is also updated using the dataset D_{train} generated through active learning, and achieves a validation loss of around 3.5×10^{-5}. To calculate the computation time for both RNN models, we carry out open-loop simulations for one sampling period $\Delta = 0.01\ hr = 36\ s$. It takes around $0.0083\ s$ and $0.0004\ s$ to solve the full-order RNN model and the reduced-order RNN model, respectively, which demonstrates that reduced-order RNN is computationally more efficient (i.e., around 20 times faster) than full-order RNN in open-loop predictions due to its simpler structure.

5.4. Closed-loop simulation results

We carry out closed-loop simulations for the nonlinear system of Eq. (19) and Eq. (20) under the MPC using reduced-order RNN model and full-order RNN model, respectively, where closed-loop performance of the full-order RNN model constructed in Section 5.3 is used as a benchmark for comparison purpose. As discussed in Section 5.1, the objective of MPC is to stabilize the temperature of flash tank separator T_3 at its steady-state. In this study, the two reactors are controlled by PI controllers and the flash tank separator is controlled by MPC. The nonlinear optimization problem of RNN-MPC is solved using python with IPOPT software package named PyIpopt, where the integration time step is $h_c = 2 \times 10^{-4}\ hr$, and the sampling period is

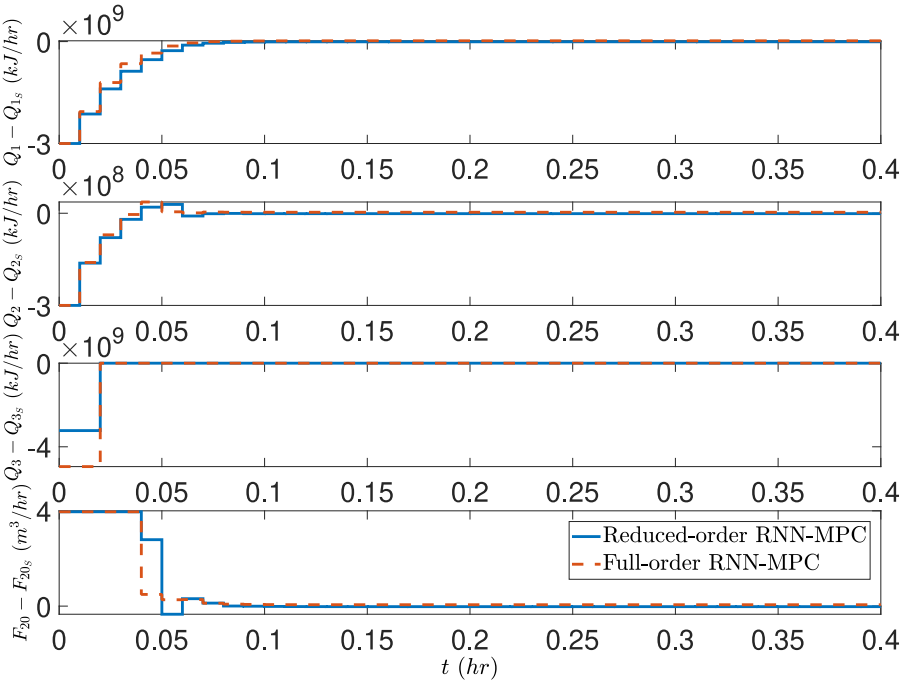

Fig. 6. Control action profiles under MPC using full-order RNN model and reduced-order RNN model.

$\Delta = 0.01$ hr Wächter and Biegler (2006). In the closed-loop simulations under MPC and PI controllers, Q_3 and F_{20} are utilized as the manipulated inputs in MPC, and Q_1 and Q_2 are manipulated by the two PI controllers, respectively.

Figure 5 shows the evolution of the closed-loop states under MPC using full-order RNN and reduced-order RNN model, respectively. It is demonstrated that the closed-loop temperatures of flash separation tank T_3 and the 2 CSTRs quickly approach zero and ultimately remain inside a neighbourhood around the origin under both MPCs. The trajectories of closed-loop mass fraction in the flash separation tank and the 2 CSTRs using both models highly overlap with each other. Fig. 6 shows the control action profiles for MPC using different models, from which it is demonstrated that both MPCs utilize aggressive control actions for Q_3 and F_{20} at the initial stage of operation to drive the states to the origin as quickly as possible. After the states enter a small neighborhood around the origin, the control actions remain at zero to avoid oscillation. Therefore, through the comparison with full-order RNN model, it is demonstrated that reduced-order modeling approach via sensitivity analysis and active learning improves computational efficiency of machine learning modeling process, while achieving desired prediction accuracy and closed-loop performance under MPC simultaneously.

6. Conclusions

In this work, we utilized sensitivity analysis and active learning to improve machine learning modeling of nonlinear processes. We first applied sensitivity analysis to identify the important input features that have a great impact on process outputs. Then, we developed a reduced-order RNN model using selected input features and implemented active learning to select the training data that could further improve model generalization performance. Based on the reduced-order RNN model, the RNN-based MPC was developed to stabilize the nonlinear system. The proposed method was applied to a chemical process example to demonstrate the improved computational efficiency and desired closed-loop performance under RNN-MPC.

Declaration of Competing Interest

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

Acknowledgments

Financial support from NUS Start-up Grant R-279-000-656-731 and ASTAR IAF-PP-A19C1a0018 is gratefully acknowledged.

References

Burbidge, R., Rowland, J.J., King, R.D., 2007. Active learning for regression based on query by committee. In: Proceedings of International Conference on Intelligent Data Engineering and Automated Learning. Springer, pp. 209–218.
Cai, W., Zhang, Y., Zhou, J., 2013. Maximizing expected model change for active learning in regression. In: Proceedings of 2013 IEEE 13th International Conference on Data Mining. IEEE, pp. 51–60.
Chen, S., Wu, Z., Christofides, P.D., 2020. A cyber-secure control-detector architecture for nonlinear processes. AIChE J. 66 (5), e16907.
Christofides, P.D., Scattolini, R., de la Pena, D.M., Liu, J., 2013. Distributed model predictive control: a tutorial review and future research directions. Comp. & Chem. Eng. 51, 21–41.
Cohn, D., Ghahramani, Z., Jordan, M., 1994. Active learning with statistical models. In: Proceedings of NIPS, Vol. 7, pp. 705–712.
Cord, M., Cunningham, P., 2008. Machine learning techniques for multimedia: case studies on organization and retrieval. Springer Science & Business Media.
Dhal, P., Azad, C., 2022. A comprehensive survey on feature selection in the various fields of machine learning. Applied Intelligence 52, 4543–4581.
El Aboudi, N., Benhlima, L., 2016. Review on wrapper feature selection approaches. In: Proceedings of 2016 International Conference on Engineering & MIS (ICEMIS). IEEE, pp. 1–5.
Hsu, H., Hsieh, C., Lu, M., 2011. Hybrid feature selection by combining filters and wrappers. Expert. Syst. Appl. 38 (7), 8144–8150.
JouanRimbaud, D., Massart, D., Leardi, R., De Noord, O.E., 1995. Genetic algorithms as a tool for wavelength selection in multivariate calibration. Anal. Chem. 67 (23), 4295–4301.
Kamalov, F., 2018. Sensitivity analysis for feature selection. In: Proceedings of 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, pp. 1466–1470.
Khalid, S., Khalil, T., Nasreen, S., 2014. A survey of feature selection and feature extraction techniques in machine learning. In: Proceedings of 2014 science and information conference. IEEE, pp. 372–378.

Kohavi, R., John, G.H., 1997. Wrappers for feature subset selection. Artif. Intell. 97 (1–2), 273–324.
Kowalski, P.A., Kusy, M., 2017. Sensitivity analysis for probabilistic neural network structure reduction. IEEE Trans. Neural. Netw. Learn. Syst. 29 (5), 1919–1932.
Krogh, A., Vedelsby, J., 1994. Neural network ensembles, cross validation, and active learning. In: Proceedings of NIPS, Vol. 7, pp. 231–238.
Ladha, L., Deepa, T., 2011. Feature selection methods and algorithms. Int. J. on Comp. Sci. and Eng. 3 (5), 1787–1797.
Maimon, O., Rokach, L., 2005. Decomposition methodology for knowledge discovery and data mining. Data mining and knowl. discov. handbook 981–1003.
Naik, D.L., et al., 2021. A novel sensitivity-based method for feature selection. J. Big Data 8 (1), 1–16.
Nikishova, A., Comi, G.E., Hoekstra, A.G., 2020. Sensitivity analysis based dimension reduction of multiscale models. Math. Comput. Simul. 170, 205–220.
Qiu, J., Wu, Q., Ding, G., Xu, Y., Feng, S., 2016. A survey of machine learning for big data processing. EURASIP J. Adv. Signal Process 2016 (1), 1–16.
Reunanen, J., 2003. Overfitting in making comparisons between variable selection methods. J. Machine Learn. Resea. 3 (Mar), 1371–1382.
Rong, M., Gong, D., Gao, X., 2019. Feature selection and its use in big data: challenges, methods, and trends. IEEE Access 7, 19709–19725.
Sarma, S.D., Deng, D.-L., Duan, L.-M., 2019. Machine learning meets quantum physics. arXiv preprint arXiv:1903.03516.
Settles, B., Craven, M., Ray, S., 2007. Multiple-instance active learning. In: Proceedings of NIPS, Vol. 20, pp. 1289–1296.
Shlens, J., 2014. A tutorial on principal component analysis. arXiv preprint arXiv:1404.1100.
Tsymbalov, E., Panov, M., Shapeev, A., 2018. Dropout-based active learning for regression. In: Proceedings of International Conference on Analysis of Images, Social Networks and Texts. Springer, pp. 247–258.
Wächter, A., Biegler, L.T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. 106, 25–57.
Wu, Z., Alnajdi, A., Gu, Q., Christofides, P.D., 2022. Statistical machine-learning-based predictive control of uncertain nonlinear processes. AlChE J. 68, e17642.
Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019. Machine learning-based predictive control of nonlinear processes. Part I: Theory. AlChE J. 65, e16729.
Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019. Machine learning-based predictive control of nonlinear processes. part II: computational implementation. AlChE J. 65, e16734.
Yang, H., Li, J., Lim, K.Z., Pan, C., Van Truong, T., Wang, Q., Li, K., Li, S., Xiao, X., Ding, M., et al., 2022. Automatic strain sensor design via active learning and data augmentation for soft machines. Nat. Machine Intell. 4, 84–94.
Zhao, T., Zheng, Y., Gong, J., Wu, Z., 2022. Machine learning-based reduced-order modeling and predictive control of nonlinear processes. Chem. Eng. Res. Des. 179, 435–451.
Zheng, Y., Wang, X., Wu, Z., 2022. Machine learning modeling and predictive control of batch crystallization process. Ind. & Engin. Chem. Resea. in press.
Zurada, J.M., Malinowski, A., Cloete, I., 1994. Sensitivity analysis for minimization of input data dimension for feedforward neural network. In: Proceedings of IEEE International Symposium on Circuits and Systems-ISCAS'94, Vol. 6. IEEE, pp. 447–450.