MACHINE LEARNING-BASED ADAPTIVE IDENTIFICATION OF NONLINEAR SYSTEMS: APPLICATION TO CHEMICAL PROCESSES

A Thesis

by

BHAVANA BHADRIRAJU VENKATA NAGA SAI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Joseph Sang-Il Kwon
Committee Members,	Costas Kravaris
	Eduardo Gildin
Head of Department,	Arul Jayaraman

May 2020

Major Subject: Chemical Engineering

Copyright 2020 Bhavana Bhadriraju Venkata Naga Sai

ABSTRACT

Recently, sparse identification of nonlinear dynamics (SINDy) has delivered promising results in identifying interpretable models using data for various process systems. However, SINDy cannot completely comprehend the dynamics of an evolving complex process without relying on impractically large data sets. Another important challenge is that at any instance of plant-model mismatch or process upset, re-training the model using SINDy is computationally expensive and cannot guarantee to catch up with rapidly changing dynamics. As a solution to this, a systematic procedure capable of identifying and predicting the nonlinear dynamics on the fly promises to provide a useful representation of the process model. Motivated by this, we propose an adaptive model identification framework that relies on the methods of sparse regression and feature selection. The proposed method is a three-step procedure: (1) identifying potential functions from a candidate library using SINDy, (2) updating coefficients of the identified model using ordinary least-squares regression, (3) selecting the most important features using stepwise regression. Initially, a baseline model is identified off-line using SINDy, and as a new data becomes available, the subsequent on-line steps are triggered based on a pre-specified tolerance to further update the model. Such an adaptive identification scheme facilitates in perceiving the model structure using a less amount of data than its off-line counterpart, SINDy.

Based on the previously proposed method, we further propose online adaptive sparse identification of systems (OASIS) framework to extend the capabilities of SINDy for accurate, automatic, and adaptive approximation of process models. The OASIS method combines SINDy algorithm and deep learning for system identification during online control of a process. First, we use SINDy to obtain multiple models from historical process data for varying input settings. Next, using these identified models and their training data, we build a deep neural network that approximates the functional relationship between SINDy coefficients and the state-input pairs in the training data. Once trained, the deep neural network is incorporated in a model predictive control framework for closed-loop operation. We demonstrate both the methods on a continuous stirred tank reactor.

DEDICATION

To my parents, grandparents and sisters, Sravya and Cherry.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my deepest gratitude and respect to my advisor Dr. Joseph Sang Il-Kwon for his endless support and encouragement throughout. His careful supervision has motivated me to be engaged in my research. I cannot thank him enough for his ideas, patience, availability, discussions, and thoughtful comments. His dedication and continuous guidance for all his students is remarkable, and I will cherish all that I have learned from him. I would also like to appreciate and thank my committee members, Dr. Costas Kravaris and Dr. Eduardo Gildin for all their assistance and timely help.

My greatest appreciation to Mr. Abhinav Narasingam and Mr. Mohammed Saad Faizan Bangi for all their guidance and help. Both of them are great mentors and I sincerely thank them for teaching me the fundamentals of my research. This work would have not been possible without their ideas, guidance, and constructive feedback. I also thank my research colleagues and friends Dr. Prashanth Siddhamshetty, Mr. Dongheon Lee, Mr. Hyun-Kyu Choi, Ms. Pallavi Kumari, Mr. Kaiyu Cao, Mr. Silabrata Pahari and Mr. Parth Shah for all their help and support both within and outside the research group. It is a memorable experience for me to work with such an inspiring research group.

A very special thanks to all my friends at Texas A&M, especially Nikhita, Debopamaa, Parth, Niranjan and Keerthi. Thank you for always being there for me and sharing this journey. I am also thankful to my beloved friends Vinaya, Aparna, Harika and Deva for all their care and encouragement.

Last but not the least, I owe my heartfelt gratitude to my parents, grandparents, sisters and cousins for all their love and care. All that I am today is because of them. I am forever indebted to my family.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Joseph Sang-Il Kwon [principal advisor] of the Department of Chemical Engineering and Texas A&M Energy Institute, Dr. Costas Kravaris of the Department of Chemical Engineering, and Dr. Eduardo Gildin of the Department of Petroleum Engineering.

All the work conducted for the thesis was completed by the student independently with support from group members.

Funding Sources

The authors gratefully acknowledge financial support from the National Science Foundation (CBET-1804407), the Department of Energy (DE-EE0007888-10-8), the Texas A&M Energy Institute, and the Artie McFerrin Department of Chemical Engineering.

NOMENCLATURE

SINDy	Sparse Identification of Nonlinear Dynamics
N4SID	Nonlinear algorithms for subspace state-space identification
MOESP	Multi-variable output-error-state-space
STLS	Sequentially Thresholded Least Squares
CSTR	Continuous Stirred Tank Reactor
MPC	Model Predictive Control
DNN	Deep Neural Network
ReLU	Rectified Linear Unit
ROM	Reduced Order Model
ODE	Ordinary Differential Equation

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Over the past decades, growing production technologies have contributed to a rise in complex processes across a major sector of industries. When dealing with such processes, first principles based modeling may be intractable and cannot be relied upon to discover the underlying governing equations, particularly when a strict constraint is imposed on the computational requirement [1]. This has guided several promising developments in the field of data-based system identification. The focus of system identification is to determine the structure of the model based on the input-output relations and provide an accurate future prediction [2]. In order to meet these goals, several data-driven methods have been established over the years based on equation-free modeling [3], artificial neural networks [4, 5, 6], empirical dynamic modeling [7], nonlinear Laplacian spectral analysis [8], automated inference from dynamics [9], and surrogate modeling [10] to name a few. Another important class of data-driven methods that have been in use for quite sometime, particularly in the areas of process control, is subspace identification. This includes methods like numerical algorithms for subspace state-space identification (N4SID) [11], multivariable output-error-state-space (MOESP) [12], and canonical variate analysis [13] which are competent in identifying simple state-space models for multivariable dynamical systems based on measured input-output data [14]. Owing to an easy accessibility to a vast amount of data and advancements in machine learning algorithms in recent times, these data-driven approaches are becoming more feasible and prominent.

Despite the proven success of the aforementioned "black-box" approaches in many applications, there has been an increasing shift in integrating data-driven methods with physics laws, especially in the case of dynamical systems. This is driven by a limitation of black-box models to extrapolate the dynamics to the entire state-space, beyond where they were sampled and constructed. Besides, it is necessary sometimes to develop a complete understanding of the physical mecha-

*Reprinted with permission from "Machine learning-based adaptive model identification of systems: Application to a chemical process" by Bhadriraju, B., Narasingam, A., and Kwon, J. S. 2019. *Chemical Engineering Research & Design*, 152, 372-383, Copyright 2019 by Elsevier.

nisms that govern the dynamical system of interest. Within this context, genetic programming-based symbolic regression has contributed in determining governing dynamics from data, along with giving an actual sense of the process [15, 16]. But in the case of large scale systems, symbolic regression can be prohibitively expensive and is prone to over-fitting. Some of the researchers have addressed these issues by developing system identification methods using sparse regression and compressive sensing. These techniques are based on the fact that only a few nonlinear terms are sufficient in determining the governing dynamics of a complex process. One such method that has recently received much attention is sparse identification of nonlinear dynamics (SINDy) algorithm developed by [17]. It has been extensively used for data-driven discovery of underlying dynamics by constraining the model structure based on a priori knowledge such as symmetries and conservation laws. The significance of SINDy has been widely researched in various fields. Some of the applications include rapid model recovery from abrupt system changes [18], simultaneous identification of both micro-scale and macro-scale dynamics [19], sparse learning of reaction kinetics [20], model-predictive control [21], developing reduced order models (ROM) for high-fidelity systems [22, 23], understanding rational function nonlinearities [24] and parameterized dynamics [25], discovering partial differential equations [26, 27], ranking the models based on Akaike Information Criterion (AIC) [28], Koopman operator based control [29, 30], and developing Galerkin regression models for fluid-flow [31]. Due to the ease of implementation and the ability to incorporate any known process knowledge, SINDy could be effectively applied for a large number of nonlinear dynamical systems. Additionally, several theoretical developments have also been established to show that the SINDy algorithm rapidly converges to a local minimizer under specific conditions [32].

1.1 Background on SINDy

This report presents a brief overview of the SINDy method and for more information, the readers can refer the original work by [17]. The SINDy algorithm is developed based on an assumption that out of all possible functions considered, only few of them govern the system dynamics. In accordance with that, a sparse regression problem is solved by balancing sparsity with accuracy.

This eliminates the intractable brute-force approach of searching for the right model among all the possible models in the given function-space. Let the governing process dynamics of the system under study be represented as

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{1.1}$$

where the vector $\mathbf{x}(t)$ denotes the states of the system at time t, $\mathbf{u}(t)$ is the vector containing the inputs applied to the system at time t, and $\mathbf{f}(\mathbf{x}, \mathbf{u})$ represents the governing equations describing the process dynamics. In some cases when the underlying dynamics are unknown and cannot be determined using physics laws, the function $\mathbf{f}$ has to be identified from measurement data. For finding the function $\mathbf{f}$, time-series data of state variables and applied inputs are collected. The time series data can be obtained either from physical sensor measurements or from numerical simulation of Eq. (1.1). The collected m snapshots of n state variables and their corresponding inputs $u(t)$ are arranged into matrices $\mathbf{X}$ and $\mathbf{U}$ as shown below.

$$\mathbf{X} = \begin{pmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{pmatrix} \quad \mathbf{U} = \begin{pmatrix} u_1(t_1) & u_2(t_1) & \cdots & u_n(t_1) \\ u_1(t_2) & u_2(t_2) & \cdots & u_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_m) & u_2(t_m) & \cdots & u_n(t_m) \end{pmatrix} \tag{1.2}$$

Next, the derivatives of state variables are either measured or numerically computed. When the time-series derivatives cannot be measured directly, they must be determined carefully for efficient working of SINDy. In general, this can be done by finite difference method. But in the presence of noise, it is suggested to use rigorous methods such as total variation regularized differentiation [33] or Knowles and Wallace variational method [34]. Using these computed derivatives, a matrix is constructed at different time points as follows:

$$\dot{\mathbf{X}} = \begin{pmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \cdots & \dot{x}_n(t_m) \end{pmatrix} \tag{1.3}$$

After obtaining the derivatives of state variables, the collected time-series data of $\mathbf{X}$ and $\mathbf{U}$ are utilized to build a candidate function library containing all possible potential functions as

$$\boldsymbol{\Theta}(\mathbf{X}, \mathbf{U}) = \begin{bmatrix} \mathbf{1} & \mathbf{X} & \mathbf{X}^2 & \cdots & \mathbf{U} & \cdots & \exp(\mathbf{X}) & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \cdots \end{bmatrix} \tag{1.4}$$

The choice of selecting the potential functions can be based on the knowledge of physics and prior information about the process. For example, as a majority of process models usually contain polynomial terms, it is useful to include them in the function library. In addition, it is recommended to populate the library with as many functions as possible like constant terms, trigonometric, and any other nonlinear functions so that the true process dynamics are well represented by the library with a higher probability. After evaluating time-derivatives of states and building the candidate function library, a regression problem is formulated as

$$\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X}, \mathbf{U})\boldsymbol{\Sigma} \tag{1.5}$$

where the vector $\dot{\mathbf{X}}$ denotes the time-series derivatives of state variables, $\boldsymbol{\Theta}(\mathbf{X}, \mathbf{U})$ is the library of possible potential functions representing the system dynamics, and $\boldsymbol{\Sigma}$ is the vector containing the function coefficients. However, solving the above problem directly using ordinary regression does not provide a parsimonious model. In order to promote sparsity in $\boldsymbol{\Sigma}$, Eq. (1.5) should be expressed in the form of a convex l_1-regularized regression as

$$\boldsymbol{\Sigma} = \underset{\boldsymbol{\Sigma}'}{\operatorname{argmin}} \|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X}, \mathbf{U})\boldsymbol{\Sigma}'\|_2 + \lambda\|\boldsymbol{\Sigma}'\|_1 \tag{1.6}$$

The above problem is solved using sequential thresholded least-squares (STLS) [17], which is similar to ordinary least-squares regression with an additional step of hard thresholding. The variable coefficients having values less than the thresholding parameter are rendered zeros and the regression problem is iteratively solved until convergence of parameter coefficients is attained. Please note that the parameter λ is crucial in eliminating the unwanted functions and its value can be evaluated using several hyperparameter tuning strategies such as grid-search [35], random search [36, 37] and Bayesian Optimization [38].

2. DATA-DRIVEN ADAPTIVE IDENTIFICATION OF NONLINEAR SYSTEMS

2.1 Motivation

Though SINDy proved to be successful in inferring the dynamics of various systems of interest, it holds the limitation of uncovering all of the underlying subtle dynamics for a complex process when only a small amount of data is available. Especially for the processes exhibiting complex nonlinear characteristics, the type regularly encountered in chemical sector, numerous samples may be required for obtaining an accurate model in the absence of enhanced sampling strategies. However, collecting such a large amount of measured data may be expensive and also, handling such massive data is computationally demanding. With this in mind, in this work, an adaptive identification method is proposed for identifying complex process dynamics with limited use of data. For nonlinear processes whose dynamics are time-varying and poorly understood, adaptive identification is a favorable approach. Most importantly, in the case of systems with parameter uncertainties and evolving dynamics, there is a need for adaptive modeling as a new data becomes available [39, 40, 41]. This is particularly useful because re-training the model may not be fast enough to cope with the real-time demands. Moreover, offline trained models can be significantly improved when they are simply updated using a new data. Therefore, for real-time applications, adaptive model identification helps in handling any plant-model mismatch that may occur during process operation. Recently, several methods contributing to the data-driven online model identification have been developed; in [42], the authors proposed an error-triggered online identification approach and in [43], a combination of event-triggered and error-triggered online identification mechanism based on recurrent neural network is discussed. Although these approaches are shown to be useful for model predictive control of real-time processes, they do not provide an interpretable model.

Apart from using a small amount of data, it is important to identify a model which is free of

*Reprinted with permission from "Machine learning-based adaptive model identification of systems: Application to a chemical process" by Bhadriraju, B., Narasingam, A., and Kwon, J. S. 2019. *Chemical Engineering Research & Design*, 152, 372-383, Copyright 2019 by Elsevier.

redundant and irrelevant variables [44]. Specifically, the features that do not contribute to the optimal model performance are considered irrelevant and the ones that are weakly relevant but can be replaced by other distinctive features are redundant. The presence of such features reduces prediction speed and accuracy. To tackle this issue, one can use feature selection techniques to eliminate the unwanted variables which do not significantly contribute to the process dynamics. These methods usually result in improved model prediction accuracy and reduced computational burden. Within the class of feature selection methods, numerous techniques are available and can be broadly grouped as filter, wrapper and embedded methods [45, 46]. Filter methods rank the features based on their correlation with the output without using any machine learning algorithm. Though these methods are computationally less expensive and evaluate the importance of each variable individually, they may fail in providing the best subset of variables as they do not actually train the model. On the other hand, wrapper and embedded methods select the best subset of features based on predictor performance. While wrapper methods use the combination of search strategies and modeling algorithm, embedded methods like LASSO [47] and RIDGE [48] have feature selection integrated within their algorithm. In this work, a wrapper-based stepwise regression is used as a part of the proposed framework [49, 50, 51].

Taking the above mentioned considerations into account, this work proposes an adaptive sparse identification method that identifies the emerging process dynamics of a complex system through a sequence of steps. First, a sparse model based on SINDy is identified offline using the initial data. In the next step, when the previous model fails to predict accurately, the coefficients of the identified functions are updated using ordinary least-squares regression. Finally, the identified model is updated by retaining only the essential features via stepwise feature selection. The main advantage of this sequential approach is that it requires a less amount of data for identifying a complex nonlinear dynamical system compared to SINDy.

The outline of this chapter is summarized as follows: In Section 2.2, a detailed description of the proposed methodology is presented, and in Section 2.3, application of the proposed method in identifying a highly nonlinear continuous stirred tank reactor (CSTR) model is described. In

Section 2.4, numerical simulations carried out to identify the CSTR dynamics using the proposed method are discussed. In the following subsections, the performance of the model identified by the proposed algorithm is analyzed, validated and then compared with the model identified by SINDy offline.

2.2 Adaptive identification of nonlinear systems

In this section, the proposed adaptive data-based model identification method is detailed. The method is executed according to the following three steps:

1. **Sparse model identification**: With the initial data available, a parsimonious model is obtained offline from a large set of candidate functions using SINDy.

2. **Re-estimation of regression coefficients**: As a new data becomes available, the coefficients of the identified functions are updated by performing ordinary least-squares regression.

3. **Feature selection**: Using stepwise regression, the best subset of functions is selected that represents the structure of the actual dynamics.

A flowchart representing each step of the proposed method is illustrated in Fig. 2.1. Instead of the conventional way of using SINDy for overall process model identification, the idea is to apply SINDy for identifying potential functions from a large library matrix. Note that, the first model is identified offline with a data available initially and is further improved online as a new data is available. At a point where the SINDy model fails, a new data is regressed onto the identified function library to update the values of the function coefficients. Furthermore, stepwise feature selection is implemented to develop a more accurate and computationally efficient model by selecting only the essential functions. As a preliminary step, it is recommended to pre-process the data for efficient regression analysis using standard techniques such as normalizing or filtering depending on the nature of the data. In the following subsections, each step of the adaptive identification method is discussed in detail.

Step 1: Sparse model identification

This is the first step in the proposed framework. In this step, a parsimonious model is obtained from some initial data (that may be collected at different operating conditions) using SINDy offline.

For example, this initial data can be obtained from process history. Identifying the correct functions along with their exact coefficients requires a large number of samples. Since Eq. (1.6) is solved only using limited data samples available initially, it is unlikely to realize an accurate model in this step. Therefore, the identified model will only be used to approximate the system until it diverges from the actual process dynamic behavior. To quantify the accuracy of the identified model, the relative error based on Frobenius norm can be used, which is calculated as

$$E(t) = \frac{\|x(t) - \hat{x}(t)\|_{fro}}{\|x(t)\|_{fro}} \tag{2.1}$$

where $\|.\|_{fro}$ denotes the Frobenius norm, $x(t)$ is the actual process value, and $\hat{x}(t)$ is the model predicted value. When the error evaluated between model prediction and process measurement exceeds a pre-specified tolerance ϵ, i.e., $E(t) > \epsilon$, Step 2 becomes functional.

Step 2: Re-estimation of regression coefficients

The objective of this step is to update the coefficients of previously determined functions. As a new data becomes available, this is done using ordinary least-squares regression. Specifically, a new library matrix is constructed considering only the functions identified in Step 1, and the new data is regressed onto this function library without any thresholding. As there is no thresholding, the regression process is computationally more attractive. Again, when the model obtained in this step performs poorly, Step 3 of the method is initiated.

Step 3: Feature selection

This step is aimed to further enhance the prediction accuracy by selecting only the essential features from the previously identified functions. For this purpose, a statistics-based approach of feature selection helps in selecting the best subset of variables without altering their representation, thus achieving a balance between model simplicity and goodness of fit. The model obtained in Step 2 is tested using stepwise forward and backward regression, by formulating a null hypothesis as

$$\gamma_j = 0 \tag{2.2}$$

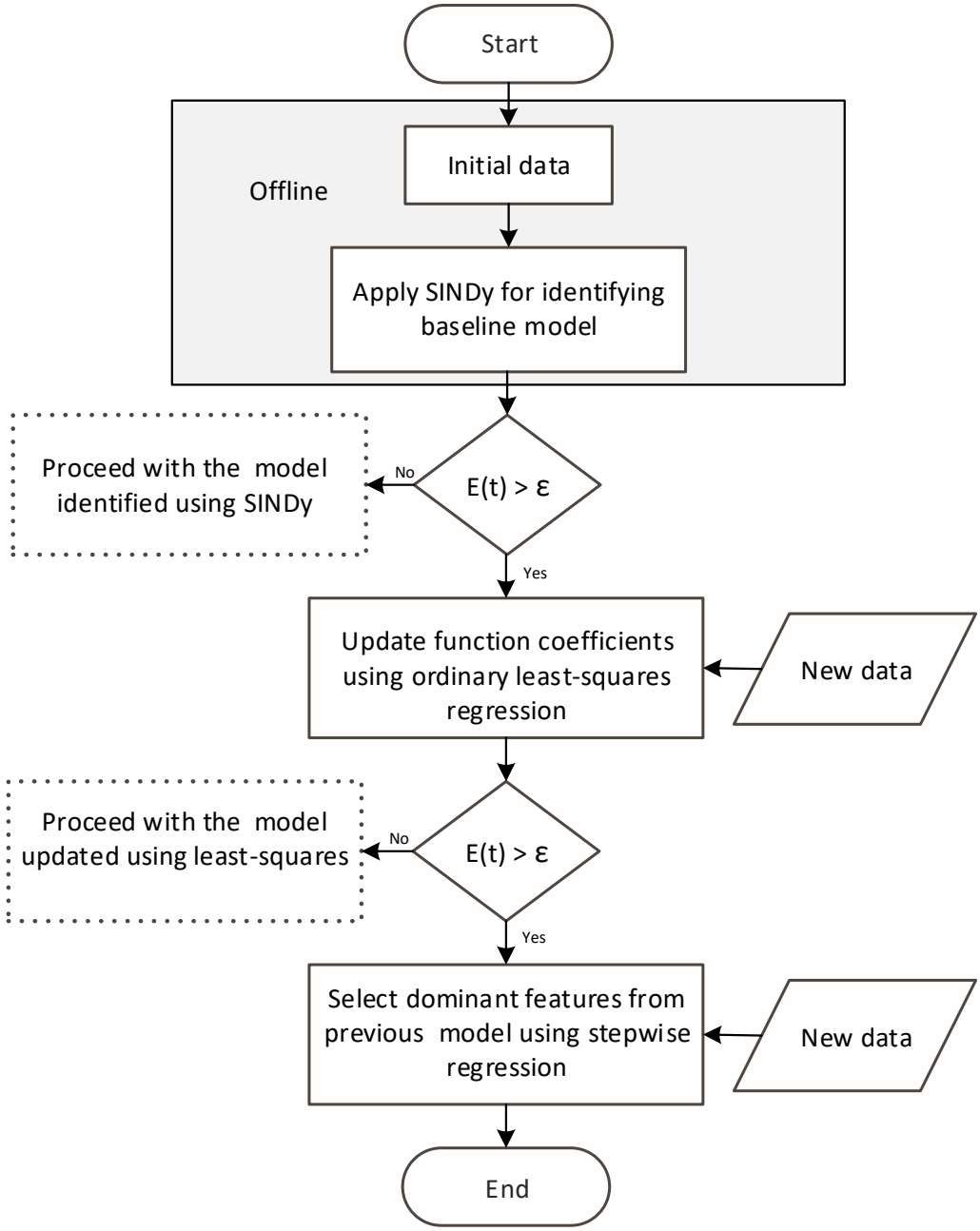


Figure 2.1: Flow chart of the proposed methodology.

where γ_j represents the estimated coefficient of a feature, z_j, considered for selection. In this work, the terms present in the model obtained from Step 2 represent the feature candidates available in

this step. Features are added or removed from the model based on a statistical criterion like p-value (the probability of a null hypothesis to be true), obtained from F-test. The order of adding features to the model is decided by measuring the correlation between the dependent variable, y, and the independent feature, z, as follows:

$$r_{zy} = \frac{\sum_{i=1}^{m}(z_i - \bar{z})(y_i - \bar{y})}{(\sum_{i=1}^{m}(z_i - \bar{z})^2)^{\frac{1}{2}}(\sum_{i=1}^{m}(y_i - \bar{y})^2)^{\frac{1}{2}}} \tag{2.3}$$

For m time-series samples considered, $\bar{z}$ and $\bar{y}$ are the mean values of the considered feature and the dependent variable, respectively. The most promising feature with the highest correlation coefficient is first added to the model and its statistical significance is examined using F-test. If this feature is significant, then the next features are added one at a time based on their partial correlation coefficient [52, 53]. At every step, the significance of all the previously selected features and the new feature to be added is evaluated. At any time during the evaluation, a previously added feature can be removed if it becomes insignificant in contributing to the desired prediction accuracy. If the p-value for a feature is less than the pre-specified significance level (i.e., α-value), then the null hypothesis is rejected indicating that the feature is valuable and is added to the model. The selection procedure stops when further addition or removal of features cannot improve goodness of fit. With this heuristic approach, only the most informative features are retained in the model, thus reducing the run time and complexity associated with more parameters. Please note that, for the cases with very few samples and more predictor variables, this particular selection technique may not deliver expected results always [54]. Fortunately, for most of the dynamic processes, the number of samples available is more than that of the predictor variables considered, including the application demonstrated in this work.

Remark 1. *For the cases where a large amount of data is available, often times it is possible to fully identify the original process model using Step 1 (SINDy) alone with a suitable value of λ (please refer the case studies presented in [17]).*

Remark 2. *Please note that measurement noise is not considered in this case study. However, in*

many practical applications, the measurement data is often contaminated by noise that may affect the performance of the proposed method. This can be addressed by denoising the data using the available nonlinear noise reduction techniques such as filtering and smoothing [55, 56, 57, 58]. Furthermore, the differentiation of data in Step 1 has to be carried out using robust methods [33, 34] that can compensate for noise. Additionally, it is important to implement a feature selection method which is robust to noisy data; for example, one can use a hybrid feature selection method combining both filter and wrapper methods [59]. In this work, stepwise regression was used as it was shown to perform well in the presence of measurement noise [60].

2.3 Application to CSTR

This section demonstrates the application of the proposed method for a perfectly mixed, non-isothermal CSTR. An exothermic, irreversible reaction A→B with the second order kinetics is considered whose reaction rate is given by

$$r = K\, C_A^2 \tag{2.4}$$

where K is the temperature dependent rate constant, and C_A is the time-varying concentration of reactant A. The reaction rate constant is determined by Arrhenius law as

$$K = K_0 exp\left(\frac{-E}{RT(t)}\right) \tag{2.5}$$

where K_0 is the pre-exponential factor, E is the activation energy of the reaction, R is the universal gas constant, and T is the time-varying reactor temperature in Kelvin. The temperature is maintained by adjusting the amount of heat transferred through the reactor jacket. The following equations obtained from mass and energy balance of the reactor are the mathematical representation of concentration and temperature dynamics in a CSTR. These equations are used to generate

simulation data for identifying the governing dynamics of the process.

$$\frac{dC_A(t)}{dt} = \frac{F}{V_r}(C_0 - C_A(t)) \; - K_0 exp\left(\frac{-E}{RT(t)}\right) C_A(t)^2 \tag{2.6}$$

$$\frac{dT(t)}{dt} = \frac{F}{V_r}(T_0 - T(t)) \; - \frac{\Delta H}{\rho c_p} K_0 exp\left(\frac{-E}{RT(t)}\right) C_A(t)^2 + \frac{Q(t)}{\rho c_p V_r} \tag{2.7}$$

In the above equations, F is the feed flow rate to the reactor, V_r is the reactor volume, ΔH is the heat of reaction, Q is the manipulated rate of heat input, and ρ and c_p are the density and specific heat capacity of the fluid in the reactor, respectively. The temperature-dependent rate constant and the coupled dynamics between temperature and concentration contribute to the complex nonlinear dynamics, making the process of system identification challenging. The objective of this case study is to develop an adaptive model that captures the evolving dynamics of concentration and temperature with a higher prediction accuracy. In the following section, the performance of models identified using the adaptive method and its offline counterpart, SINDy, is evaluated.

2.4 Simulation results

This section presents the results obtained from the numerical experiments carried out for model identification of the CSTR dynamics. The characteristics of the models developed using the proposed method and the original SINDy method are compared on the basis of prediction accuracy and the total number of data samples required. In this work, all the simulations were performed using MATLAB R2018b programming platform.

The input-output data required for training the models is generated by solving open-loop simulations of the mathematical models, Eq. (2.6) and Eq. (2.7), using the `ode45` solver. The process is subjected to a random heat input profile with signals varying between $-6 \times 10^4 \; KJ/h$ to $10 \times 10^4 \; KJ/h$. A simulation time step of $1 \times 10^{-6} \; h$ is considered within the solver, and the data is collected with a sampling time step of $1 \times 10^{-4} \; h$. Assuming full state measurements are available, the process outputs are C and T. The parameter values considered for numerical simulation are shown in Table 2.1. It is expected that the function coefficients of the identified model must be

Table 2.1: Parameter values for simulation.

Parameter	Units	Value
Flowrate, F	m^3/h	5
Arrhenius pre-exponential factor, K_0	$1/h$	8.46×10^6
Reactor volume, V_r	m^3	1
Gas constant, R	$KJ/Kmol{\cdot}K$	8.314
Inlet temperatue, T_0	K	300
Initial concentration, C_0	$Kmol/m^3$	4
Activation energy, E	$KJ/Kmol$	5×10^4
Enthalpy change, ΔH	$KJ/Kmol$	-1.15×10^4
Fluid density, ρ	Kg/m^3	1000
Specific heat, c_p	$KJ/Kg{\cdot}K$	0.231

approximately in the same range as the true values shown in Table 2.2. In the following subsections, the adaptive sparse identification and SINDy based models are identified, validated and then compared.

2.4.1 Adaptive model identification

As mentioned earlier, the proposed algorithm is a three-step method having different goals in each step as:

Step 1) Identify an initial set of potential functions.

Step 2) Update the identified function coefficients.

Step 3) Select the best combination of essential functions.

In Step 1, the original SINDy algorithm is applied to identify the governing functions of the concentration and temperature dynamics. To this end, a candidate library matrix is built with 22 functions as represented in Eq. (2.8). The advantage of SINDy, which is to incorporate a priori knowledge such as the temperature dependence of the rate constant via Arrhenius law, is utilized by including a temperature-dependent exponential term in the function library. The library developed

Table 2.2: True coefficients.

Functions	$\frac{dC_A}{dt}$	$\frac{dT}{dt}$
1	20	1500
C	-5	0
T	0	-5
$exp\left(\frac{-E}{RT}\right)C^2$	-8.46×10^6	4.21×10^8
Q	0	4.33×10^{-3}

in this step is an $m \times 22$ matrix, where m indicates the size of the time series data. In the subsequent steps, as the model gets updated, the column size of the library may vary. The simulated outputs of concentration, temperature, and the manipulated heat input are represented as x_1, x_2 and u, respectively.

$$\mathbf{\Theta}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} 1 & {x_i}^n & \exp\left(\frac{-E}{Rx_2}\right) {x_1}^2 & u & u^2 & x_i u & \sin(x_i) & \cos(x_i) \end{bmatrix} \tag{2.8}$$

In the above equation, the subscript i = $1, 2$ corresponds to the concentration and temperature variables, respectively, and n = $1, \cdots, 6$ indicates the degree of the polynomial. This step is performed offline using the available historical data of m = 5×10^3 samples. This historical data is obtained by simulating the process starting at initial conditions C = 1.9 $kmol/m^3$ and T = 400 K for a total duration of t = 0.5 h. Please note that, in this specific application, the concentration and temperature values are different by two orders of magnitude. This disparity in the scales can prompt poor sparsity, especially in the scenario of dealing with many functions. This issue is handled by normalizing the concentration data. Specifically, the concentration values are multiplied with the ratio of the mean values of temperature and concentration. Also, the magnitude of the exponential term in the candidate library is relatively higher than that of the other functions present, and this leads to a scaling issue. To solve this problem, each library element is divided by the mean of the corresponding library column. After pre-processing the data, the samples are

differentiated by finite difference method and are used to solve the sparse regression problem as presented in Eq. (1.6). For this purpose, the STLS method is used with 100 iterations to ensure optimum convergence of coefficients. The value of the thresholding parameter, λ, affects the degree of sparsity observed [61]. Different models are obtained for different values of λ and increasing λ results in a more sparse model. But as the degree of sparsity increases, many functions are disregarded and because of this, the error computed between the predicted value and the measured value increases. Therefore, the λ value is selected by balancing sparsity and accuracy. In this case, the relative error given by Eq. (2.1) is considered as a measure for quantifying model accuracy. As shown in Fig. 2.2, the Pareto front analysis gives the optimum value of thresholding parameter as 0.22. For this value of λ, the model identified in this step performs well with respect to training data as presented in Fig. 2.3.

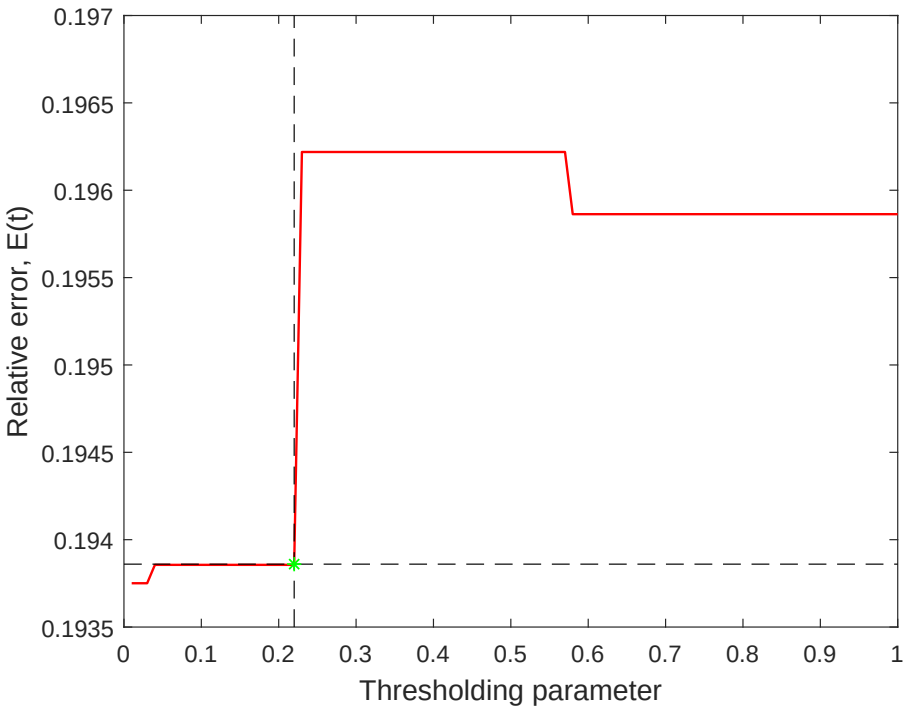
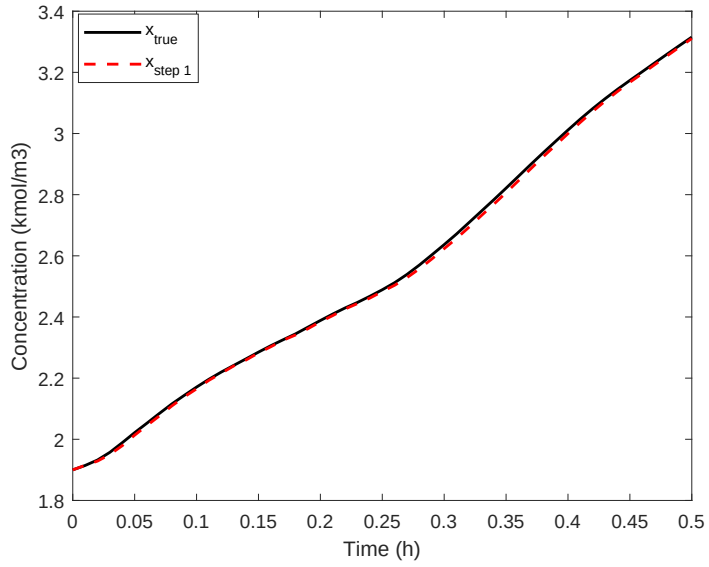
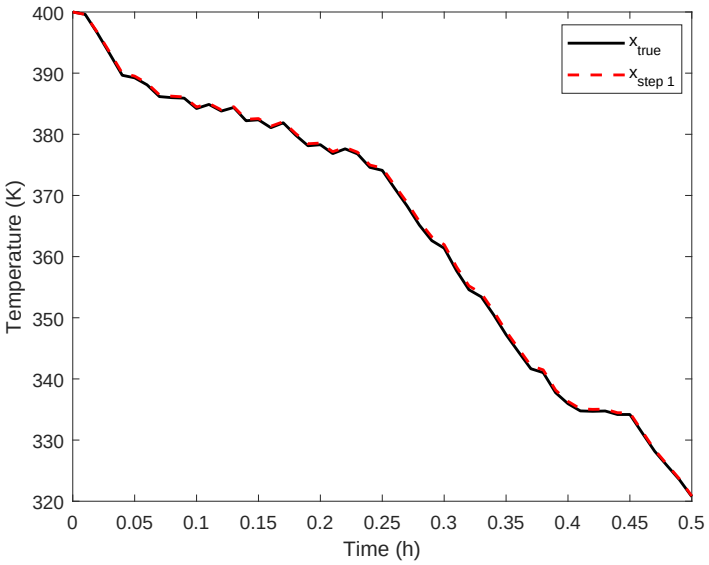


Figure 2.2: Relative error vs thresholding parameter.



(a)



(b)

Figure 2.3: Results obtained in Step 1 using the training data for (a) concentration, and (b) temperature profiles.

In reference to the model structure, Step 1 identifies only 10 out of 22 functions as the potential candidates and the results are shown in Table 2.3. From the table it can be observed that by solving Eq. (1.6), all the original functions present in Eq. (2.6) and Eq. (2.7) are correctly identified along

with some additional functions which are not part of the original system. However, the values of the identified function coefficients are not close to the actual model and thus, the model needs to be updated in the subsequent steps.

Once the model is obtained offline through Step 1, it is used as a baseline model to predict the dynamics of a process starting at initial conditions C = 3.31 $kmol/m^3$ and T = 320.75 K. An error tolerance value of $\epsilon = 5 \times 10^{-3}$ is considered for both temperature and concentration and any model having an error exceeding this value is deemed poor. The divergence point where the obtained model prediction deviates from the actual measurement serves as an indication to start Step 2. In Fig. 2.4(a), the model obtained from Step 1 predicts well from t = 0 h to t = 0.18 h, and after that it begins to deviate and can no longer be used for predicting the future states. The relative error computed between the predicted output and the measured data is illustrated in Fig. 2.4(b). It can be observed that the relative error exceeds the tolerance at t = 0.18 h and at this point, Step 2 of the proposed framework is initiated.

In Step 2, ordinary least-squares regression is performed for updating the coefficients of the previously identified functions. The amount of data utilized in this step is 5×10^3 samples, which are collected from the process between t = 0 to t = 0.18 h. The library matrix is re-constructed using only the 10 functions identified in Step 1 and their coefficients values are determined by solving Eq. (1.5), without any thresholding. The results obtained from Step 2 are presented in Table 2.4(a). From the table, it can be seen that the coefficients for concentration are nearly identical to the original model and T^4 term is observed to play no role in the reactor dynamics as seen from its zero coefficient value. Therefore, only the remaining 9 functions are taken into account for improving the model further. From Fig. 2.4(a) it can be observed that at t = 0.34 h the temperature profile deviates from the actual model, i.e., the error exceeds the pre-specified tolerance, ϵ, and this triggers Step 3. Additionally, the overall performance of the model updated through Step 2 is better than the model identified in Step 1, as can be seen from the plots depicted in Fig. 2.4(b). Note that, as the concentration prediction fits well with the actual behavior (Table 2.4(a)), Step 3 is performed to update the temperature model only.

Table 2.3: Sparse coefficients estimated in Step 1.

Functions	$\frac{dC_A}{dt}$	$\frac{dT}{dt}$
1	-85.937	-2.95×10^7
C	-0.30311	-9.99×10^4
T	1.781	4.95×10^5
C^2	-4.510	1.12×10^5
T^2	-0.012	-3458.71
C^3	2.294	-6.55×10^4
T^3	0	12.854
$exp\left(\frac{-E}{RT}\right) C^2$	-8.46×10^6	8.98×10^8
Q	0	4.14×10^{-3}
C^4	0	0
T^4	0	-0.0268
C^5	0	0
T^5	0	0
C^6	0	0
T^6	0	0
CQ	0	0
TQ	0	0
Q^2	0	0
$sin(C)$	0	0
$cos(C)$	0	0
$sin(T)$	0	0
$cos(T)$	0	0

Table 2.4: Regression coefficients estimated in (a) Step 2, and (b) Step 3.

(a) Model obtained in Step 2.

Functions	$\frac{dC_A}{dt}$	$\frac{dT}{dt}$
1	19.896	-1.05×10^5
C	-4.982	7.81×10^3
T	7.10×10^{-4}	1.02×10^3
C^2	-5.31×10^{-3}	-2345.24
T^2	0	-3.881
C^3	5.34×10^{-4}	237.284
T^3	0	6.05×10^{-3}
$exp\left(\frac{-E}{RT}\right) C^2$	-8.46×10^6	-6.41×10^8
Q	0	4.3×10^{-3}
T^4	0	0

(b) Model obtained in Step 3.

Functions	$\frac{dC_A}{dt}$	$\frac{dT}{dt}$
1	19.896	1311.4
C	-4.982	16.993
T	7.10×10^{-4}	-4.587
C^2	-5.31×10^{-3}	0
T^2	0	0
C^3	5.34×10^{-4}	0
T^3	0	0
$exp\left(\frac{-E}{RT}\right) C^2$	-8.46×10^6	4.06×10^8
Q	0	4.3×10^{-3}

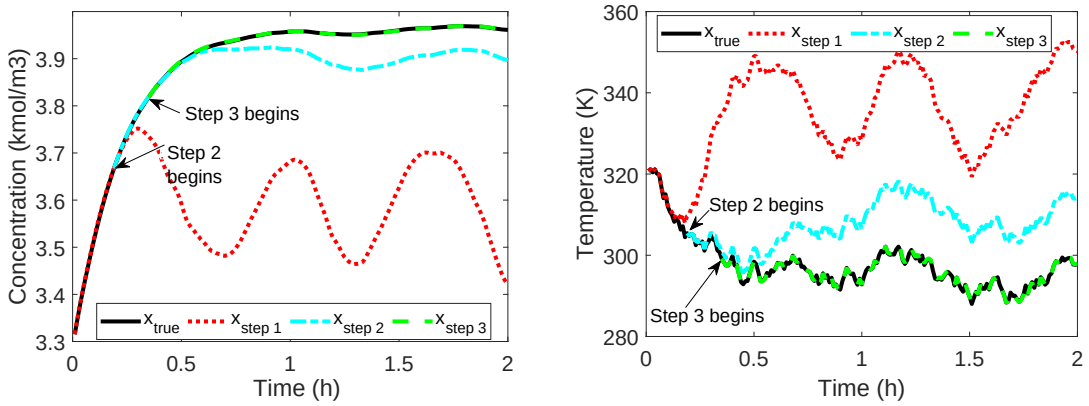
In Step 3, only the functions that contribute most significantly towards improved prediction accuracy are selected using feature selection. To do this, a feature matrix is constructed using the available 5×10^3 time samples till t = 0.34 h. In this case, the products of the 9 identified functions and their updated coefficients from Step 2 represent the features to be considered for selection. Within the feature selection algorithm, the standard significance level of α = 0.05 is specified as the criterion to reject the null hypothesis, i.e., a feature is added to the model if its p-value is less than 0.05. Among the 9 features considered, only 5 of them are selected in this step as the dominating ones (Table 2.4(b)), resulting in a more concise and efficient model. It can be observed from Table 2.4(b) that the identified model is very close to the true model both in terms of the governing functions as well as their coefficients. As shown in Fig. 2.4(a), both the temperature and concentration profiles obtained from Step 3 are nearly identical to the actual process, and the relative error of the updated model is within the tolerance limits (Fig. 2.4(b)). Thus, in terms of predictive performance, the model identified in Step 3 is superior to the previously obtained models as it contains essential features only.

In real-time applications, a major challenge of model adaptation is to rapidly update the model to capture all of the changing dynamics. Therefore, analyzing the computational time of each step is important. For the case study presented in this work, the model is updated from a point where it diverges from the actual behavior of the system. The computational times taken for updating the models in Step 2 and Step 3 of the proposed method are 0.137 s and 1.862 s, respectively. From the results it can be interpreted that in these steps the model structure is improved almost instantaneously since only a limited amount of data is used in both the steps. Note that, during this time of model update, the model identified in the previous steps continues to be in use until a new model is identified.

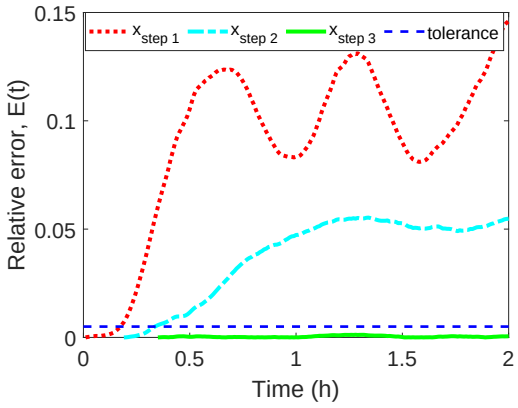
Remark 3. *The number of samples used for training the model is one of the most important factors for any data-driven model identification methods. In this work, multiple numerical simulations were performed considering varying number of data samples in order to train the model. However, not all of the results obtained are reported in this work. Only the results of the best model*

identified and the corresponding data size is discussed in the manuscript. The proposed algorithm is observed to perform well for any dataset having the size larger than this optimum value.

Remark 4. *Please note that the presence of noise may lead to frequent model updates if not handled appropriately. Once the data is cleaned using the previously mentioned techniques, the effects of measurement noise can be mitigated preventing frequent model updates.*



(a)

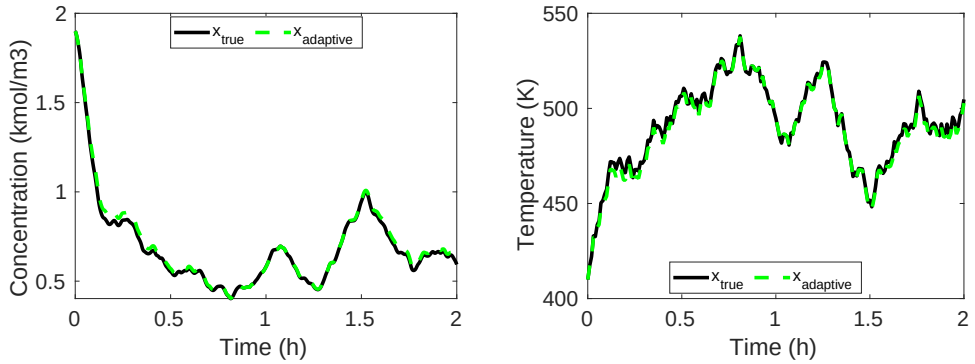


(b)

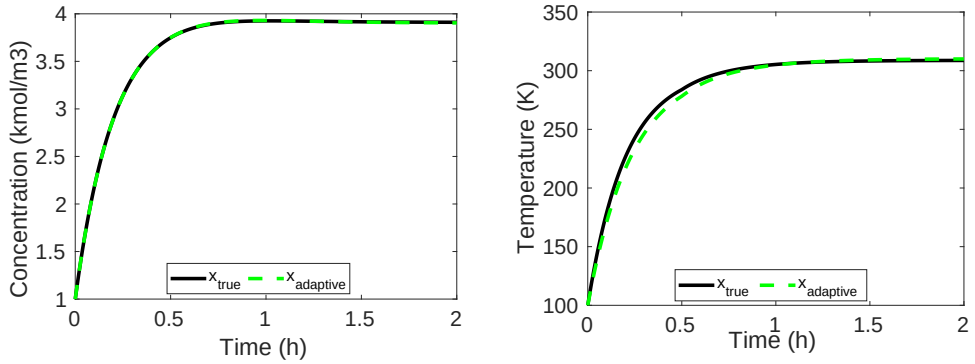
Figure 2.4: (a) Comparing the future behavior of individual models obtained using the proposed method for concentration and temperature profiles, and (b) relative errors of the models identified using the proposed method with respect to time.

2.4.2 Validation of the adaptive method

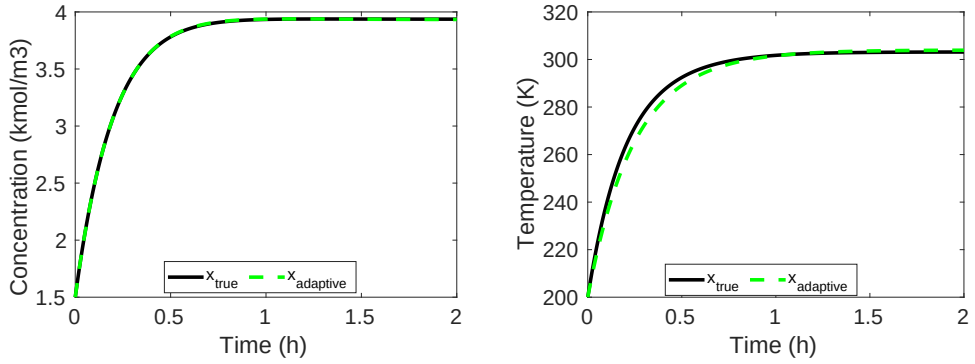
To evaluate the quality of the final model derived using the proposed method, it is validated against various sets of input profiles at different operating conditions.



(a) Random input.



(b) Step input.



(c) Sinusoidal input.

Figure 2.5: Open-loop validation of concentration and temperature profiles described by the adaptive model for three different input settings.

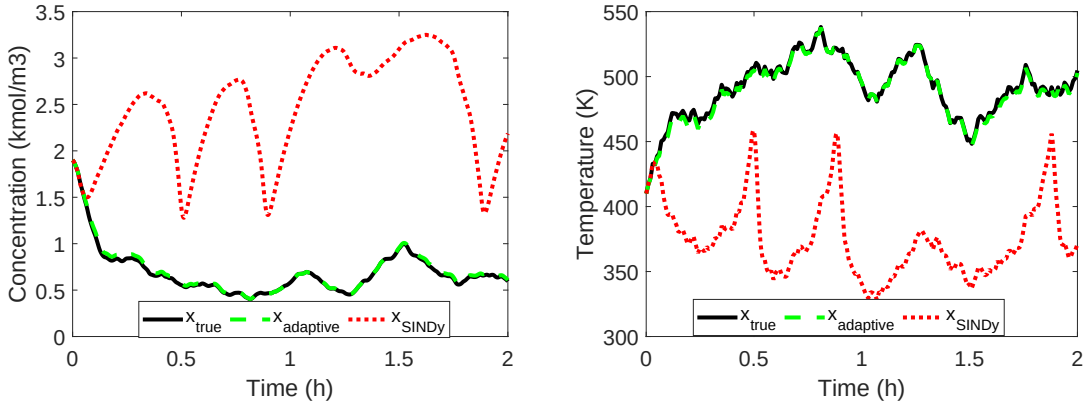
The validation datasets are generated using three kinds of heat input profiles; a random input with signals varying between -1.2×10^5 KJ/h and 2×10^5 KJ/h, an input with a step of 5×10^3 KJ/h, and a sinusoidal input with an amplitude and frequency of 6 KJ/h and 2 h^{-1}. The sampling and simulation time steps are the same as that of the training data, and the validation results are presented in Fig. 2.5. The results observed from the figure show that in all the three cases, the adaptive model predicts the process dynamics accurately.

2.4.3 Comparison with SINDy

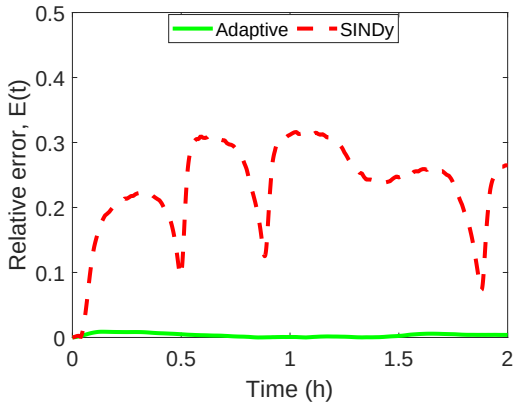
In this subsection, the proposed method is compared with SINDy in terms of future state prediction. In order to do this, the input-output dataset used in obtaining the adaptive model is considered. For the comparison to be consistent, the number of samples used to perform SINDy is taken to be the same as the total number of samples used for the proposed method (i.e., all three steps combined). The resulting sparse regression problem is solved using STLS with a thresholding parameter of 0.22, and 100 iterations are employed for proper convergence. Once the models are discovered by the proposed method and SINDy, their prediction performance is compared using validation data generated with a random input profile. From the results presented in Fig. 2.6(a), it can be observed that the model identified by SINDy fails to interpret the actual dynamics while the adaptive model (final model obtained after implementing Step 3) accurately represents both the concentration and temperature dynamics. Furthermore, the final model identified by the proposed method has a very low relative error at each time point (Fig. 2.6(b)). For the same number of samples, the proposed adaptive method could approximately predict the real model (Table 2.4), while an additional amount of data may be required for SINDy to fully identify the system. To generalize the validation results across different input settings, both the models are compared using 100 different random input profiles. The relative errors based on Frobenius norm are calculated for each of the input profiles and their average values are shown in Table 2.5, highlighting the superior performance of the proposed method.

Table 2.5: Average relative error computed for 100 different datasets.

Model	**Relative error**
Adaptive	3.60×10^{-3}
SINDy	2.51×10^{-1}



(a)

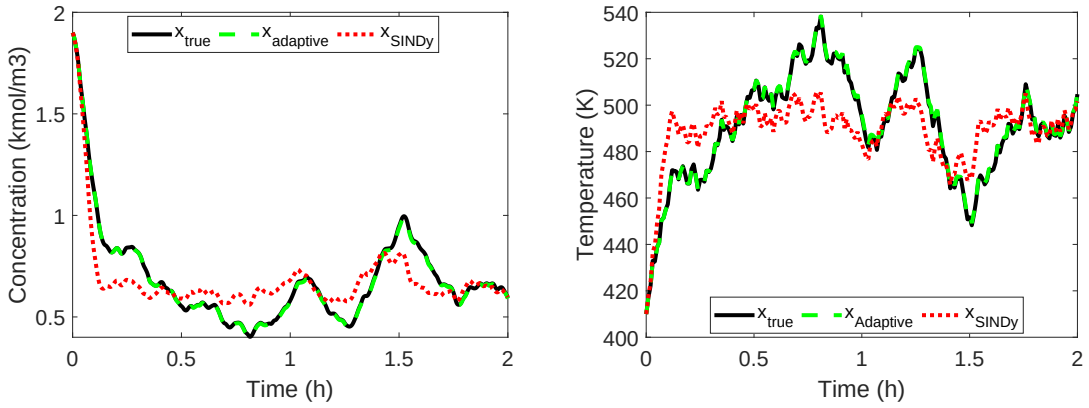


(b)

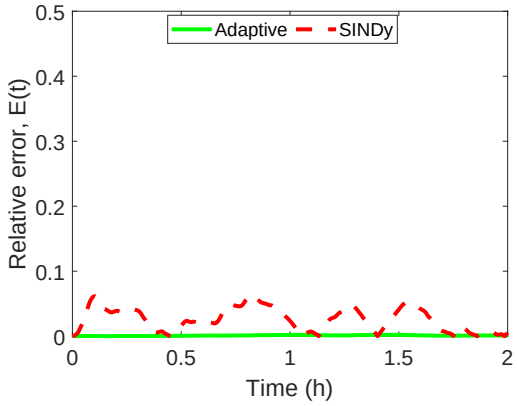
Figure 2.6: (a) Open-loop validation of the adaptive and SINDy models identified using the same number of samples for concentration and temperature profile, and (b) comparison of the relative errors for the adaptive and SINDy models identified using the same amount of data.

Additionally, the performance of both the methods is tested for the case when the SINDy model is trained using a large amount of training data. Specifically, 6 times the total number of samples used in identifying the adaptive model is used in training the SINDy model. In Fig. 2.7(a), the

response of the final model previously identified by the proposed method is compared with the model identified by SINDy with respect to the concentration and temperature variables. Also, the relative errors estimated at each time point for both the models are compared in Fig. 2.7(b). Although the performance of the model identified by SINDy using a high number of samples is improved when compared to the previous case (using a less number of samples), the accuracy of the adaptive model still outperformed SINDy. Overall, the results ascertain the usefulness of the proposed approach in obtaining a reasonable model using a less amount of data.



(a)



(b)

Figure 2.7: (a) Open-loop validation of the adaptive model and SINDy model identified using a large amount of data for concentration and temperature profiles, and (b) comparison of the relative errors for the adaptive model and SINDy model identified using a large amount of data.

3. ONLINE ADAPTIVE SPARSE IDENTIFICATION OF SYSTEMS (OASIS)

3.1 Motivation

Despite the simplicity of SINDy algorithm, it may require a large amount of data to discover the governing equations representing complex process dynamics. Therefore, it is challenging to use SINDy for model-based control, especially in the presence of any parameter uncertainties or changing process dynamics, because recovering the model by solving a sparse regression problem online is computationally expensive. To address this, we previously proposed an adaptive model identification framework that utilizes a small amount of data to identify and recover the model in real-time [62]. Though the method provides a direction to apply SINDy for adaptive modeling, it is useful to have a robust framework that guarantees to adapt well with the changing process dynamics. In line with this methodology, we now propose an algorithm that leverages the advantages of SINDy for online adaptive modeling using a deep neural network (DNN).

In the last two decades, DNNs have been widely used for various chemical engineering applications such as process control [63, 64, 65, 66, 67], fault diagnosis [68], system identification [69, 70, 71], sensor data analysis [72], and process design and simulation [73]. This success can be attributed to their ability to learn and approximate any underlying complex nonlinearities using a simple architecture. Moreover, advancements in parallel computing technology, and the ease of implementation supports the use of DNNs. Specifically, a lot of studies have been done to apply DNNs to model predictive control (MPC) as it is widely used in the industry because of its capabilities in dealing with output constraints and multi-variable processes [74]. In MPC, control action is prompted by solving an online optimization problem that requires a nonlinear model with high prediction accuracy and good generalization properties. To meet this demand, several researchers have incorporated DNNs within an MPC framework in different ways [75, 42, 43]. Generally, there are two ways to integrate a NN into an adaptive control structure [76]. One is a direct adaptive scheme which does not require a model; instead a DNN acts as a controller whose parameters,

the weights and biases, are updated online according to the control objective. However, this approach is challenging and time consuming when there are too many network parameters to handle [77]. The other method is an indirect adaptive scheme which uses a DNN to model the process, based on which the control action is determined [78]. In this case, both model update and control take place simultaneously within the MPC scheme. In this paper, we are particularly interested in applying DNNs for modeling and indirect adaptive control of nonlinear dynamical systems.

In this work, we propose an adaptive modeling and control procedure based on the SINDy algortihm and deep learning. The key novelty is to combine the usefulness of SINDy in discovering nonlinear dynamics with DNNs to adaptively model and control the process dynamics in real-time. The proposed method is implemented in two steps: system identification and controller design. For the system identification, we utilize several sets of process historical data that are available for various input settings and identify their corresponding models using SINDy. Next, we train a DNN using the previously collected historical datasets and their respective models such that the DNN approximates the relationship between process data and SINDy models. We use this trained DNN to design a controller wherein the DNN predicts the model to estimate the future behavior of the process. In this way, the proposed approach supports the application of SINDy for real-time prediction and control. For application purposes, we used the proposed online adaptive sparse identification of system (OASIS) framework to identify and control the nonlinear dynamics of a CSTR system.

The remainder of this chapter is structured as follows: in Section 3.2 a brief introduction of the SINDy algorithm is provided, followed by a short description of the DNN. Next, the proposed OASIS framework is presented in Section 3.3. In the following subsections, the results obtained from numerical simulations performed in identifying models via SINDy, developing DNNs, and designing a model-based controller for a CSTR are discussed.

3.2 Deep neural networks

A DNN consists of an input layer, an output layer, and a series of hidden layers that learn the input-output relationship in a dataset. Each layer contains multiple individual units called nodes

or neurons. The connection between two nodes in two consecutive layers is characterized by a parameter called weight, w. In a feed-forward DNN, as represented in Fig. 3.1, the input to a layer is the weighted sum of the outputs of the nodes in the previous layer. Another important parameter is bias, b, which is added to the weighted sum of the inputs to control the output of a node. Inside each node, its input is processed through an activation function, σ, in order to learn the underlying nonlinearities. Some of the popular activation functions are Hyperbolic tangent, Sigmoid, Rectified Linear Unit (ReLU), and Leaky ReLU. Suppose there is a particular layer p having $\mathbf{W}^p$ and $\mathbf{B}^p$ as its weight vector and bias vector, respectively. If the output vector from the previous layer is $\mathbf{O}^{p-1}$, then the output of the layer p, $\mathbf{Y}^p$, is given as

$$\mathbf{Z}^p = \mathbf{W}^p\mathbf{O}^{p-1} + \mathbf{B}^p \tag{3.1a}$$

$$\mathbf{Y}^p = \sigma(\mathbf{Z}^p) \tag{3.1b}$$

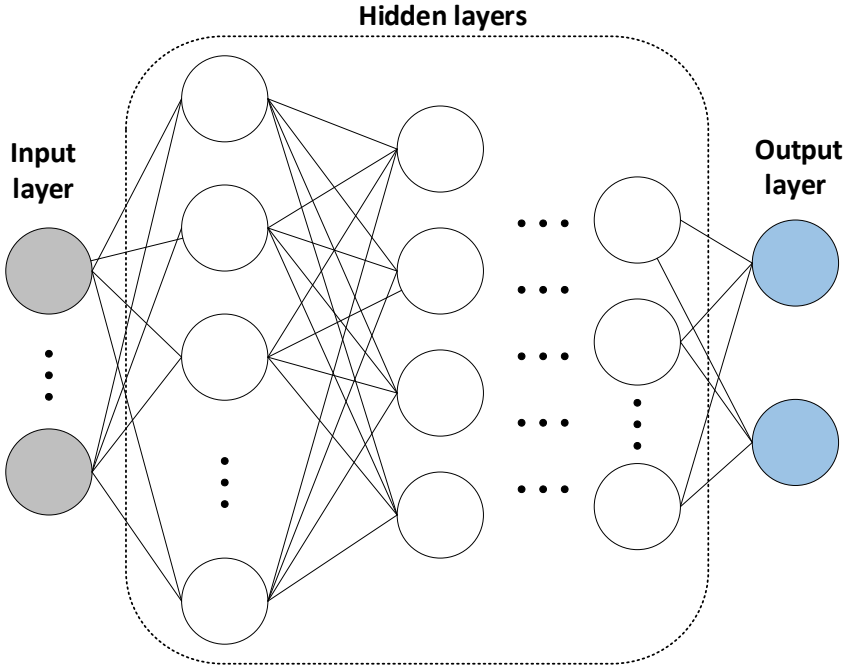


Figure 3.1: Deep neural network.

The accuracy of the network is evaluated based on the difference between the output predicted by the DNN and the actual output, called loss function, H. Hence, the objective during the DNN training step is to minimize this loss function. This is achieved by updating the weights and bias iteratively through an optimization algorithm. Each cycle of computing H, and updating w and b is called an epoch. There are many methods such as Gradient descent [79, 80], Newton method [81], Conjugate gradient [82], Quasi-Newton method [83], Levenberg-Marquardt algorithm [84, 85, 86], and Bayesian regularization [87, 88] that are popularly used as learning algorithms. Because of these advancements in learning algorithms, in recent years DNN-based models have performed better than the existing state-of-the-art models [89, 90].

3.3 OASIS methodology

The proposed OASIS framework combines the SINDy algorithm with deep learning to identify $\mathbf{f}(\mathbf{x}, \mathbf{u})$. The schematic of the OASIS method is presented in Fig. 3.2. The blue colored section in the figure represents offline training using SINDy and DNN, and the black one represents online implementation of the DNN for controller design. The system learning step is performed offline wherein we use n sets of historical time-series process data available either from experimental measurements or numerical simulations of high-fidelity equations. In this step, we identify several models offline, i.e., SINDy coefficients, using these multiple training datasets obtained at various operating conditions of the system and applied inputs. Please note that we use a limited number of samples for every case, and therefore, each of the models obtained by SINDy is not expected to be applicable beyond the training data (i.e., a typical extrapolation issue with any data-based models). On the contrary, considering multiple input trajectories improves the availability of models for a broad range of operating conditions. For an i^{th} dataset with $i = 1, \ldots, d$, let $\mathbf{X_i}$ represent the state variables, $\mathbf{U_i}$ be the applied inputs, and $\mathbf{\Sigma_i}$ be the model coefficients identified by SINDy. The SINDy algorithm is applied to these d datasets individually in order to identify function coefficients, $\mathbf{\Sigma_i}$, using which the corresponding model, $\mathbf{\Theta_i \Sigma_i}$, is obtained. We use these d pairs of $(\mathbf{X_i}, \mathbf{U_i})$ and $\mathbf{\Sigma_i}$ data to train the DNN that learns the relationship between them. For effective learning, it is recommended to consider a sufficient number of data samples.

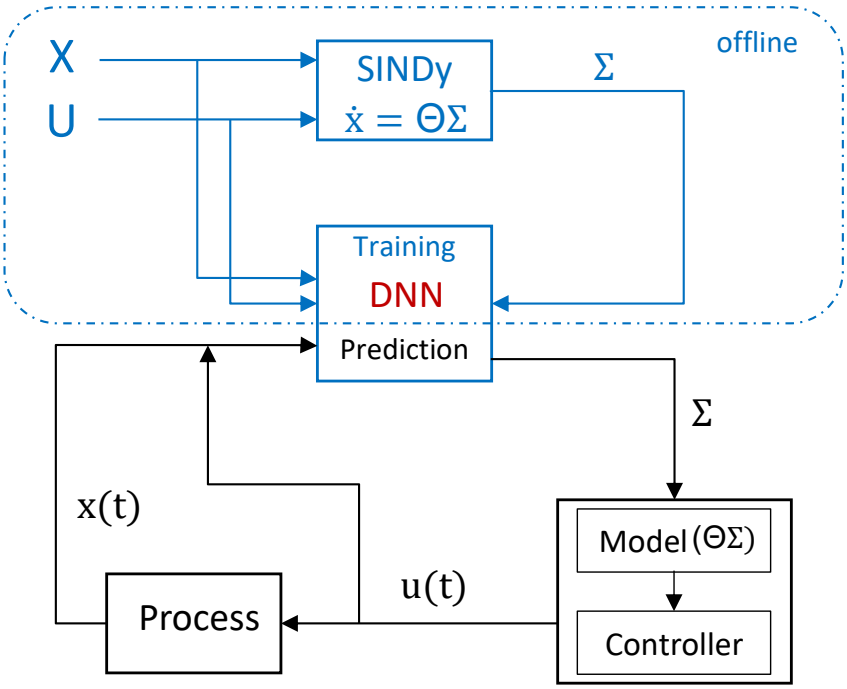


Figure 3.2: OASIS methodology.

Once the network is well-trained, it is used in a model-based controller to predict the future behavior of the process. The model predicted by the DNN is used within the controller to predict the states and calculate the control actions. As the process begins, a baseline SINDy model is determined by the DNN using the initial process conditions. This initial model is used for prediction until the next sampling time at which the current state, $\mathbf{x}(t)$, and the control input, $\mathbf{u}(t)$, are then fed to the DNN to update the model. The resulting model is further used in the next control step. In such a manner, the sequence of control action is obtained through the nonlinear models predicted by the DNN. Hence, the DNN serves as a tool to successfully use SINDy for adaptive modeling and control applications. The advantages of the proposed method are twofold: First, the nonlinear models predicted by the DNN continue to preserve the physical meaning of the process, since the DNN is developed using SINDy. Second, it is easy to update the models using the DNN online without loss of performance. The summary of the OASIS framework is briefly presented in

Algorithm 1.

Algorithm 1 OASIS methodology

1: Collect n sets of time-series data, $\mathbf{X}$ and $\mathbf{U}$.
2: Compute the derivatives of $\mathbf{X}$ and build a $\dot{\mathbf{X}}$ matrix.
3: Construct a candidate library, $\mathbf{\Theta}$, containing a variety of functions including any prior knowledge.
4: Solve the sparse regression problem, $\mathbf{\Sigma} = \underset{\mathbf{\Sigma}'}{\operatorname{argmin}} \|\dot{\mathbf{X}} - \mathbf{\Theta}(\mathbf{X}, \mathbf{U})\mathbf{\Sigma}'\|_2 + \lambda\|\mathbf{\Sigma}'\|_1$.
5: Perform the above steps for all the datasets to obtain multiple models for multiple inputs.
6: Train a DNN to learn the relationship between $\mathbf{\Sigma_i}$ and $(\mathbf{X_i}, \mathbf{U_i})$.
7: Design a model-based controller using the trained DNN that identifies and updates the models for process control.

Remark 5. *Please note that the optimization problem mentioned in Eq. (1.6) can be solved using sparsity promoting methods such as STLS, LASSO [47], and RIDGE [48]. The sufficient conditions related to the convergence of the SINDy algorithm are discussed in [32] The authors prove that the SINDy algorithm converges to an optimum solution in a finite number of steps. Additionally, in [91], the authors provided a condition to guarantee convergence when the data used in identifying governing equations are collected from multiple sources. Also, the convergence of sparse relaxed regularization problems is detailed in [92] Following such theoretical developments guarantees to achieve convergence of the sparse solution using SINDy. Hence, in our proposed method, SINDy can readily converge to a local minimizer for every dataset.*

3.4 Simulation results

In this section, we present the numerical experiments related to system identification and control of CSTR dynamics using OASIS. We first demonstrate the OASIS architecture in training the DNN using SINDy, and subsequently use the trained DNN to design a controller. All the simulations are performed using MATLAB R2018b programming platform. The data required to train the models using SINDy and to develop the DNN is numerically generated by solving Eqs. (2.6) -

(2.7) using the `ode45` solver. We used $1 \times 10^{-2}\ h$ and $1 \times 10^{-4}\ h$ as the measurement sampling and numerical integration time steps, respectively.

3.4.1 Model identification using SINDy

This is the first step in the proposed framework where we identify SINDy models for concentration and temperature dynamics using STLS algorithm offline. To generate training data, we introduced 100 random heat input profiles in the range of $-3.54 \times 10^5\ KJ/h$ to $1.96 \times 10^5\ KJ/h$ to the process for a total duration of $1\ h$. In this manner, 100 time-series datasets with varying trajectories are obtained. Also, using multiple trajectories ensures variation within the training datasets and is helpful in generalizing the DNN. After preparing for the data, we built a candidate library matrix using the following 9 functions:

$$\boldsymbol{\Theta}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_2^2 & x_1^3 & x_2^3 & \exp\left(\frac{-E}{Rx_2}\right) {x_1}^2 & u \end{bmatrix} \tag{3.2}$$

where x_1, x_2, and u represent concentration, temperature, and rate of heat input, respectively. Based on the prior knowledge of the system, i.e., the reaction rate constant exponentially varying with temperature, we included an exponential function in the library. The size of the library matrix developed is $m \times 9$ with m as 10000 in this case. In such a manner, 100 library matrices are developed for all of the datasets. For the case study presented, the order of magnitude of the exponential function in the library is relatively higher than the other terms present. This dissimilarity in scales affects the regression calculations. Hence, we normalized all the library elements to a same scale by dividing each element with the mean of the corresponding column in the library. In the next step, we calculated the derivatives using central difference method. This sequence of collecting and pre-processing data, building candidate library, and computing derivatives is repeated for all the 100 datasets. For each of the datasets, we solved the sparse regression problem presented in Eq. (1.6) iteratively using STLS algorithm until the coefficient values converge. Thereby, we identified 100 parsimonious models for all the 100 datasets. As discussed in Section 2, λ plays an important role in balancing sparsity with accuracy. The chance of not identifying the required

functions increases with λ, and this affects the prediction accuracy of the model. For this reason, we used 100 different values of λ between 0.01 to 10 to train the models using STLS. Amongst all the models obtained, we selected the one having the lowest value of the l_2-norm-based error, e, given as follows:

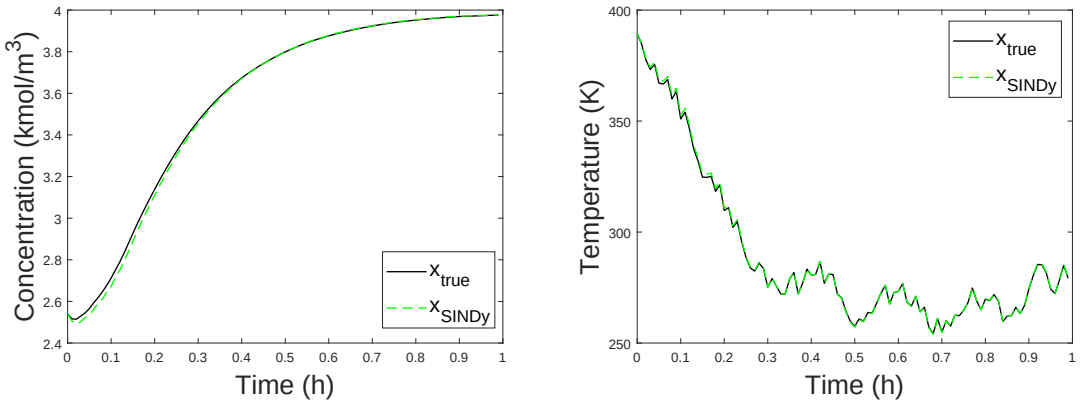
$$e = \sum_{i=1}^{m} \|\mathbf{x}(t_i) - \hat{\mathbf{x}}(t_i)\|_2 \tag{3.3}$$

In the above equation, $\mathbf{x}(t)$ denotes the actual states of the system as seen in the data, and $\hat{\mathbf{x}}(t)$ represents the model predicted value. Please note that each of the models is identified using 10000 data samples only. Hence, they may not approximate well for a different dataset which is not used in its training. To qualitatively prove this point, we selected a model from these 100 models and compared its prediction performance when an input other than its training input is applied to it.

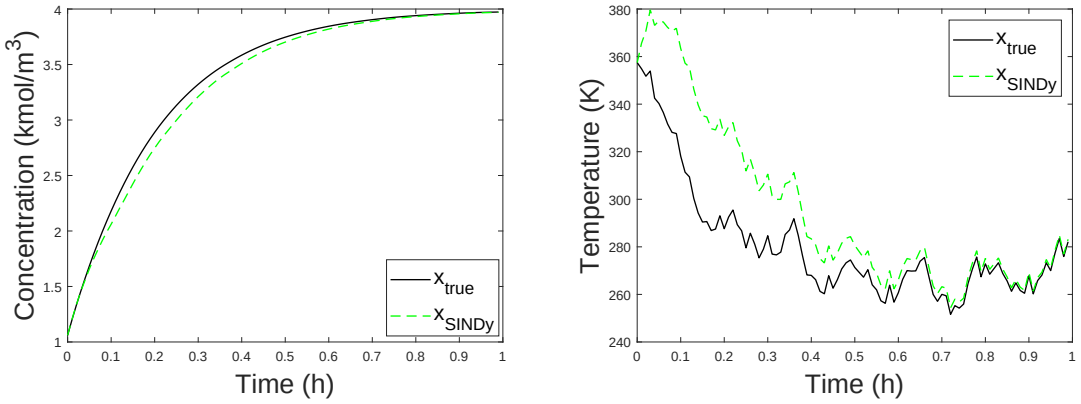
The results are presented in Fig. 3.3, and it can be observed that the model performs well in replicating the dynamics of its training dataset used in its identification, but could not approximate the dynamics of the test dataset to a satisfactory degree. Therefore, the 100 models, derived using SINDy are not completely identical to each other and this variation is advantageous in training the DNN.

3.4.2 Training the DNNs

In this step, the objective is to build a DNN that can predict the SINDy coefficients online, precisely describing the changing process dynamics. For the purpose of DNN training, we utilized the 100 models identified using SINDy and their corresponding training states and inputs. These datasets are normalized before using them for DNN training. Instead of building a single DNN that predicts all the SINDy coefficients, we developed two feed-forward DNNs, i.e., one each for the concentration and temperature variables. This is because having individual structures for each of the states enables the DNN to learn more effectively. Please note that we used the structure with 5 layers including 1 input layer with 3 nodes, 1 output layer with 9 nodes, and 3 hidden layers with 2, 10, and 10 nodes, respectively. From each of the 100 datasets, we selected 100 data points, and thus, a total of 10000 samples were utilized in training the DNNs.



(a)



(b)

Figure 3.3: Performance of a model identified by SINDy with respect to the (a) same training input used in its identification, and (b) testing input.

These samples containing the time-series data of C_A, T, and u are the training inputs to the DNNs. The training outputs are the coefficient values of the 9 functions representing the SINDy models. Hence, the size of input and output training data for both the DNNs is 3×10000 and 9×10000, respectively. For every input, i.e., C_A, T, and u, the DNN predicts the SINDy coefficients as illustrated in Fig. 3.4. Once the structure of the DNNs is defined, we trained both the DNNs individually using MATLAB's Deep Learning toolbox. To begin the DNN learning, the parameters w and b are randomly initialized, and an initial output is predicted. Based on the output, loss function is computed and accordingly the parameters w and b are further updated using Bayesian regularization algorithm. Generally, the training process aims at minimizing the

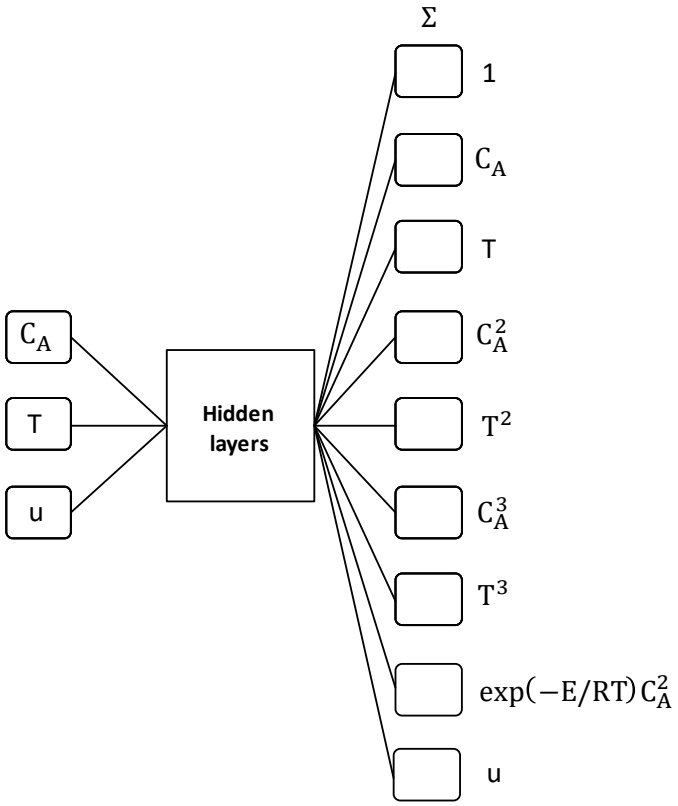


Figure 3.4: Structure of the DNN.

sum of squared errors as the loss function. But the Bayesian regularization algorithm includes an additional regularization term containing the sum of square of weights in the loss function, H, as

$$\mathrm{H} = \sum_{i=1}^{k} \alpha(\mathbf{x_i} - \hat{\mathbf{x}}_\mathbf{i})^2 + \beta {w_i}^2 \tag{3.4}$$

where α and β are the parameters that decide the relative importance of terms in the loss function, $\mathbf{x_i}$ and $\hat{\mathbf{x}}_\mathbf{i}$ are the true and predicted values, respectively. This step of penalizing weights improves the DNNs' ability to generalize well for new inputs. Moreover, having small weights will allow the response of the network to be smooth [93] and prevents the network from over-fitting. Bayesian

regularization follows the same update rule as Levenberg-Marquardt algorithm given as

$$w_{i+1} = w_i - (J_i^T J_i + \mu I)^{-1} J_i E_i \tag{3.5a}$$

$$b_{i+1} = b_i - (J_i^T J_i + \mu I)^{-1} J_i E_i \tag{3.5b}$$

where J is the Jacobian matrix containing the gradients of loss function with respect to network parameters; specifically, μ is the combination coefficient, and I is the identity matrix. The main difference of this algorithm from the Levenberg-Marquardt algorithm is that the former aims at minimizing a linear combination of the sum of squared errors and the sum of squared weights, while the latter minimizes only the sum of squared errors. To activate the network layers, we used sigmoid function for hidden layers and linear function for the output layer. One important point to note is that we trained the DNNs by feeding the data in a series of 5 batches with 2000 samples (i.e., 2000 epochs per batch). This is because training in large batches or using all the data at once does not improve the performance of the model and can cause over-fitting [94, 95]. On the other hand, training in small batches combines the information from both old and new data, and has an advantage of faster convergence [96]. Such a method of sequential training improves the learning process, and this can be seen in Fig. 3.5 which shows the performance of the DNNs for the training data. The prediction accuracy is evaluated by calculating relative error, RE(t), based on l_2-norm as

$$\text{RE(t)} = \frac{\|x(t) - \hat{x}(t)\|_2}{\|x(t)\|_2} \tag{3.6}$$

It can be observed that both the DNNs for C_A and T perform well in reproducing the dynamics for training data. Note that these results only indicate the learning curve of the DNNs and do not prove the generalizing abilities of the networks. We evaluate the generalization properties of the DNNs using test data as explained in the following section.

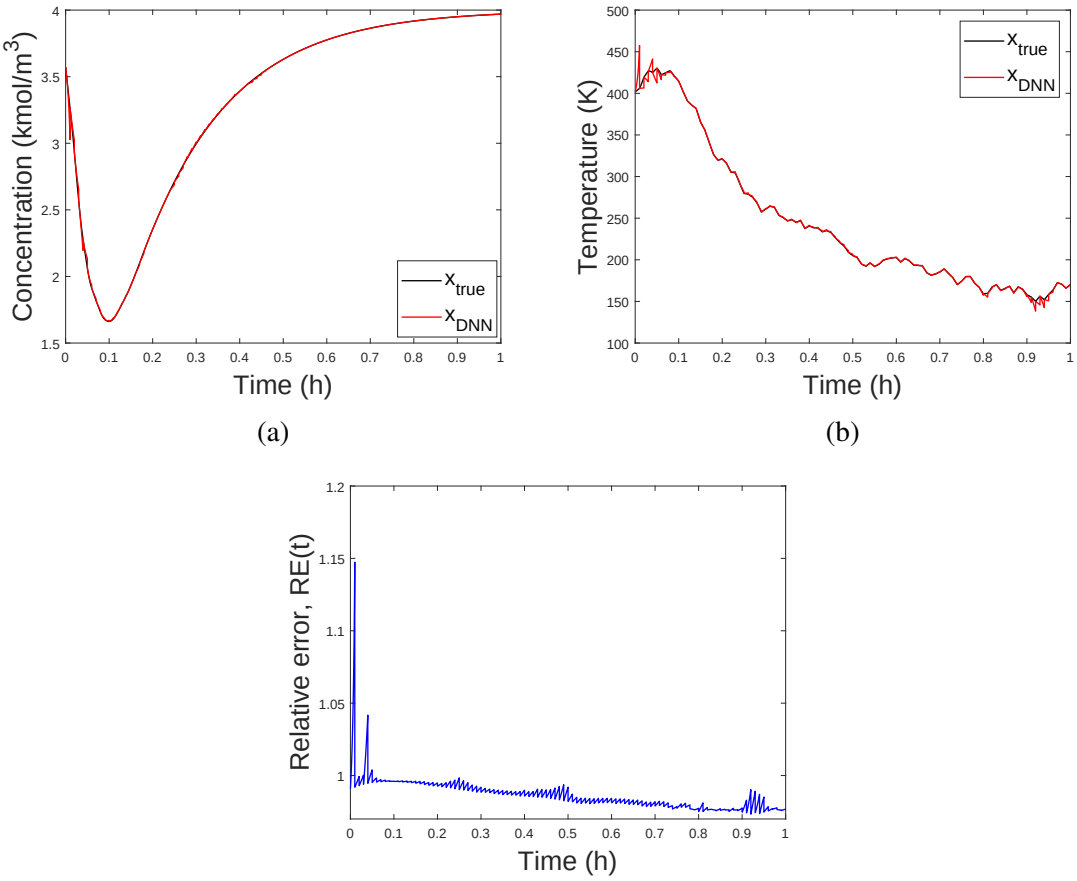
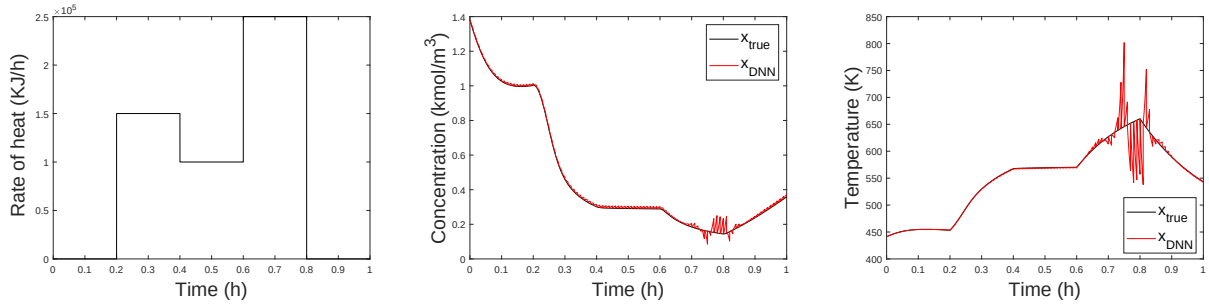


Figure 3.5: (a) Prediction of concentration and temperature dynamics using the DNNs with respect to the training data, and (b) the relative error of the DNNs' predictions with respect to the training data.

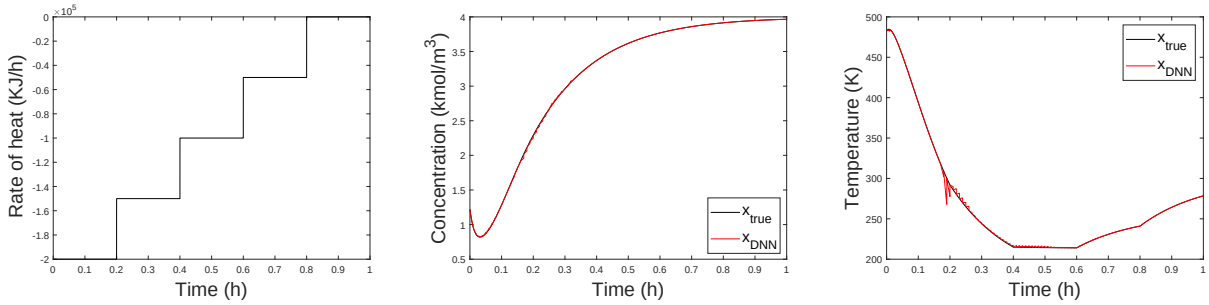
3.4.3 Testing the DNNs

To check the performance of DNNs, we tested the networks using data generated at different operating conditions. The test data are obtained by solving Eqs. (2.6)-(2.7) using a step change in heat input, with the simulation and sampling times identical to the training data.

The main objective of network training is to achieve local generalization. In this work, techniques such as regularization-based learning algorithm and small-batch training helped with the generalization in terms of interpolation. The ranges of C_A, T, and u in the training data are presented in Table 3.1. Based on this range, we evaluated the performance of DNNs, and the results



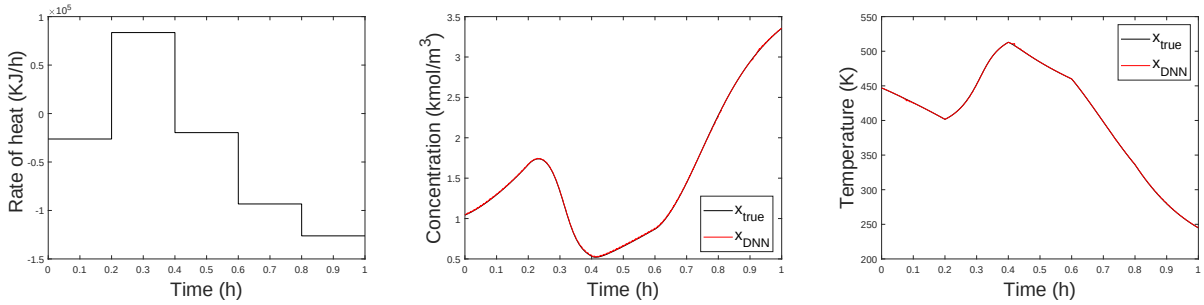
(a)



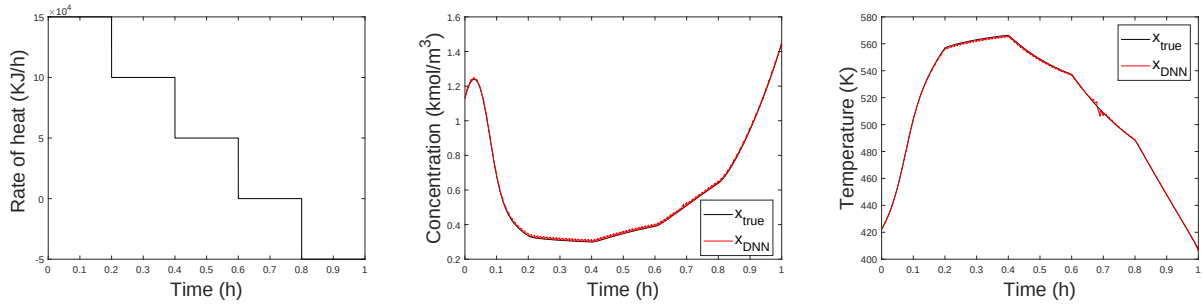
(b)

Figure 3.6: (a) Validation of the performance of the models obtained from DNNs (a) when the rate of heat input violates the upper bound of the constraints, i.e., 1.5×10^5 KJ/h (due to the high value of heat input, the temperature may exceed training values for the initial conditions considered), and (b) when the rate of heat input violates the lower bound of the constraints, i.e., -1.5×10^5 KJ/h.

are presented in Figs. 3.6-3.7. From Fig. 3.6 it can be seen that there are fluctuations in the predicted output. This happens because high rates of heat input can cause the reactor temperature to go beyond the training data span, i.e., $589\ K$. Furthermore, prediction inaccuracies are observed in the case of lower values of heat input. Hence, to ensure the states remain within the training range and improve the prediction accuracy, we selected the constraints for heat input as $u \in [-1.5 \times 10^5, 1.5 \times 10^5]$. Consequently, these constraints will be used in designing a model-based controller, which is discussed in the next section. The test results of the DNNs with respect to the input constraints are presented in Fig. 3.7. It can be observed that in both the test cases, the DNNs perform well with respect to their interpolation within the span of the training samples. This proves the generalization abilities of the DNNs in predicting the nonlinear dynamics when inputs other than the training inputs are given to the system. In the following section, we present the use



(a)



(b)

Figure 3.7: Evaluating the performance of the models obtained from DNNs when the rate of heat input is in the range of [-1.5 $\times$ 10^5, 1.5 $\times$ 10^5] KJ/h.

of these trained DNNs in controlling the reactor dynamics. Please note that at every sampling time, the feedback from the process is utilized by the DNNs to update the models for all the prediction results presented in this work (Fig. 3.8).

3.4.4 Model predictive control

In this section, we demonstrate the application of the OASIS framework in designing a model predictive controller for the CSTR system described earlier. In the first step of OASIS, we trained DNN architectures using SINDy offline. In the second step, the obtained DNNs are used online to predict the SINDy coefficients, using which the controller takes action to move the process towards the desired set-point value. The schematic of DNN-based predictive control is illustrated in Fig. 3.8.

In this case study, the control objective is to operate the reactor at its unstable steady-state condition, T_s = 401.87 K and C_{As} = 1.95 $Kmol/m^3$. To meet this objective, we formulated the

Table 3.1: Training data range for DNN inputs.

Variable	**Units**	**Range**
Temperature, T	K	[142, 589]
Concentration, C_A	$Kmol/m^3$	[0.26, 4]
Rate of heat input, u	$KJ/Kmol$	$[-3.54 \times 10^5, 1.96 \times 10^5]$

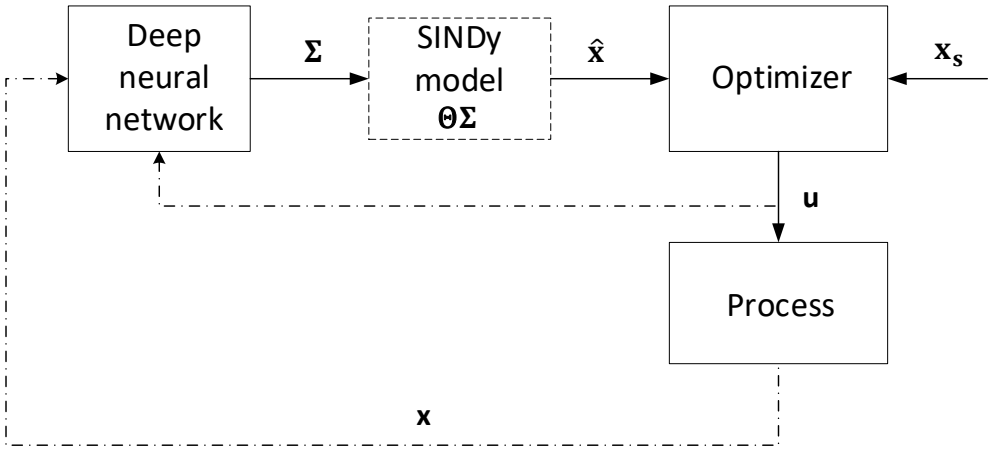


Figure 3.8: Model predictive control using the proposed OASIS algorithm.

MPC problem as follows:

$$\min_{u(k),\ldots,u(k+N-1)} \quad \sum_{j=1}^{N} (\hat{\mathbf{x}}(k+j) - \mathbf{x_s})^{\mathbf{T}} \mathbf{Q_c} (\hat{\mathbf{x}}(k+j) - \mathbf{x_s}) \tag{3.7a}$$

$$\text{s.t} \quad \boldsymbol{\Sigma}(k) = \text{DNN}(\mathbf{x}(k), u(k-1)) \tag{3.7b}$$

$$\hat{\mathbf{x}}(k+j) = \boldsymbol{\Theta}(\hat{\mathbf{x}}(k+j-1), u(k+j-1))\boldsymbol{\Sigma}(k), \qquad \forall j = 1, \ldots, N \tag{3.7c}$$

$$u_{min} \leq u(k+j-1) \leq u_{max}, \qquad \forall j = 1, \ldots, N \tag{3.7d}$$

where $\hat{\mathbf{x}}$ and $\mathbf{x}$ are the vectors containing the states available from the DNN models and process,

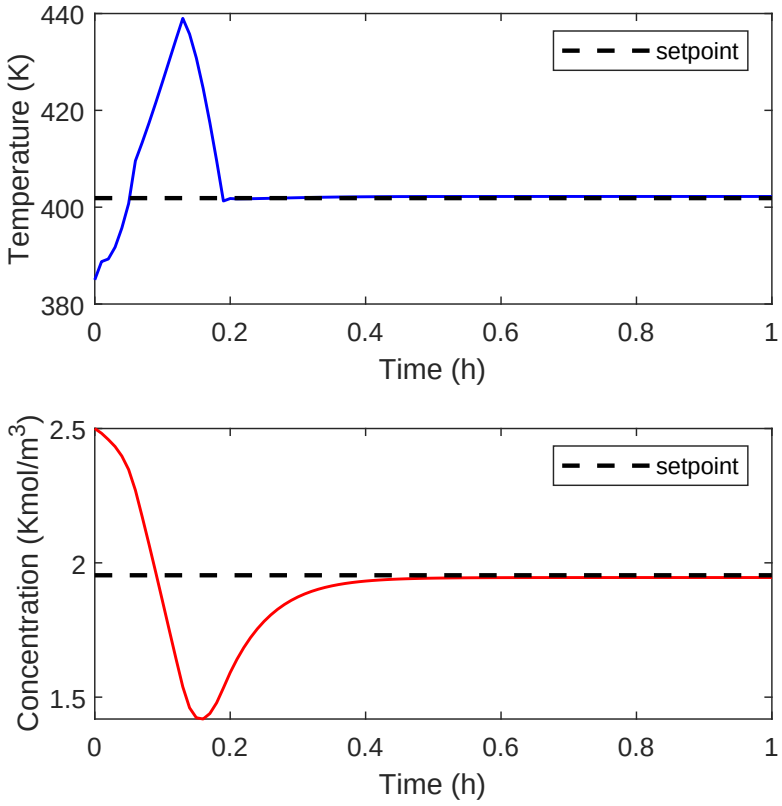


Figure 3.9: Closed-loop simulation results under the proposed OASIS-based MPC.

respectively, $\mathbf{x_s}$ are the desired set-point values for C_A and T, $\mathbf{Q_c}$ is a positive definite weighting matrix that weighs the importance of the states variables, k is the current time step, and u is the rate of heat input which is manipulated to control the state variables. We considered the above problem as a receding horizon MPC with horizon length as N at any step k. Also, the control horizon is the same as the prediction horizon. In the above mentioned formulation, the objective of the optimization problem is to minimize the squared deviation of process variables from their desired set-point values. The optimization problem is solved utilizing the models predicted by the pre-trained DNNs for C_A and T states. We selected the constraints for MPC based on the span of the training data. Specifically, the input should vary between the minimum and maximum values, i.e., u_{min} and u_{max}. The optimization problem in Eq. (3.7) is solved in a closed-loop manner with the sampling time of 1×10^{-2} h for a total duration of 1 h. The MPC is initialized with initial

conditions for C_A and T as 2.5 $Kmol/m^3$ and 385 K, respectively. At every sampling time, the measurement values are collected by solving Eqs. (2.6)-(2.7) using the parameters mentioned in Table 2.1. These measurements and control inputs are fed to the DNNs to obtain an updated SINDy model (as in Eq. (3.7b)) which is used in the MPC (as in Eq. (3.7c)) to compute the control inputs until the next sampling time. Specifically, at every step k, the MPC is evaluated for a prediction horizon of length $N = 10$, computing an input profile as $[u(k), \ldots, u(k+N-1)]$. The first input, $u(k)$, is then applied as the control action to the CSTR process, and the SINDy model is re-estimated at the next sampling time. This procedure is followed throughout the duration of the process for every sampling time.

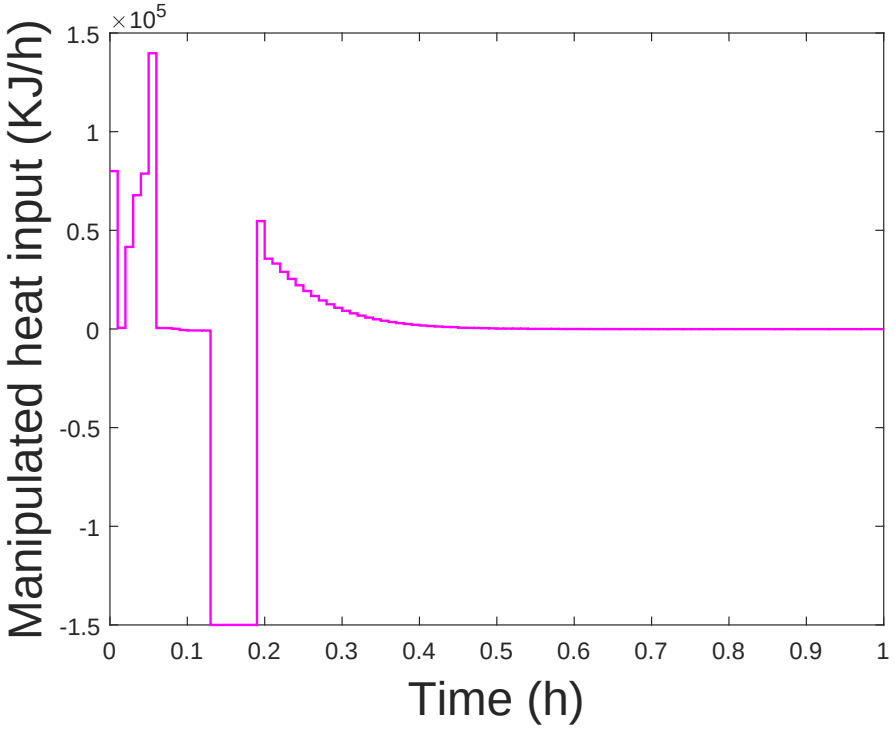


Figure 3.10: The manipulated heat input profile under the proposed MPC.

From the closed-loop simulation results in Fig. 3.9, it can be observed that the designed controller could drive both the states to their target set-points. This is due to the fact that the DNNs

performed reasonably well in updating the models with respect to feedback from the process. The corresponding heat input profile that is manipulated to achieve steady-state operation is presented in Fig. 3.10. These results proved the capabilities of the OASIS framework in predicting nonlinear systems for a defined window of operating conditions.

Remark 6. *Please note that in practical applications, the measurements are usually corrupted by noise, and therefore, it is important to have a model that counters the effect of noise. Fortunately, the proposed method has the potential to perform well even in the presence of noise when compared to the methods that solely rely on neural networks. This is because in the OASIS framework, we use the SINDy algorithm to obtain sparse models and it was proven that models identified by sparse regression methods such as SINDy and its variants are robust to noise [97, 98, 21]. Moreover, as the DNN is trained using SINDy, when we integrate the proposed method with an MPC framework, it is possible for the DNN to successfully identify models that are capable in dealing with noisy data.*

4. SUMMARY AND CONCLUSIONS

This thesis proposes two different methods for adaptive model identification of nonlinear systems. The first method proposed in Chapter 2 seeks to identify and improve the model following the three steps. First, a set of potential candidate functions is identified using sequentially thresholded sparse regression. In the following step, the coefficients of these identified functions are updated using least-squares regression, and lastly, stepwise regression is implemented for selecting the best combination of the most important features. The choice of candidate library functions, the thresholding parameter value considered, the feature selection criterion for stepwise regression, and the number of samples used at every step significantly affect the performance of this adaptive approach. The effectiveness of the proposed methodology was demonstrated on a CSTR system with second-order kinetics. For a less amount of data, the adaptive model successfully identified the coupled dynamics between concentration and temperature variables. In all the cases tested, the prediction accuracy of the model identified by the proposed algorithm was much higher than that of its off-line counterpart, SINDy. Furthermore, the final model was observed to perform well when it is validated with different operating conditions, making it a viable representation of the actual dynamics. To conclude, the adaptive method presented proves to be useful in predicting complex process dynamics from a less amount of data.

Improving upon the previous method, in Chapter 3, we proposed an online model identification framework based on SINDy and deep learning for nonlinear process systems. The novel aspect of this proposed method is to utilize the potential of SINDy in identifying an interpretable model for adaptive modeling and control applications through a DNN. Following this approach, we first used SINDy to identify multiple models from historical/simulation process data. Then, we trained DNNs using the SINDy models and their corresponding process data. Subsequently, we implemented the resulting network in a MPC framework for identifying and updating the models in real-time in order to obtain optimal control performance. We demonstrated the applicability of this method to a non-isothermal chemical reactor. One of the key challenges is to train the DNN that

generalizes well and predicts accurately. To achieve this, we trained two individual DNNs for concentration and temperature models using Bayesian-regularization learning algorithm. Additionally, the DNN training was carried out in a small-batch fashion to prevent over-fitting. Further, the test results indicated that the trained DNNs perform well with a high prediction accuracy, and thus, are suitable for designing a model predictive controller. The closed-loop results showed that the proposed OASIS framework can be effectively used in adaptive modeling and control of nonlinear processes.

REFERENCES

[1] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective.," *Information Sciences*, 2013.

[2] X. Hong, R. J. Mitchell, S. Chen, C. J. Harris, K. Li, and G. W. Irwin, "Model selection approaches for non-linear system identification: a review.," *International journal of systems science*, 2008.

[3] J. Cisternas, C. W. Gear, S. Levin, and I. G. Kevrekidis, "Equation-free modelling of evolving diseases: coarse-grained computations with individual-based models.," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 2004.

[4] K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop, "Neural networks for control systems—a survey.," *Automatica*, 1992.

[5] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, "Machine learning-based predictive control of nonlinear processes. part i: Theory," *AIChE Journal*, 2019.

[6] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, "Machine learning-based predictive control of nonlinear processes. part ii: Computational implementation," *AIChE Journal*, 2019.

[7] H. Ye, R. J. Beamish, S. M. Glaser, S. C. Grant, C.-h. Hsieh, L. J. Richards, J. T. Schnute, and G. Sugihara, "Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling.," *Proceedings of the National Academy of Sciences*, 2015.

[8] D. Giannakis and A. J. Majda, "Nonlinear laplacian spectral analysis for time series with intermittency and low-frequency variability.," *Proceedings of the National Academy of Sciences*, 2012.

[9] B. C. Daniels and I. Nemenman, "Automated adaptive inference of phenomenological dynamical models.," *Nature communications*, 2015.

[10] J. C. Brigham and W. Aquino, "Surrogate-model accelerated random search algorithm for global optimization with applications to inverse material identification," *Computer Methods in Applied Mechanics and Engineering*, 2007.

[11] P. Van Overschee and B. De Moor, "N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, 1994.

[12] M. Verhaegen and P. Dewilde, "Subspace model identification part 2. analysis of the elementary output-error state-space model identification algorithm," *International Journal of Control*, 1992.

[13] W. E. Larimore, "Canonical variate analysis in identification, filtering, and adaptive control," in *29th IEEE Conference on Decision and control*, (Piscataway, NJ), IEEE, 1990.

[14] M. Viberg, "Subspace methods in system identification," *IFAC Proceedings Volumes*, 1994.

[15] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems.," *Proceedings of the National Academy of Sciences*, 2007.

[16] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data.," *Science*, 2009.

[17] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems.," *Proceedings of the National Academy of Sciences*, 2016.

[18] M. Quade, M. Abel, J. Nathan Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for rapid model recovery.," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 2018.

[19] K. P. Champion, S. L. Brunton, and J. N. Kutz, "Discovery of nonlinear multiscale systems: Sampling strategies and embeddings.," *SIAM Journal on Applied Dynamical Systems*, 2019.

[20] M. Hoffmann, C. Fröhner, and F. Noé, "Reactive sindy: Discovering governing reactions from concentration data.," *The Journal of Chemical Physics*, 2019.

[21] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit.," *Proceedings of the Royal Society A*, 2018.

[22] A. Narasingam and J. S.-I. Kwon, "Data-driven identification of interpretable reduced-order models using sparse regression.," *Computers & Chemical Engineering*, 2018.

[23] J.-C. Loiseau, B. R. Noack, and S. L. Brunton, "Sparse reduced-order modelling: sensor-based dynamics to full-state estimation.," *Journal of Fluid Mechanics*, 2018.

[24] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Inferring biological networks by sparse identification of nonlinear dynamics.," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2016.

[25] S. Li, E. Kaiser, S. Laima, H. Li, S. L. Brunton, and J. N. Kutz, "Discovering time-varying aerodynamics of a prototype bridge by sparse identification of nonlinear dynamical systems," *Physical Review E*, 2019.

[26] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations.," *Science Advances*, 2017.

[27] H. Schaeffer, "Learning partial differential equations via data discovery and sparse optimization.," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2017.

[28] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor, "Model selection for dynamical systems via sparse regression and information criteria.," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2017.

[29] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of koopman eigenfunctions for control," *Bulletin of the American Physical Society*, 2017.

[30] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control.," *PloS one*, 2016.

[31] J.-C. Loiseau and S. L. Brunton, "Constrained sparse galerkin regression.," *Journal of Fluid Mechanics*, 2018.

[32] L. Zhang and H. Schaeffer, "On the convergence of the sindy algorithm," *Multiscale Modeling & Simulation*, 2019.

[33] R. Chartrand, "Numerical differentiation of noisy, nonsmooth data.," *ISRN Applied Mathematics*, 2011.

[34] I. Knowles and R. J. Renka, "Methods for numerical differentiation of noisy data.," *Electronic Journal of Differential Equations*, 2014.

[35] A. Abraham, F. Pedregosa, M. Eickenberg, P. Gervais, A. Mueller, J. Kossaifi, A. Gramfort, B. Thirion, and G. Varoquaux, "Machine learning for neuroimaging with scikit-learn.," *Frontiers in Neuroinformatics*, 2014.

[36] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of Machine Learning Research*, 2012.

[37] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, (New York, USA), 2011.

[38] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown, "Towards an empirical foundation for assessing bayesian optimization of hyperparameters," in *NIPS workshop on Bayesian Optimization in Theory and Practice*, (Lake Tahoe, USA), 2013.

[39] A. Varshney, S. Pitchaiah, and A. Armaou, "Feedback control of dissipative pde systems using adaptive model reduction.," *AIChE journal*, 2009.

[40] K. J. Åström and P. Eykhoff, "System identification—a survey.," *Automatica*, 1971.

[41] D. Seborg, T. F. Edgar, and S. Shah, "Adaptive control strategies for process control: a survey.," *AIChE Journal*, 1986.

[42] A. Alanqar, H. Durand, and P. D. Christofides, "Error-triggered on-line model identification for model-based feedback control," *AIChE Journal*, 2017.

[43] Z. Wu, F. D. Rincón, and P. D. Christofides, "Real-time adaptive machine-learning-based predictive control of nonlinear processes," *Industrial & Engineering Chemistry Research*, 2019.

[44] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, (Opatija, Croatia), IEEE, 2015.

[45] G. Chandrashekar and F. Sahin, "A survey on feature selection methods.," *Computers & Electrical Engineering*, 2014.

[46] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection.," *Journal of machine learning research*, 2003.

[47] R. Tibshirani, "Regression shrinkage and selection via the lasso.," *Journal of the Royal Statistical Society: Series B (Methodological)*, 1996.

[48] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems.," *Technometrics*, 1970.

[49] M. L. Thompson, "Selection of variables in multiple regression: Part i. a review and evaluation.," *International Statistical Review/Revue Internationale de Statistique*, 1978.

[50] S. Billings and W. Voon, "A prediction-error and stepwise-regression estimation algorithm for non-linear systems.," *International Journal of Control*, 1986.

[51] C.-F. Tsai, "Feature selection in bankruptcy prediction.," *Knowledge-Based Systems*, 2009.

[52] N. R. Draper and H. Smith, *Applied regression analysis*. New York, USA: John Wiley & Sons, 2014.

[53] V. Klein, J. G. Batterson, and P. C. Murphy, "Determination of airplane model structure from flight data by using modified stepwise regression," 1981.

[54] M. Lewis, "Stepwise versus hierarchical regression: Pros and cons.," *Online Submission*, 2007.

[55] T. Schreiber, "Extremely simple nonlinear noise-reduction method," *Physical Review E*, 1993.

[56] L. A. Aguirre and S. Billings, "Identification of models for chaotic systems from noisy data: implications for performance and nonlinear filtering," *Physica D: Nonlinear Phenomena*, 1995.

[57] S. P. Lalley *et al.*, "Beneath the noise, chaos," *The Annals of Statistics*, 1999.

[58] E. J. Kostelich and T. Schreiber, "Noise reduction in chaotic time-series data: A survey of common methods," *Physical Review E*, 1993.

[59] R. Kumar, V. K. Jayaraman, and B. D. Kulkarni, "An svm classifier incorporating simultaneous noise reduction and feature selection: illustrative case examples," *Pattern Recognition*, 2005.

[60] T. J. Burkholder and R. L. Lieber, "Stepwise regression is an alternative to splines for fitting noisy data," *Journal of Biomechanics*, 1996.

[61] S. K. Goharoodi, K. Dekemele, L. Dupre, M. Loccufier, and G. Crevecoeur, "Sparse identification of nonlinear duffing oscillator from measurement data.," *IFAC-PapersOnLine*, 2018.

[62] B. Bhadriraju, A. Narasingam, and J. S.-I. Kwon, "Machine learning-based adaptive model identification of systems: Application to a chemical process," *Chemical Engineering Research and Design*, 2019.

[63] S. Ge, C. Hang, and T. Zhang, "Nonlinear adaptive control using neural networks and its application to cstr systems," *Journal of Process Control*, 1999.

[64] T. D. Knapp, H. M. Budman, and G. Broderick, "Adaptive control of a cstr with a neural network model," *Journal of Process Control*, 2001.

[65] V. A. Akpan and G. D. Hassapis, "Nonlinear model identification and adaptive model predictive control using neural networks," *ISA Transactions*, 2011.

[66] S. Spielberg, R. Gopaluni, and P. Loewen, "Deep reinforcement learning approaches for process control," in *6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, (Taipei, Taiwan), IEEE, 2017.

[67] G.-H. Liu and E. A. Theodorou, "Deep learning theory review: An optimal control and dynamical systems perspective," *arXiv preprint arXiv:1908.10920*, 2019.

[68] F. Lv, C. Wen, Z. Bao, and M. Liu, "Fault diagnosis based on deep learning," in *2016 American Control Conference (ACC)*, (Boston, MA), IEEE, 2016.

[69] J. C. Hoskins and D. Himmelblau, "Artificial neural network models of knowledge representation in chemical engineering," *Computers & Chemical Engineering*, 1988.

[70] V. Prasad and B. W. Bequette, "Nonlinear system identification and model reduction using artificial neural networks," *Computers & Chemical Engineering*, 2003.

[71] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans, "Nonlinear systems identification using deep dynamic neural networks," *CoRR*, 2016.

[72] C. Shang, F. Yang, D. Huang, and W. Lyu, "Data-driven soft sensor development based on deep learning technique," *Journal of Process Control*, 2014.

[73] I. Malkiel, M. Mrejen, A. Nagler, U. Arieli, L. Wolf, and H. Suchowski, "Plasmonic nanostructure design and characterization via deep learning," *Light: Science & Applications*, 2018.

[74] R. Hedjar, "Adaptive neural network model predictive control," *International Journal of Innovative Computing, Information and Control*, 2013.

[75] E. Aggelogiannaki and H. Sarimveis, "Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models," *Computers & Chemical Engineering*, 2008.

[76] M. A. Hussain, *Review of the applications of neural networks in chemical process control—simulation and online implementation*, 1999.

[77] M. Nikravesh, A. Farell, and T. Stanford, "Model identification of nonlinear time variant processes via artificial neural network," *Computers & Chemical Engineering*, 1996.

[78] T. Chovan, T. Catfolis, and K. Meert, "Neural network architecture for process control based on the rtrl algorithm," *AIChE Journal*, 1996.

[79] S. Ruder, "An overview of gradient descent optimization algorithms.," *arXiv preprint arXiv:1609.04747*, 2016.

[80] S.-I. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, 1993.

[81] R. Battiti, "First-and second-order methods for learning: between steepest descent and newton's method," *Neural Computation*, 1992.

[82] C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," *IEE Proceedings G (Circuits, Devices and Systems)*, 1992.

[83] B. Robitaille, B. Marcos, M. Veillette, and G. Payre, "Modified quasi-newton methods for training neural networks," *Computers & Chemical Engineering*, 1996.

[84] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, 1944.

[85] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, 1963.

[86] H. Yu and B. M. Wilamowski, "Levenberg-marquardt training," *Industrial Electronics Handbook*, 2011.

[87] D. J. MacKay, "Bayesian interpolation," *Neural Computation*, 1992.

[88] K. Hirschen and M. Schäfer, "Bayesian regularization neural networks for optimizing fluid flow processes," *Computer Methods in Applied Mechanics and Engineering*, 2006.

[89] N. B. Karayiannis and A. N. Venetsanopoulos, "Fast learning algorithms for neural networks," in *Artificial Neural Networks*, Boston, MA: Springer, 1993.

[90] G. B. Goh, N. O. Hodas, and A. Vishnu, "Deep learning for computational chemistry," *Journal of Computational Chemistry*, 2017.

[91] H. Schaeffer, G. Tran, and R. Ward, "Learning dynamical systems and bifurcation via group sparsity.," *arXiv preprint arXiv:1709.01558*, 2017.

[92] P. Zheng, T. Askham, S. L. Brunton, J. N. Kutz, and A. Y. Aravkin, "A unified framework for sparse relaxed regularized regression: Sr3," *IEEE Access*, 2018.

[93] F. D. Foresee and M. T. Hagan, "Gauss-newton approximation to bayesian learning," in *Proceedings of International Conference on Neural Networks (ICNN'97)*, (Houston, TX), IEEE, 1997.

[94] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *5th International Conference on Learning Representations*, (Toulon, France), 2017.

[95] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018.

[96] A. Robins, "Sequential learning in neural networks: A review and a discussion of pseudorehearsal based methods," *Intelligent Data Analysis*, 2004.

[97] H. Schaeffer and S. G. McCalla, "Sparse model selection via integral terms," *Phys. Rev. E*, 2017.

[98] G. Tran and R. Ward, "Exact recovery of chaotic systems from highly corrupted data.," *Multiscale Modeling & Simulation*, 2017.