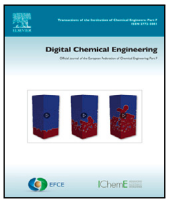
Contents lists available at ScienceDirect

Digital Chemical Engineering

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



Original article

Machine learning-based predictive control of an electrically-heated steam methane reforming process

Yifei Wang[a], Xiaodong Cui[a], Dominic Peters[a], Berkay Çıtmacı[a], Aisha Alnajdi[b], Carlos G. Morales-Guio[a], Panagiotis D. Christofides[a,b,*]

[a] *Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA, 90095-1592, USA*
[b] *Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095-1592, USA*

ARTICLE INFO

Keywords:
Model predictive control (MPC)
Recurrent neural networks (RNN)
Steam methane reforming (SMR)
Transfer learning
Disturbance rejection
Offset-free model predictive control
Real-time online retraining
Hydrogen production

ABSTRACT

Hydrogen plays a crucial role in improving sustainability and offering a clean and efficient energy carrier that significantly reduces greenhouse gas emissions. However, the primary method of industrial hydrogen production, steam methane reforming (SMR), relies on the combustion of hydrocarbons as the heating source for the reforming reactions, resulting in significant carbon emissions. To address this issue, an experimental setup of an electrically-heated steam methane reformer (e-SMR) has been constructed at UCLA, and a lumped first-principle dynamic process model was built based on parameters estimated from the experimental data in a previous study. Subsequently, the first-principle dynamic process model was implemented into the computational model predictive control (MPC) scheme, successfully driving the hydrogen production rate to the desired setpoint. While these works are important and pave the way for developing MPC for large-scale e-SMR processes, the first-principle process model may not accurately reflect the actual process behavior, particularly as the process behavior changes with time. Therefore, the development and establishment of an adaptive data-driven approach for implementing model predictive control in the e-SMR process is necessary. To address this need, the present work investigates the construction of recurrent neural network (RNN) models for an e-SMR process in-depth, utilizing data from an experimentally-validated first-principle model. Specifically, a long short-term memory (LSTM) layer was utilized in the RNN model to effectively capture the complex correlations present in long-term sequential data. Subsequently, this LSTM-based RNN process model was employed to design an MPC, and its performance was evaluated through comparison with proportional–integral (PI) control. To address potential disturbances and variability in a typical e-SMR process, three distinct approaches were developed: MPC with an integrator, MPC with real-time online retraining (transfer learning), and offset-free MPC. These approaches effectively eliminated the offset caused by disturbances. Overall, this study underscores the effectiveness of utilizing RNN models to capture process dynamics in an experimental e-SMR process. It also outlines strategies for employing RNN-based control and multiple approaches to address disturbances in general processes with partially infrequent and delayed measurement feedback. This approach is particularly valuable in scenarios where developing first-principle models for a new process may be challenging.

1. Introduction

Hydrogen (H_2) is recognized for its clean and efficient emissions properties and is crucial in addressing global climate change and supporting the urgent energy revolution (Zhou et al., 2022). As an ideal energy carrier (Lubitz and Tumas, 2007), hydrogen substantially contributes to the establishment of environmentally friendly industries, offering a sustainable alternative to traditional fossil fuels. Moreover, its pivotal roles in transportation (Tanç et al., 2019), manufacturing (Green, 1982), energy storage and production (Rasul et al., 2022) underscore its significant contribution to decarbonization. However, in modern industry, nearly 95% of H_2 is produced through steam methane reforming (SMR) (Nieva et al., 2014). Steam methane reforming is a classic industrial process used to produce hydrogen from natural gas. During the traditional SMR reaction process, the combustion of fossil fuels creates the high temperatures required to heat the reactions in large-volume industrial reactors, which significantly contributes to greenhouse gas accumulation and climate change. To address this issue, a novel electrically-heated SMR (e-SMR) system is considered as an alternative green hydrogen production method. Instead of utilizing

* Corresponding author at: Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA, 90095-1592, USA.
E-mail addresses: moralesguio@ucla.edu (C.G. Morales-Guio), pdc@seas.ucla.edu (P.D. Christofides).

https://doi.org/10.1016/j.dche.2024.100173
Received 7 July 2024; Received in revised form 21 July 2024; Accepted 21 July 2024
Available online 23 July 2024

heat from natural gas combustion, electricity is used as the energy source to heat the reformer. This electricity is generated from various clean energy sources, including solar energy, wind power, and hydropower (Pazheri et al., 2014). Therefore, substantially decreased carbon emissions are achieved since no carbon is generated to supply the necessary heat for the e-SMR. Additionally, when compared to conventional SMR reactors, Meloni et al. (2022) also mentioned enhanced catalytic performance in an e-SMR setup due to direct heating of the catalyst. Moreover, electrification can eliminate the need for a combustion furnace, resulting in a significantly reduced reactor volume (Wismann et al., 2019).

To build an energy plant based around an e-SMR unit, a controller must be designed to maintain a stable hydrogen production rate, addressing challenges such as start-up procedures, catalyst deactivation, and setpoint modulation. At UCLA, the experimental setup for the e-SMR was constructed for e-SMR-based hydrogen production in an experimental process, and research work was conducted to achieve the desired hydrogen production using model predictive control (MPC) based on first-principle models, leading to a successful experimental implementation and desired control behavior (Çıtmacı et al., 2024a,b; Cui et al., 2024). While the integration of the first-principle model with the MPC controller effectively drives the output to the desired setpoint in previous studies, the predictions of the first-principle model may not fully reflect the complexities of real experimental processes. The discrepancies arise from assumptions made in constructing the first-principle model, which neglected crucial factors inherent in real reaction processes, such as mass transfer phenomena and catalyst deactivation. These uncertainties emphasize the necessity of developing alternative approaches to accurately simulate the real process. Therefore, a data-based approach for MPC implementation in the e-SMR process is investigated in this study.

With the advancements in computational power and the development of sophisticated algorithms in the era of big data, the evolution of machine learning has gained significant attention in the process modeling area. Over the past decades, the machine learning technique has evolved from a research-oriented area to a field with various real-world applications, such as healthcare, finance, manufacturing, etc (Sarker, 2021). Among various architectures of machine learning techniques, neural networks have emerged as one of the most powerful due to their ability to learn and model non-linear and complex data patterns. The common types of neural networks include feed-forward neural networks (FNNs), recurrent neural networks (RNNs), and convolutional neural networks (CNNs) (Wu et al., 2019b). Specifically, RNNs have gained significant popularity for modeling a broad class of nonlinear dynamical systems. The origins of the RNN model date back to the 1980s, marked by the invention of Hopfield networks for the purpose of pattern recognition (Hopfield, 1982). Since then, various types of RNNs have been extensively developed for applications such as pattern recognition and natural language processing. Today, with the rapid advancement of computational resources and the availability of open-source neural network libraries and frameworks like TensorFlow and Keras, RNNs have been leveraged as one of the most effective methods to solve regression and classification problems in engineering fields (Schmidhuber, 2015). For example, Wu et al. (2018) describes the development of a neural network-based detection system designed to identify cyberattacks in chemical processes. Zheng et al. (2022) used RNNs and other networks in an MPC scheme to improve the efficiency and quality of a cooling crystallization process. Xiao et al. (2021) introduced an RNN-based MPC framework for a plasma etching process.

Compared to feed-forward and other types of neural networks with single-directional connections between input and output vector, RNNs exhibit advantages due to their additional recurrent connections formed between sequential data, which preserve the memory of the previous layer. This characteristic allows RNNs to effectively process and learn the information between data points while accounting for time dependency, making them particularly suited for tasks such as natural language processing, time series prediction, and pattern recognition (Wu et al., 2019b). Moreover, the distinctive architecture leverages RNNs to capture the dynamic behavior of the target process in a way conceptually similar to the nonlinear state-space ordinary differential equation models, making them a valid alternative to the first-principle model (Miljanovic, 2012). Although a basic RNN model can be more effective for simulating time-dependent processes than other neural network models, their structural simplicity may be insufficient to accurately capture highly complex processes with long-term data dependencies. Today, various types of modern RNN-based process models have been developed and incorporated into MPC schemes to enable long-term control actions on sequential data. The common architectures of the modern RNN-based process model are long short-term memory (LSTM), gated-recurrent unit (GRU), and Encoder–decoder. LSTM and GRU use a similar gate mechanism to regulate information flow and manage the long-term time dependencies in sequential data. According to a study conducted by Zarzycki and Ławryńczuk (2021), an LSTM and GRU model accurately predicted process outputs and achieved desired control behaviors to a similar extent when integrated with an MPC scheme for controlling two chemical reactors. The Encoder–Decoder is a type of special RNN model incorporating two RNN models in series. Compared to traditional RNNs, the Encoder–Decoder RNN performs better in processing long-term sequential data by using a more flexible time window size of input and output (Cho et al., 2014). Moreover, a study conducted by Zhang et al. (2021a) found that the Encoder–Decoder outperforms LSTM-based and GRU-based RNNs in simulating dynamic processes that involve numerous long-term dependencies. Even though the Encoder–Decoder neural network model overcomes certain limitations of traditional RNN models, it also has a higher computational cost, while an LSTM-based RNN model is adequate at capturing most of the dynamic process behavior (Ren et al., 2022).

In the field of chemical engineering, machine learning techniques can also make significant contributions. In a study conducted by Çıtmacı et al. (2023), the LSTM-based RNN process model trained on experimental data was implemented into a multi-input-multi-output (MIMO) control scheme to regulate ethylene and carbon monoxide production in a rotating cylinder electrode reactor for carbon dioxide (CO_2) electrochemical reduction, achieving effective closed-loop control and approved the feasibility of RNNs in the chemical engineering context. Nevertheless, there has been limited research on using RNNs to simulate the e-SMR processes, despite their ability to accurately approximate complex dynamics and non-linearities without the knowledge of the underlying physics and chemistry principles behind the process. Thus, this area presents an opportunity for further exploration, which could make further advancements.

Based on these considerations, RNN-based model predictive control emerges as a promising approach for the e-SMR process. However, various unforeseen phenomena can occur during the e-SMR process. Notably, carbon formation (coking) is a prevalent issue that can deteriorate catalyst performance (Zhang et al., 2021b; Ginsburg et al., 2005; Ashik et al., 2017). These changes introduce disturbances to the actual e-SMR process, resulting in inaccuracies in the RNN model. Therefore, strategies designed to address disturbances in an e-SMR process MPC scheme are required.

To address disturbances in an MPC scheme, several approaches have been developed. One widely used method in early MPC algorithms is MPC combined with an integrator, which eliminates offset by adding extra control action derived from error integration (Yang et al., 2015). Wang et al. (2009) discussed embedding integrators within designed models, while Qin and Badgwell (2003) applied integrators in dynamic matrix control (DMC). The offset-free approach is a more recent innovation for eliminating disturbance-induced offset compared to conventional methods. Various disturbance models have been explored and incorporated into this strategy (e.g., Muske and

Badgwell (2002) and Pannocchia and Rawlings (2003)). Specifically, the offset-free strategy employs additional disturbance states and a disturbance observer to estimate and counteract disturbances (Maeder et al., 2009), thereby effectively addressing offsets. Beyond these general methods, online retraining using transfer learning offers another solution based on the properties of RNNs. This approach involves retraining the pre-trained RNN model with real-time data, adapting it to current conditions to manage offset (Wu et al., 2019a; Fekri et al., 2021). By continuously updating the RNN model with new data, the control system can effectively respond to disturbances and maintain desired performance levels. These methodologies handle disturbances within an MPC scheme sufficiently and can be applied to the e-SMR process.

In this study, we aim to explore the application of a machine learning-based model to replace the first-principle model within an MPC framework for process control. Specifically, an RNN model will be constructed based on the data generated from the first-principle model to capture the underlying pattern, which introduces a solid theoretical background of the actual e-SMR experiment. Subsequently, the RNN-based process model will be employed in an MPC scheme to estimate the initial condition of the controller and forecast the future state values of the process variables as a predictive model within time horizons. Considering disturbances in an e-SMR process, three approaches of RNN-based MPC schemes are emphasized for eliminating the final offset: MPC combined with an integrator, MPC with transfer learning, and offset-free MPC. These approaches will be analyzed based on their control performance. Our primary objective is to investigate the application of the RNN-based process models in the context of hydrogen production and control.

2. Preliminaries

2.1. Notations

The notation $\boldsymbol{x} \in \mathbb{R}^5$ represents the vector of state variables, involving flow rates with derivation form $(F_i - F_{sp})$ of methane (CH_4), carbon monoxide (CO), hydrogen, carbon dioxide and the averaged reactor temperature $(T - T_{sp})$. The notation $\boldsymbol{x}^\intercal$ represents the transpose of the vector $\boldsymbol{x}$. The notation $\hat{x}$ is the predicted vector of state variables. The notation $\bar{x}$ is the predicted vector of state variables combined with the error tracking terms $(\boldsymbol{\theta})$. The notation $\boldsymbol{u} \in \mathbb{R}^1$ represents the vector of the control action, involving the electric current with derivation form $(I - I_{sp})$. The notation $\boldsymbol{u}^\intercal$ represents the transpose of the vector $\boldsymbol{u}$. A function $f(\boldsymbol{x})$ belongs to the class C^1 if it exhibits continuous differentiability along its domain. $F(\cdot)$ is the model to be used to estimate state values. $\bar{F}(\cdot)$ is the modified model to be used to estimate state values. The notation $\hat{\boldsymbol{y}}$ represents the deviation form of the target output vector predicted by the model. The notation $\tilde{\boldsymbol{y}}$ represents the deviation form of measured target output.

2.2. Process overview

In our previous work, an electrically heated steam methane reformer was built to convert methane to hydrogen gas carbon emission-free from heating (Çıtmacı et al., 2024a). In the electrically heated SMR setup, methane, water steam, and argon (Ar) gas are fed to a tubular reactor under different temperatures and pressures to react and produce hydrogen gas. The overall chemical reactions can be written as follows:

$$\text{Steam methane reforming}: CH_4 + H_2O \rightleftharpoons 3H_2 + CO, \quad \Delta H_{298} = 206.1 \text{ kJ mol}^{-1} \tag{1a}$$

$$\text{Water gas shift}: CO + H_2O \rightleftharpoons CO_2 + H_2, \quad \Delta H_{298} = -41.15 \text{ kJ mol}^{-1} \tag{1b}$$

where steam methane reforming and water gas shift reactions are involved. Steam methane reforming is a strongly endothermic reaction that converts methane and water to hydrogen and carbon monoxide, and the water gas shift is a slightly exothermic reaction that converts the carbon monoxide and water to carbon dioxide and hydrogen gas. Large amounts of energy are needed to initialize the steam methane reforming reaction (Wei and Iglesia, 2004a,b,c,d), and a highly active Ni-based catalyst is used in the reactor under high reaction temperatures to reduce activation energies and increase the net SMR reaction rate.

The e-SMR process of interest outlined in Fig. 1 describes the transformation of methane and steam reactants to carbon monoxide, carbon dioxide, and hydrogen products. Though the experimental argon input is neglected in the RNN model developed in this study, the inert gas is used as a tracer in the experiments for volumetric flow rate monitoring. Temperature measurements are taken at the inlet and outlet of the reformer and are recorded on a per-second basis with a set of two thermocouples (TC). The average of these experimental temperature measurements is utilized for modeling the reactor temperature. A DC power supply sends electrical energy through the outer wall of the experimental reformer which generates the required energy input for the SMR reactions in the form of resistive-heat, also known as joule-heating. Temperature control in the reformer is achieved with a proportional–integral (PI) controller that modulates the electrical current input using a first-order algorithm that has been derived and tested by Çıtmacı et al. (2024a). A nickel-embedded zirconia washcoat catalyst resides along the inner walls of the reformer to lower the activation energies of the SMR and WGS reactions. The synthesis procedure for the washcoat was developed and modeled by Çıtmacı et al. (2024b). After the product gas mixture exits the reformer tube, the mixture enters a steam condenser that removes unreacted water vapor to prepare the gasses for analysis in a gas chromatography (GC) device. Mole fractions of each product species are measured in a TCD column as a means for quantifying the compositions of the gas products.

To capture the dynamic behavior of the e-SMR process, a lumped-parameter dynamic model was constructed in Çıtmacı et al. (2024a) based on the reaction kinetics developed in Xu and Froment (1989). SMR, WGS, and gas species adsorption kinetic parameters were experimentally validated in Çıtmacı et al. (2024a,b). Due to the relatively small scale of the experimental setup, the first-principle model is built based on a simplified modeling approach, which approximates the tubular reactor as a continuously stirred tank reactor (CSTR). In the case of a lumped parameter model, the first-principle model is built based on the mass balance of each chemical species to simulate a gas-phase CSTR. Due to the temperature dependency of the reaction rates, the energy balance was also employed in the first-principle model. Additionally, $Pq = FRT$ is held at all times to satisfy the ideal gas law within the flow reactor system, where P is the pressure, q is the volumetric flow rate, F is the molar flow rate, R is the universal gas constant and T is the reactor temperature. The details about the reaction kinetics and the mass balance equations can be found in Çıtmacı et al. (2024a).

Remark 1. The flow rate of each gas species exiting the reactor is measured using GC and quantified in the unit of standard cubic centimeters per minute (SCCM). Considering the analysis time and cooling period of the GC, an 18-min sampling interval and a 15-min delay are factored into the process. Consequently, the gas measurement is characterized by infrequent and delayed data acquisition.

2.3. Model predictive control

Model predictive control works by using a mathematical model of a system to predict its future behavior over a set time horizon. At each control step, MPC solves an optimization problem to determine the sequence of control actions that will minimize a predefined cost

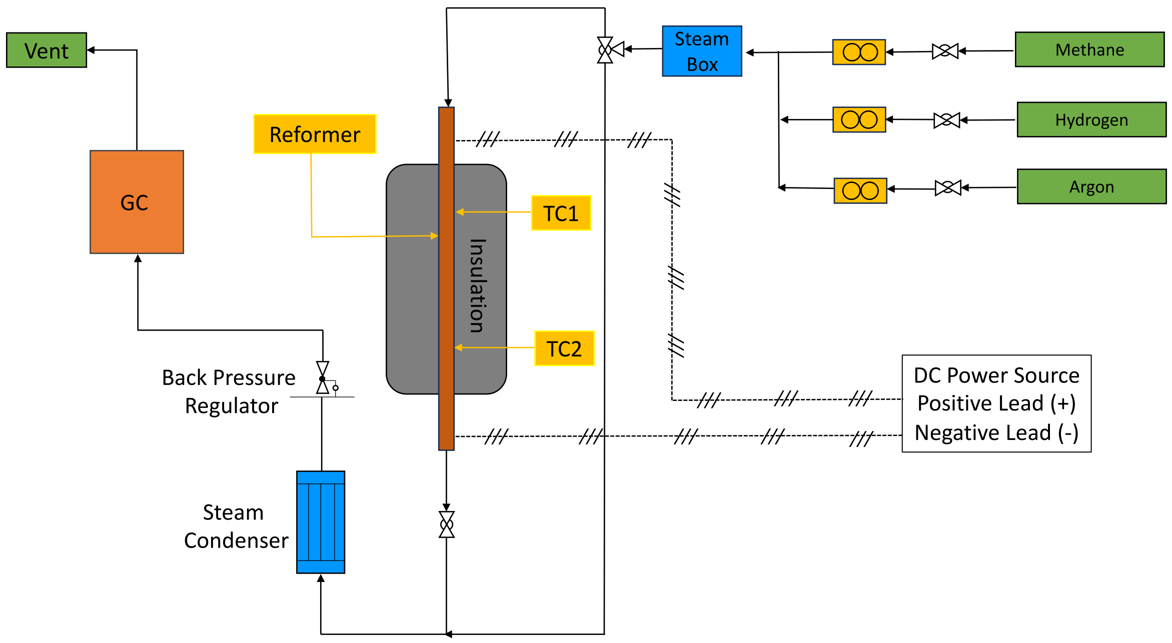


Fig. 1. Process schematic of the electrified steam methane reformer at UCLA.

function, which typically includes terms for tracking desired outputs and minimizing control efforts, while satisfying any constraints on inputs and outputs. Once the optimal sequence is determined, only the first control action is implemented, and the process is repeated at the next time step with updated system information, effectively moving the prediction horizon forward and continuously adjusting the control actions (Schwenzer et al., 2021). One form of the MPC mathematical formulation is represented by the following equations:

$$\mathcal{J} = \min_{\boldsymbol{u}} \int_{t_k}^{t_{k+N_h}} L(\hat{\boldsymbol{y}}(t), \boldsymbol{u}(t))\, \mathrm{d}t \tag{2a}$$

$$\text{s.t.}\ \ \dot{\hat{\boldsymbol{x}}}(t) = F(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) = f(\hat{\boldsymbol{x}}(t)) + g(\hat{\boldsymbol{x}}(t))\boldsymbol{u}(t),\ \ \hat{\boldsymbol{x}}(t_k) = \boldsymbol{x}(t_k) \tag{2b}$$

$$\hat{\boldsymbol{y}}(t) = h(\hat{\boldsymbol{x}}(t)) \tag{2c}$$

$$L(\hat{\boldsymbol{y}}(t), \boldsymbol{u}(t)) = \hat{\boldsymbol{y}}^{\intercal}(t) A \hat{\boldsymbol{y}}(t) + \boldsymbol{u}^{\intercal}(t) B \boldsymbol{u}(t) \tag{2d}$$

$$t \in [t_k, t_{k+N_h}) \tag{2e}$$

$$\|\boldsymbol{u}(t_k) - \boldsymbol{u}(t_{k-1})\| \le u_c \tag{2f}$$

$$\boldsymbol{u}(t) \in U \quad \forall t \in (t_k, t_{k+N_h}) \tag{2g}$$

where $\boldsymbol{u}$ is the control input vector, which is also a variable in the optimization problem, N_h is the horizon length, L is the objective function to be optimized, which measures the difference between the setpoint and controlled output prediction over the horizons, $\hat{\boldsymbol{x}}$ is the predicted vector of all state variables by the model, $\hat{\boldsymbol{y}}$ is the target output vector predicted by the model and $F(\cdot)$ is the model to be used to estimate state values over the horizon. A and B are positive definite weight matrices for the output target values and manipulated control input, respectively. The objective of the MPC is to find the optimal control input to minimize the difference between the model predictions and setpoint. It is crucial to emphasize that this optimization problem is solved under the rate of change and magnitude constraints on the control input vector (Eqs. (2f), (2g)). Specifically, Eq. (2f) is employed to ensure that the control input change remains within a practically desired range, while Eq. (2g) is implemented to ensure the calculated control action values adhere to specific limits.

2.4. Offset-free model predictive control

Offset-free model predictive control is a control strategy that aims to regulate the output from the system to a desired reference while simultaneously estimating and compensating for unknown disturbances or offsets. Compared with the traditional MPC scheme, an additional term (θ) is incorporated to augment the system model by tracking the accumulation of the error between the real data and estimated values, which solves the steady-state errors resulting from model-plant mismatch or disturbances. Hence, the model is modified by this error accumulation term as shown below (Maeder et al., 2009; Wallace et al., 2016):

$$\dot{\hat{\boldsymbol{x}}}(t) = F(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) + G_\theta \boldsymbol{\theta}(t) \tag{3a}$$

$$\dot{\boldsymbol{\theta}}(t) = 0 \tag{3b}$$

where $\boldsymbol{\theta}$ is the error accumulation term with its corresponding coefficient, G_θ. This augmented model can be further written as follows:

$$\dot{\bar{\boldsymbol{x}}}(t) = \bar{F}(\bar{\boldsymbol{x}}(t), \boldsymbol{u}(t)) \tag{4a}$$

$$\dot{\bar{\boldsymbol{x}}}(t) = \begin{bmatrix} \dot{\hat{\boldsymbol{x}}}(t) \\ \dot{\boldsymbol{\theta}} \end{bmatrix} \tag{4b}$$

$$\bar{F}(\bar{\boldsymbol{x}}(t), \boldsymbol{u}(t)) = \begin{bmatrix} F(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) + G_\theta \boldsymbol{\theta}(t) \\ 0 \end{bmatrix} \tag{4c}$$

This modification subsequently utilizes the updated model in the control scheme. To estimate the current augmented term in real time and improve the estimation of other state variables in the state vector, a Luenberger observer is employed.

$$\dot{\bar{\boldsymbol{x}}}(t) = \bar{F}(\bar{\boldsymbol{x}}(t), \boldsymbol{u}(t)) + K\left[\tilde{\boldsymbol{y}}(t_n) - \bar{\boldsymbol{y}}(t_n)\right] \tag{5a}$$

$$K = \begin{bmatrix} K_y \\ K_\theta \end{bmatrix} \tag{5b}$$

where K is the gain matrix of the Luenberger observer, $\tilde{\boldsymbol{y}}$ is the measured output, $\hat{\boldsymbol{y}}$ is the estimated output, and t_n is the measurement time instant. A constant error between measurement and estimation is assumed for the interval between two consecutive measurements, and the $\boldsymbol{\theta}$ can be deemed as the integral of this error. Consequently, the model undergoes continuous correction until no mismatch between measurement and estimation is achieved. The estimated state vector is utilized for the initial values of the predictions in the MPC. In the present work, the offset-free MPC concept is combined with the use of neural network models in MPC and it is applied to the electrified SMR model under disturbances in a later section.

3. Recurrent neural network model

The recurrent neural network is a type of machine learning-based process model that has been widely applied for modeling nonlinear dynamic processes. Unlike other neural network (NN) process models, the RNN process model can memorize the information from a previous input in the sequence and predict the output based on the ordinal correlation between different time steps. Due to this advantage, the RNN process models are more suitable to simulate dynamic processes involving time-dependent data compared to other NN models. Furthermore, the feedback loop in the RNN models that enables the capture of the model's dynamic behavior over time is similar to how non-linear first-principle models describe the process behavior (Ren et al., 2022). Therefore, the RNN models are better options than other NN process models for simulating time-dependent dynamic processes. In our study, an LSTM-based RNN process model is built to capture the behavior of a first-principle-based model built in our prior work (Çıtmacı et al., 2024a), which is a model constructed to simulate the dynamic behavior of the e-SMR process. Additionally, the LSTM-based RNN process model is used as an initial condition estimator for the MPC controller, and also as a predicting model in the MPC schemes to predict the flow rates of hydrogen products within prediction horizons based on previously known reaction states, which will be discussed in later chapters. In this section, the architecture of an LSTM-based RNN process model is discussed in detail in the following order: Section 3.1 describes the data generation and preprocessing, Section 3.2 describes model construction and hyperparameter tuning, Section 3.3 describes the model training and evaluation process, and Section 3.4 describes the transfer learning method.

3.1. Data generation and preprocessing

3.1.1. Data generation

In our previous study published in Çıtmacı et al. (2024a), a first-principle-based process model was built based on chemical engineering fundamental principles that included a mole balance and energy balance around a CSTR control volume. In this work, the first-principle-based process model was used to simulate the real experimental process in our MPC scheme. Thus, the data used to train, validate, and test the RNN model is generated from the open-loop simulation using the first-principle model. Specifically, the first-principle model generates reaction states at each time step within a defined time interval, starting from different initial conditions. Subsequently, the data is transformed into the desired format to train the RNN model. To consider different scenarios and improve the generalization of the model, a wide range of initial conditions (10,000 initial conditions of 7 input variables: concentrations of CH_4, CO, CO_2, H_2O, H_2, Ar, and the reactor temperature (T)) are applied to the first-principle model to conduct the open-loop simulations for twenty seconds. These results are collected as the dynamic behavior of the simulated experimental process starting at different initial conditions. Several limits and bounds for each input variable were applied to the data generalization process to make the 10,000 initial conditions conceptually reasonable according to the understanding of the e-SMR process principles. For example, the ideal gas law principle and its derivative correlation between the mole fraction and the concentration of each chemical species involved in the reactor are used to determine the initial concentrations of each chemical species, as shown in Eq. (6) below:

$$\frac{P}{R \cdot T} = C_{\text{total}} \tag{6a}$$

$$C_i = C_{\text{total}} \cdot X_i \quad \text{where } i = \mathrm{CH_4, CO, CO_2, H_2O, H_2, Ar} \tag{6b}$$

$$\sum X_i = 1 \quad \text{where } i = \mathrm{CH_4, CO, CO_2, H_2O, H_2, Ar} \tag{6c}$$

where P, n, and T are the pressure, number of moles, and temperature within the reactor. C_i is the concentration of the species i, and X_i is the mole fraction of the species i. The mole fraction of each species takes value between 0 and 1. The initial conditions of the electric current are 24 and 31 A, and the initial temperature conditions are constrained between 482 °C and 743 °C. According to Eq. (6), the sum of the initial concentrations of each chemical species is equal to the total concentration, which requires the sum of the fraction of each chemical species to be 1 to obey the ideal gas law principle. Moreover, the first-principle process model utilizes concentrations as variables whereas the RNN model uses flow rates as variables. The RNN model is structured to accommodate flow rate variables as both inputs and outputs, aligning with the experimental process where flow rates serve as data points. Additionally, the control system relies on species flow rate variables as a control objective to achieve the system behavior. The first principle model utilizes concentrations as variables because they are directly involved in the mole balance equations, which are foundational for constructing an accurate model. Thus, the concentration variables generated from the first-principle model are converted to flow rates before being applied to the RNN model. This conversion from concentration to flow rates involves another important factor that affects the quality of training data for the RNN: the volumetric flow rate. The equation of volumetric flow rate (q) is as follows:

$$q = \frac{F_{T0} + 2r_1 W}{\frac{P}{RT}} + \frac{V_R}{T}\frac{\mathrm{d}T}{\mathrm{d}t} \tag{7}$$

where F_{T0} is the total inlet molar flow rate, r_1 is the reaction rate of Eq. (1a), V_R is the reactor volume. The details of the equation can be found in Cui et al. (2024). Utilizing this calculated q, the flow rate of each species can be obtained by Eq. (8):

$$F_i = qC_i \quad \text{where } i = \mathrm{CH_4, CO, CO_2, H_2O, H_2, Ar} \tag{8}$$

where F_i is the flow rate of species i. Based on the volumetric flow rate equation (Eq. (7)), an additional condition requiring the calculated volumetric flow rate to be positive is applied to the initial condition-generating process. The open-loop simulation data-generation process yields 200,000 data points of the flow rates of CH_4, CO, CO_2, H_2O, H_2, Ar, reactor temperature, and electric current values.

3.1.2. Data preprocessing

For a sequential forecasting task, the data must be processed in the model in batches rather than single sequences (Ren et al., 2022). The sliding window technique is employed to separate and extract the RNN input and output data from the 200,000 data points generated from the first-principle model and arrange them into the desired dimensions for the training process. The sliding function is a data processing technique that is widely used for preprocessing time-dependent data sets. In our case, the window size is 10, which is set to equal the time length of the RNN input, and the step size is 1. Specifically, the sliding-window technique is applied to the dataset by shifting a ten-second time window one-time step at a time across the 200,000 data points collected from the first-principle-based process model. The fixed time window of ten seconds allows the sliding window technique to extract the reaction states from i seconds to $i + 9$ s time step as a ten-second scale RNN input data, and the reaction states at $i + 10$ s time step as a one-second scale output data, where $i \in \{0, 1, 2, 3, \ldots, n - 11\}$ for n data. Through this approach, the data points are separated into the size of (10^5, 10, 6) as an RNN input and (10^5, 1, 5) as an output, where the first index refers to the number of ten-second scale RNN input data sets generated using the sliding window technique, the second index represents the size of the time window, and the third index represents the number of RNN input/output variables. Following this data arrangement, the model will learn to predict the state variables at the immediate time step based on the known data from the previous ten seconds as the RNN input.

When constructing an RNN model, it is essential to separate the dataset into training and testing sets. Additionally, a separate validation set is recommended for tuning hyperparameters to optimize model performance. In our study, the data is split into training, validation, and testing datasets by the train-testing split technique. A split ratio of 70/15/15 gives the RNN model 70,000 training inputs/outputs to update the weights and biases, 15,000 validation inputs/outputs to adjust the hyperparameter of the model, and 15,000 testing inputs/outputs to evaluate the model performance on unseen data. The data split helps to prevent data leakage, which can occur when information outside of the training data, such as mean or standard deviation, influences the training process and eventually results in a false reflection of the model performance on unseen data (Ren et al., 2022).

Data normalization is an essential preprocessing step that transforms data to a reasonable scale, enhancing the quality of the training process. Variations in the dataset can significantly disrupt the learning process, leading to poor model performance due to exploding or vanishing gradients. Normalization mitigates these variations, stabilizing gradients, improving model generalization, and increasing the convergence rate during optimization. In our case, the dataset is normalized using the Min–Max scaling method, which scales the data between 0 and 1 according to the following equation:

$$z_{normalzied} = \frac{z - z_{min}}{z_{max} - z_{min}}(f_{max} - f_{min}) + f_{min} \tag{9}$$

where z_{max} is the maximum value within the dataset, z_{min} is the minimum value within the dataset, z is the target data point, f_{max} is the user-defined maximum value of the data after scaling, and f_{min} is the user-defined minimum value of the data after scaling. In our case, f_{max} is 1 and f_{min} is 0. The Min–Max scaler is fitted to the training dataset and subsequently applied to transform the training, testing, and validation datasets. This process scales the original data to a range between 0 and 1, eventually improving the training process while preserving the original data distribution.

Remark 2. The data preprocessing process in our study remains simple due to the high quality of data generated by the validated first-principle model. In scenarios where a first-principle model is unavailable and noisy experimental data must be utilized, additional data processing techniques may be necessary to combat the noise and improve the training efficiency.

Remark 3. It is important to point out that the available experimental data are insufficient for directly training an RNN model. Specifically, the flow rates of the chemical species, which serve as the training data for the RNN construction, are recorded only once every 18 minutes. Although polynomial functions are employed to estimate additional data points between these measurements, the resulting dataset remains insufficient compared to the extensive simulation data required to train an RNN model accurately. Thus, the approach of using a first-principles model for parameter estimation and subsequently developing the RNN model in an MPC scheme is utilized to address the need for extensive data input in building the RNN model.

3.2. Model construction and hyperparameter tuning

3.2.1. Model structure

Depending on the complexity of the dynamic behavior of the process to capture, different neural network models may be applied. Among neural networks, the FNN model is the simplest neural network model that computes the weights and outputs of neurons by propagating the information in one direction from input to output. The simple learning algorithm and basic architecture make it easy to construct and implement FNNs; however, this type of neural network is limited to capturing complex dynamic behavior involving ordinal datasets. Compared to the FNN model, the RNN model computes the sequence of hidden states and updates the weights by including time as an additional factor to account for the time-dependent property. In other words, the RNN model is considered a two-dimensional model that includes time as an additional factor while the FNN model is one-dimensional (Ren et al., 2022). Like any other type of neural network model, the learning algorithm behind the RNN model consists of two steps: the forward pass and the backward pass. In the forward pass, the RNN model regulates the flow of information between layers, between different time steps, and passes the information down to the output layer to make a final prediction. In the backward pass, the RNN model back propagates the difference between the predicted output and the true value of the target to update weight parameters. For this reason, the RNN model has been a popular option to study time series data and is also suitable for this study. However, as the sequence length of input data increases and the weight matrices become larger in the standard RNN model, the output may diverge, leading to either extremely large (gradient exploding) or small (gradient vanishing) gradient estimations during backpropagation. One solution to solve this issue is to truncate the gradient at certain time steps to prevent large matrix computations and divergent eigenvalues, yet this may result in losing information from earlier steps (Ren et al., 2022). The LSTM-based RNN model is a special type of RNN model that replaces the normal recurrent units with LSTM units to solve the gradient vanishing/exploding issue by utilizing an additional cell state and achieve better performance compared to standard RNN models (Çıtmacı et al., 2023). According to Zarzycki and Ławryńczuk (2021), the LSTM-based RNN model performed better than a standard RNN model as a process model in the MPC scheme to stimulate the experiment process. Thus, an LSTM-based RNN model is used in our study to capture the nonlinear dynamic behavior of the first-principle model. In this section, the construction, learning algorithm, and hyperparameter tuning of an LSTM-based RNN model are explained.

The RNN process model used in our study is constructed by four layers: an input layer, an LSTM layer, an output layer, and a reshape layer, as shown in Fig. 2. The input layer serves as the entry of the RNN model, accepting sequential data in the form of time-dependent inputs. The size of our input data is (10,6), representing six input variables (I, T, F_{CH_4}, F_{H_2}, F_{CO_2}, and F_{CO}) over ten seconds. The LSTM layer is composed of 180 units to capture the long-term time dependencies of the sequential data. The reason for using multiple LSTM units in our study is to capture long-term data dependencies more effectively. Using a simple LSTM unit might result in under-fitting, providing insufficient capacity to learn the complex patterns underlying the data. Thus, using multiple LSTM units helps prevent under-fitting during model training, thereby improving overall model performance and achieving a more accurate simulation of the process dynamics. The selection of the number of neurons will be discussed in detail in Section 3.2.2. After the LSTM layer, the dense layer receives all the active neurons from the output of the LSTM layer and transforms them into the final outputs. Eventually, the reshape layer transforms the final output into the desired data size (1,5), representing five output variables including T, F_{CH_4}, F_{H_2}, F_{CO_2}, and F_{CO} at one-time step.

Among these layers, the LSTM layer is the key factor in capturing the time-dependent data pattern (Luo et al., 2023). LSTM units are designed to overcome the limitations of traditional RNN models, such as the vanishing and exploding gradient problems (Hochreiter and Schmidhuber, 1997). Compared to the basic recurrent units, the LSTM unit uses an additional memory cell state, a candidate cell state, and a gate mechanism involving three gates (forget gate, input gate, and output gate) to control the flow of information and update the parameters between different time steps (Wu et al., 2019b). A structure diagram of the LSTM unit showing the correlation between states and gates can be seen in Fig. 3. In the LSTM unit, the memory cell state and candidate cell state help to save the relevant information and allow the

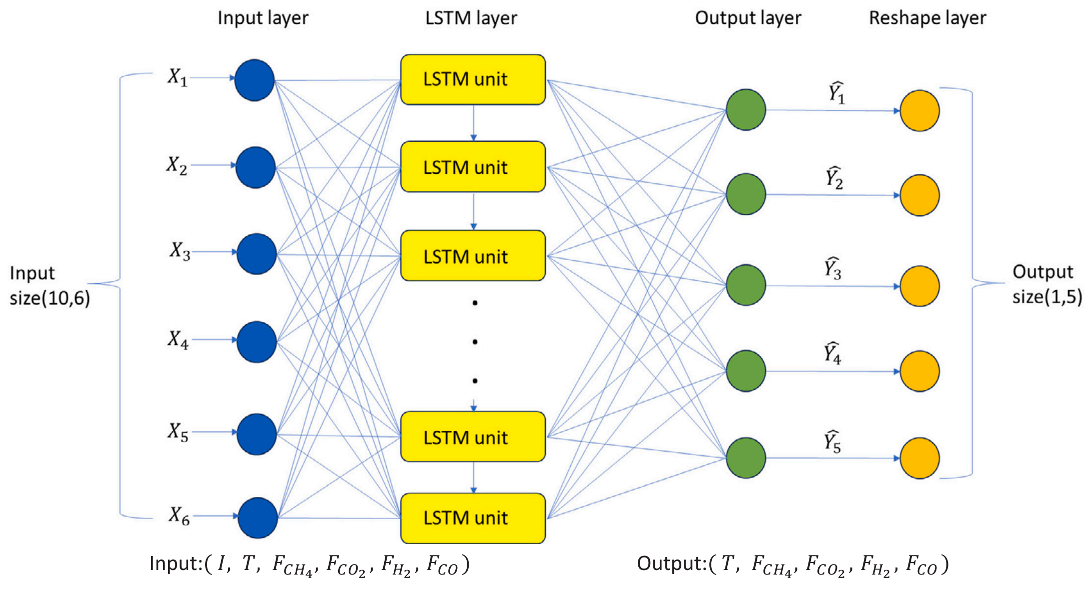


Fig. 2. The structure of the RNN model.

LSTM layers to learn long short-term dependencies in sequential data more efficiently than traditional recurrent units. The gate mechanism involving the forget, input and output gate is also crucial in the LSTM unit to determine the extent of the information to memory or discard. According to Fig. 3, the C_{t-1}^l and H_{t-1}^l are the memory cell state and hidden state from the previous time step, X_t, C_t^l and H_t^l are the input, cell state, and hidden state at the current time step. For a better understanding of concepts, the math expression of the memory cell state C_t, candidate cell state, and hidden state of the LSTM unit can be shown as follows:

$$\hat{C}_t = \tanh(X_t W_x + H_{t-1}^l W_h + b_c) \tag{10a}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t \tag{10b}$$

$$H_t = o_t \cdot \tanh(C_t) \tag{10c}$$

where W_c and b_c are the weight matrix and the bias term of the candidate cell state. Based on Fig. 3 and Eq. (10), the candidate cell state $\hat{C}_t$ is determined by applying a hyperbolic tangent activation function (tanh) to the weighted sum of the RNN input vector at current time step X_t, the hidden state at previous time step H_{t-1} and the candidate cell-corresponded biased term b_c. The cell state at the current time steps C_t is computed based on the cell state at the previous time step C_{t-1} and $\hat{C}_t$, which is the candidate cell state at the current time step. Additionally, f_t is the forget gate, and i_t is the input gate. The hidden state at the current time step H_t is computed based on the current cell state C_t and o_t output gate. The hyperbolic tangent activation function used to compute the candidate cell state and hidden state will return a value between −1 and 1, which helps to stabilize the training process and also captures both positive (increasing) and negative (decreasing) changes from the current RNN input and previous hidden states. Unlike the cell states and hidden states, math formulations of the forget, input, and output gates use a sigmoid activation and incorporate the current RNN input vector and previous hidden states according to the equation:

$$f_t = \sigma(X_t W_{xf} + H_{t-1}^l W_{hf} + b_f) \tag{11a}$$

$$i_t = \sigma(X_t W_{xi} + H_{t-1}^l W_{hi} + b_i) \tag{11b}$$

$$o_t = \sigma(X_t W_{xo} + H_{t-1}^l W_{ho} + b_o) \tag{11c}$$

where f_t, i_t, and o_t are the forget gate, input gate, and output gate. According to Fig. 3 and Eq. (11), the forget gate is determined by the weighted sum of the current input vector X_t and previous hidden states H_{t-1}, where each is multiplied by their corresponding weight parameters W_i, and is supplemented by the bias term b_i, then passed through a sigmoid activation function, which eventually returns a value between 0 and 1. The forget gate determines how much information is kept or discarded from the previous cell state. The input gate and output gate share the same math formulation as the forget gate which involves the current RNN input vector, previous hidden states, corresponding weights, and biases. Different from the forget gate, the input gate decides how much new information from the current candidate cell state should be added to the cell state, whereas the output gate decides how much of the information restored in the cell state should be passed to the next hidden states. Depending on the value returned by the sigmoid function, each gate decides how much of the information will be discarded or passed down, where 0 indicates all the information will be discarded and 1 means all the information will be kept and passed to the next step.

Remark 4. In our study, a simple RNN structure with one LSTM layer is used to adequately capture the relatively simple behavior of the process. Typically, the number of layers required in an RNN depends on the complexity of the target process. For a more complex process, more than one layer might be necessary to accurately capture the underlying dynamics. While increasing the number of layers can improve the model performance based on large datasets, it also raises the risk of over-fitting, where the model learns the training data too well and performs poorly on new data. Using fewer layers, on the other hand, may result in poor model performance which is incapable of capturing the actual correlation between data points. Thus, finding the optimal balance in the number of layers is crucial for achieving strong model performance and preventing over-fitting.

Remark 5. The dropout layer, which randomly deactivates neurons to improve model performance on unseen data, is a common regularization technique to prevent over-fitting (Srivastava et al., 2014). In our study, various RNN models were tested with different dropout rates. The best model performance was observed with a dropout rate of zero, resulting in a loss function error at the scale of 10^{-8}. This minimal error suggests that the model learns more effectively without dropout. Therefore, the dropout layer was not utilized in our model. In scenarios where there are high levels of data noise, the dropout layer may be crucial to be utilized in the model.

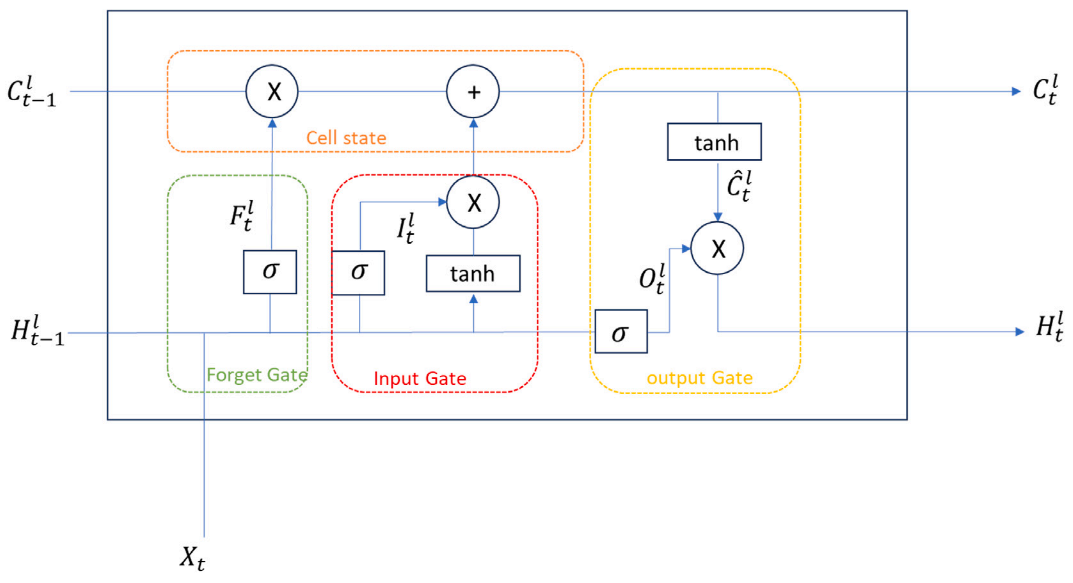


Fig. 3. The structure of an LSTM unit.

3.2.2. *Hyperparameter tuning*

Hyperparameter tuning is another important step in constructing the architecture of the RNN model. Hyperparameters are the parameter settings defined by users in the machine learning model. For example, the number of layers, neurons, epochs, type of optimizer, batch size, and many other parameters are used to build the model. Searching for the optimal hyperparameter is crucial in improving the model performance. Common methods for hyperparameter tuning include random search, grid search, Bayesian optimization, etc. (Feurer and Hutter, 2019). Grid search and random search are two common hyperparameter tuning approaches. Grid search finds the optimal hyperparameter by evaluating the model performance based on all combinations of possible hyperparameters within a user-defined region and saving the one with the best performance. Random search, on the other hand, evaluates the model based on a few randomly selected combinations of hyperparameters. According to Bergstra and Bengio (2012), random search is more efficient than grid search because it prioritizes tuning hyperparameters that have a greater impact on model performance, optimizing the search process by focusing on key parameters rather than all possible combinations. Thus, the random search method is used to determine the number of parameters in this study due to the advantage of its lower computational power demands and higher efficiency compared to other methods. In this section, various hyperparameters and their approach are discussed.

The number of neurons is critical in determining the learning capacity and complexity of the model, where more neurons yield a model that can capture more complex processes, yet require more computational time and power and also may encounter the issue of vanishing gradient (Ren et al., 2022). Conversely, fewer neurons simplify the model, requiring less computation yet possibly limiting its ability to capture complex patterns. In the LSTM layer, 180 neurons are used to capture the time dependencies of the data, which is determined by the random search method. According to Table 1, models containing 64, 128, 180, and 200 neurons are constructed, and the mean squared error (MSE) of training and testing data are compared. The model with the smallest training and testing MSE (3.39×10^{-8} and 3.74×10^{-8}, respectively) is found with 180 neurons, which is decided to be the optimal unit number used in the LSTM layer in our model.

In the output layer, six dense units are used corresponding to the number of output variables. The hyperbolic tangent is used as the activation function in the LSTM layer to regulate the cell state updates. The sigmoid activation function is used as the activation function in the dense layer to interpret the RNN input in terms of probability, and eventually transform the probability into meaningful output. The number of epochs used in the RNN model refers to the number of iterations of a complete training process through the entire data set. One epoch indicates the model has learned every training data point one time. The model can be evaluated through epochs based on validation loss, or MSE value, to determine whether the model is still improving. Since the EarlyStop function is used in our model, the learning process will end early when the training MSE stops improving for 15 epochs. In our case, the maximum number of epochs used is 100. The batch size refers to the number of data samples processed by RNN to make predictions and update weights in one forward and backward pass of the model. Typically, a small batch size provides more frequent updates to the weights but results in noisier gradient estimates. In contrast, a large batch size updates the weights less frequently but yields more accurate gradient estimates. In our case, the default batch size for TensorFlow/Keras (32 samples) is used. The number of layers, neurons, mini-batch, and epochs are determined by the random search method.

Table 1
Testing and training MSE for different neurons.

Number of neurons	Training mean squared error	Validation mean squared Error
64	1.62×10^{-7}	1.69×10^{-7}
128	4.59×10^{-8}	4.89×10^{-8}
180	3.39×10^{-8}	3.74×10^{-8}
200	4.53×10^{-8}	5.10×10^{-8}

The optimizer plays an important role in the model structure as a hyperparameter. It updates the weights in the neural network to minimize the value of the loss function in the training process. Common optimizers include stochastic gradient descent (SGD), RMSprop, and Adam (Ren et al., 2022). Adam optimizer is a type of optimizer that combines the advantages of SGD and RMSprop optimizer, enabling it to adapt the learning rate for different parameters based on the gradient and also momentum to smooth the optimization process (Kingma and Ba, 2014). Thus, Adam optimizes the process more robustly and efficiently compared to other optimization methods. In our study, we used the Adam optimizer, shown in the equation form:

$$m_t = \beta m_{t-1} + (1-\beta)\nabla E \tag{12a}$$

$$v = \gamma v + (1-\gamma)\nabla^2 E \tag{12b}$$

$$\omega = \omega - \frac{\alpha}{\sqrt{v+\epsilon}} m \tag{12c}$$

where ω is the weight, v is the velocity term, m is the momentum of the gradient, γ and β are the hyperparameters of the momentum and decay rate, ϵ is a coefficient (10^{-8}), α is the learning rate, and E is the cost function. According to equation Eq. (12), the momentum improves the convergence rate in the learning process by accelerating the gradient vector in the correct direction based on the past gradients. The velocity term introduces the ability of the adaptive learning rate to adjust the weight-updating process according to the magnitude of the gradient by incorporating a squared gradient. Thus, the Adam optimizer can find the optimal learning rate according to past gradient estimation while conducting a faster convergence.

Remark 6. The learning rate refers to the step size to update the weights in the direction of gradients in each epoch during optimization. A small learning rate will slow the convergence process, while a large learning rate might skip the optimal points. Therefore, Adam optimizer can be a good option to find the optimal learning rate based on past gradient estimation which is affected by the batch size, due to its adaptive learning rate characteristic.

3.3. Model training and evaluation process

After preprocessing the data and determining the right model architecture, the training process of the RNN model proceeds to learn the underlying pattern of the training data. As previously mentioned, the RNN model captures the information from training data and passes them down layer by layer to make the predictions in the forward pass and update the weights to minimize the error in the backward pass during the training process. With the validation data used in the training process, the model performance on the unseen data can be monitored during the training process. This validation data evaluation can help to determine whether the model is over-fit or under-fit based on the validation loss, which can also guide the adjustment of the hyperparameters. In the training process, we used 70,000 input/output, 15,000 input/output, and 15,000 input/output as training data, validation data, and testing data, respectively.

To evaluate the model performance and guide the weight updating in the training process, the loss function is applied in the RNN model to determine the error of the training model in terms of the average squared difference between the model predictions and actual results. During the training process, the loss function is minimized to improve the model accuracy by updating the weights in the opposite direction of the gradient in the backpropagation step. The common types of loss functions are mean absolute error (MAE), mean squared logarithmic error (MSLE), and mean square error (MSE) (Ren et al., 2022). In our model, the MSE loss function is used to evaluate and improve model performance based on training and validation data, where the MSE loss function could be shown in the equation:

$$MSE = \frac{1}{N_t} \sum_{i=1}^{N_t} \left(y_{(i)} - \hat{y}_{(i)} \right)^2 \tag{13}$$

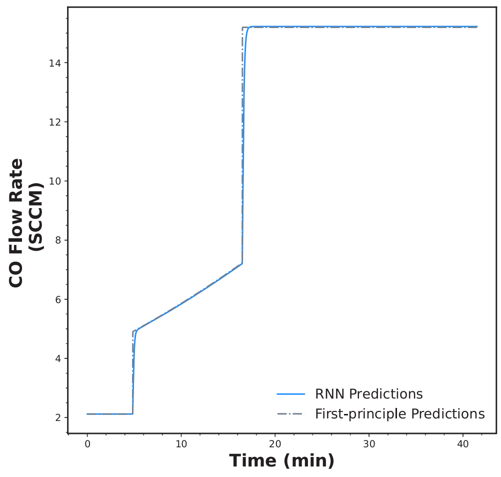
where N_t is the number of the training data, $y_{(i)}$ is the value of the training data, and $\hat{y}_{(i)}$ is the predicted value of the training data. MSE is a common type of loss function for regression tasks. Like other loss functions, the MSE loss function computes the difference between the predicted value and the actual value in the forward pass and minimizes the difference by adjusting the weights according to the gradient of the loss function in the backward pass.

During the training process, the training MSE loss decreases as more epochs pass. The MSE loss function also plays an important role in solving over-fitting and saving the best model by incorporating EarlyStopping and ModelCheckpoint functionalities. The over-fitting issue occurs when the training MSE loss decreases yet the validation MSE loss increases during the training process, indicating poor model performance on unseen data. EarlyStopping is a technique that monitors the validation MSE as a metric and stops the training process once it does not improve. Our training process stopped at 77 epochs, with no MSE loss improvement from the previous 15 epochs. ModelCheckpoint is a technique that records the model during the training process and saves the best model based on metrics. In our case, the ModelCheckpoint saves the best model after the MSE validation loss ceases to improve.

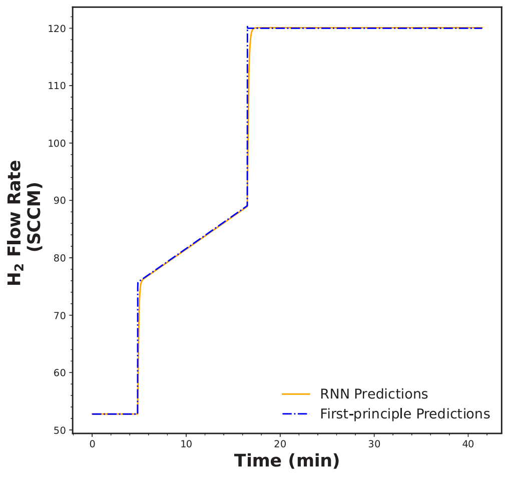
To prevent over-fitting, L_1 and L_2 regularization techniques are commonly applied in training neural network models (Salehin and Kang, 2023). Both L_1 and L_2 techniques effectively solve the over-fitting issue by applying different L_p norms to penalize the cost function and reduce the large weight parameter. L_1 regularization typically eliminates very small weights making the weight matrix more sparse, while L_2 regularization adds a penalty term to the cost function to prevent large weights (outliers) from dominating (Ren et al., 2022). L_2 regularization is typically more effective with larger datasets, maintaining model complexity and improving generalization without eliminating any values. In our study, we applied various L_2 parameters within the LSTM layer using a random search method to evaluate its effectiveness. The best model performance is achieved with a L_2 regularization parameter set to 0, indicating a sufficient model performance without the need for additional regularization in our study. This outcome is attributed to the high quality and abundance of data in our training process. In scenarios where data is limited in amounts and highly affected by noise, additional regularization techniques may be considered to prevent over-fitting.

The training and validation MSE are 3.39×10^{-8} and 3.74×10^{-8}, respectively. The similarity between the tiny errors of training and testing data exhibits the successful training process and good model performance on the unseen data. More testing data for each RNN input variable (I, T, F_{CH_4}, F_{H_2}, F_{CO_2}, and F_{CO}) generated in open-loop simulations using the first-principle model to build step change, ramp change, and sinusoidal data trends, are employed to further evaluate the model performance. The comparisons between the RNN and first-principle predictions in a closed-loop simulation are shown in Figs. 4 and 5. In the beginning, the RNN model is initiated by the testing data. For each time step, the flow rate of each gas species and temperature obtained from the RNN output is fed back to the RNN model. The electric current is updated using the testing data at each time step. In Figs. 4 and 5, the RNN model is evaluated based on step change, ramp change trends, and sinusoidal type data. The close alignment between the RNN predictions and the reference value illustrates the high accuracy of the model. Small differences between RNN predictions and first-principle predictions can be observed at step changes in Fig. 4. For a step change, the first-principle prediction reaches the steady state immediately, whereas the RNN prediction takes longer to reach the steady-state conditions. The discrepancy between the RNN and first-principle model predictions arises from the designated input shape of the RNN model and the approach of data extraction in the data preprocessing process. Since RNN and first-principle predictions closely align, with small averaged relative errors (0.32% for Fig. 4 and 1.05% for Fig. 5), the difference in the model errors is negligible.

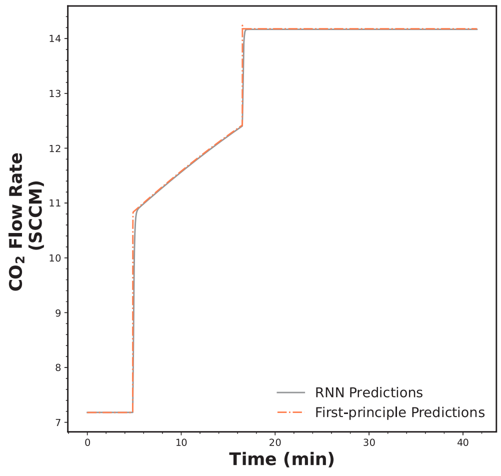
To test the feasibility of this RNN model in the MPC scheme, the prediction of the RNN was compared to the first-principle model in a scenario of the MPC scheme. In Fig. 6, the RNN predictions of each output variable are compared to the first-principle predictions within two prediction horizons with a 5-s sampling time to align with the scope and duration of the control actions in our MPC scheme. Also, the electric current in the RNN input is increased by 0.01 A (maximum limit change per control action) at the beginning of the second prediction horizon to simulate the condition of the electric current as a control variable in our MPC scheme. Based on Fig. 6, the comparisons of the flow rates of CH_4, H_2, CO_2, CO, and temperature are made between two models. The close alignments between the RNN predictions and the first-principle predictions demonstrate the reliability of the RNN



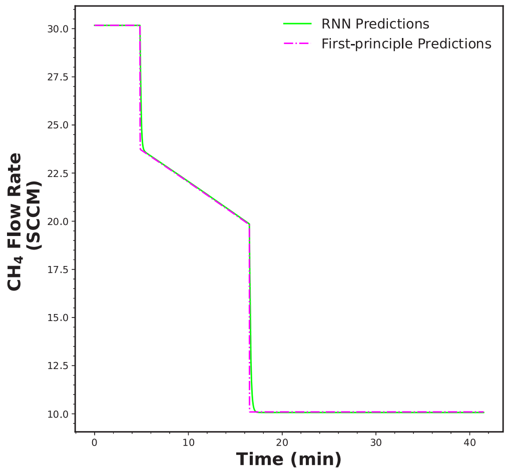
(a) Flow rates of CO.



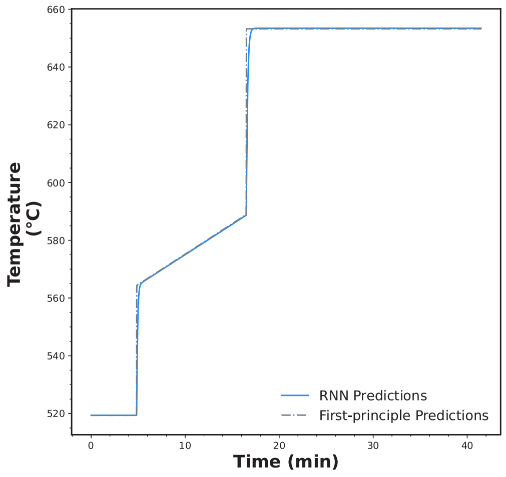
(b) Flow rates of H_2.



(c) Flow rates of CO_2.

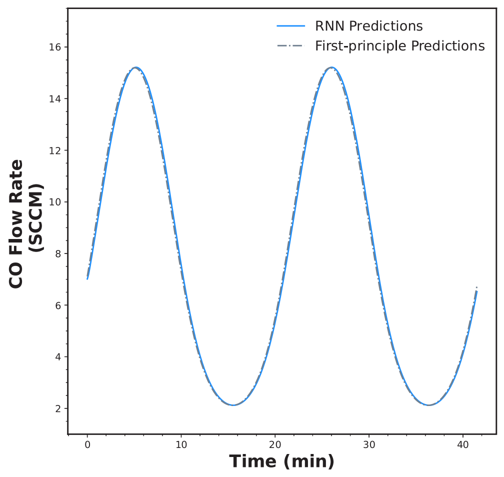


(d) Flow rates of CH_4.

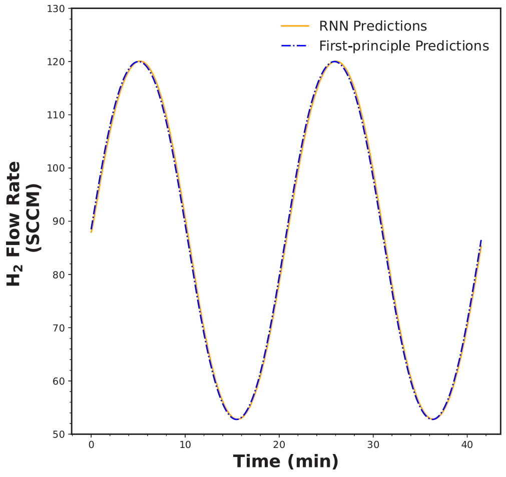


(e) Temperature.

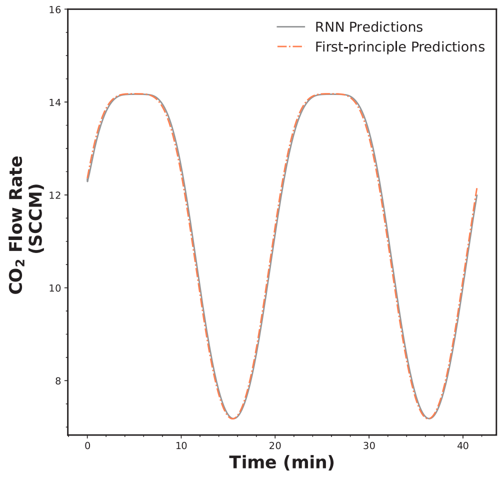
Fig. 4. RNN model predictions of the flow rates of CH_4, H_2O, H_2, CO and temperature compared to first-principle model predictions based on step change and ramp change reference data.



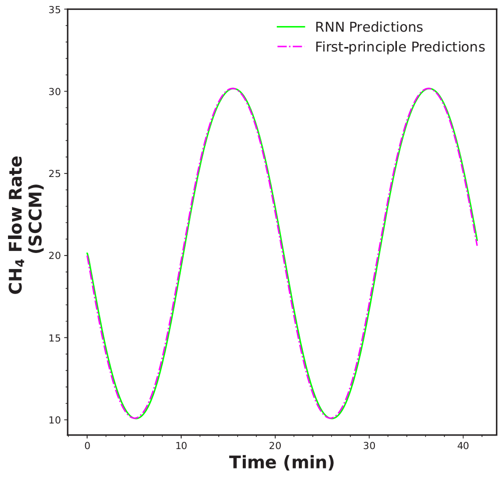
(a) Flow rates of CO.



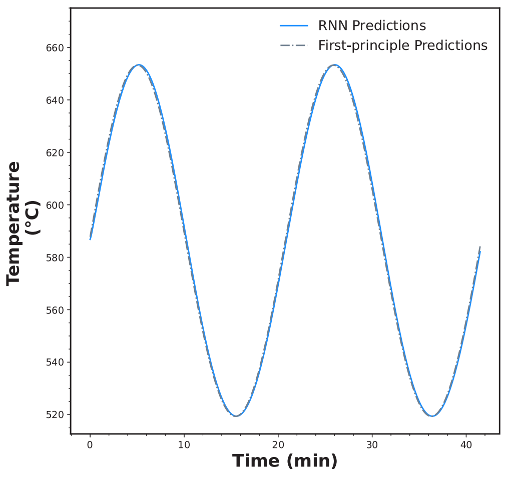
(b) Flow rates of H_2.



(c) Flow rates of CO_2.



(d) Flow rates of CH_4.



(e) Temperature

Fig. 5. RNN model predictions of the concentrations of CH_4, H_2O, H_2, CO and temperature compared to first-principle model predictions based on sinusoidal reference data.

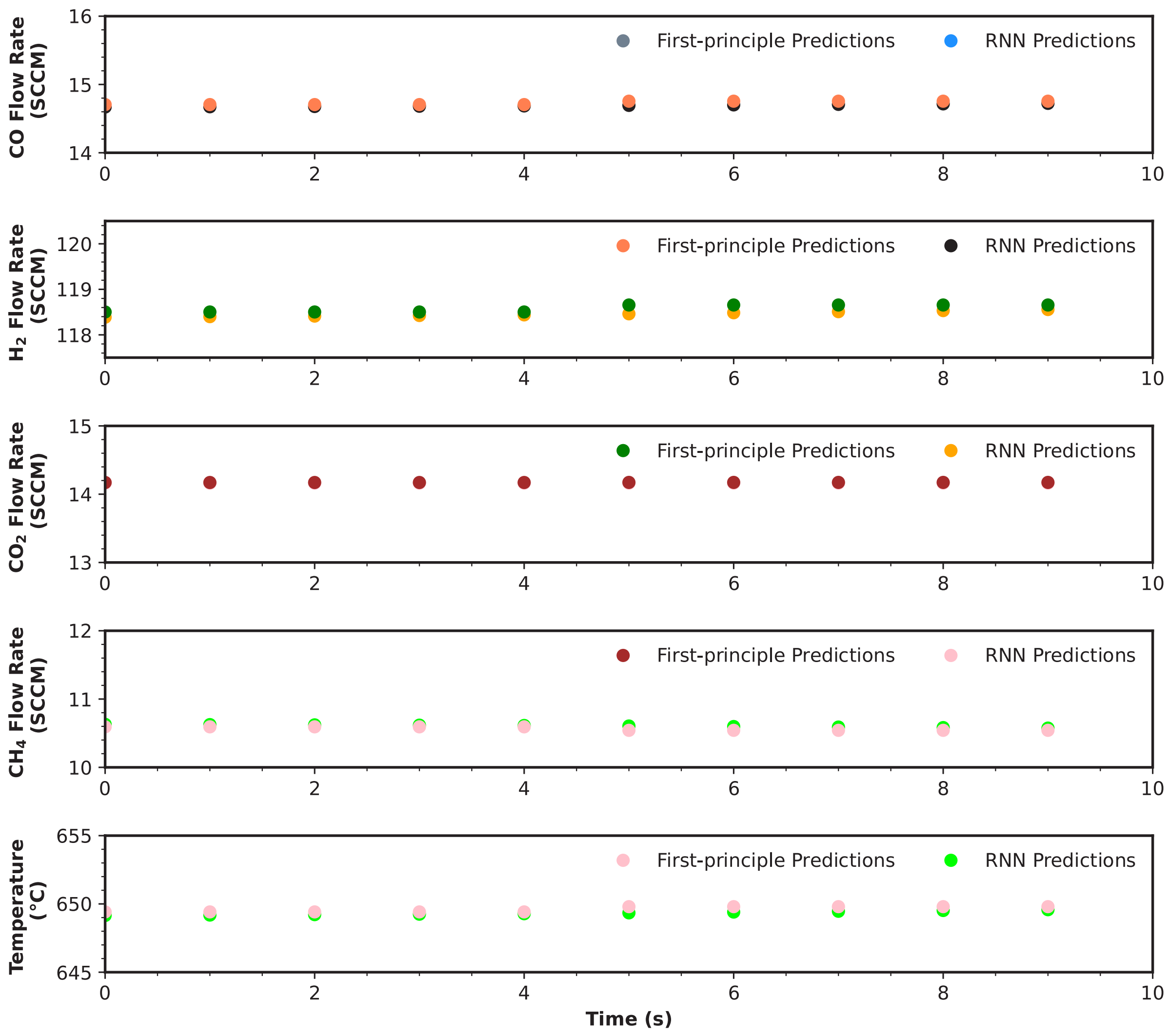


Fig. 6. Comparison between RNN model and first-principles model predictions across two model predictive control horizons, spanning a total of 10 s with distinct control actions implemented in each 5-s interval.

model to be utilized in MPC. Moreover, both RNN and first-principle predictions slightly increase at the beginning of the second horizon (at 5 s), in response to the electric current increase. However, it should be noted that the maximum difference occurs at the beginning of each horizon, which is noticeably observed at $t = 0$ s and $t = 5$ s in the figures of hydrogen flow rate and temperature. This difference gradually reduces over time as the RNN predictions converge to the first-principle predictions at the end of each horizon. This observation shows a similarity to the variations between RNN and first-principle model observed in Figs. 4 and 5 when the electric current undergoes a step change, confirming that our RNN model takes a slightly longer lag to response compared to the first-principle model with the electric current change. Nevertheless, the difference between the predictions of the two models is sufficiently small (within an average relative error of 0.3%) and can be considered insignificant. To evaluate the generalization capability of the RNN model, five different initial conditions are applied to the scenario. Table 2 presents the average relative errors of the five output variables for the RNN. The small average relative error for all five initial conditions demonstrates the good generalization capability of our RNN model. Overall, the results demonstrate the accuracy of the RNN model, indicating its learning capacity to fully capture the underlying pattern of the data and the validation of using the RNN model to predict the system behavior and aim control actions in the MPC scheme.

Table 2
Average relative errors of F_{CH_4}, F_{H_2}, F_{CO_2}, F_{CO}, and T started at different initial conditions.

Initial conditions	Average relative error (%)
Initial condition 1	0.216
Initial condition 2	0.219
Initial condition 3	0.163
Initial condition 4	0.209
Initial condition 5	0.188

3.4. Transfer learning method

After the training and evaluation process, the predictions of the LSTM-based RNN model and the first-principle process model exhibit close correspondence according to the small MSE value determined based on training and testing data, indicating the accuracy of the RNN model in capturing the dynamic behavior of the first-principle model. However, the discrepancies between the true experiment results and the first-principle model predictions can be observed due to several factors, including simplifications and assumptions made in the mathematical formulations of the first-principle model, or unknown phenomena of the real experiment. Ideally, an RNN model should be built using real experimental data to capture the actual process behavior, yet the amount of experimental data is insufficient to build an RNN model most of the time. The transfer learning method is an approach used in machine-learning modeling that utilizes prior knowledge from a pre-trained model and combines it with new, albeit limited, data to train a new model. Thus, the transfer learning method is employed to refine the pre-trained RNN model with limited new data and align the model predictions closer with the actual experiment results. The common application of the transfer learning method is to improve model performance in detecting operational faults of industrial processes, particularly in multimode chemical processes where the experiment results and the number of available data points vary for different initial conditions (Wu and Zhao, 2020; Zhou et al., 2023). The utilization of the transfer learning method requires the new target process to share similar configurations with the process captured by the pre-trained model. By transferring the prior knowledge from the pre-trained model, the transfer learning method can generalize the model and achieve new tasks while saving training time. In this study, the transfer learning method is used in the real-time online retraining MPC scheme to reduce the discrepancies between the predictions of the RNN model and actual experiment results. In this section, the transfer learning method is conducted on the pre-trained RNN model with a small set of experimental data to ensure its effectiveness in the real-time online retraining MPC scheme.

In the transfer learning method, the weight initialization is an important step to utilize the prior information by initializing the weights and biases of the new model using the weights and biases of the pre-trained model (Yosinski et al., 2014). If the number of the state variables or control variables changes in the new model, the weights of the pre-trained model need to be modified before being used to initialize the parameters of the new model. In such cases, new fully-connected layers need to be added before and after the input and output layers to adapt to the new data dimensions. In our objective, the transfer learning method is used to compensate for the disturbance that caused the unexpected discrepancies between the pre-trained model and the real process. Thus, the dimensions of the input and output of the new model are the same as the pre-trained model, meaning the weights of the pre-trained model can be used without further modifications (Alhajeri et al., 2024). In the transfer learning process, the pre-trained RNN model is fine-tuned with experimental data directly, without adding new layers. Over 30,000 experimental data points, which are generated by the first-principle model including disturbance, are used for fine-tuning the new model with a learning rate of 10^{-6}. All the hyperparameters remained the same as the pre-trained RNN model. After the fine-tuning process, the result is shown in Fig. 7, where the predictions of the transfer learning-based RNN model are compared to those from the pre-trained RNN model, and the simulation results under disturbances. According to Figs. 7(a) to 7(d), the flow rates of each chemical species predicted by the transfer learning-based RNN model show closer modeling of the simulation results under disturbances compared to the pre-trained model predictions, indicating the transfer learning-based RNN model effectively captures the process behavior. In Figs. 7(e) and 7(f), the temperature predictions also show a better correspondence to the target values compared the pre-trained model predictions at the final steady state. This result validates the effectiveness of the transfer learning method in solving the disturbance encountered in our case.

Remark 7. When additional state and control variables are taken into account in the new task, new fully connected layers are added before and after the input and output layers of the pre-trained model to create a new model that adopts the new variable dimension. Typically, the procedure of the transfer-learning method starts with only training the newly added layers by setting the pre-trained layers to non-trainable. This step saves the prior information of the pre-trained model in the new model while introducing the new data trends into the newly added layers. Subsequently, the fine-tuning process further trains the new model by setting the pre-trained layers to be trainable and the newly added layers to be non-trainable. To prevent dramatically losing the prior information from the pre-trained layers, the learning rate used in the fine-tuning process is tuned to be small, typically 10^{-5} (Gulli and Pal, 2017). As mentioned in previous sections, a small learning rate causes a slow convergence rate and eventually results in a long period of the training process. Therefore, small experiment data sets are used to fine-tune the pre-trained layers.

4. Recurrent neural network model-based predictive control

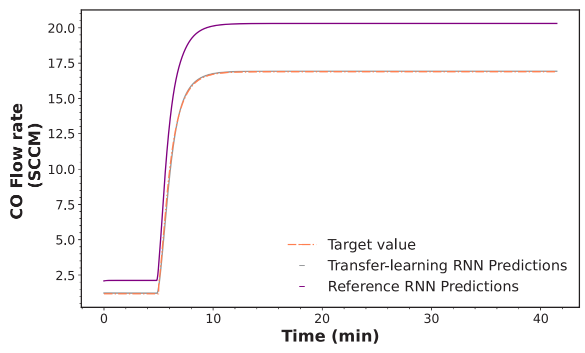
In this section, an RNN-based model predictive controller is specifically designed to regulate the H_2 production rate by adjusting the electric current input through the DC power supply. A method to address the challenges of infrequent and delayed measurement feedback is developed. The performance of this model predictive control strategy is then evaluated by comparing it to a PI controller.

The mathematical formulation of MPC is shown in Eq. (2). Specifically, derivation forms of flow rates of CH_4, CO, H_2, and CO_2 measured by the GC, and the average reactor temperature measured by a TC are set as the vector of state variables, represented as $\boldsymbol{x}$. The manipulated electric current with derivation form actuated by DC power is the control action represented as u. This MPC is designed with predictions of two horizons, each lasting 5 s. In each horizon, the RNN predicts the vector of state variables based on the previous vector of state values and control actions. y is the process output target with deviation form required to be controlled, which is the vector of H_2 production rate. u and y are scalars instead of vectors since u and y only refer to one element for each.

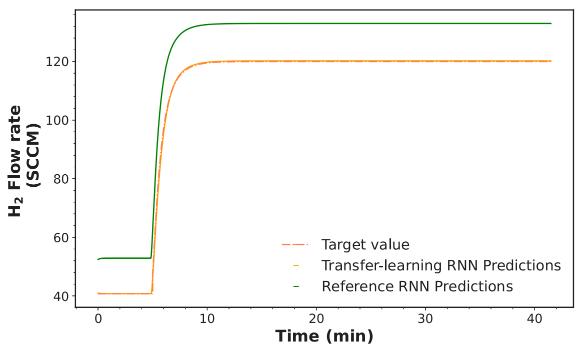
As discussed in Section 1, catalyst sintering is a common phenomenon in an e-SMR process with a Ni-based catalyst. This issue is accelerated by higher reactor temperature change rates. Therefore, it is crucial to set a limit to the reactor temperature. According to Cui et al. (2024), the corresponding electric current constraint is defined in Eq. (2f), where u_c is set to 0.01 A per control action interval, to restrict the temperature change rate not exceeding 6 °C/min. Eq. (2g) ensures that the electric current magnitude is larger than 0 A and smaller than 40 A, maintaining the feasibility of the operational electric current range and ensuring the safe working of the entire system. Based on the constraint and bound, the optimization problem is conducted to minimize the difference between the H_2 production rate and electric current with their corresponding setpoints within receding horizons (Eq. (2d)).

This optimization problem is solved by sequential quadratic programming (SQP), a type of optimization technique widely used for solving constrained non-linear optimization problems. By using the SQP method, the original non-linear optimization problem is transformed into a series of quadratic programming sub-problems. The constraints are linearized at each iteration. By solving these new convex quadratic programming sub-problems with linearized constraints in each iteration, the search direction is determined. This search direction is then used to update the solution estimates, which progressively lead to the optimal solution. Due to its strong local convergence properties, SQP can accurately approximate the optimal solution with high efficiency, especially when provided with good initial guesses. Thus, the SQP is a good option for saving computational time in solving optimization problems that have well-defined constraints and a reasonable initial guess. In particular, a 5-s control time interval is selected to ensure the optimization process completes before the subsequent control action calculation.

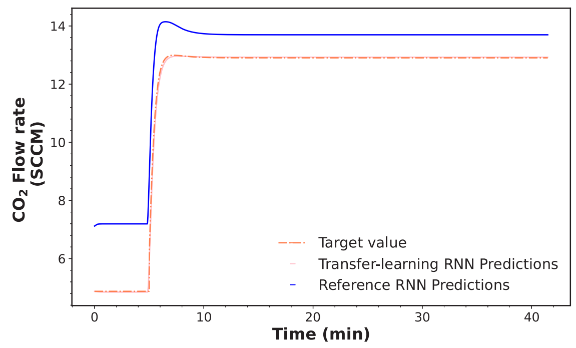
To implement this designed controller in regulating H_2 production rate for an e-SMR process, a close-loop MPC scheme is developed, which is illustrated in Fig. 8. The electric current (I_{mpc}) computed by the RNN-based model predictive controller is implemented in the actual process, with the resulting behavior detected by the GC and TC. In the simulation, the real process is modeled using the lumped parameter-based dynamic model described in 2.2. Given an 18-min



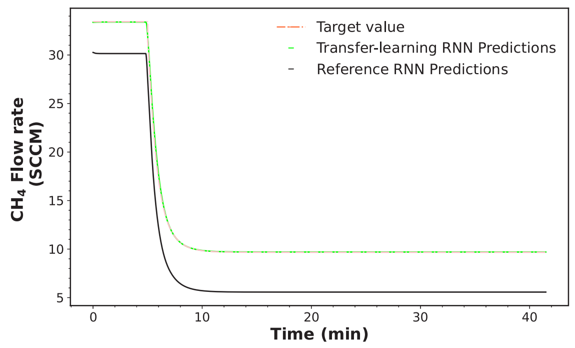
(a) Flow rates of CO.



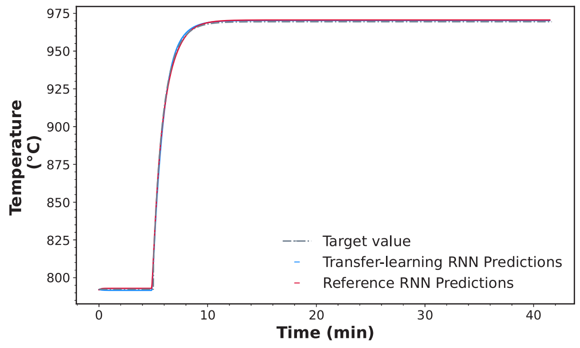
(b) Flow rates of H_2.



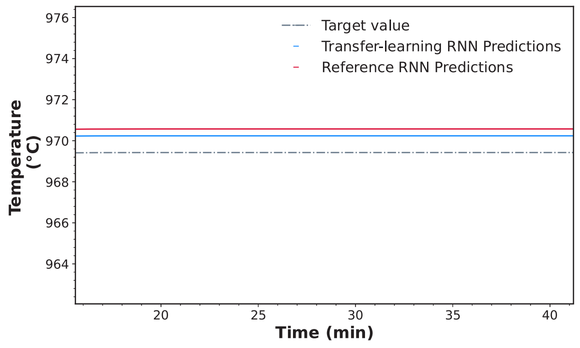
(c) Flow rates of CO_2.



(d) Flow rates of CH_4.



(e) Temperature.



(f) Temperature zoom in.

Fig. 7. Impact of the transfer learning method on a reference RNN model for capturing real experimental trends, with the flow rates of CH_4, H_2O, H_2, CO and temperature as the output variables.

sampling interval of the GC, frequent species flow rate feedback from the measurement cannot be provided to the MPC. Therefore, the RNN is employed to estimate flow rates every second, based on the same implemented control action (I_{mpc}). For each second, the flow rate predictions are based on the flow rates of CH_4, CO, H_2, and CO_2 from previous RNN estimations, and reactor temperature values are extracted from the TC, which has a sampling time of 1 s. These estimated flow rates incorporated with the temperature measurements from TC serve as feedback information for the MPC. The simulated flow rate measurement data from the GC are utilized to correct the RNN estimation and feedback to MPC. However, the input of the RNN requires data per second, while the measurement data is reported only once every 18 min.

To address this issue, the flow rate data between each discrete measurement point (18 min) is estimated on a per-second basis, leveraging the temperature measurements recorded at each second. This estimation is determined based on the strong correlation between the behaviors of all flow rates and the reactor temperature at each time interval. Given the rapid dynamics of the designed e-SMR process according to Çıtmacı et al. (2024a) and Cui et al. (2024), the flow rates quickly reach a steady state corresponding to the new temperature value whenever the temperature changes, which indicates a strong correlation between temperature and flow rates and demonstrates that temperature can be used to estimate flow rates accurately. In detail, a polynomial regression method is utilized and conducted by the following steps (Saini et al., 2021):

- Data preparation: The GC measurement data obtained every 18 min are collected in a dataset. Additionally, the corresponding reactor temperature values for each GC measurement are also recorded, considering a 15-min time delay. Consequently, the dataset containing all simulated GC measurements and the dataset containing their corresponding reactor temperatures are established. To ensure the accuracy of the polynomial regression, the data generation process is triggered when the number of GC collection times exceeds three.

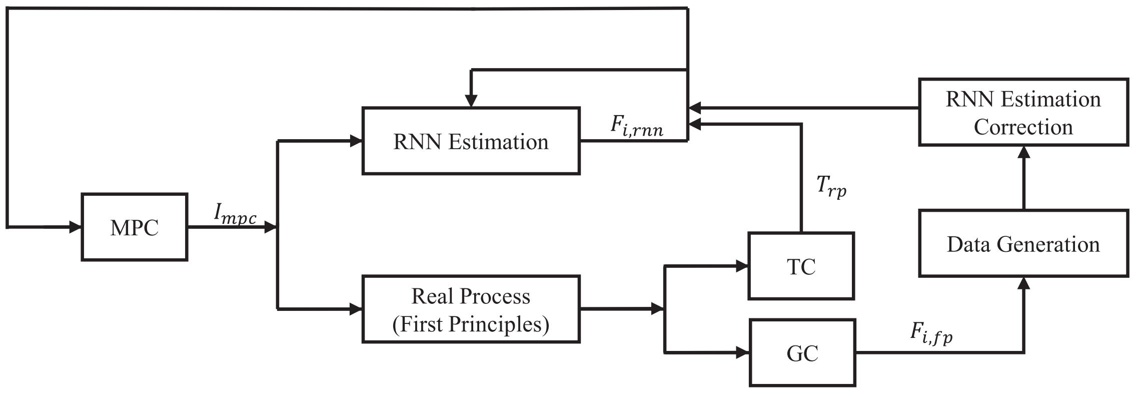


Fig. 8. Closed-loop RNN-based model predictive control scheme to regulate the H_2 production rate for the e-SMR process.

- Feature engineering: Feature engineering is the process of creating new features or transforming existing features to improve the performance of fitting. Specifically, the reactor temperature values are transformed to the third-order polynomial. This third-order polynomial is selected according to the complexity of the system.
- Linear regression: After transforming the feature, a linear regression method is employed to predict the trend of H_2 production rate behavior concerning the variations in the transformed temperature values. The prediction of this trend will certainly be more accurate if more GC data are collected and utilized.
- Data generation: Based on the linear relationship observed between the featured temperature and H_2 production rate, and real-time temperature data captured every second across three consecutive GC datasets, the H_2 production rate is computed for each interpolated data point. This computation results in a total of 1080 data points for flow rates of each gas species.

Following these steps, the pre-trained RNN model utilized in MPC optimization problem and in the initial condition estimations will be replaced by the retrained RNN. Additionally, based on this retrained RNN, flow rates of all gas species per second can be estimated. Considering the time delay, the RNN input at the time step of 15 min ago can be obtained by combining the last 10 consecutive estimated flow rates of all gas species (10 consecutive seconds) from the polynomial function results with the 10 temperature measurements and 10 electric currents from the same time step (10 consecutive seconds 15 min ago). The retrained RNN model subsequently simulates forward in 15-min intervals, using these data as the initial condition, continuously updating its predictions by feeding its output back into the input until it reaches the present time step. These predictions at the current time step are further utilized as the corrected MPC initial conditions and corrected RNN input

To evaluate the performance of the RNN-based model predictive controller, the MPC is executed over 120 min, with the control action starting at 2 min. The setpoint for the H_2 production rate is 120 SCCM. The initial conditions for the RNN model are 25.0 A, 514 °C, 30.2 SCCM, 2.12 SCMM, 52.8 SCMM, and 7.18 SCMM, corresponding to the six input variables (I, T, F_{CH_4}, F_{CO_2}, F_{H_2}, and F_{CO}). The initial conditions of flow rates and temperature are defined by the steady-state value collected when 25.0 A of electric current is applied to the simulated real process. The MPC result is shown in Fig. 9, where the flow rates of CH_4, CO_2, H_2, CO, and temperature are compared between the experiment results and RNN predictions. According to Fig. 9, the H_2 production flow rates of both RNN predictions and results from the simulated real-process initially maintain a steady state value for a brief period before gradually increasing towards the setpoint, corresponding to the time interval before the control action starts. Subsequently, the H_2 production rates and temperature gradually increase, which is consistent with our specified constraint on the electric current changes between consecutive control actions. At the settling time of 33.4 min, the H_2 production flow rates reached the setpoint within 1% offset. The difference in H_2 production rates between the RNN predictions and simulated real-process at the final steady state is 0.02 SCCM, representing a 0.02% deviation from the experiment results. Overall, the RNN predictions strongly agree with the simulated real-process, with the largest variation being 0.374 SCCM, which exhibits outstanding model performance. Consequently, the effectiveness of the control actions demonstrated in the MPC results validates the accuracy of the RNN model, and also its suitability for implementation in the MPC scheme to achieve the desired control results.

The PI controller was implemented to provide a comparison with the MPC scheme. It incorporates a combination of proportional and integral control terms, according to the equation:

$$u = K_c \cdot \left[\left(y_{sp} - \tilde{y} \right) - \frac{1}{\tau_I} \cdot \int_0^t \left(y_{sp} - \tilde{y} \right) \, dt \right] \tag{14a}$$

$$I = u + I_{sp} \tag{14b}$$

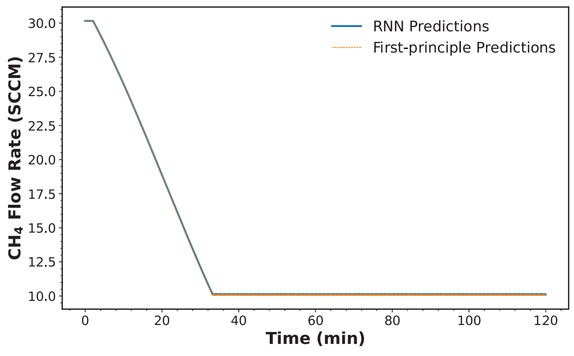
$$y_{sp} = F_{H_2,\,sp} - F_{H_{2,s}} \tag{14c}$$

$$\tilde{y} = F_{H_2} - F_{H_{2,s}} \tag{14d}$$

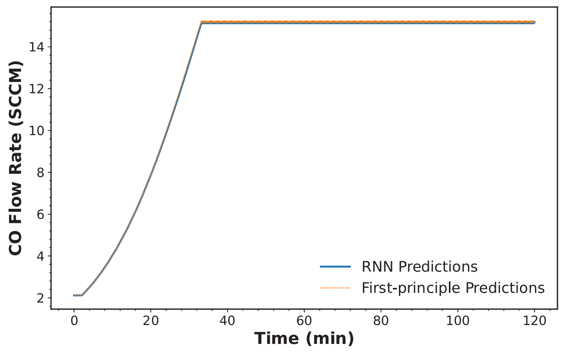
$$0\ \text{A} < I < 40\ \text{A} \tag{14e}$$

where u is the deviation form of the manipulated control input ($I - I_{sp}$), $\tilde{y}$ is the deviation form of the measured target output, and y_{sp} is the deviation form of the setpoint. K_c and τ_i are the gain and integral time constant, respectively. $F_{H_{2,s}}$ and I_s are the initial steady state value of the H_2 production rate and electric current, respectively. The proportional term adjusts the measured output concerning the error signal, while the integral term reduces the error of the output by accumulating the past error. The integral time constant τ_I and gain K_c are tuned to be 80 s and 0.0014 A/SCCM to drive the hydrogen production rate to the setpoint without overshooting. This PI controller parameter settings also ensure the quick response of electric current while satisfying the constraint on temperature change rate that remains below 6 °C/min. The comparison between the PI and MPC process behavior is shown in Fig. 10, where electric currents and H_2 production rates of two control strategies are compared. The H_2 production rates of both control strategies reached the setpoint. However, the MPC took a settling time of 33.4 min to reach the setpoint, while the PI controller took 122 min, which shows the higher efficiency of the MPC strategy. Additionally, the MPC offers the advantage of ensuring the electric current constraint for each control step, as this constraint is integrated into the optimization problem (Eq. (2f)), while there is no guarantee that each step of the PI control will satisfy the electric current or temperature change constraints.

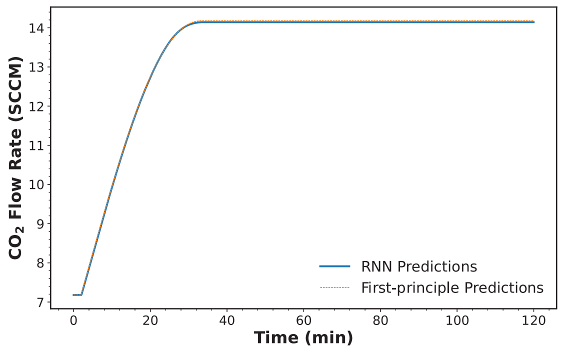
To test the reliability of the polynomial regression method used for data generation, the flow rate data, generated using polynomial



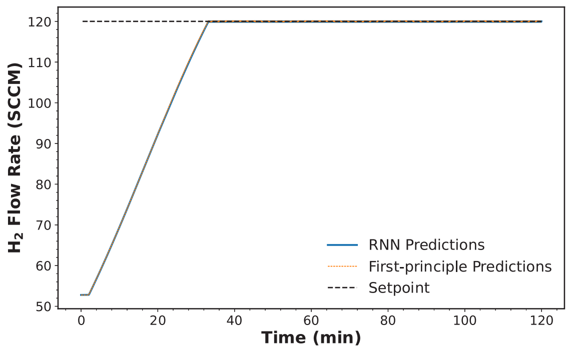
(a) CH_4 profile under RNN-based MPC.



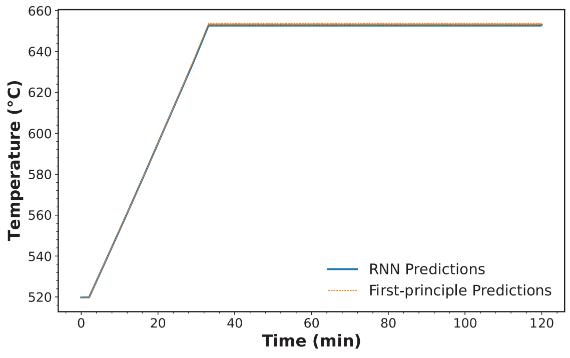
(b) CO profile under RNN-based MPC.



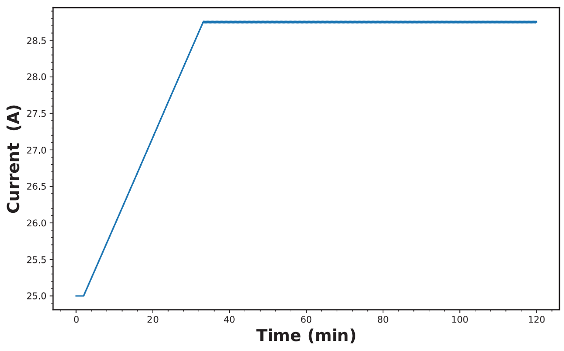
(c) CO_2 profile under RNN-based MPC.



(d) H_2 profile under RNN-based MPC.

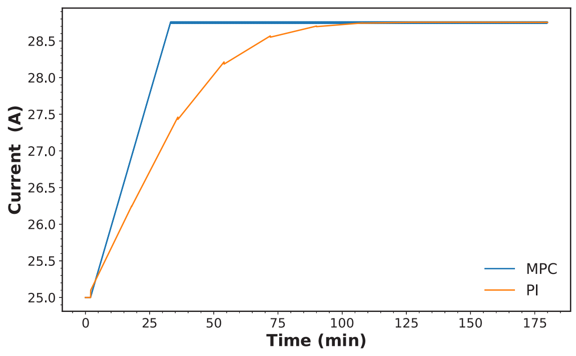


(e) Temperature profile under RNN-based MPC.

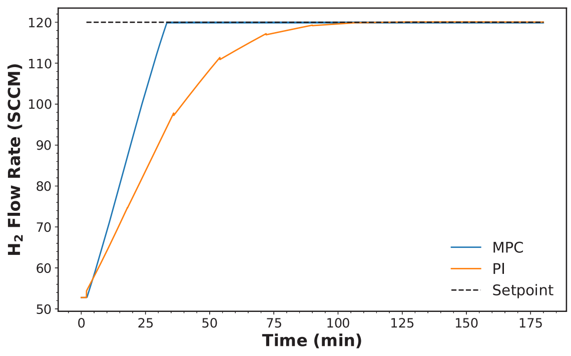


(f) Current profile under RNN-based MPC.

Fig. 9. Setpoint tracking control of the H_2 production rate with an RNN-based model predictive controller for the e-SMR process.



(a) Current profile under PI control.



(b) H_2 production profile under PI control.

Fig. 10. Current and H_2 production rate behaviors under PI control and MPC for the e-SMR process.

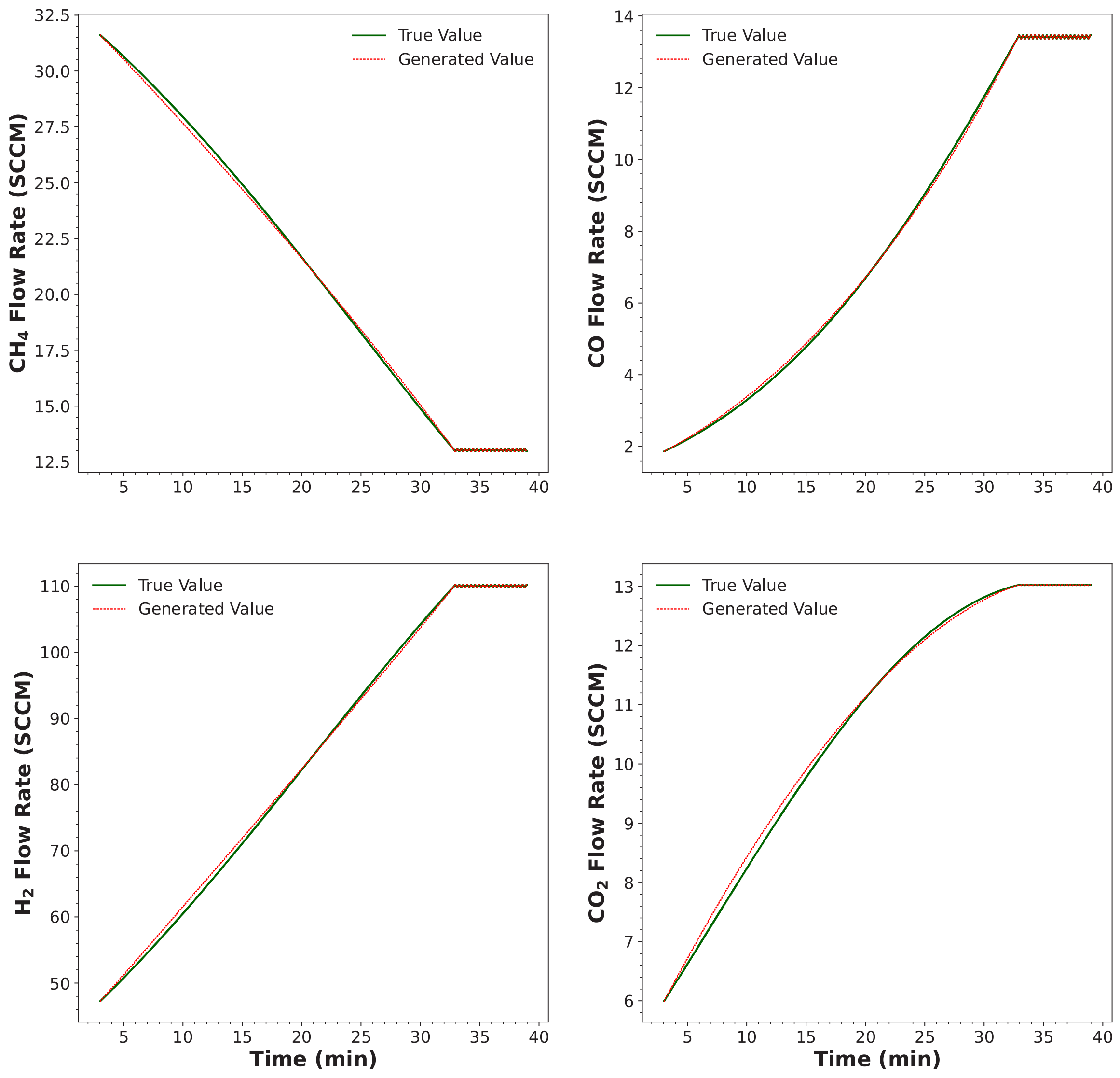


Fig. 11. Comparison between the generated data from polynomial regression and simulated measurement data for all gas species. Measurement data used for data generation are the first 3 simulated GC data points during the MPC run.

regression from the first three consecutive simulated measurement datasets during the MPC simulation, is compared with simulated real values for each second in Fig. 11, where the flow rates of CH_4, CO, H_2, and CO_2 are analyzed. The discrepancies between the generated data and the simulated real-process data are 0.585%, 1.14%, 0.671%, and 0.815% for CH_4, CO, H_2, and CO_2, respectively. These results demonstrate the reliability of the polynomial regression method, as evidenced by the close alignment between the generated data and the simulated real-process data.

Remark 8. To address the possibility that the optimizer might not obtain the optimal electric current value within 5 s in real time, a proportional controller is designed to serve as a backup controller.

Remark 9. Polynomial regression relies on three consecutive simulated GC measurements: the latest GC measurement combined with the two preceding ones. Consequently, both data generation and correction estimation via RNN can only proceed when a minimum of three simulated GC measurements are available.

Remark 10. In practical settings, noise commonly arises from fluctuations in the processing system or inherent variability properties of the sensors, affecting the accuracy of measurement feedback information. Such variations lead to the generation of noisy signals fed to the controller, potentially resulting in inaccurate control actions. Moreover, measurement noise adversely impacts the data utilized for the online retraining of the RNN, posing additional challenges in real-world applications. Given our experience with the experimental electrified SMR process at UCLA, noise in the measurements has not been significant enough to influence the stability and performance of the control systems used.

5. Disturbance compensation

In this section, a common disturbance for the real e-SMR process is considered, resulting in a mismatch between the real process and the RNN model. To address this issue, various approaches for handling model inaccuracy in the presence of disturbances are explored. Each method is applied, and their respective outcomes are analyzed.

5.1. Model predictive control under disturbance

For the e-SMR process, a key challenge is catalyst deactivation (Saeidi et al., 2023). Zhang et al. (2021b) stated that the main causes of catalyst deactivation are coking and sintering, leading to a reduced ability of the catalyst to effectively lower the activation energies of the SMR and WGS reactions. Consequently, higher activation energies for SMR reactions (Eq. (1)) are considered as a disturbance for the e-SMR process. Specifically, a larger value of activation energy (E_a) is used in the simulated real experimental process in our case. Therefore, a mismatch between the RNN model and the first-principle-based real process model is observed.

In Fig. 12, the MPC results under a disturbance (2% increase in E_a for each reaction) are presented. While the RNN estimation is approximately the same as in the no-disturbance case, the simulated real-process variables change significantly due to the disturbance. Initially, the electric current is set to be 25.0 A in both scenarios. Compared with the no-disturbance case, the temperature and H_2 production after adding the disturbance results in different values at the initial stage. The H_2 production decreases from 52.7 SCCM to 45.4 SCCM due to the reduced catalyst performance. Moreover, the reactor temperature increases from 519 to 522 °C due to lower methane consumption. Towards the final state, the electric current stabilizes around 28.7 A for both scenarios. However, in contrast to the no-disturbance case, the final simulated real H_2 production stabilizes around 110 SCCM, resulting in an offset between the setpoint and the simulated real-process, while the RNN estimation can be driven to the setpoint (120 SCCM). This discrepancy arises as the RNN is trained using data that does not include disturbances, resulting in a mismatch between its estimation and the real process which is subjected to disturbances. Since the RNN estimation is utilized as the initial condition in the MPC, the MPC can only drive the RNN estimation to the setpoint instead of the actual process output. Thus, additional strategies are required to achieve offset-less output tracking.

5.2. Model predictive control combined with an integrator

One approach to solve the offset issue is combining the model predictive controller with an integrator. To correct the control action signal from the model predictive controller, an integrator is added to the control action signal as follows:

$$u_I = \frac{1}{\tau_I'} \int_0^t (F_{H_2,sp} - \hat{F}_{H_2}(t_n))\, d\tau \tag{15a}$$

$$u = u_{MPC} + u_I \tag{15b}$$

$$I = u + I_s \tag{15c}$$

where $\boldsymbol{u}$ is the deviation form of the control action vector (electric current, I) from the newly designed controller, $\boldsymbol{u}_{MPC}$ is the deviation form of the control action vector from the model predictive controller, I_s is the initial steady state value of the electric current, τ_I' is the tuning parameter for the integrator, and $\hat{F}_{H_2}(t_{n,H_2})$ is the simulated H_2 production at $t = t_{n,H_2}$, where t_{n,H_2} is the corresponding time instant for the measured value from the simulated GC, involving delay. The integral term of Eq. (15) can compensate for the offset between the setpoint and the real-process H_2 production by considering the error accumulation, utilizing the same function as the integrator in PI control. In this approach, the initial control action from the integrator (u_I) is numerically much smaller than the initial control action obtained from MPC (u_{MPC}), thus showing less influence on the overall control action effectiveness compared to MPC action at the beginning of the closed-loop implementation. Therefore, the H_2 production rate is driven primarily by MPC actions initially until the u_{MPC} control action reaches a plateau, indicating alignment with the setpoint reached by the H_2 production rate estimated by the RNN model integrated with the MPC. After the u_{MPC} stabilizes at a constant value, the remaining offset between the H_2 production rate estimated by the new controller and the setpoint will be eliminated by the integrator term u_I. Additionally, a smaller limit for the control action change rate in the model predictive control algorithm (u_c in Eq. (2f)) and a sufficiently large τ_I' should be implemented to ensure the absolute value of the temperature change rate is less than 6 °C/min. Based on this consideration, the limit for the change in control action relative to its previous value is set to be 0.009 A and the τ_I' is set to be 2.71×10^5 A s^{-1} SCCM^{-1}.

Table 3
Settling time and maximum overshoot for different values of activation energies (E_a) under the model predictive controller combined with an integrator scheme.

Activation energy	Settling time (min)	Maximum overshoot (%)
1.00 E_a	563	9.06
1.01 E_a	499	6.47
1.02 E_a	354	3.47
1.03 E_a	47	0
1.04 E_a	377	0.19

The performance of this approach is shown in Fig. 13. Overall, the RNN estimation reaches the setpoint rapidly (35.9 min within 1% offset), which is similar to the MPC results under disturbance conditions without the integrator. Compared to the final offset observed in Fig. 12, the simulated real H_2 production rate reaches the setpoint (top-left figure in Fig. 13), which demonstrates the successful control outcomes. Moreover, the temperature behaviors observed in the bottom two figures in Fig. 13 show that the controller action is within the temperature change rate constraint (6 °C/min in the bottom right figure in Fig. 13), showing the feasibility of this approach. However, unexpected results are also presented in Fig. 13. Even though the simulated real H_2 production rate reaches the setpoint, it requires 354 min to achieve the simulated real H_2 production rate within 1% offset, which is significantly longer than the time required in previous control simulations (33.4 min). Additionally, 3.47% overshoot occurs for the simulated real H_2 production rate.

To further analyze this approach, various e-SMR process models under disturbances, simulated by the first-principle model with different activation energies (E_a), are evaluated under the designed controller and compared. The results for each disturbance case are illustrated in Fig. 14, and Table 3. As evident in the results reported in Table 3, this approach performs best when the activation energy is increased by 3%, resulting in a much shorter settling time and no overshoot. In contrast, other cases exhibit longer settling times and larger overshoots. The reason behind this is as follows: in the early stage of the closed-loop system simulation, the offset between the H_2 production rate is large compared to the offset when the u_{MPC} is constant. Therefore, the change of control action by the integrator is mainly from the offset when the H_2 production rate is mainly driven by MPC, multiplied by the integrator coefficient (τ_I'). If this additive control action can make the H_2 production rate get close to the setpoint when the u_{MPC} is constant, the time required to further eliminate the offset should be shorter. In the case of a 3% increment in E_a for the disturbance case, u_I calculated by the integration of error multiplied by the coefficient (τ_I') can make the real H_2 production rate around the setpoint when u_{MPC} stops changing. Therefore, there is little effort required for the integrator to drive the real H_2 production rate to the setpoint, indicating much less time and no overshoot. For the cases of 0%, 1%, and 2% increments in E_a for the disturbance, the additive control actions make the H_2 production rate to be higher than the setpoint when the u_{MPC} is constant. Therefore, an overshoot and a long settling time cannot be avoided for each of these cases. In the case of a 4% increment in E_a for the disturbance, the real H_2 production rate is lower than the setpoint when the u_{MPC} is constant. Therefore, even though there is nearly no overshoot, a long time to reach the setpoint is still required.

Remark 11. Tuning the τ_I' in Eq. (15) is challenging for the scheme of MPC with an integrator. On one side, τ_I' cannot be too small to prevent substantial overshoots in the H_2 production rate. Additionally, a small τ_I' can significantly increase the integrator term, causing step changes

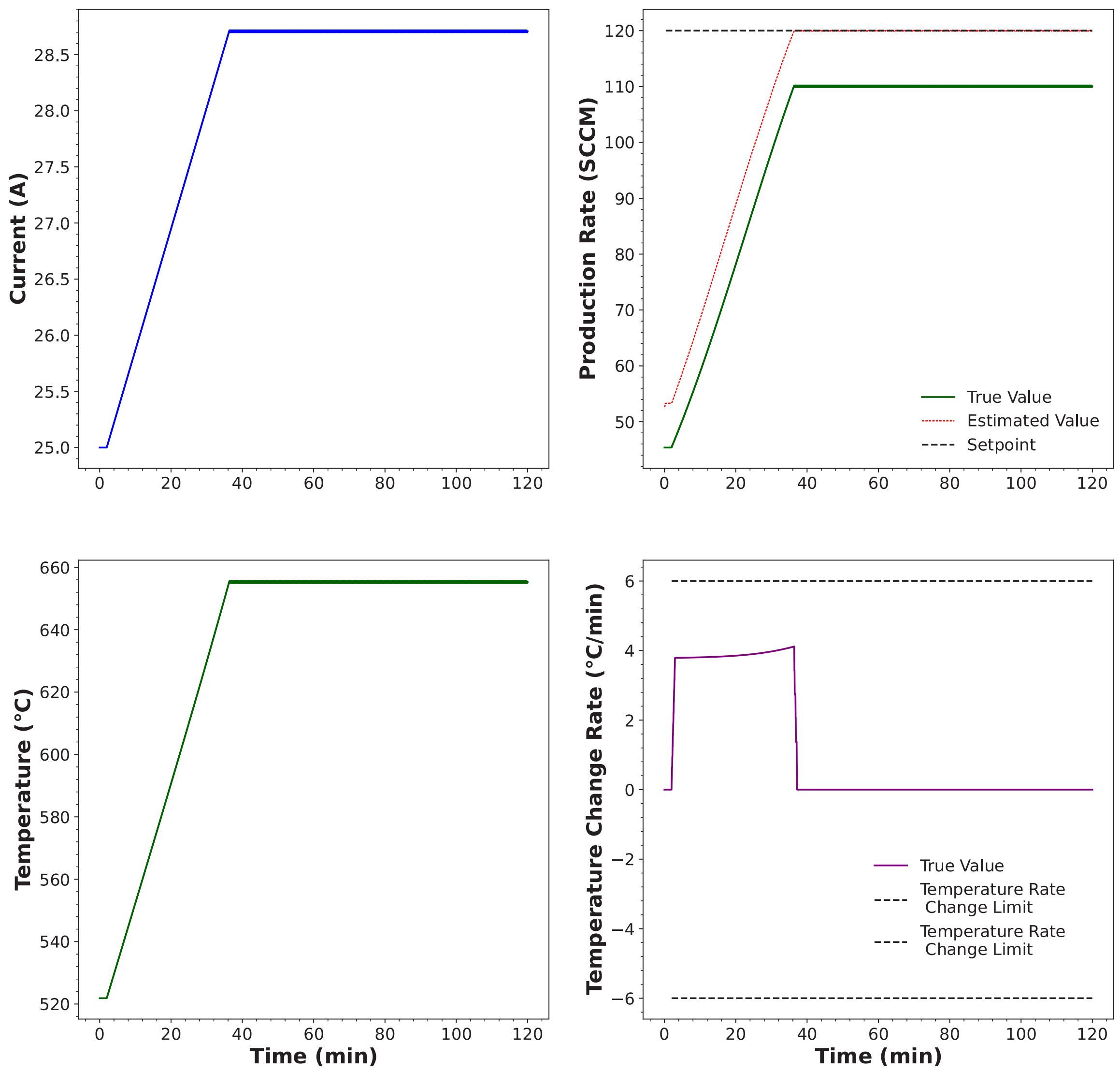


Fig. 12. Setpoint tracking control of the H_2 production rate with an RNN-based model predictive controller for the e-SMR process under disturbances.

in the electric current that exceeds the constraint of the electric current changing rate. On the other side, τ_I' cannot be too large either, since a large value of τ_I' can lead to minimal changes of electric current, leading to a much longer time to eliminate the offset.

Remark 12. Even though a sufficiently large τ_I' is utilized to make the electric current change rate within the constraint, there remains a risk that the overall control action (u) could still violate the constraints, indicating a potential concern for applying this method in real e-SMR processes.

5.3. Model predictive control with RNN real-time online retraining

Instead of modifying the control actions of an MPC scheme, which is built based on a mismatched RNN model with an additional controller term, the real-time online retraining method can be utilized to correct this mismatched RNN model directly. Specifically, the mismatched RNN model will be corrected by using real-time data from the past. Eventually, the corrected RNN model will be implemented within the MPC scheme and estimation of the initial condition of MPC. Therefore, the obtained control action from MPC can be more accurate and the offset between the RNN model and the simulated real process can be eliminated.

For the control scheme employed in this approach, the components and processing flow of the online RNN retraining-based MPC are depicted in Fig. 15. Generally, the left orange box represents the MPC of this system every 5 s, while the right blue box signifies the online retraining and re-estimation process occurring every sampling time of the GC (18 min). Retraining and re-estimation are initiated only upon receiving new GC data. The MPC for the e-SMR process follows a similar scheme as shown in Fig. 8, but utilizes a simulated process model under disturbances.

For the online retraining and re-estimation process shown in the blue box, this process is launched when new simulated measurement data becomes available. These new data are first used for generating the flow rates for each second by applying two consecutive past simulated measurement data points into the polynomial function (detailed descriptions of this process are shown in Section 4). These generated data have two applications. First, the last 10 generated data points are used as the initial conditions from 15 min ago (delayed time, t_d) which are utilized to estimate the corrected 10 data points for the current time step by the RNN model. Subsequently, these 10 data points are used to estimate the corrected initial condition of the RNN model and also serve as the corrected feedback to the controller. Another application for this data is the adaptive online retraining of the RNN.

The adaptive online retraining process involves several key steps:

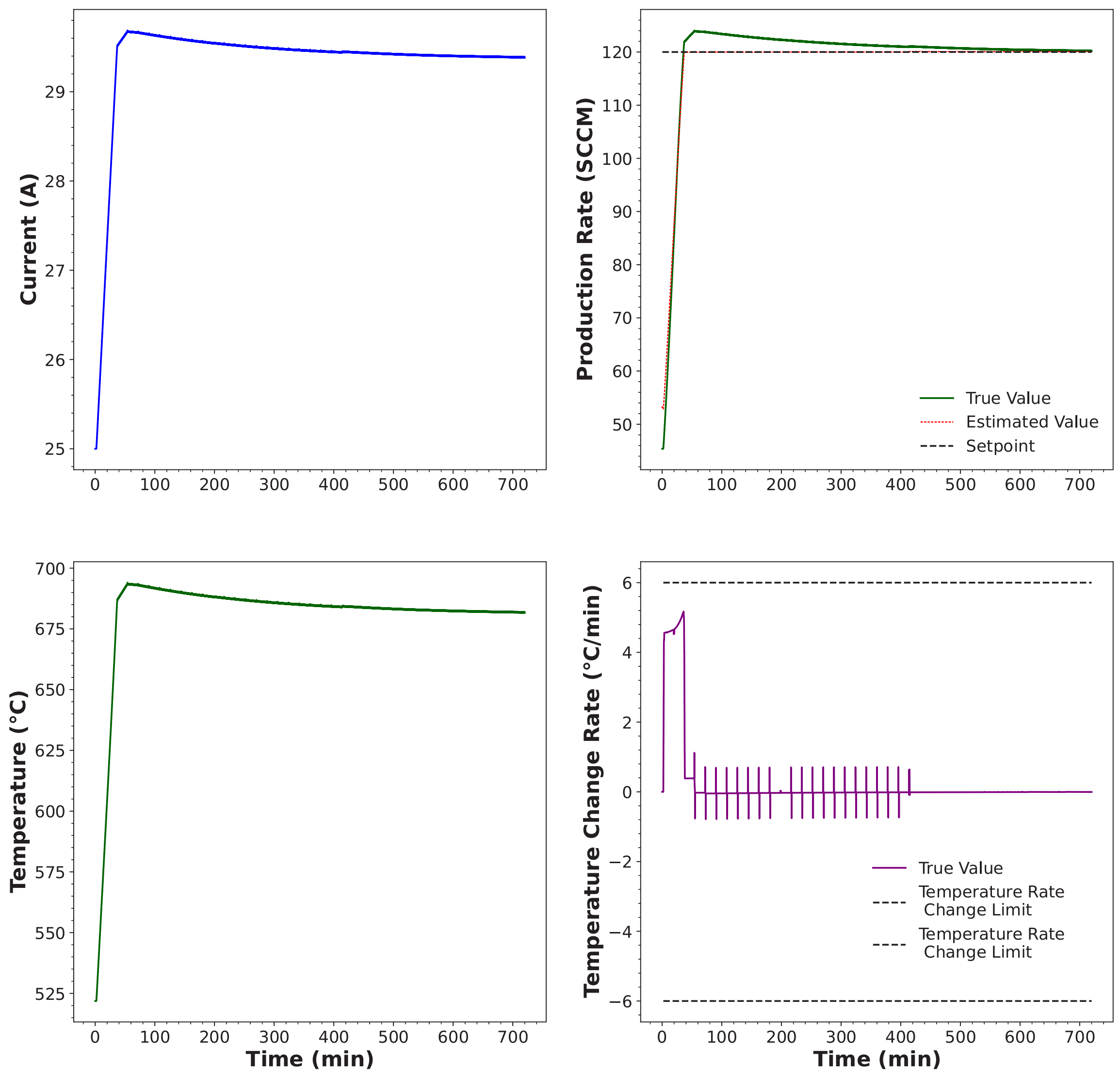


Fig. 13. Setpoint tracking control of the H_2 production rate with an RNN-based model predictive controller with an integrator for the e-SMR process under disturbances.

- Buffering module preparation (Fekri et al., 2021): For the online adaptive RNN strategy, the buffering module is utilized to identify and store useful data batches when the RNN model has poor performance. Additionally, this buffering module can also have an initial data batch to prevent unexpected behavior of the RNN model after retraining. However, this initial batch should not contain a large number of data, as the real-time data is typically limited in practice. A large number of data in the buffering module would reduce the retraining impact of the real-time data. In general, the size of the buffering module can be designed and constrained by the data drift concept. Specifically, by using the data drift concept, data with the lowest error will be replaced by the newly collected data when the buffer module reaches its maximum size. In our case, the data drift concept is not utilized since the designed maximum size (10,800) of the buffering model has not been reached.
- Data collection and processing: For real-time control, reactor temperatures from the TC are obtained every second, while data from the GC are collected every 18 min. The data generation process uses polynomial regression to estimate flow rates for each gas species every second. However, data collected from the GC and TC must be denoised before being fitted by the polynomial regression. Additionally, any infeasible data should be removed to ensure high data quality for training the RNN.
- Error-triggering mechanism: An error-triggering mechanism (Abdullah and Christofides, 2023) is applied to evaluate whether the estimated data set generated by polynomial regression should be incorporated into the buffering module. This mechanism compares the error between the model and measurement data against a predefined threshold to determine if online retraining should be initiated. The Mean Relative Error (MRE) serves as the metric for this comparison, calculated as follows:

$$MRE_n = \frac{1}{N_k} \cdot \frac{\sum_{i=0}^{N_k-1} \left| F_{H_2,RNN}(t_k - \Delta_{GC} \cdot i - t_{d,GC}) - F_{H_2,rp,n-i} \right|}{F_{H_2,rp,ave}} \tag{16}$$

where MRE_n is the mean relative error when the nth measurement data are received, N_k is the number of measurement data points that are utilized for calculating the error (in our case, $N_k = 3$), $F_{H_2,RNN}$ is the RNN estimation of H_2 production rate, t_k is the time moment for the newly received data point, Δ_{GC} is the GC sampling time (18 min), $t_{d,GC}$ is the time delay for the GC measurement (15 min), $F_{H_2,rp,n-i}$ is the $(n-i)$th real H_2 production rate measurement, $F_{H_2,rp,ave}$ is the average of last 3 consecutive real H_2 production rate measurements. If the MRE is less than the threshold (C), the generated data will not be

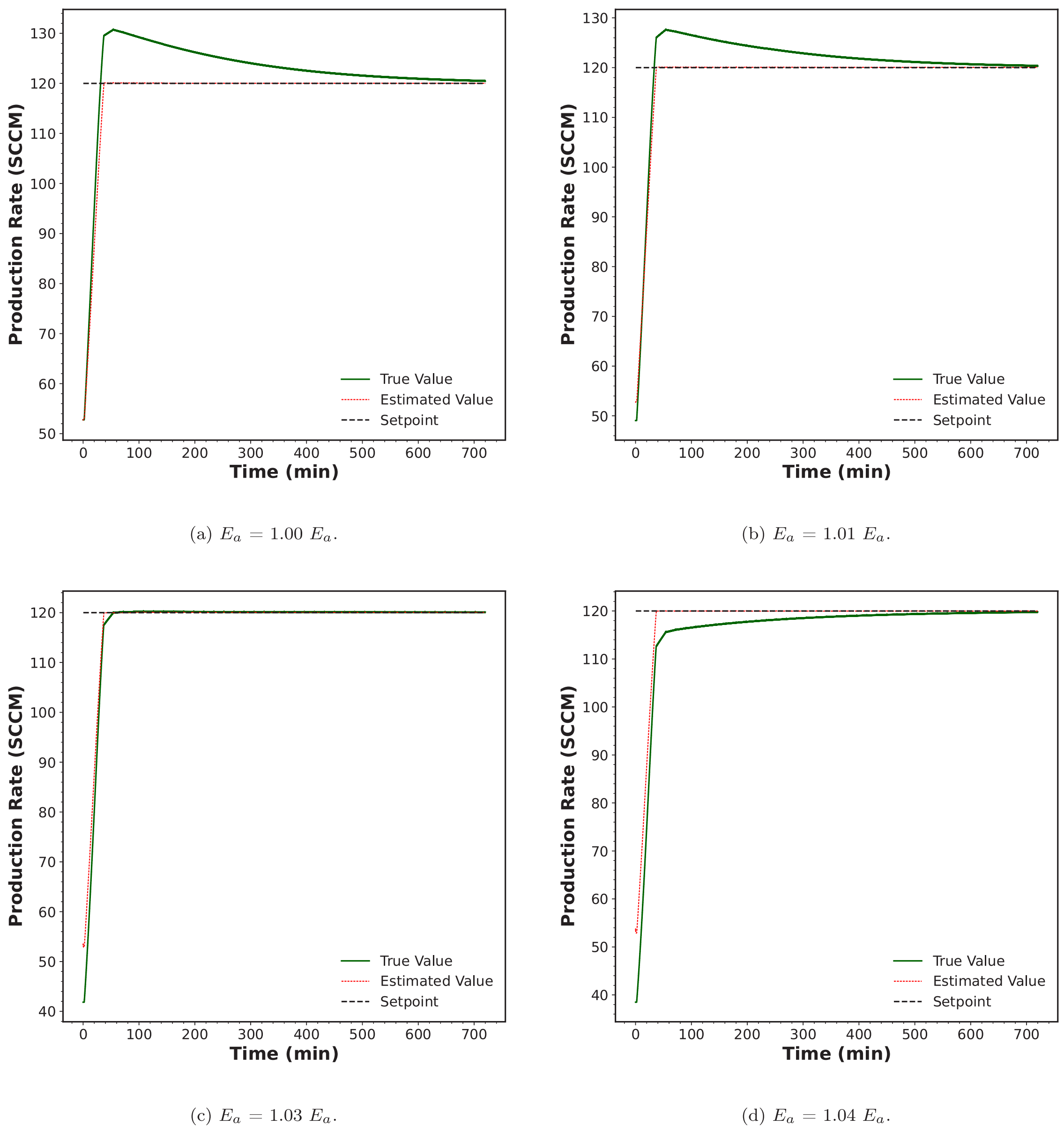


Fig. 14. Setpoint tracking control of the H_2 production rate with RNN-based model predictive controller combined with an integrator under different disturbances.

added to the buffering module, indicating that the RNN model is accurate. Conversely, if the MRE exceeds the threshold, it signifies that the RNN model is inaccurate and requires retraining based on newly generated data. In this case, the threshold is chosen as the 2% of the average of the 3 consecutive GC measurement data:

$$C = 2\% \cdot \frac{\sum_{i=0}^{N_k - 1} F_{H_2,rp,n-i}}{N_k} \tag{17}$$

- Retraining: The retraining process, also known as the transfer learning process, adjusts the RNN model based on real-time data. The training process is detailed in Section 3.4. Before retraining, the data in the buffering module must be preprocessed by the window-sliding technique and normalization techniques. It is crucial to ensure that data from different batches are not mixed within the same sliding window. Additionally, a much lower learning rate ($\alpha = 6 \times 10^{-6}$) and fewer epochs (10) are utilized to prevent the dramatic changes in the behavior of the RNN model resulting from retraining with new data.

Upon completion of retraining, the new RNN model will replace the existing RNN model that is used for state variable estimation as inputs to the MPC, and also the RNN model integrated within the MPC which predicts state variables within the prediction horizons. Additionally, the new RNN model is used to re-estimate all initial state values for the MPC and the initial state values for the RNN model.

The simulation results of this control scheme are presented in Fig. 16. At the initial electric current value of 25.0 A, there is a significant difference between the RNN estimation of the H_2 production rate (52.7 SCCM) and the actual process (45.5 SCCM), as shown in the top-right sub-figure. This discrepancy arises because the RNN model fails to reflect the real process under process conditions subjected to disturbances. This difference persists until the first retraining process is triggered when the third GC sample is obtained at $t = 54$ min. Subsequently, the RNN estimation aligns more closely with the actual process value, indicating improved accuracy. After the first retraining process, the feedback and predicted values of the H_2 production rate fall below the setpoint, successfully reflecting the process behavior under disturbances due to the RNN model parameter adaptation. Consequently, the electric current increases, as shown in the top-left

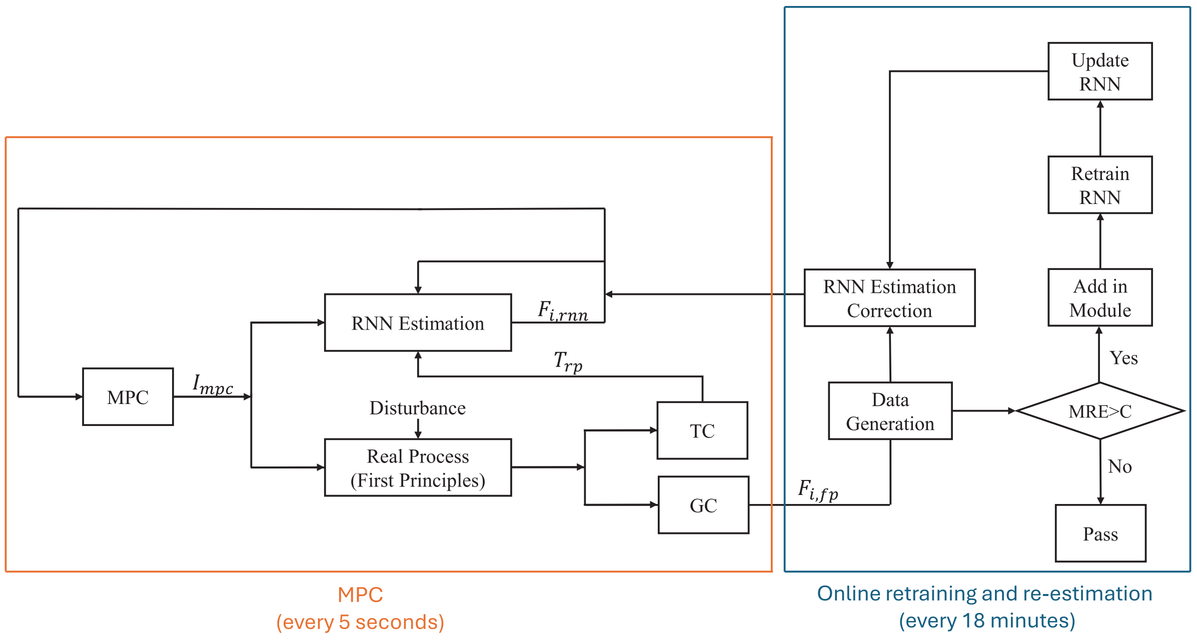


Fig. 15. Setpoint tracking control of the H_2 production rate with an RNN-based model predictive controller using online RNN retraining by utilizing real-time data for the e-SMR process under disturbances: components and processing flow. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sub-figure, leading to a rise in the simulated reactor temperature, depicted in the bottom-left sub-figure. These adjustments occur every 18 min until the MRE, calculated by Eq. (16), falls below the threshold determined by Eq. (17). Ultimately, the H_2 production rate reaches the setpoint with a settling time of 58.9 min and zero overshoot. The bottom-right sub-figure demonstrates that the control is maintained within the temperature constraint, preventing damage to the Ni-based catalyst.

Remark 13. In the case of real-time control, the online retraining process and MPC simulation must be executed concurrently when the retraining event is triggered to ensure the MPC operates within the five-second sampling time. In the Python implementation, the command “subprocess.Popen” is utilized to execute a separate Python script, thereby preventing the retraining process from interfering with the control process. A 2-min sampling time is applied to the retraining process to ensure its completion, assuming that this duration is sufficient for the training process. Specifically, the RNN is replaced with the newly trained model 2 min after the beginning of the retraining process. After the replacement, subsequent RNN estimations are generated from this retrained RNN model

5.4. Offset-free model predictive control

The offset-free model predictive controller is also designed to eliminate the offset. The detailed algorithm for this approach is shown in Section 2.4. Specifically, in Eq. (3), $F(\cdot)$ is the RNN model and θ is used to compensate for the error between the RNN estimation and the experimental data. This error vector can be combined with the vector of state variables to form a new vector of state variables, $\bar{x}$ in Eq. (4). Additionally, the new corrected model can also be written as Eq. (4c). These new versions of the state variable vector and the model are utilized in the MPC.

The Luenberger observer is employed to improve the estimation of state variables and minimize error accumulation (Eq. (5)). Specifically, the differences in temperature and H_2 production rate between the process measurements and model estimates are used as the error.

Due to the sampling frequency of the GC, the temperature is updated every second to compensate for the discrepancy, whereas updates for the H_2 production rate difference occur every 18 min. This difference in sampling time may lead to discrepancies in the H_2 production rate. To account for this potential discrepancy, the coefficients of the observer terms (K) are tuned sufficiently small to ensure that the state values do not change excessively as a result of the correction term. Eventually, these gain parameters result in model estimation that closely matches the simulated process data. Specifically, the parameters are chosen as given by Eqs. (18a) and (18b) in Box I where $G_\theta^\intercal$ is the transpose of the coefficient matrix for the argument term (θ) and $K^\intercal$ is the transpose of the coefficient matrix for the differences in the H_2 production rate and reactor temperature terms. Results for this offset-free model predictive control of this specific e-SMR process are illustrated in Fig. 17. At $t = 0$ min, the same electric current (I_{MPC}) is utilized for the RNN estimation model and the process under disturbances. In the top-right figure of Fig. 17, the corresponding offset is observed initially. While the H_2 production rate estimated by RNN reaches the setpoint (within 1% offset) at $t = 33.4$ min, the H_2 production rate from the simulated process gradually approaches the setpoint due to the correction of the model by the offset-free MPC approach, indicating an improved control performance compared to the constant offset shown in Fig. 12.

As the model utilized in the MPC and the initial state value estimation is progressively corrected, the I_{MPC} calculated from the MPC is gradually corrected accordingly. The adjustment of I_{MPC} shown in the top-left figure of Fig. 17, results in the gradual alignment of the simulated real-process H_2 production rate with the setpoint, as shown in the top-right figure of Fig. 17. At $t = 88.2$ min, the H_2 production rate of the simulated real-process reaches the setpoint with a deviation within 1%. The bottom two figures illustrate the reactor temperature behaviors, demonstrating that the applied electric current from the MPC is safe for the catalyst, as the temperature change rate remains within the permissible limit of 6 °C/min.

Remark 14. With the accumulation of the θ, the H_2 production rate from the model estimation also changes. However, since the coefficients

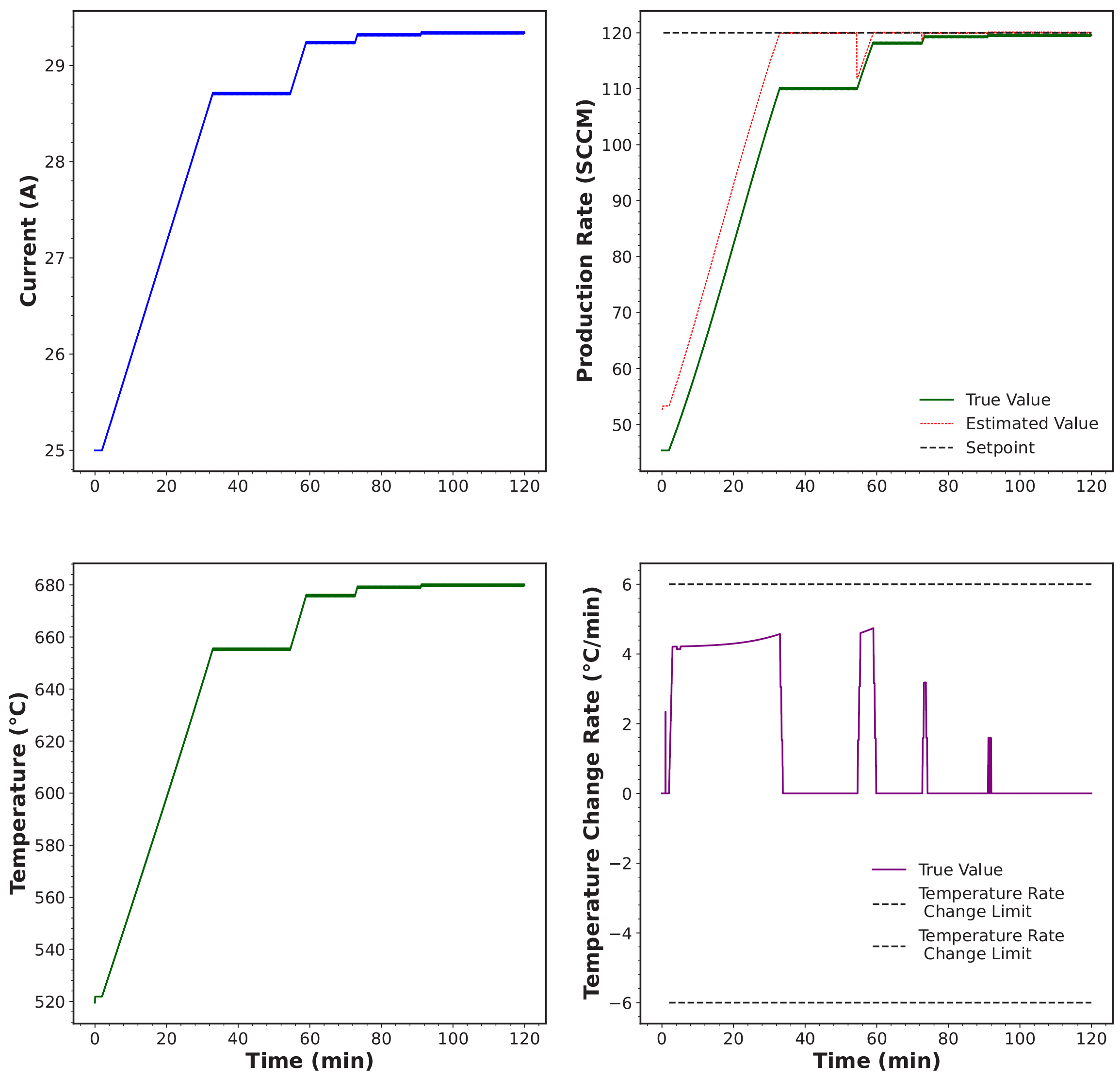


Fig. 16. Setpoint tracking control of the H_2 production rate with an RNN-based model predictive controller using online RNN retraining by utilizing real-time data for the e-SMR process under disturbances: simulation results.

$$G_\theta^T = [0 \quad -0.004 \quad 0.003 \quad 0.025 \quad 0.001] \tag{18a}$$

$$K^T = \begin{bmatrix} 0 & -0.01 & 0.01 & 0.01 & 0.01 & 0 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0 & 0.01 & 0 & -0.01 & 0.01 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{18b}$$

Box I.

(K and G_θ) are small and the corresponding MPC responses are frequent, the model estimation values remain nearly constant within its sampling time.

6. Conclusion

In this work, a recurrent neural network model was developed to capture the dynamic behavior of an electrically heated steam methane reforming process using time-series data from an experimentally-validated lumped parameter dynamic model. Specifically, the LSTM layer was utilized to learn the long-term dependencies in sequential data. The accuracy of this model was further validated. To regulate the H_2 production rate, a setpoint tracking control was implemented using an RNN-based predictive control strategy, which effectively handled infrequent and delayed flow rate measurements. This control strategy demonstrated significantly better performance compared to a PI control scheme. Building on the developed RNN-based MPC strategy, three distinct approaches were successfully developed to manage the e-SMR process under disturbance conditions: MPC combined with an integrator, MPC with transfer learning applied to the RNN model, and offset-free MPC. These approaches have been demonstrated to effectively eliminate the offset caused by disturbances.

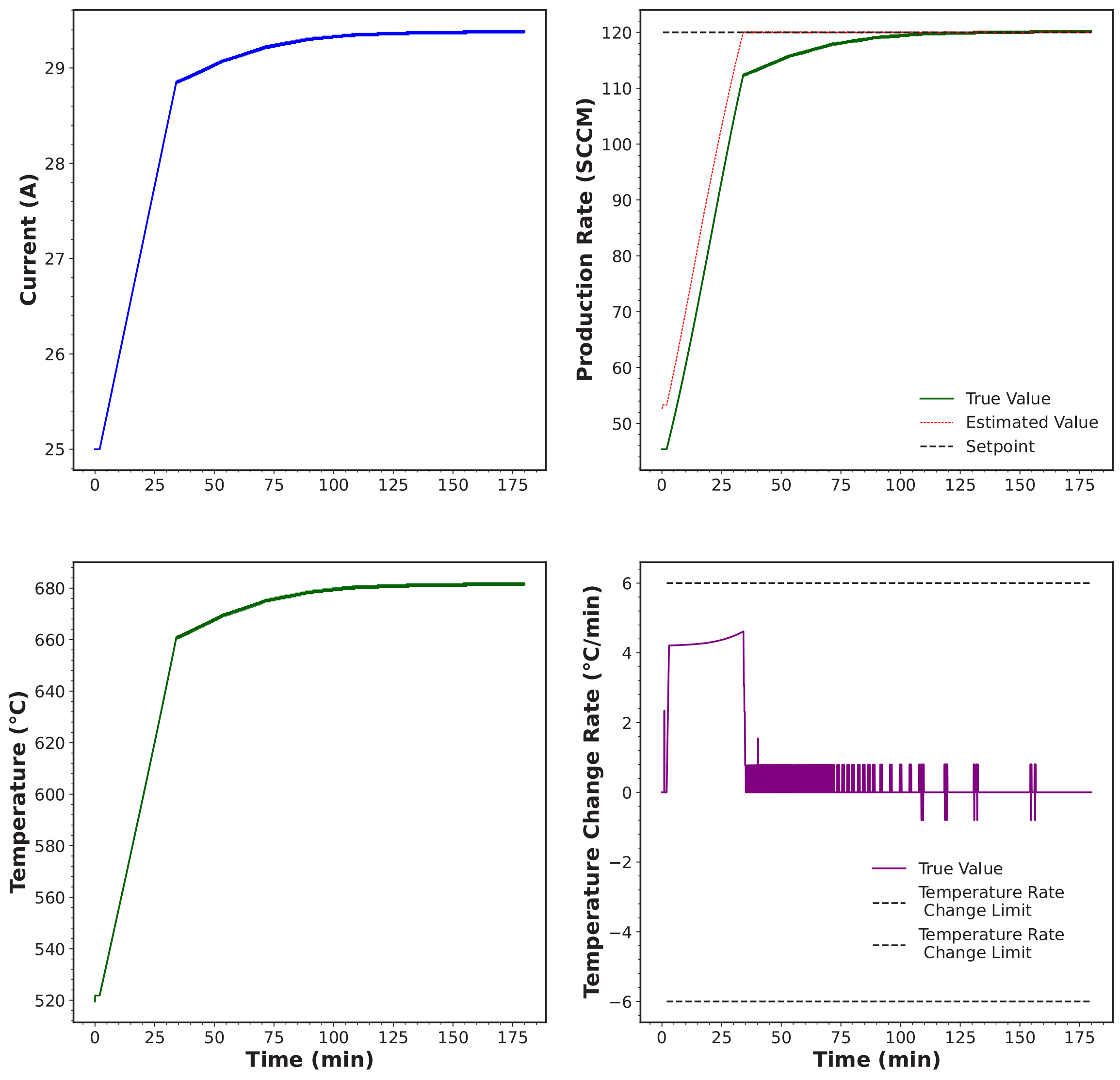


Fig. 17. Setpoint tracking control of the H_2 production rate with an RNN-based offset-free model predictive controller for the e-SMR process under disturbances.

CRediT authorship contribution statement

Yifei Wang: Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Xiaodong Cui:** Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Dominic Peters:** Writing – original draft, Methodology, Investigation. **Berkay Çıtmacı:** Methodology, Investigation. **Aisha Alnajdi:** Methodology, Investigation. **Carlos G. Morales-Guio:** Supervision, Methodology. **Panagiotis D. Christofides:** Writing – review & editing, Supervision, Methodology, Investigation, Conceptualization.

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

We would like to gratefully acknowledge financial support from the U.S. Department of Energy, through the Office of Energy Efficiency and Renewable Energy (EERE), under the Advanced Manufacturing Office Award Number DE-EE0007613. Financial support from the National Science Foundation, United States is also gratefully acknowledged.

References

Abdullah, F., Christofides, P.D., 2023. Real-time adaptive sparse-identification-based predictive control of nonlinear processes. Chem. Eng. Res. Des. 196, 750–769.
Alhajeri, M.S., Ren, Y.M., Ou, F., Abdullah, F., Christofides, P.D., 2024. Model predictive control of nonlinear processes using transfer learning-based recurrent neural networks. Chem. Eng. Res. Des. 205, 1–12.
Ashik, U., Daud, W.W., Abbas, H.F., 2017. Methane decomposition kinetics and reaction rate over Ni/SiO$_2$ nanocatalyst produced through CO-precipitation cum modified Stöber method. Int. J. Hydrog. Energy 42, 938–952.
Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, 281–305.
Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.
Çıtmacı, B., Cui, X., Abdullah, F., Richard, D., Peters, D., Wang, Y., Hsu, E., Chheda, P., Morales-Guio, C.G., Christofides, P.D., 2024a. Model predictive control of an electrically-heated steam methane reformer. Digit. Chem. Eng. 10, 100138.
Çıtmacı, B., Luo, J., Jang, J.B., Morales-Guio, C.G., Christofides, P.D., 2023. Machine learning-based ethylene and carbon monoxide estimation, real-time optimization, and multivariable feedback control of an experimental electrochemical reactor. Chem. Eng. Res. Des. 191, 658–681.
Çıtmacı, B., Peters, D., Cui, X., Abdullah, F., Almunaifi, A., Chheda, P., Morales-Guio, C.G., Christofides, P.D., 2024b. Feedback control of an experimental electrically-heated steam methane reformer. Chem. Eng. Res. Des. 206, 469–488.
Cui, X., Çıtmacı, B., Peters, D., Abdullah, F., Wang, Y., Hsu, E., Chheda, P., Morales-Guio, C.G., Christofides, P.D., 2024. Estimation-based model predictive control of an electrically-heated steam methane reforming process. Digit. Chem. Eng. 11, 100153.

Fekri, M.N., Patel, H., Grolinger, K., Sharma, V., 2021. Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network. Appl. Energy 282, 116177.

Feurer, M., Hutter, F., 2019. Hyperparameter optimization. In: Automated Machine Learning: Methods, Systems, Challenges. Springer International Publishing, pp. 3–33.

Ginsburg, J.M., Piña, J., El Solh, T., De Lasa, H.I., 2005. Coke formation over a nickel catalyst under methane dry reforming conditions: thermodynamic and kinetic models. Ind. Eng. Chem. Res. 44, 4846–4854.

Green, Jr., L., 1982. An ammonia energy vector for the hydrogen economy. Int. J. Hydrog. Energy 7, 355–359.

Gulli, A., Pal, S., 2017. Deep Learning with Keras. Packt Publishing Ltd.

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

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. 79, 2554–2558.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Lubitz, W., Tumas, W., 2007. Hydrogen: An overview. Chem. Rev. 107, 3900–3903.

Luo, J., Çıtmacı, B., Jang, J.B., Abdullah, F., Morales-Guio, C.G., Christofides, P.D., 2023. Machine learning-based predictive control using on-line model linearization: Application to an experimental electrochemical reactor. Chem. Eng. Res. Des. 197, 721–737.

Maeder, U., Borrelli, F., Morari, M., 2009. Linear offset-free model predictive control. Automatica 45 (10), 2214–2222.

Meloni, E., Iervolino, G., Ruocco, C., Renda, S., Festa, G., Martino, M., Palma, V., 2022. Electrified hydrogen production from methane for PEM fuel cells feeding: A review. Energies 15 (10), 3588.

Miljanovic, M., 2012. Comparative analysis of recurrent and finite impulse response neural networks in time series prediction. Indian J. Comput. Sci. Eng. 3, 180–191.

Muske, K.R., Badgwell, T.A., 2002. Disturbance modeling for offset-free linear model predictive control. J. Process Control 12, 617–632.

Nieva, M.A., Villaverde, M.M., Monzón, A., Garetto, T.F., Marchi, A.J., 2014. Steam-methane reforming at low temperature on nickel-based catalysts. Chem. Eng. J. 235, 158–166.

Pannocchia, G., Rawlings, J.B., 2003. Disturbance models for offset-free model-predictive control. AIChE J. 49, 426–437.

Pazheri, F., Othman, M., Malik, N., 2014. A review on global renewable electricity scenario. Renew. Sustain. Energy Rev. 31, 835–845.

Qin, S.J., Badgwell, T.A., 2003. A survey of industrial model predictive control technology. Control Eng. Pract. 11, 733–764.

Rasul, M., Hazrat, M., Sattar, M., Jahirul, M., Shearer, M., 2022. The future of hydrogen: Challenges on production, storage and applications. Energy Convers. Manage. 272, 116326.

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

Saeidi, S., Sápi, A., Khoja, A.H., Najari, S., Ayesha, M., Kónya, Z., Asare-Bediako, B.B., Tatarczuk, A., Hessel, V., Keil, F.J., et al., 2023. Evolution paths from gray to turquoise hydrogen via catalytic steam methane reforming: Current challenges and future developments. Renew. Sustain. Energy Rev. 183, 113392.

Saini, A., Panday, S., Gupta, N., et al., 2021. Polynomial based linear regression model to predict COVID-19 cases. In: 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology. RTEICT, IEEE, Bangalore, India, pp. 66–69.

Salehin, I., Kang, D.-K., 2023. A review on dropout regularization approaches for deep neural networks within the scholarly domain. Electronics 12 (14), 3106.

Sarker, I.H., 2021. Machine learning: Algorithms, real-world applications and research directions. SN Comput. Sci. 2, 160.

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

Schwenzer, M., Ay, M., Bergs, T., Abel, D., 2021. Review on model predictive control: An engineering perspective. Int. J. Adv. Manuf. Technol. 117 (5), 1327–1349.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1929–1958.

Tanç, B., Arat, H.T., Baltacıoğlu, E., Aydın, K., 2019. Overview of the next quarter century vision of hydrogen fuel cell electric vehicles. Int. J. Hydrog. Energy 44, 10120–10128.

Wallace, M., Pon Kumar, S.S., Mhaskar, P., 2016. Offset-free model predictive control with explicit performance specification. Ind. Eng. Chem. Res. 55 (4), 995–1003.

Wang, L., et al., 2009. Model Predictive Control System Design and Implementation Using MATLAB. Vol. 3, Springer.

Wei, J., Iglesia, E., 2004a. Isotopic and kinetic assessment of the mechanism of methane reforming and decomposition reactions on supported iridium catalysts. Phys. Chem. Chem. Phys. 6 (13), 3754–3759.

Wei, J., Iglesia, E., 2004b. Mechanism and site requirements for activation and chemical conversion of methane on supported Pt clusters and turnover rate comparisons among noble metals. J. Phys. Chem. B 108, 4094–4103.

Wei, J., Iglesia, E., 2004c. Reaction pathways and site requirements for the activation and chemical conversion of methane on Ru-based catalysts. J. Phys. Chem. B 108, 7253–7262.

Wei, J., Iglesia, E., 2004d. Structural and mechanistic requirements for methane activation and chemical conversion on supported iridium clusters. Angew. Chem. Int. Ed. 43, 3685–3688.

Wismann, S.T., Engbæk, J.S., Vendelbo, S.B., Bendixen, F.B., Eriksen, W.L., Aasberg-Petersen, K., Frandsen, C., Chorkendorff, I., Mortensen, P.M., 2019. Electrified methane reforming: A compact approach to greener industrial hydrogen production. Science 364, 756–759.

Wu, Z., Albalawi, F., Zhang, J., Zhang, Z., Durand, H., Christofides, P.D., 2018. Detecting and handling cyber-attacks in model predictive control of chemical processes. Mathematics 6, 173.

Wu, Z., Rincon, D., Christofides, P.D., 2019a. Real-time adaptive machine-learning-based predictive control of nonlinear processes. Ind. Eng. Chem. Res. 59, 2275–2290.

Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019b. Machine-learning-based predictive control of nonlinear processes. Part II: Computational implementation. AIChE J. 65 (11), e16734.

Wu, H., Zhao, J., 2020. Fault detection and diagnosis based on transfer learning for multimode chemical processes. Comput. Chem. Eng. 135, 106731.

Xiao, T., Wu, Z., Christofides, P.D., Armaou, A., Ni, D., 2021. Recurrent neural-network-based model predictive control of a plasma etch process. Ind. Eng. Chem. Res. 61, 638–652.

Xu, J., Froment, G.F., 1989. Methane steam reforming, methanation and water-gas shift: I. Intrinsic kinetics. AIChE J. 35, 88–96.

Yang, J., Zheng, W.X., Li, S., Wu, B., Cheng, M., 2015. Design of a prediction-accuracy-enhanced continuous-time MPC for disturbed systems via a disturbance observer. IEEE Trans. Ind. Electron. 62, 5807–5816.

Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks? Adv. Neural Inf. Process. Syst. 27, 3320–3328.

Zarzycki, K., Ławryńczuk, M., 2021. LSTM and GRU neural networks as models of dynamical processes used in predictive control: A comparison of models developed for two chemical reactors. Sensors 21 (16), 5625.

Zhang, H., Sun, Z., Hu, Y.H., 2021b. Steam reforming of methane: Current states of catalyst design and process upgrading. Renew. Sustain. Energy Rev. 149, 111330.

Zhang, B., Zou, G., Qin, D., Lu, Y., Jin, Y., Wang, H., 2021a. A novel encoder-decoder model based on read-first LSTM for air pollutant prediction. Sci. Total Environ. 765, 144507.

Zheng, Y., Wang, X., Wu, Z., 2022. Machine learning modeling and predictive control of the batch crystallization process. Ind. Eng. Chem. Res. 61, 5578–5592.

Zhou, Y., Jia, L., Zhang, Y., 2023. A transfer learning approach using improved copula subspace division for multi-mode fault detection. Can. J. Chem. Eng. 101 (12), 7015–7030.

Zhou, Y., Li, R., Lv, Z., Liu, J., Zhou, H., Xu, C., 2022. Green hydrogen: A promising way to the carbon-free society. Chin. J. Chem. Eng. 43, 2–13.