Contents lists available at ScienceDirect

Chemical Engineering Research and Design

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

Process structure-based recurrent neural network modeling for predictive control: A comparative study

Mohammed S. Alhajeri[a,e], *Junwei Luo*[a], *Zhe Wu*[c], *Fahad Albalawi*[d], *Panagiotis D. Christofides*[a,b,*]

[a] *Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095-1592, USA*
[b] *Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095-1592, USA*
[c] *Department of Chemical and Biomolecular Engineering, National University of Singapore, 117585 Singapore, Singapore*
[d] *Department of Electrical Engineering, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*
[e] *Department of Chemical Engineering, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait*

ARTICLE INFO

Article history:
Received 9 December 2021
Received in revised form 27 December 2021
Accepted 29 December 2021
Available online 5 January 2022

Keywords:
Process control
Model predictive control
Nonlinear processes
Machine learning
Recurrent neural networks
Aspen Plus Dynamics

ABSTRACT

Recurrent neural networks (RNN) have demonstrated their ability in providing a remarkably accurate modeling approximation to describe the dynamic evolution of complex, nonlinear chemical processes in several applications. Although conventional fully-connected RNN models have been successfully utilized in model predictive control (MPC) to regulate chemical processes with desired approximation accuracy, the development of RNN models in terms of model structure can be further improved by incorporating physical knowledge to achieve better accuracy and computational efficiency. This work investigates the performance of MPC based on two different RNN structures. Specifically, a fully-connected RNN model, and a partially-connected RNN model developed using a prior physical knowledge, are considered. This study uses an example of a large-scale complex chemical process simulated by Aspen Plus Dynamics to demonstrate improvements in the RNN model and an RNN-based MPC performance, when the prior knowledge of the process is taken into account.

1. Introduction

Mathematical models that describe the relationship between the manipulated inputs and the process outputs are essential for constructing model-based control systems for industrial applications. Currently, the development of a process model is based on either first-principles or process data under various assumptions depending on the process of interest. However, certain limitations on model performance exist. For example, linearized models of nonlinear processes are only valid around a vicinity of the operating point that is used in the linearization. Furthermore, due to the dynamical nature alongside with the inherent nonlinear behavior, and high complexity of most of chemical processes, it is usually not an easy task to find an accurate first-principles model.

To address this problem, the investigation of utilizing artificial intelligence (AI) techniques in chemical engineering has been carried out continuously. The AI technology has provided classic and powerful modeling tools such as fuzzy logic in the 1960s (Zadeh, 1968), expert systems in the 1980s (Liao, 2005; Lee, 1990), and machine learning (ML) in the 1990s (Vepa, 1993). Moreover, the implementation of ML techniques in the modeling of complex systems comes with a successful history in different chemical processes applications (Banerjee et al.,

* *Corresponding author at*: Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095-1592, USA.
E-mail address: pdc@seas.ucla.edu (P.D. Christofides).
https://doi.org/10.1016/j.cherd.2021.12.046

2017; Singh et al., 2017; Wong et al., 2018; Dias et al., 2017). For example in Banerjee et al. (2017), an artificial neural network (ANN) model is developed for a bio-diesel production process. The ANN model provided an approximation of the percentage of fatty acid methyl ester yield within ±8% deviation from the experimental data. Additionally, among various ML modeling techniques, recurrent neural networks (RNN) have been broadly employed for modelling a general class of dynamical systems for control and state estimation purposes (Pan and Wang, 2011). In Singh et al. (2017), a RNN model of a continuous binary distillation column (BDC) was trained and validated using experimental data, and the study demonstrated that the RNN model prediction can outperform a first-principles model for large-scale, complex, nonlinear process, due to its high degree of freedom to solve the complex non-linear regression problem using the process dataset.

With the continuous improvement of data availability and accessibility, machine learning (ML) based model predictive control (MPC) methods receive increasing attention as the next generation of control systems. Conceptually, an MPC contains three major components: a predictive model, an objective function and constraints, and a process optimizer (Camacho and Bordons, 2013). By using the neural network as the predictive model, the MPC can capture the process dynamics and accordingly make smart decisions: approaching the target states efficiently, automatically, and economically. Furthermore, recent works have demonstrated that ML-based MPC can be utilized to deal with various challenging tasks to improve manufacturing processes such as suppressing measurement noise and searching optimum economic benefits, for which classical control techniques are incapable to accomplish (Ellis et al., 2017; Wu et al., 2021).

Fully-connected RNN model (i.e., densely relate all the inputs to all the outputs) is the typical candidate to analyze time-series data in a black-box manner. However, such an approach is not always optimal, especially for complex chemical processes. For instance, in an integrated chemical plant the upstream units affect the downstream units but not the other way around. Therefore, to further improve the RNN model accuracy, various works (De Azevedo et al., 1997; Kahrs and Marquardt, 2007; Stephanopoulos and Han, 1996) have investigated the gray-box modeling, also known as hybrid modeling, by introducing a prior physical knowledge into the development of neural network models of chemical processes. Recently, Patel et al. (2020) proposed a method for integrating data-driven modeling with first-principles knowledge, that specifically allows for including information on known gains between certain inputs and outputs. This suggested method can be utilized in large processes with prior knowledge of interconnections. Also, the work of Lu et al. (2017), formulated a partially-connected RNN model, where the outputs were connected to the impacting inputs only. The resulting RNN model was demonstrated to outperform a fully-connected model. Additionally, Wu et al. (2020) used a partially-connected RNN in the framework of ML-based MPC. It was demonstrated that the partially-connected RNN model was able to improve the MPC performance compared to a fully-connected RNN model.

Taking the aforementioned considerations into account, the present work evaluates the performance of a partially-connected RNN-based MPC using a high-fidelity process simulator of a nonlinear chemical process. First, we construct a simulation model for a chemical plant used to produce Ethylbenzene with two continuous stirred tank reactors (CSTR) in series via the Aspen Plus and Aspen Plus Dynamics simulators. Subsequently, we train a fully-connected and a partially-connected RNN model, respectively, to capture the dynamics of the process using the same datasets obtained from extensive open-loop simulations. Finally, we compare each model's open-loop and closed-loop performance to demonstrate the advantages of using a partially-connected neural network in MPC.

The rest of this manuscript is organized as follow: In Section 2, the class of process systems, mathematics notations, and assumption of stabilizing control law are discussed. In Sections 3 and 4, concepts and methods used to construct fully-connected and partially-connected RNN models, respectively, are presented. Furthermore, a Lyapunov-based model predictive controller (LMPC) integrated with a RNN model is developed and discussed in Section 5. In Section 6, the open-loop and closed-loop performances of MPCs using different RNN model structures are evaluated using the chemical process application.

2. Preliminaries

2.1. Notations

Through this manuscript the notation $|x|$ represents the Euclidean norm of a vector x. The notation $L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$ denotes the standard Lie derivative. For set subtraction " $-$" is used, i.e., $A - B = \{x \in \mathbf{R}^n | x \in A, x \notin B\}$. A function $f(x)$ is of class C^1 if it is continuously differentiable.

2.2. Class of systems

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

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

where the state vector of the system is $x = [x_1, \ldots, x_n]^T \in \mathbf{R}^n$, $y = [y_1, \ldots, y_q]^T \in \mathbf{R}^q$ is the output vector, and the manipulated input vector is $u = [u_1, \ldots, u_m]^T \in \mathbf{R}^m$. $F(x, u)$ represents a nonlinear vector function of x and u which is assumed to be sufficiently smooth functions of its arguments. The constraints on control inputs are given by $u \in U := \{u_i^{min} \le u_i \le u_i^{max}\}$. The functions $f(\cdot)$, and $g(\cdot)$ are nonlinear vector and matrix functions of $n \times 1$ and $n \times m$ dimensions, respectively.

2.3. Stabilizability assumption

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

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

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

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

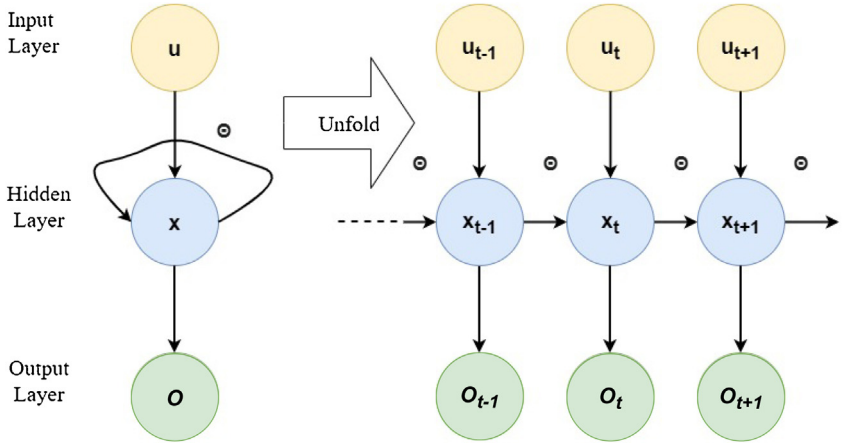

Fig. 1 – A schematic of a recurrent neural network.

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

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

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

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

3. Recurrent neural networks (RNN) models

As mentioned in the introduction, RNN models are suitable for modeling time-series data. The recursive action in the hidden layer neurons allows the RNN to hold the memory of the previous states, such that it can adequately approximate a time-series dataset. In this work, the RNN model used to approximate the nonlinear system of Eq. (1) using the process operational data can be represented as:

$$\dot{\bar{x}} = F_{rnn}(\bar{x}, u) := A\bar{x} + \Theta^T \quad y \tag{4}$$

where $\bar{x} = [\bar{x}_1, \ldots, \bar{x}_n]$ is the state vector of the RNN, and the manipulated input vector is $u = [u_1, \ldots, u_m]$. As a vector of both $\bar{x}$ and u, the vector y is defined as $[y_1, \ldots, y_n, y_{n+1}, \ldots, y_{m+n}] = [\sigma(\bar{x}_1), \ldots, \sigma(\bar{x}_n), u_1, \ldots, u_m] \in \mathbf{R}^{n+m}$. The notation $\sigma(\cdot)$ represents the nonlinear activation function (e.g., a hyperbolic tangent function) used in the hidden layers. $A = diag[-a_1, \ldots, -a_n]$ is a diagonal negative coefficient matrix where $a_i > 0$ such that each state $\bar{x}$ is stable in the sense of bounded-input bounded-state stability. The notation $\Theta = [\theta_1, \ldots, \theta_n] \in \mathbf{R}^{(n+m)\times n}$ is a matrix containing associated weights to be optimized during the neural network training process. Therefore, the vector $\theta_i = b_i[w_{i1}, \ldots, w_{i(n+m)}]$ is an element of Θ where b_i is a constant, and w_{ij} stands for the weight on the connection between the jth input to the ith neuron where $j = 1, \ldots, (n+m)$ and $i = 1, \ldots, n$ (Fig. 1).

Subsequently, the RNN is trained following a standard learning procedure as discussed in Alhajeri et al. (2021). The datasets for training, validation and testing are generated from extensive open-loop simulations of the process model under sufficient variation of initial conditions and control actions. In particular, the continuous-time system of Eq. (1) is numerically integrated using the explicit Euler's method with an appropriately small integration time step h_c, and the control actions u are implemented in a sample-and-hold fashion, i.e., $u(t) = u(t_k)$, $\forall\, t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$ and Δ denotes sampling period. Since the RNN is known for its ability to capture nonlinear process dynamic behavior from time-series data (Chen and Chen, 1995; Park and Sandberg, 1991), the RNN model can be trained using all or some of the integration step data points (i.e., data at each integration time step h_c) within each sampling time to be able to capture the state evolution. Furthermore, the RNN model needs to satisfy a sufficiently small modeling error ν (i.e., $|\nu| = |F(x,u) - F_{rnn}(\bar{x},u)| \leq \gamma|x| \leq \nu_{min}$, where γ, $\nu_{min} > 0$) during the model training process, and thus it can well represent process dynamics within the considered operating region.

Remark 1. The modeling error ν is not a constant under different inputs and states. However, by limiting the operation to be within the stability region Ω_ρ, the inputs and states are both bounded. Therefore by training the RNN model using the datasets generated within Ω_ρ, the modeling error can be upper bounded by a sufficiently small positive value ν_{min} for all the states within the stability region.

4. Partially-connected RNN model

A neural network model which takes all accessible process inputs to provide prediction of the outputs of interest is favored in developing a dynamic model for a nonlinear process. Developing a dynamic model for such processes can be easily implemented using open-source machine learning packages, and this model would be capable to account for all possible relationships between every input and every output of the underling process. In Fig. 2, the illustration on the left side represents the general structure of a fully-connected RNN with an input layer, hidden layers, and an output layer. Hence, fully-connected RNN models are usually the prime candidate for processes with no prior knowledge.

The expression process structure knowledge refers to a physical understanding of the underlying process that exists in advance to the development of first-principles model process model. It includes but not limited to the model's intended purpose, soft or hard physical constraints imposed on the

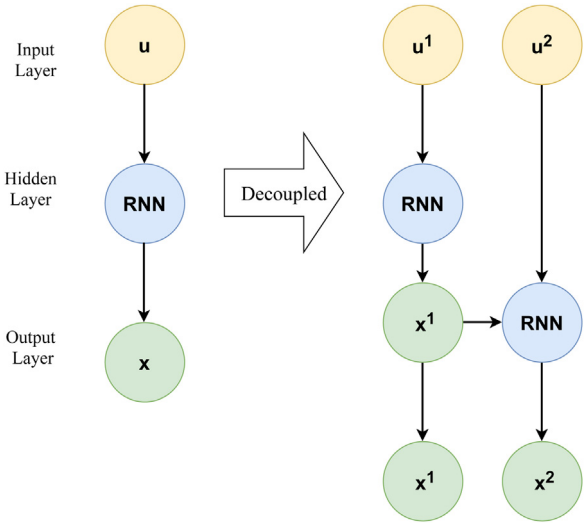

Fig. 2 – Fully-connected and partially-connected RNN structure, where $\mathbf{u}=[u^1, u^2]$ and $\mathbf{x}=[x^1, x^2]$.

process by design considerations, and process structure. In this work, we focus on process structure knowledge in terms of relationships between the process input and output variables. Particularly, in chemical process industry, the physical relations among the process variables can be straightforward. For instance, the upstream processes affect the downstream stage processes, while such an effect does not exist in the opposite direction. This connection between upstream and downstream stages is often reflected in a mathematical model obtained via first-principles. Therefore, incorporating process knowledge that describes physical relations among the underlying process inputs and outputs into the neural network structural modeling will improve its performance as discussed in Thompson and Kramer (1994). This modeling methodology is called partially-connected neural network (Wu et al., 2020).

In this study, a partially-connected RNN model, as illustrated in Fig. 2 on the right side, is developed for the nonlinear system of Eq. (1). We consider that $\mathbf{u}=[u^1 \in \mathbf{R}^{m_1}, u^2 \in \mathbf{R}^{m_2}]$ and $\mathbf{x}=[x^1 \in \mathbf{R}^{n_1}, x^2 \in \mathbf{R}^{n_2}]$ where $m=m_1+m_2$ and $n=n_1+n_2$, and that only the input vector u^1 affects the state vector x^1, while x^2 is affected by the two input vectors u^1 and u^2. By adjusting the RNN structure to explicitly exclude the connection between x^1 and u^2, physical knowledge is integrated into the RNN modeling of the nominal system. As a result, an improved approximation is achieved. For example, as discussed in Wu et al. (2020), the partially-connected RNN models can remarkably reduce the required number of hidden neurons, and weight parameters to achieve the desired performance compared to fully-connected RNN model. Moreover, partially-connected RNNs may be able to capture the process dynamics using a smaller training dataset, since the use of process knowledge may simplify the optimization process of an RNN model by revealing the correct search direction.

The development of partially-connected RNN models follows the development framework of fully-connected RNN models, but with more specifications. A dataset for training/validation can be constructed either from experimental and industrial sources, or by extensive open-loop simulation as discussed in the second paragraph of the previous section. The rule of thumb of splitting the collected dataset to 70% training and 30% validating can be employed, or more sophisticated methods such as cross validation may be used. The input vectors u^1, u^2 and the output vectors x^1, x^2 should be specified before RNN models training. This step is also known as data pre-processing.

In this work, we used the open-source library 'Keras' in Python to construct and train the RNN models. Specifically, our models are constructed with an input layer, an output layer, and two hidden layers activated by the hyperbolic tangent function. To generate a partially-connected RNN model, rather than taking the input vector $\mathbf{u}$ as in the fully-connected model, the input vectors u^1 and u^2 are fed separately using different input layers with respect to the process structure as illustrated in Fig. 2. The first hidden layer takes the input vector u^1 and predicts the output vector x^1. Subsequently, u^2 and x^1 are merged and sent to the second hidden layer to predict x^2. Eventually, the constructed model can provide prediction for both output vectors x^1 and x^2. The Pseudocodes 1 and 2 below summarize the construction procedure of the partially-connected RNN models, and may be useful to colleagues who may pursue the approach.

Remark 2. Both the fully-connected and the partially-connected RNN models are developed for the nominal system described in Eq. (1) assuming no disturbances. In the presence of time-varying disturbances, the prediction of RNN models may under-perform due to model mismatch. To resolve this issue, the RNN models can be updated online using recent process measurements to capture the process-model mismatch caused by the disturbances.

Remark 3. Note that partially-connected RNN models only reflect process structure on its internal network connection without explicit expression. Therefore this model is still a "black-box" model which is different from the hybrid models. For discussion on hybrid modeling of chemical processes, the interested readers may refer to Alhajeri et al. (2021), Zhang et al. (2019), Ghosh et al. (2019).

Remark 4. The classes and features of layers in Pseudocodes are named in the Keras manner. The naming could be different for other machine learning application programming interface (API), such as Tensorflow and Pytorch. The number of input and output features, and the number of data points in the input data sequence should be specified in the data prepossessing step as mentioned in this section and the previous section, respectively.

Pseudocode 1: Partially-connected RNN Construction

input layer-1:
{
layer class: Input
units: (number of input features in vector u^1)
input shape: (number of data points in the input data sequence, number of input features in vector u^1)
connected to: hidden layer-1
}

hidden layer-1:
{
layer class: Long Short-term Memory
units: (Number of inputs) × (Number of outputs)
return sequences: true
activation function: tanh
recurrent activation function: sigmoid
recurrent initializer: orthogonal
use bias: true
connected to: output layer-1
}

output layer-1:
{
layer class: Dense
units: number of outputs
activation function: linear
output shape: (number of data points in the output data sequence , number of outputs in vector x^1)
connected to: merge layer
}

input layer-2
{
layer class: Input
units: (number of input features in vector u^2)
input shape: (number of data points in the input data sequence , number of input features in vector u^2)
connected to: merge layer
}

merge layer:
{
layer class: Concatenate
connected to: hidden layer-2
}

Pseudocode 1: Partially-connected RNN Construction (continued)

hidden layer-2
{
layer class: Long Short-term Memory
units: (Number of inputs) × (Number of outputs)
return sequence: true
activation function: tanh
recurrent activation function: sigmoid
recurrent initializer: orthogonal
use bias: true
connected to: output layer-2
}

output layer-2
{
layer class: Dense
units: number of output
activation function: linear
output shape: (number of data points in the output data sequence , number of outputs in vector x^2)
}

Pseudocode 2: Partially-connected RNN Training

model compile
{
optimizer: adam (candidate optimizer: RMSprop, SGD, etc.)
loss function: mean squared error
}

early stop
{
monitor: validation loss
early stop condition: 1×10^{-8}
}

model fit
{
training:(x_t, y_t) *:*
x_t*: python list (input training set for input layer 1, input training set for input layer 2)*
y_t*: python list (output training set for output layer 1, output training set for output layer 2)*
batch size: 32 (defaults value)
epochs: 50 (user choice, other numbers can be used)
validation:(x_v, y_v) *:*
x_v*: python list (input validation set for input layer 1, input validation set for input layer 2)*
y_v*: python list (output validation set for output layer 1, output validation set for output layer 2)*

callbacks: early stop
}

5. RNN-based model predictive control

In this section, we incorporate an RNN model in a Lyapunov-based model predictive control (LMPC) strategy. Specifically, the RNN model (with partially-connected or fully-connected structure) provides state predictions to solve the MPC optimization problem, which is formulated as follows:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+P}} L(\tilde{x}(t), u(t))dt \tag{5a}$$

$$\text{s.t.} \quad \tilde{x}(t) = F_{rnn}(\tilde{x}(t), u(t)) \tag{5b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{5c}$$

$$u(t) \in U, \forall t \in [t_k, t_{k+P}) \tag{5d}$$

$$\dot{V}(x(t_k), u) \leq \dot{V}(x(t_k), \Phi(x(t_k))), \quad \text{if} \quad x(t_k) \in \Omega_\rho - \Omega_{\rho_{rnn}} \tag{5e}$$

$$V(\tilde{x}(t)) \leq \rho_{rnn}, \quad \forall \ t \in [t_k, t_{k+P}), \quad \text{if} \quad x(t_k) \in \Omega_{\rho_{rnn}} \tag{5f}$$

where predicted state trajectory is $\tilde{x}$. $S(\Delta)$ denotes the set of constant piecewise functions with period Δ, and the prediction horizon is P. The function $\dot{V}(x, u)$ in Eq. (5e) is the time-derivative of Lyapunov function V (i.e., $\frac{\partial V(x)}{\partial x}(F_{rnn}(x, u))$). The LMPC computes the optimal inputs series $u^*(t)$ over the specified prediction horizon $t \in [t_k, t_{k+P})$. The first optimal inputs $u^*(t)$ of the each prediction horizon is sent to the system to be implemented for the next sampling period. Then, the new state measurements are fed back to the LMPC and the control optimization problem is resolved again with the new state measurements at the next sampling period. Moreover, the objective of the MPC optimization problem is to minimize the time integral cost function $L(\tilde{x}, u)$ as represented in Eq. (5a) over the prediction horizon while satisfying the constraints of Eqs. (5b)–(5f). The first constraint of Eq. (5b) is the RNN model from Eq. (4) that is utilized to predict the evolution of the closed-loop state. In Eq. (5b), $x(t_k)$ is used to update the initial condition of the prediction $\tilde{x}(t_k)$. As for the inputs, the constraints are represented by Eq. (5d), which are applied throughout the entire prediction horizon.

To maintain the closed loop stability, the contractive constraint of Eq. (5e) is activated when $x(t_k) \in \Omega_\rho - \Omega_{\rho_{rnn}}$. This constraint forces the Lyapunov function of the closed-loop states to decrease, and as a result, the actual state will approach the steady-state in finite time. Furthermore, if the last state $x(t_k)$ enters the desired region $\Omega_{\rho_{rnn}}$, then the predicted closed-loop state will be maintained within this region for the entire prediction horizon. The work of Wu et al. (2019) demonstrated that when using RNN-based LMPC as in Eq. (5) to control a nonlinear system as given in Eq. (1), the closed-loop state is guaranteed to be bounded within the stability region Ω_ρ for all times, and ultimately it will converge to a very small neighborhood around the origin under the assumption that the modeling error ν is sufficiently small.

Remark 5. In the case where $x(t_k)$ are not fully available online, a state observer may be used to estimate the unmeasured states from the measured ones. The previous work Alhajeri et al. (2021) developed two different machine learning based state estimators in the framework of ML based LMPC for nonlinear processes. It was demonstrated that both the ML-based and the hybrid-model based estimators achieved accurate state estimation, and that all state trajectories initiating from various initial conditions converged to the steady-state under the LMPC.

6. Application to a chemical process modeled in Aspen Plus

In this section, we use a large-scale chemical process to evaluate the proposed partially-connected RNN-based LMPC. Firstly, we develop two dynamic models for a chemical process using the Aspen Plus Dynamics V11 and first-principles modeling principles, respectively. Subsequently, a process time-series dataset is collected to train and test the RNN models via extensive open-loop simulation. Finally, open-loop simulations and closed-loop simulations under RNN-model based MPC are carried out and discussed.

6.1. *Dynamic model in Aspen Plus Dynamics*

The Ethylbenzene (EB) production process using Ethylene (E) and Benzene (B) as raw materials is considered. The main reaction for this process is a second-order, exothermic, and irreversible reaction, and it is taking place along with two other side reactions as described in Eq. (6) below in two non-isothermal, well mixed continuous stirred tank reactors (CSTR). The chemical reactions are as follows:

$$C_2H_4 + C_6H_6 \rightarrow C_8H_{10} \tag{6a}$$

$$C_2H_4 + C_8H_{10} \rightarrow C_{10}H_{14} \tag{6b}$$

$$C_6H_6 + C_{10}H_{14} \rightarrow 2\ \ C_8H_{10} \tag{6c}$$

In this work, the two CSTRs are placed in series, and the process model is developed using Aspen Plus and Aspen Plus Dynamics V11, known as a high-fidelity software for complex chemical processes. Initially, the process model is constructed in Aspen Plus where a steady-state simulation is performed and checked based on material and energy balances. Subsequently, we carry out dynamic simulation of this process in Aspen Plus Dynamics to analyse and control its dynamic performance. We construct the steady-state and the dynamical models through the following procedure:

(1) Inlet streams specification: The raw materials are fed to each reactor as Hexane solutions by the flow rates F_1 and F_2. Hexane solution is used to ensure that the inlet flows remain in liquid phase under the feeding temperature. The concentration of Ethylene, Benzene, Ethylbenzene, and Di-Ethylbenzene are denoted by C_E, C_B, C_{EB}, and C_{DEB}, respectively. T_i, ρ_i, and V_i are the temperature, mass density, and the liquid volume of $CSTR_i$, $i = 1, 2$. The mass heat capacity of the liquid mixture is denoted by C_p, and it is assumed to be constant. The values of process parameters used are listed in Table 1, where the subscript "o" indicates the initial state, and "s" represents the steady-state.
(2) Pressure drop selection: To establish a dynamic model for Aspen Plus Dynamics, valves are essential as connectors of parts and fluid flow by manipulating pressure drop throughout the process. With a proper pressure drop, the simulation runs smoothly and the model is able to specify the flow direction, and if inadequate pressure drop is selected, it will return a simulation error. In our model, the pressure drop in valves v_1, v_2, v_3, and v_4 are chosen to be 5, 5, 2, and 14 bars, respectively.
(3) Reactor setting: Each $CSTR_i$ is associated with a heating/cooling jacket which supplies/removes heat at rate Q_i, $i = 1, 2$. The initial pressure of both CSTRs are set to be 15 bar, and the initial temperature of the first and second CSTR are 400 K and 450 K, respectively, to keep the reactants and products in liquid phase during the process. Those values will be automatically adjusted by performing build-in steady-state simulations. After setting up the reactions in the two CSTRs, steady-state simulation is executed for the purpose of analysing the plant behavior.
(4) Thermodynamic parameters & reactor geometric: Before exporting the steady-state model from Aspen Plus to Aspen Plus Dynamics, the thermodynamic parameters and the reactor geometry need to be specified. For our model, the vessels type is vertical, the head type is flat, and the length of each CSTR is ten meters. The thermodynamics parameters are listed in Table 1.
(5) Pressure checking: To make sure that the dynamic model is set properly, we run the steady-state simulation again and perform pressure checking via the built-in Aspen Plus pressure checker without encountering errors. Subsequently, the steady-state model is exported to Aspen Plus Dynamics.
(6) Dynamic model initialization: A direct-acting level controller is added to each reactor to maintain the reactors at half capacity. The level controller can be designed and added following the default setting in Aspen Plus before exporting the steady-state model, or it can be developed manually in Aspen Plus Dynamics. Following the level controllers configuration, we apply a steady-state simulation to obtain the steady-state values of the dynamical model, which gives $Q_{1s} = -911.455$ kW and $Q_{2s} = -6835.270$ kW.
(7) Data type configuration: In order to allow the outside control of the manipulated variables (i.e., Q_1 and Q_2) during the dynamic simulation, the heating type of the two reactors are changed to constant duty. Also, the volumetric flow rates F_1 and F_2 are set as fixed constants, and a steady-state simulation is performed again to ensure that the dynamic model remains at steady-state after the data type configuration. Hence, building the process dynamical model is completed, and the model's flow sheet is illustrated in Fig. 3.

Table 1 – Parameter and steady-state values of the Aspen Plus model.

$T_{1_o} = 350$ K	$T_{1_s} = 310.523$ K
$T_{2_o} = 350$ K	$T_{2_s} = 430.542$ K
$F_1 = 43.2$ m^3/h	$F_2 = 91.079$ m^3/h
$C_{E_1} = 4.2455$ kmol/m^3	$C_{E_2} = 0.3254$ kmol/m^3
$C_{B_1} = 5.3532$ kmol/m^3	$C_{B_2} = 1.3841$ kmol/m^3
$C_{EB_1} = 0.1854$ kmol/m^3	$C_{EB_2} = 3.8744$ kmol/m^3
$C_{DEB_1} = 9.1426 \times 10^{-7}$ kmol/m^3	$C_{DEB_2} = 0.0058$ kmol/m^3
Heat transfer option	Dynamics
Medium temperature	298 K
Temperature approach	77.33 K
Heat capacity of coolant	4200 J/kg K
Medium holdup	1000 kg
$C_p = 2.411$ kJ/kg K	$\rho_1 = 639.1530$ kg/m^3
$V_1 = V_2 = 60$ m^3	$\rho_2 = 607.5040$ kg/m^3

Open-loop simulation is performed using the constructed dynamical model with pseudo-random input signals generated by a MATLAB script. To link Aspen Plus Dynamics with MATLAB, a local message passing interface (MPI) is created, such that the dynamical model is able to automatically read the input signals from MATLAB and then implement them in the dynamic simulations. In particu-

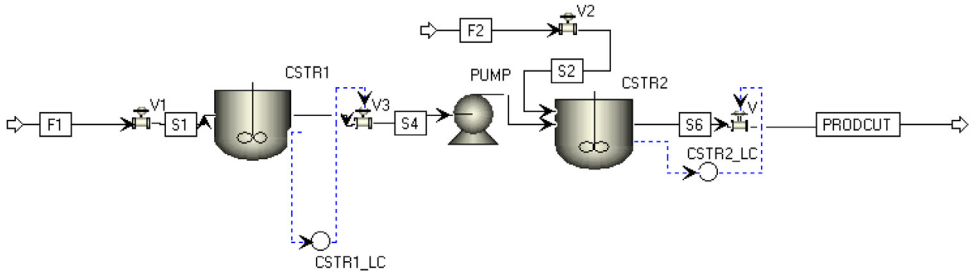

Fig. 3 – Aspen Plus model flow sheet of two reactors in series.

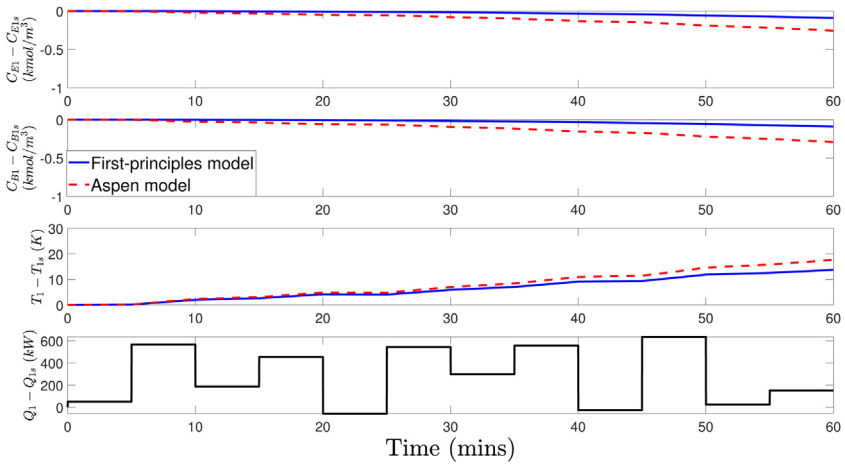

Fig. 4 – Open-loop state and manipulated input profiles for $CSTR_1$.

lar, the MATLAB script generates the manipulated variables in deviation form against their steady-state values (i.e., $u_1 = Q_1 - Q_{1s}$ and $u_2 = Q_2 - Q_{2s}$). The two manipulated variables randomly vary within the range of $[-1 \times 10^4$ kW, 1×10^3kW], and $[-1.5 \times 10^4$ kW, 5×10^3 kW] respectively, and are implemented to the dynamic simulation in a sample-and-hold manner that the values are updated every five minutes. All input values and output states (e.g., C_E, C_B, and T) are recorded in time-series to establish the training/validating dataset.

6.2. *First-principles model development*

Aspen Plus Dynamics is a highly efficient software that allows chemical engineers to simulate, and to optimize chemical process performance and profitability. However, due to their long computational time, typically the Aspen Plus models are not the optimal option to generate data for deep-learning models which require comparatively large amount of data. To overcome this issue, first-principles models with simplifying assumptions are well-established candidates to generate data for machine learning (Takahashi and Tanaka, 2016; Chun et al., 2021). By applying the concepts of mass and energy balances, the first-principles models for the CSTRs are developed. Specifically, the dynamic model of the first CSTR is represented by the following ODEs:

$$\frac{dC_{E_1}}{dt} = \frac{F_1 C_{Eo_1} - F_{out_1} C_{E_1}}{V_1} - r_1 - r_2 \tag{7a}$$

$$\frac{dC_{B_1}}{dt} = \frac{F_1 C_{Bo_1} - F_{out_1} C_{B_1}}{V_1} - r_1 - r_3 \tag{7b}$$

$$\frac{dC_{EB_1}}{dt} = \frac{-F_{out_1} C_{EB_1}}{V_1} + r_1 - r_2 + 2r_3 \tag{7c}$$

$$\frac{dC_{DEB_1}}{dt} = \frac{-F_{out_1} C_{DEB_1}}{V_1} + r_2 - r_3 \tag{7d}$$

$$\frac{dT_1}{dt} = \frac{(T_{1_o} F_1 - T_1 F_{out_1})}{V_1} + \sum_{j=1}^{3} \frac{-\Delta H_j}{\rho_1 C_p} r_j + \frac{Q_1}{\rho_1 C_p V_1} \tag{7e}$$

The dynamic model of the second CSTR is comprised of the following ODES:

$$\frac{dC_{E_2}}{dt} = \frac{F_2 C_{Eo_2} + F_{out_1} C_{E_1} - F_{out_2} C_{E_1}}{V_2} - r_1 - r_2 \tag{8a}$$

$$\frac{dC_{B_2}}{dt} = \frac{F_2 C_{Bo_2} + F_{out_1} C_{B_1} - F_{out_2} C_{B_1}}{V_2} - r_1 - r_3 \tag{8b}$$

$$\frac{dC_{EB_2}}{dt} = \frac{F_{out_1} C_{EB_1} - F_{out_2} C_{EB_2}}{V_2} + r_1 - r_2 + 2r_3 \tag{8c}$$

$$\frac{dC_{DEB_2}}{dt} = \frac{F_{out_1} C_{DEB_1} - F_{out_2} C_{DEB_2}}{V_2} + r_2 - r_3 \tag{8d}$$

$$\frac{dT_2}{dt} = \frac{(T_{2_o} F_2 + T_1 F_{out_1} - T_2 F_{out_2})}{V_2} + \sum_{j=1}^{3} \frac{-\Delta H_j}{\rho_2 C_p} r_j + \frac{Q_2}{\rho_2 C_p V_2} \tag{8e}$$

where the reaction rates are calculated by the following expressions:

$$r_1 = k_1 e^{\frac{-E_1}{RT_i}} C_{E_i} C_{B_i} \tag{9a}$$

$$r_2 = k_2 e^{\frac{-E_2}{RT_i}} C_{EB_i} C_{E_i} \tag{9b}$$

$$r_3 = k_3 e^{\frac{-E_3}{RT_i}} C_{DEB_i} C_{B_i}, \quad i = 1, 2 \tag{9c}$$

The parameters used in the first-principles models are listed in Table 1. Figs. 4–5show open-loop simulations of the

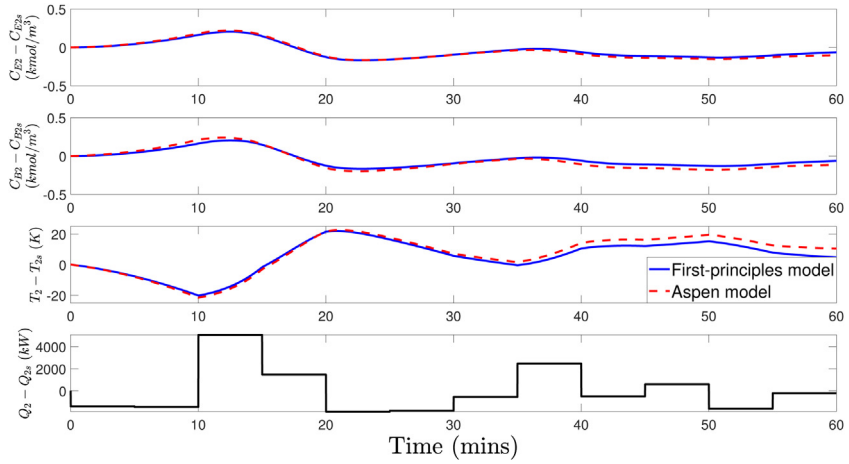

Fig. 5 – Open-loop state and manipulated input profiles for $CSTR_2$.

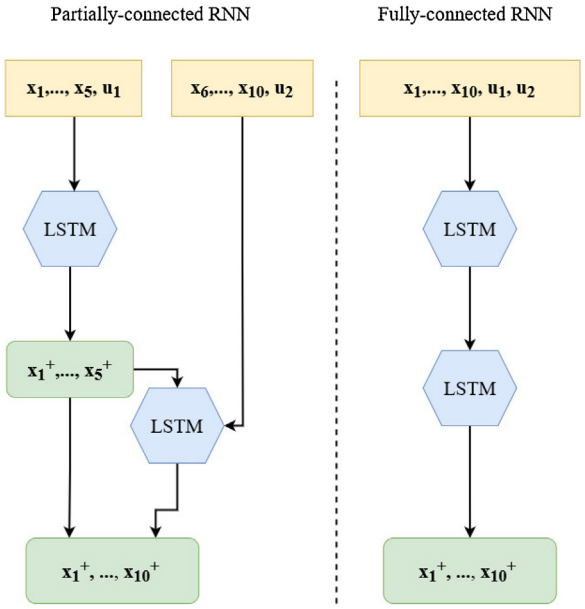

Fig. 6 – RNN modeling structures for ethylbenzene production in two CSTRs in series, where x_i and x_i^+ are defined at $t = t_k$ and $t = t_k + \Delta$, respectively.

first-principles model and of the Aspen Plus model under the same time-varying inputs and initial conditions. This simulation illustrate the good agreement between the two models within the operating region.

6.3. *Data generation and RNN models development*

In this work, we create a dataset that contains open-loop simulation data from both the Aspen Plus and the first-principles models to develop the two RNN models. Using Keras library, the two RNN models are constructed following the diagram shown in Fig. 6. The fully-connected and partially-connected RNN models are designed as follows: they have two long short term memory (LSTM) layers with 50 neurons in each, and they are activated by hyperbolic tangent functions (i.e., $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$). For the training part, the fully-connected and partially-connected RNN models are both developed based on the same dataset and via using Keras library with the same neural network parameters as follows: two hidden layers with 50 neurons in every layer, and hyperbolic tangent as the activation function. The output layer is activated by a linear activation function, which gives estimation for 10 states, and there are 12

Table 2 – Input and output states of the RNN models.

Notation	State (in deviation form)
x_1	Concentration of ethane for $CSTR_1$
x_2	Concentration of benzene for $CSTR_1$
x_3	Concentration of ethylbenzene for $CSTR_1$
x_4	Concentration of butylbenzene for $CSTR_1$
x_5	Temperature of the reactor for $CSTR_1$
x_6	Concentration of ethane for $CSTR_2$
x_7	Concentration of benzene for $CSTR_2$
x_8	Concentration of ethylbenzene for $CSTR_2$
x_9	Concentration of butylbenzene for $CSTR_2$
x_{10}	Temperature of the reactor for $CSTR_2$
u_1	Heating/cooling duty of the reactor for $CSTR_1$
u_2	Heating/cooling duty of the reactor for $CSTR_2$

Inputs	Outputs
$x_1(t_k)$	$x_1(t_k + \Delta)$
$x_2(t_k)$	$x_2(t_k + \Delta)$
$x_3(t_k)$	$x_3(t_k + \Delta)$
$x_4(t_k)$	$x_4(t_k + \Delta)$
$x_5(t_k)$	$x_5(t_k + \Delta)$
$x_6(t_k)$	$x_6(t_k + \Delta)$
$x_7(t_k)$	$x_7(t_k + \Delta)$
$x_8(t_k)$	$x_8(t_k + \Delta)$
$x_9(t_k)$	$x_9(t_k + \Delta)$
$x_{10}(t_k)$	$x_{10}(t_k + \Delta)$
$u_1(t_k)$	
$u_2(t_k)$	

inputs for both neural network models. The input and output variables are listed in Table 2. We use the input data that covers the five minute sampling period to predict the states evolution for the next five minutes. Rather than using the conventional gradient decent optimization algorithm, we use Adam optimizer which is a combination of two algorithms, the gradient descent with momentum and the RMSprop. Moreover, to produce more robust models, we apply five-fold cross validation, and select the model with the least mean squared error (MSE) for each RNN structure.

We train the two models starting with 50 epochs, and the partially-connected RNN model reaches the early stopping criteria (i.e., *validation loss* $= 1 \times 10^{-8}$) at the 20th epoch. Thus, we use 20 epochs to train the fully-connected RNN model for comparison consistency. The training and validating summary for the two models is illustrated in Figs. 7 and 8. We can see that the two developed models yield high accuracy in both training and validating processes, where the accuracy is demonstrated

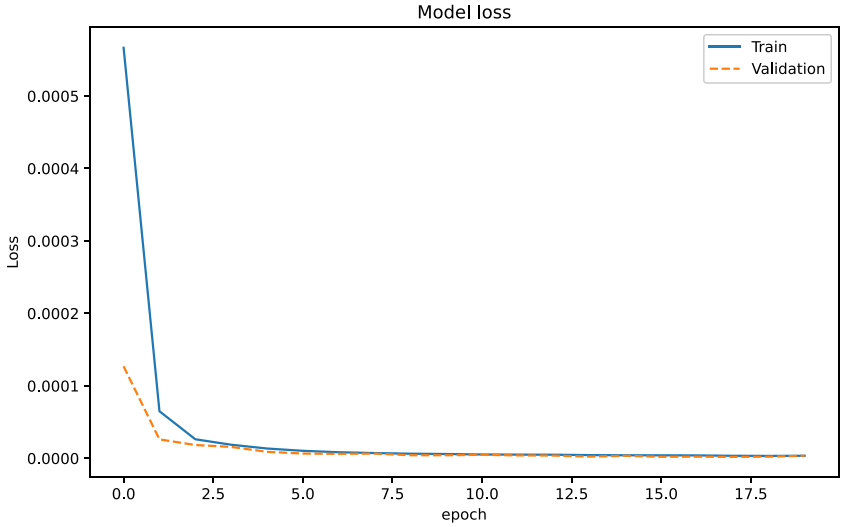

Fig. 7 – Partially-connected RNN model training and validation loss functions.

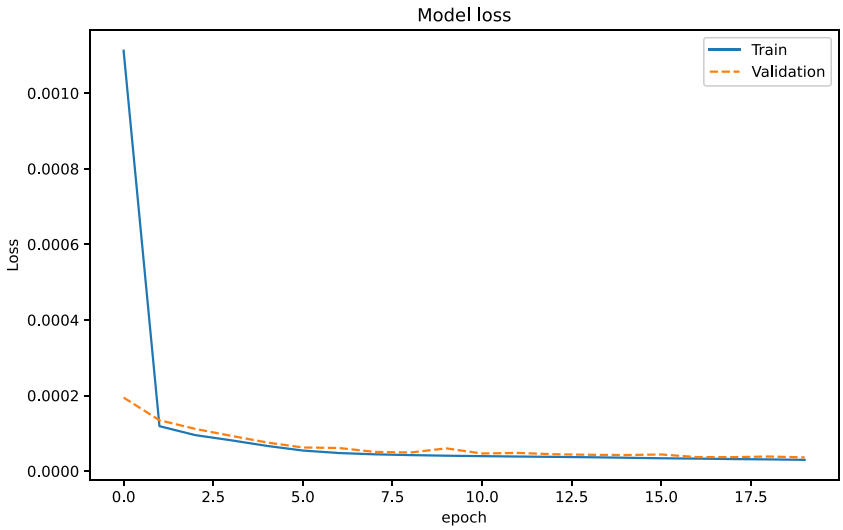

Fig. 8 – Fully-connected RNN model training and validation loss functions.

Table 3 – MSE comparison of the open-loop prediction results between the RNN models and the Aspen Plus simulation model.

	Partially-connected RNN	Fully-connected RNN
T_1	2.52×10^{-3}	6.93×10^{-1}
C_{E_1}	6.626×10^{-7}	5.51×10^{-5}
T_2	1.837×10^{-1}	1.723
C_{E_2}	1.996×10^{-2}	1.988×10^{-2}

in terms of MSE between the model prediction and the reference value.

For the open-loop simulation, we designate a message passing interface (MPI) for files exchanging to adopt random control actions from a Python script to the Aspen Plus simulation. The control actions are simultaneously applied to the RNN models. Therefore, this simulation fulfills two objectives: (a) checking the connection between Python and Aspen Plus, and (b) testing for the open-loop prediction of the two RNN models.

Table 3 summarizes the simulation results, and presents the MSE between the predicted states from each RNN model and the corresponding Aspen Plus Dynamics model outputs as reference. Furthermore, the open-loop responses predicted by the partially-connected and the fully-connected RNN models under a step change in u_2 are displayed in Fig. 9. The figure demonstrates that the partially-connected RNN model has an improved model identification of the process dynamics. These results indicate that both RNN models provide accurate prediction, yet, the partially-connected RNN model approximates the Aspen process model more accurately.

6.4. *Closed-loop simulation: first-principles process model*

Subsequently, we develop LMPCs based on the fully-connected RNN model and the partially-connected RNN model respectively, to perform the closed-loop simulation with the confidence that both RNN models provide high accuracy approximation for the process outputs. In order to develop the LMPCs, we use the Python version of the interior point optimizer (IPOPT) package to solve the nonlinear optimization problem of the LMPC for each sampling time Δ. This optimizer is an open source package which can be used for solving large-scale nonlinear optimization problems. It employs an interior point line search filter technique which intends to find a local solution of nonlinear programming problems. The LMPC objective function is defined as $L(x, u) = x^T \mathbf{Q}\, x + u^T \mathbf{R}\, u$, where $\mathbf{Q}$ and $\mathbf{R}$ are diagonal penalty matrices for the setpoint error and control actions, respectively. The two matrices are critically

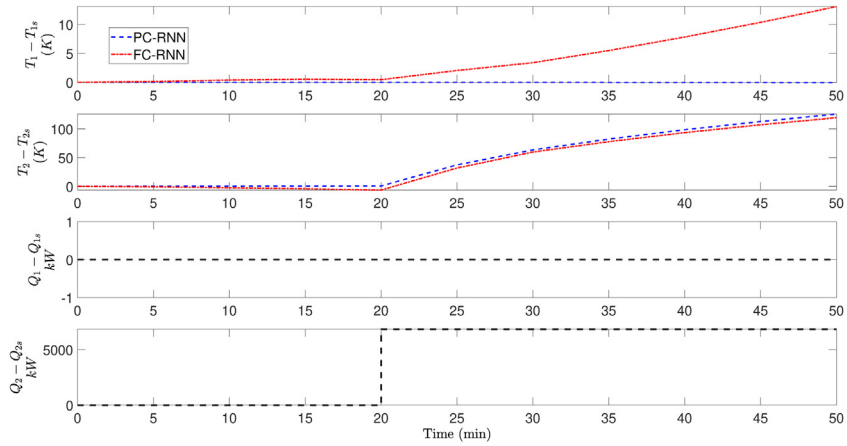

Fig. 9 – Open-loop simulation under step change in ($Q_2 - Q_{2s}$) of the two RNN models: partially-connected RNN (denoted by PC-RNN in dashed line), and the fully-connected RNN (denoted by FC-RNN in dash-dotted line).

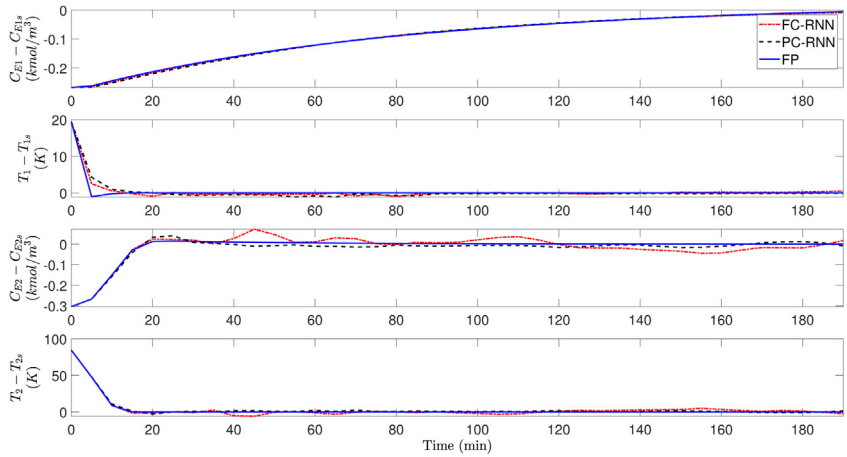

Fig. 10 – State profiles of the closed-loop simulation of the first-principles process model under the LMPC using three models: first-principles (denoted by FP in solid line), partially-connected RNN (denoted by PC-RNN in dashed line), and fully-connected RNN (denoted by FC-RNN in dash-dotted line).

impacting the performance of the LMPC and require proper tuning, hence, the tuning guidelines discussed in Alhajeri and Soroush (2020) is followed. Lastly, we choose $V(x) = x^T P x$ as the Lyapunov function, where P is a positive definite matrix obtained by applying grid search.

Under the LMPC, we first perform the closed-loop simulation using the first-principles process model of Eq. (7)–(8) integrated by explicit Euler method with times step $h_c = 0.05$ min, and the results are shown in Fig. 10–11. From those figures, both LMPCs, each based on its predictive RNN model, are able to stabilize the process by driving the states close to the steady-state values. However, state trajectories resulting from the partially-connected RNN based LMPC are smoother and do not exhibit oscillation around the steady-state. This simulation is used to find the MPC parameters, such as parameters in the cost function, that would deliver a desired closed-loop performance. Furthermore, it is important to check the closed-loop state evolution before applying the proposed LMPC to the high-fidelity Aspen Plus Dynamics model.

Remark 6. The MPI implements information exchange by defining a digital platform that allows access from the python-based MPC and the Aspen dynamic software. Specifically, a python script is developed to automatically upload the MPC output to the shared platform during the simulation. Simultaneously, the Aspen dynamic simulation can search and find the correct input data to update its control states via the build-in script function. Subsequently, the Aspen dynamic simulation can send the real-time measurement to the MPC using the same scripts and platform to carry on the next iteration. Commercial platforms, such as Google Drive and Dropbox, can be efficient candidates to adopt for data exchanging if it is not necessary to construct a specified database for users' projects.

6.5. *Closed-loop simulation: Aspen Plus Dynamic model*

Finally, we carry out closed-loop simulations of the Aspen Plus Dynamic model under the proposed LMPCs, and the results are shown in Fig. 12–13. Both LMPCs stabilize the process at the steady-state exhibiting similar dynamic performance. The responses under the fully-connected RNN-based LMPC exhibits noticeable oscillation, while the ones under the LMPC that utilizes partially-connected RNN model are smoother. This is expected since the fully-connected RNN assumes that every input affects every possible output, which interferes with the prediction accuracy. We note that an ensemble of RNN models may be developed from the training date set and used to make average predictions of the future state evolution (Wu et al., 2019), but this approach was not pursued in the present work as the MPCs implemented using the developed RNN models produced desired closed-loop responses.

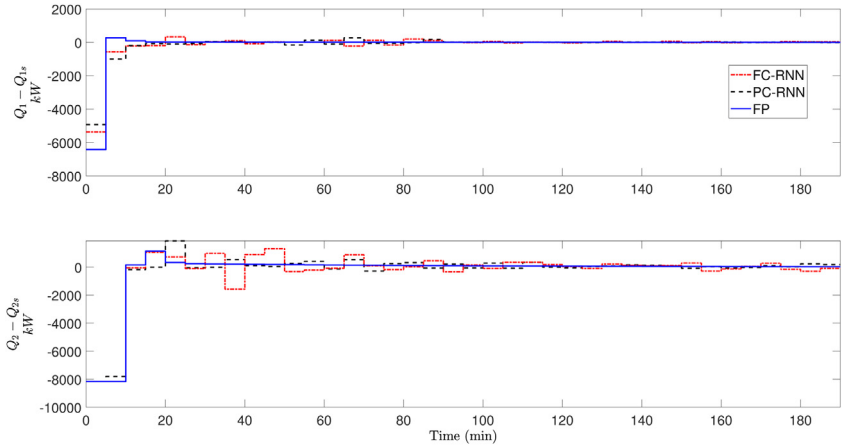

Fig. 11 – Input profiles of the closed-loop simulation of the first-principles process model under the LMPC using three models: first-principles (denoted by FP in solid line), partially-connected RNN (denoted by PC-RNN in dashed line), and fully-connected RNN (denoted by FC-RNN in dash-dotted line).

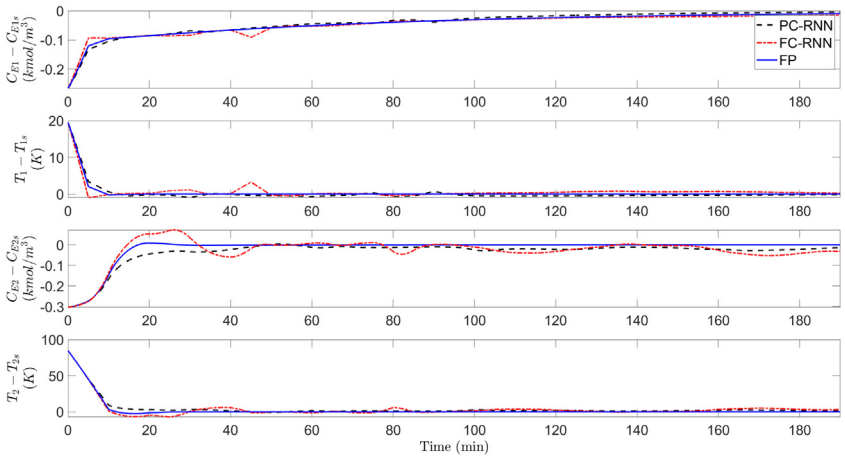

Fig. 12 – State profiles of the closed-loop simulation of the Aspen Plus Dynamics model under the LMPC using three models: first-principles (denoted by FP in solid line), partially-connected RNN (denoted by PC-RNN in dashed line), and fully-connected RNN (denoted by FC-RNN in dash-dotted line).

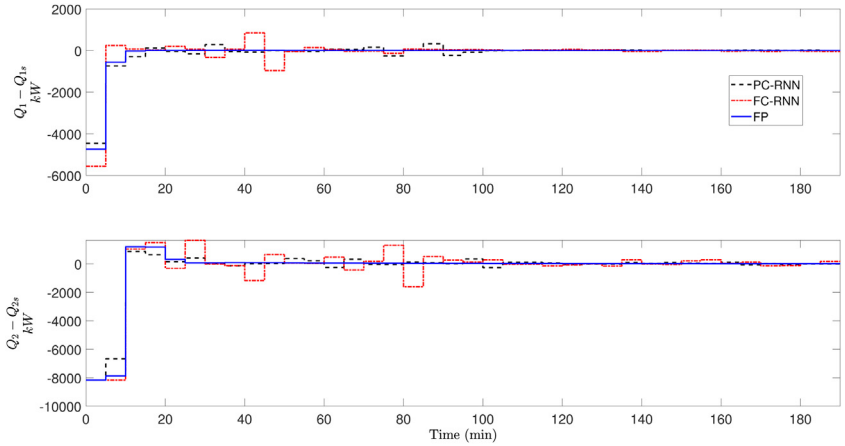

Fig. 13 – Input profiles of the closed-loop simulation of the Aspen Plus Dynamics model under the LMPC using three models: first-principles (denoted by FP in solid line), partially-connected RNN (denoted by PC-RNN in dashed line), and fully-connected RNN (denoted by FC-RNN in dash-dotted line).

Another critical performance metric is the computational time needed to calculate the control action, which impacts the feasibility of the controller in real-time operation. To evaluate this we run closed-loop simulation, and record the computational time to solve for the optimum controller actions in each sampling period for the two different RNN-based LMPCs. The computational time ratio of the fully-connected RNN-LMPC to the partially-connected RNN-LMPC is presented in Fig. 14. As illustrated in Fig. 14, the fully-connected RNN-LMPC requires longer computational time to find the optimal solution in 32 out of the 39 calculations. On average, the computational times for the fully-connected RNN-LMPC and the

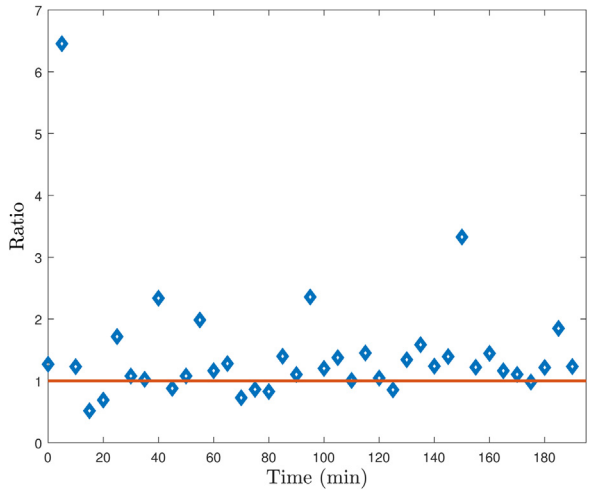

Fig. 14 – Ratio of computational time needed to calculate the control actions by LMPC using fully-connected RNN and partially-connected RNN at each sampling time Δ.

Table 4 – Statistical analysis of the computational times (in minutes) needed to calculate the control actions using the two RNN structures in LMPC.

	Partially-connected RNN	Fully-connected RNN
μ	1.65305	2.1161
σ	0.6185	0.87922
Range (min–max)	(0.38 – 3.77)	(1.21 – 4.8)

partially-connected RNN-LMPC are 2.1161 and 1.65305 min, respectively. Furthermore, the computation times mean μ, standard deviation σ, and range using each RNN model are listed in Table 4. From the table, the three parameters indicate that the computational times for the fully-connected RNN-LMPC are longer than the partially-connected RNN-LMPC. In particular, using the partially-connected RNN structure in the LMPC, the overall computational time has been reduced by approximately 22%.

7. Conclusion

In this work, a partially-connected RNN model, which integrates a priori process-structure knowledge into the RNN modeling, was developed and utilized as a predictive model in an LMPC, and evaluated by a dynamic simulation for a chemical process using Aspen Plus Dynamic. It was then compared with a fully-connected RNN-based LMPC. The open-loop simulations demonstrated the superiority of the partially-connected RNN by yielding smaller prediction errors. Furthermore, the closed-loop simulations demonstrated that the chemical process under the partially-connected RNN-based LMPC had a smoother state trajectory and the optimal control action calculations required smaller computational time. Finally, the partially-connected RNN based LMPC gave a steady control signal after the process reached steady-state, which eliminated the states variance when under a fully-connected RNN-based LMPC.

Declaration of Competing Interest

The authors report no declarations of interest.

Acknowledgements

Financial support from the National Science Foundation and the Department of Energy is gratefully acknowledged. Mohammed S. Alhajeri would like to express his genuine appreciation to Kuwait University for its support via the KU-scholarship program. Fahad Albalawi acknowledges Taif University for their support via Taif University Researchers Supporting Project (TURSP-2020/97).

References

Alhajeri, M., Soroush, M., 2020. Tuning guidelines for model-predictive control. Ind. Eng. Chem. Res. 59, 4177–4191.
Alhajeri, M.S., Wu, Z., Rincon, D., Albalawi, F., Christofides, P.D., 2021. Machine-learning-based state estimation and predictive control of nonlinear processes. Chem. Eng. Res. Des. 167, 268–280.
Banerjee, A., Varshney, D., Kumar, S., Chaudhary, P., Gupta, V.K., 2017. Biodiesel production from castor oil: Ann modeling and kinetic parameter estimation. Int. J. Ind. Chem. 8, 253–262.
Camacho, E., Bordons, C., 2013. Model Predictive Control, 2nd ed. Springer Science & Business Media, London.
Chen, T., Chen, H., 1995. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Trans. Neural Netw. 6, 911–917.
Chun, H., Lee, E., Nam, K., Jang, J., Kyoung, W., Noh, S., Han, B., 2021. First-principle-data-integrated machine-learning approach for high-throughput searching of ternary electrocatalyst toward oxygen reduction reaction. Chem. Catal. 1, 855–869.
De Azevedo, S.F., Dahm, B., Oliveira, F., 1997. Hybrid modelling of biochemical processes: a comparison with the conventional approach. Comput. Chem. Eng. 21, S751–S756.
Dias, A.C.S.R., da Silva, W.B., Dutra, J.C.S., 2017. Propylene polymerization reactor control and estimation using a particle filter and neural network. Macromol. React. Eng. 11, 1700010.
Ellis, M., Liu, J., Christofides, P.D., 2017. Economic Model Predictive Control, vol. 5. Springer, Switzerland.
Ghosh, D., Hermonat, E., Mhaskar, P., Snowling, S., Goel, R., 2019. Hybrid modeling approach integrating first-principles models with subspace identification. Ind. Eng. Chem. Res. 58, 13533–13543.
Kahrs, O., Marquardt, W., 2007. The validity domain of hybrid models and its application in process optimization. Chem. Eng. Process.: Process Intensif. 46, 1054–1066.
Lee, C., 1990. Fuzzy logic in control systems: fuzzy logic controller. I. IEEE Trans. Syst., Man, Cybern. 20, 404–418.
Liao, S., 2005. Expert system methodologies and applications – a decade review from 1995 to 2004. Expert Syst. Appl. 28, 93–103.
Lin, Y., Sontag, E.D., 1991. A universal formula for stabilization with bounded controls. Syst. Control Lett. 16, 393–397.
Lu, Y., Rajora, M., Zou, P., Liang, S., 2017. Physics-embedded machine learning: case study with electrochemical micro-machining. Machines 5, 4.
Pan, Y., Wang, J., 2011. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. IEEE Trans. Ind. Electron. 59, 3089–3101.
Park, J., Sandberg, I.W., 1991. Universal approximation using radial-basis-function networks. Neural Comput. 3, 246–257.
Patel, N., Nease, J., Aumi, S., Ewaschuk, C., Luo, J., Mhaskar, P., 2020. Integrating data-driven modeling with first-principles knowledge. Ind. Eng. Chem. Res. 59, 5103–5113.
Singh, A., Singh, H.P., Mishra, S., 2017. Validation of ANN-based model for binary distillation column. Proceeding of International Conference on Intelligent Communication, Control and Devices, 235–242.
Stephanopoulos, G., Han, C., 1996. Intelligent systems in process engineering: a review. Comput. Chem. Eng. 20, 743–791.
Takahashi, K., Tanaka, Y., 2016. Material synthesis and design from first principle calculations and machine learning. Comput. Mater. Sci. 112, 364–367.
Thompson, M.L., Kramer, M.A., 1994. Modeling chemical processes using prior knowledge and neural networks. AIChE J. 40, 1328–1340.
Vepa, R., 1993. A review of techniques for machine learning of real-time control strategies. Intell. Syst. Eng. 2, 77–90.
Wong, W.C., Chee, E., Li, J., Wang, X., 2018. Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. Mathematics 6, 242.
Wu, Z., Luo, J., Rincon, D., Christofides, P.D., 2021. Machine learning-based predictive control using noisy data: evaluating performance and robustness via a large-scale process simulator. Chem. Eng. Res. Des. 168, 275–287.
Wu, Z., Rincon, D., Christofides, P.D., 2020. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. J. Process Control 89, 74–84.
Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019. Machine learning-based predictive control of nonlinear processes. Part I: Theory. AIChE J. 65, e16729.
Zadeh, L.A., 1968. Probability measures of fuzzy events. J. Math. Anal. Appl. 23, 421–427.
Zhang, Z., Wu, Z., Rincon, D., Christofides, P.D., 2019. Real-time optimization and control of nonlinear processes using machine learning. Mathematics 7, 890.