I&EC research
Industrial & Engineering Chemistry Research

pubs.acs.org/IECR Article

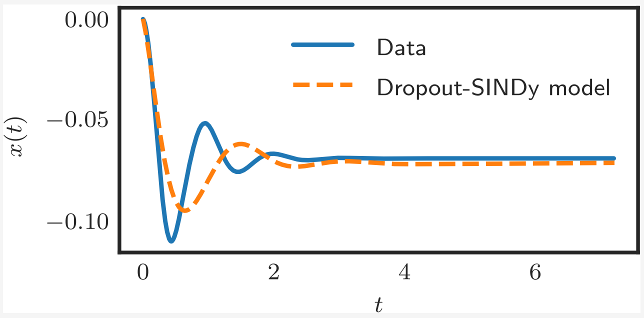
Modeling and Control of Nonlinear Processes Using Sparse Identification: Using Dropout to Handle Noisy Data

Fahim Abdullah, Mohammed S. Alhajeri, and Panagiotis D. Christofides*

Cite This: *Ind. Eng. Chem. Res.* 2022, 61, 17976−17992 Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: Sparse identification of nonlinear dynamics (SINDy) is a recent nonlinear modeling technique that has demonstrated superior performance in modeling complex time-series data in the form of first-order ordinary differential equations (ODEs), which are explicit and continuous in time. However, a crucial step in the SINDy algorithm involves estimating the time derivative of the states from the discrete, measured data. Therefore, the presence of noise can greatly deteriorate the performance if it is not carefully considered and accounted for. In this work, SINDy is used with ensemble learning, where multiple models are identified to improve the overall/final nonlinear model's performance. Specifically, in the SINDy algorithm, a fraction of the library functions considered for the ODE model representation are randomly dropped out in each submodel to favor model sparsity and stability at the possible risk of lowering the model accuracy. This trade-off is controlled by manipulating the fraction of the library functions dropped out and the total number of models generated, both of which are considered as hyperparameters to be tuned in the proposed algorithm. Data from open-loop simulations of a large-scale chemical plant are generated using the well-known high-fidelity process simulator, Aspen Plus Dynamics, and corrupted with substantial sensor noise to be implemented in the newly proposed algorithm, dropout-SINDy. The dropout-SINDy models obtained from training with the noisy data are then tested in open-loop simulations to demonstrate accurate identification of the steady-state and reasonably close transient behavior under a variety of initial conditions and manipulated input values. Finally, the constructed models are used in a Lyapunov-based model predictive controller to control the large-scale Aspen process, meeting desired closed-loop stability and performance specifications.



1. INTRODUCTION

A significant amount of effort has been directed toward the development of mathematical models that describe the physical laws that govern various systems of relevance in engineering and the physical sciences. Although this has traditionally been achieved using mathematical and first-principles frameworks, data-driven approaches have attracted much attention in more recent endeavors. A large proportion of these physical laws are in the form of ordinary differential equations (ODEs) or partial differential equations (PDEs), which are dynamical time-varying models. The Navier−Stokes equations in the domain of fluid mechanics are an ubiquitous model of this form that is widely used in applications in chemical and mechanical engineering. Such time-series predictive models are vital to the design of advanced model-based control methodologies such as model predictive control (MPC) and preemptive maintenance in industrial engineering applications. In predictive maintenance, the health of process equipment is modeled to minimize downtime and increase manufacturing efficiency. An MPC uses a dynamic model to estimate process variables over a finite prediction horizon to calculate optimal control actions. Consequently, several recent articles have focused on the building and integration of data-based models in MPC, e.g., refs [1−13]. The bank of data-driven system identification algorithms is expansive, ranging from traditional methods such as singular value decomposition[14] and numerical algorithms for Subspace State Space System Identification (N4SID)[15] to more recent advances such as auto-regressive models with exogenous inputs (ARX/ARARX).[16−18] However, due to the unprecedented increase in computational capacity and algorithmic development over the last two decades, machine learning (ML) has become a popular option for modeling classical engineering systems due to its efficiency in providing inferences on big data and nonlinear behavior, both of which are characteristics of industrial process systems. This advantage may be attributed

Received: July 22, 2022
Revised: November 10, 2022
Accepted: November 15, 2022
Published: November 30, 2022

to the many tunable hyperparameters in the structure of ML models. In practical regression problems, some ML models that are commonly used are support vector regression, deep neural networks, and sparse identification (SINDy), among others. In the present article, sparse identification, conceptualized in ref 19, is further developed. While the original intent of the SINDy method was to try to learn governing laws from data as interpretable ODEs consistent with the physics, e.g., refs 19 and 20, the use case of SINDy is not restricted to learning governing laws from data and can be used to build computationally-efficient, closed-form models for control. Furthermore, the ODE models obtained from the SINDy method may or may not be consistent with physical laws as there is no guarantee that the search over the set of nonlinear basis functions would lead to a dynamic model providing such physical insight. In the present work, we employ the SINDy method as a system identification method to build dynamic models from noisy data using standard error metrics to determine their goodness of fit. With respect to noise, numerous efforts have been made to improve the sparse identification in many aspects, as seen in refs 21−30, but most of these endeavors consider noise-free data from theoretical simulations. Hence, the practical challenge of handling the inevitable measurement sensor noise present in industrial data remains largely unresolved at the moment.

ML methods have mostly been pioneered in the field of computer science, where either high-fidelity data is often readily available or "noisy data" is used to refer to improper labels in classification rather than regression problems. Hence, many ML methods do not readily generalize to the case of solving regression problems in the field of engineering applications, where noise refers to numerical inaccuracies that are inherent in measurement devices. Thus, the performance of ML methods may be unexpectedly poor when applied to such regression problems and noisy data sets without consideration of the type and magnitude of noise present. This possible drawback was reconfirmed in ref 31, where machine learning and common statistical techniques both performed poorly without any data preprocessing, but yielded satisfactory results following feature engineering. Hence, classical system identification methods recognize this challenge and have expanded the traditional methods to counteract the issue of noise. For example, with respect to linear systems, extended ARX/ARARX methods to the case of additive white noise in input/output data were proposed in ref 32, principal component analysis (PCA) was used in ref 33 to estimate the noise term, subspace identification methods were applied to closed-loop operational data in ref 34, and Gaussian white noise in linear dynamical systems with the Kalman filter was handled in refs 35 and 36. However, as linear approximations may not perform adequately for nonlinear systems, methods such as the extended Kalman filter and moving-horizon estimation for the case of nonlinear systems were proposed in ref 35. Nevertheless, many of the aforementioned methods restrict the class of systems and distribution of noise that may be present, constricting their widespread use in an industrial setting. Consequently, the problem of nonlinear process modeling using noisy time-series data from sensors remains an open challenge for active researchers and control practitioners in the field.

In the recent literature surrounding SINDy, the work on the modeling front has been enumerated and was described in detail in ref 37. This includes the use of an autoencoder and Kalman filter to recover the denoised states,[38] the use of the total-variation regularized derivative, the application of a smoother such as the Savitzky−Golay filter to prefilter the data,[39] updating rather than reidentifying new models upon the availability of new data,[40] the use of Laplace transforms to rewrite ODE models using integral terms,[41] the use of splines as basis functions to estimate the parameters in the already known dynamic model structure,[42] the use of Duhamel's integral for mass-spring systems,[43] exploiting spectral methods[44,45] or Tikhonov regularization[46] to estimate the derivatives, the incorporation of weights in the least squares algorithm,[46] subsampling, and Bayesian regression,[47] among others. A common inadequacy identified in many of the articles, e.g., refs 38 and 48, includes the addition of noise to the derivatives after its computation rather than directly to the noisy data. However, when using SINDy, a significant part of the challenge of handling noisy data, perhaps the most, is the estimation of the derivatives from noisy data.

Beyond noise, a second challenge where a gap existed in the literature was the consideration of the sampling rate of the data. Data in industrial process systems must be measured at finite time intervals, known as the sampling time. This is an inherent limitation of sensors and can be dependent on the variable being measured. For example, a high-end thermocouple can measure the temperature every 0.01 s, while a gas chromatograph can only yield interpretable results on the composition of the substance every 15 min. The sampling time is also a key factor affecting the accuracy of SINDy, as the numerical estimation of the derivative is closely linked to the spacing of the data points in the time domain. Several articles that advanced sparse identification further used data sampled at extremely small rates, such as 10^{-5} s, which is very rarely found in industrial processes. The combination of noisy data and larger sampling times introduces new challenges, where existing methods failed. Therefore, the dual effect of Gaussian noise and large sampling times on sparse-identified models was studied in-depth and a novel algorithm that combined subsampling from statistics with co-teaching from the neural network context in was proposed in ref 37. Specifically, subsampling refers to using a fraction of the data to identify a model with the aim of eliminating outliers, while co-teaching takes advantage of noise-free data obtained by simulating simplified first-principles models that can be derived theoretically. The resulting method significantly outperformed the state of the art when sensor noise levels were very high.

However, first-principles models simplify the differential equations, thereby providing an imperfect representation of the real process. Industrial process systems may often be far too complex with highly nonlinear behavior and input−output interactions that are impossible to reasonably derive and capture in such a simplified physics-based first-principles model, leading to an exorbitant plant−model mismatch. While the initial transient behavior of a first-principles model may agree with the industrial process, often, the long-term dynamics may deviate entirely, rendering the first-principles model ineffective for long-term predictions. While certain methods such as neural networks may only train the model to predict the next sampling time i.e., capture short-term dependencies, for a method like SINDy where the resulting model is an ODE that must be integrable from a given initial condition for sufficiently long while remaining within the close agreement of the true trajectory, the method of co-teaching is not applicable if the plant−model mismatch is too large in the long-term. This is especially relevant when the final steady-state value itself is not the same between the two scenarios. In such a case, both the long-term dynamics and the final steady-state values of the first-principles model are unrepresentative of the industrial process that is desired to be captured. Hence, the goal of this article is to overcome the combined challenges of modeling data that is (1) industrial, (2) highly noisy, and (3) coarsely sampled using sparse identification with ensemble learning. To obtain such a data set, a chemical process simulator is used.

Chemical process simulators are a broad category of software that are used widely in industry to design and optimize processes by conducting reliable steady-state and dynamic simulations. Their superiority over first-principles simulations may be attributed to their built-in packages that handle unit operations, thermodynamic properties, molecular interactions, as well as other features.[49] Chemical process simulators can be broadly divided into equation-oriented and sequential modular approaches such as EMSO software and Aspen Plus, respectively.[50] Moreover, Sharifzadeh investigated the integration of process control systems within process simulators since the two units share decisions and information.[51] Therefore, in this work, a large-scale chemical process is designed in the high-fidelity process simulator, Aspen Plus Dynamics, and then dynamically simulated to generate time-series industrial data. The data set is then corrupted with sufficiently high Gaussian white noise to investigate the enhanced performance of sparse identification with ensemble learning when standard sparse identification is inadequate and yields unstable models. The remainder of this manuscript is outlined as follows: in Section 2, the general class of nonlinear process systems under consideration as well as the theoretical background of stabilization, sparse identification, and ensemble learning are presented in brief. In Section 3, the development of SINDy with dropout to produce an ensemble of models is explained. Finally, a large-scale chemical process is used to assess the modeling and control performance of the proposed algorithm in Section 4.

2. PRELIMINARIES

2.1. Notation. Given a vector x, its transpose and weighted Euclidean norm are denoted by x^T and $|x|_Q$, respectively, where Q is a positive definite matrix. $L_fV(x)$ is used to denote the standard Lie derivative $L_fV(x) := \frac{\partial V(x)}{\partial x}f(x)$. The "\" operator denotes set subtraction such that $A\backslash B := \{x \in \mathbb{R}^n | x \in A, x \notin B\}$. A function $f(\cdot)$ belongs to class C^1 if it is continuously differentiable in its domain.

2.2. Class of Systems. We consider the general class of continuous-time nonlinear systems of the form

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

$$y = x + w \tag{1b}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the manipulated input vector, $y \in \mathbb{R}^n$ is the discretely sampled vector of state measurements, $w \in \mathbb{R}^n$ represents sensor noise, and $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$ and $n \times m$, respectively. Without loss of generality, throughout the article, the initial time t_0 and the initial condition $f(0)$ are taken to be 0, the latter of which implies that the origin is a steady state of the nominal system of eqs 1a and 1b, i.e., $(x_s^*, u_s^*) = (0, 0)$, where x_s^* and u_s^* represent the steady-state and input vectors, respectively.

2.3. Stability Assumption. For the nominal system of eqs 1a and 1b under full state feedback consisting of noise-free state measurements, it is assumed that a stabilizing control law $u = \Phi(x) \in \mathcal{U}$ exists that can render the origin of the closed-loop system of eqs 1a and 1b exponentially stable. Converse Lyapunov theorems[52−54] then imply that there exist a C^1 control Lyapunov function, $V(x)$, and four positive constants, c_1, c_2, c_3, c_4, such that $\forall x$ in an open neighborhood D around the origin

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

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

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

where $\dot{V}$ represents the time derivative of the Lyapunov function and $F(x, \Phi(x))$ represents the nominal system of eqs 1a and 1b under a candidate controller $\Phi(x)$ such as the universal Sontag control law.[55] We first characterize a set of states $\phi_u = \{x \in \mathbb{R}^n | \dot{V}(x) = L_fV + L_gVu < -kV(x), u = \Phi(x) \in \mathcal{U}, k > 0\} \cup \{0\}$ under the controller $u = \Phi(x) \in \mathcal{U}$ that satisfies the conditions of eqs 2a−2c. Subsequently, we define the closed-loop stability region Ω_ρ[56] for the nominal system of eqs 1a and 1b to be a sublevel set of V inside ϕ_u, i.e., $\Omega_\rho := \{x \in \phi_u | V(x) \le \rho\}$, where $\rho > 0$ and $\Omega_\rho \subset \phi_u$.

2.4. Sparse Identification. Sparse identification is a recent breakthrough in nonlinear system identification that has been shown to be effective in numerous examples from various engineering disciplines.[57−63] The goal of SINDy is to use only discrete measurement data from a physical system to identify a first-order ODE of the form

$$\dot{\hat{x}} = \hat{f}(\hat{x}) + \hat{g}(\hat{x})u \tag{3}$$

where $\hat{x} \in \mathbb{R}^n$ is the state vector of the sparse-identified model, while $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ are the vector fields that capture the physical laws governing the system.

The key assumption in SINDy is that, for most physical systems, the right-hand side of eq 3 contains only a few nonlinear terms. Consequently, if a large bank of possible nonlinear basis functions are considered for $\hat{f}$ and $\hat{g}$, only a few terms will be active and will have nonzero premultiplying coefficients associated with them. Since the space of the candidate basis functions is then sparse, efficient convex optimization algorithms can be used to calculate the coefficients, which is the sparse identification problem. The application of SINDy starts with sampling real-time data from the process to be identified. This may be collected via sensors from an open-loop experimental/industrial process or generated from open-loop computer simulations using theoretical models, either from first-principles or chemical process simulators. The sampled data set is then arranged into the compact data matrix X and the input matrix U

$$X = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix} \tag{4a}$$

$$U = \begin{bmatrix} u_1(t_1) & u_2(t_1) & \cdots & u_r(t_1) \\ u_1(t_2) & u_2(t_2) & \cdots & u_r(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_m) & u_2(t_m) & \cdots & u_r(t_m) \end{bmatrix} \tag{4b}$$

where $x_i(t_l)$ and $u_j(t_l)$ represent the i^{th} state measurement and j^{th} input measurement at the l^{th} sampling time, respectively, with $i = 1, \cdots, n$, $j = 1, \cdots, r$, and $l = 1, \cdots, m$, and $\dot{X}$ represents the first-order time-derivative of X, which may be possible to measure directly in some applications, but is otherwise numerically estimated. Importantly, this step of obtaining $\dot{X}$ is the key challenge when using noisy data for sparse identification since, based on the approximation method, numerical estimations of the derivative may not be stable when using noisy data. From the data matrices X and U, a function library $\Theta(X, U)$ is created, which contains p columns corresponding to the p nonlinear basis functions considered for the terms in $\hat{f}$ and $\hat{g}$. Given the universality of polynomials and trigonometric functions in the field of engineering, an example of a typical function library matrix when identifying process systems is as follows

$$\Theta(X, U) = \begin{bmatrix} | & | & | & | & | & | & | \\ \mathbf{1} & X & X^{P_2} & \sin(X) & e^X & U & UX^2 \\ | & | & | & | & | & | & | \end{bmatrix} \tag{5}$$

where X^{P_2} concisely represents all possible quadratic nonlinearities

$$X^{P_2} = [x_1^2 \ x_1x_2 \ \cdots \ x_2^2 \ x_2x_3 \ \cdots \ x_n^2] \tag{6}$$

However, the preliminary basis set chosen can and should be updated if required based on the performance and available process structure knowledge. Each candidate basis function in eq 5 associated with each variable or row of eq 3 is assigned a coefficient, with all of the coefficients being stored in the matrix $\Xi \in \mathbb{R}^{p\times n}$, which is the output of the sparse identification algorithm calculated by solving the following equation

$$\dot{X} = \Theta(X, U)\Xi \tag{7}$$

Popular methods for solving eq 7 include sequential thresholded least squares (STLSQ) and sparse relaxed regularized regression, the latter of which is based on the least absolute shrinkage and selection operator (LASSO).[64] In STLSQ, a hard threshold called the sparsification knob λ is first specified, after which all coefficients in Ξ smaller than λ are set to zero. The resulting equation is then solved to yield Ξ. This procedure is repeated until all nonzero coefficients converge. The sparse nature of the matrix Ξ makes the iterations fast and efficient. The final values of Ξ obtained are then used to construct the continuous-time ODE

$$\dot{x} = \Xi^{\mathrm{T}}(\Theta(x^{\mathrm{T}}, u^{\mathrm{T}}))^{\mathrm{T}}$$

where $\Theta(x^{\mathrm{T}}, u^{\mathrm{T}})$ is not a matrix of data, but a column vector of symbolic functions of x and u from the library of considered functions.

Remark 1: The sparse identification-based modeling approach is a technique to develop closed-form nonlinear, first-order ODE models for process systems using time-series measurement data. It should be viewed similarly to other system identification techniques for developing dynamic models from data. The central advantage of the sparse identification modeling technique is the construction of a closed-loop form nonlinear model that can be efficiently numerically integrated when used in the context of MPC—there is a significant computational efficiency advantage when a closed-form model is used in MPC as opposed to recurrent neural network models that are more computationally demanding in both training and implementation phases. This is the key reason for exploring the use of sparse identification modeling in the context of MPC. In ref 65, for a similar process system, i.e., CSTRs modeled via Aspen Plus Dynamics, it was demonstrated that the average time to solve the optimization problem in the MPC took 2.1161 min (127 s), while the proposed dropout-SINDy-based MPC took an average of 42 s. While the system in ref 65 is not identical to the one studied here, the number of control actions to be computed by the MPC and the computational power of the computer were very similar, thereby expecting similar computational efficiency results.

2.5. Ensemble Learning.

To improve the performance of machine learning models, one simple and intuitive practice is the use of multiple models to make predictions, which is known as ensemble learning. All of the models are identified from the same data set, but the model structure differs from one model in the ensemble to the next, which introduces flexibility and allows for better generalization. Specifically, ensemble learning potentially reduces the variance of the algorithm without increasing the bias for individual models[66] and also enables the accounting of uncertainty during the model selection process.[67] Ensemble learning algorithms are broadly classified into two categories: homogeneous and heterogeneous. In homogeneous ensembles, one base learning method/algorithm is used multiple times on different, random subsets of the entire data set to generate several models that are slightly different due to the different data subsets used. The concept is similar to subsampling. In contrast, heterogeneous ensemble learning involves using a diverse array of machine learning methods such as linear regression, artificial neural networks, support vector regressors, XG Boost, etc. to improve generalization and diversity within the ensemble. Whether the ensemble is homogeneous or heterogeneous, the final output of the model is taken as either a central tendency of all of the models in the ensemble or selected by cross-validation. Common central tendencies include the mean (bagging) or median (bragging) of all of the model parameters/outputs.

2.6. Model Predictive Control Using Sparse Identification.

To achieve closed-loop stability and performance, we propose a Lyapunov-based model predictive control (LMPC) scheme using the SINDy model, formulated as follows

$$\mathcal{J} = \min_{u\in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u(t))\mathrm{d}t \tag{8a}$$

$$\text{s. t.} \quad \dot{\tilde{x}}(t) = F_{\mathrm{si}}(\tilde{x}(t), u(t)) \tag{8b}$$

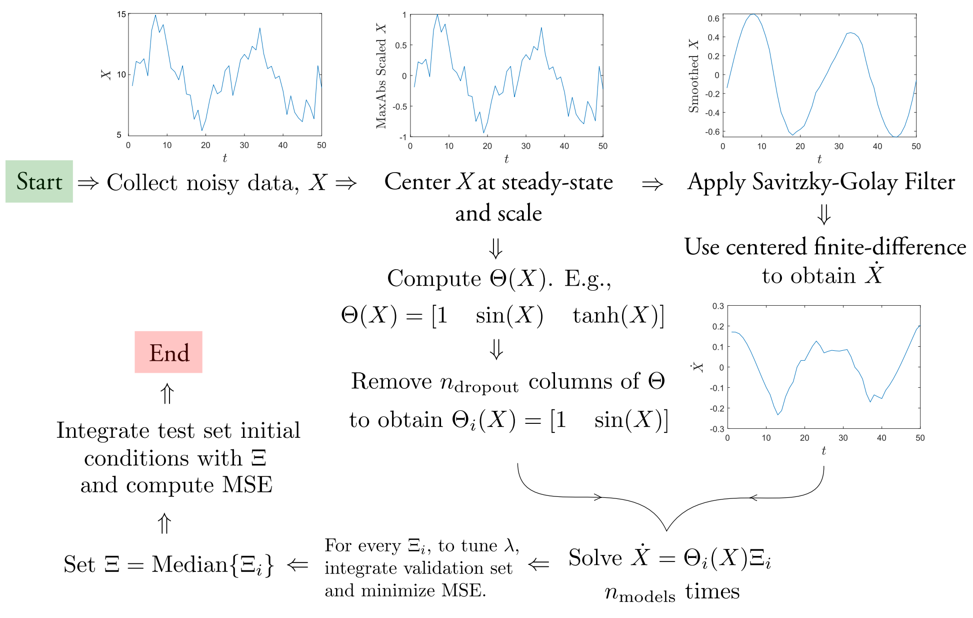

Start
Collect noisy data, X
Center X at steady-state and scale
Apply Savitzky-Golay Filter
Use centered finite-difference to obtain $\dot{X}$
Compute $\Theta(X)$. E.g., $\Theta(X) = [1 \quad \sin(X) \quad \tanh(X)]$
Remove n_{dropout} columns of Θ to obtain $\Theta_i(X) = [1 \quad \sin(X)]$
Solve $\dot{X} = \Theta_i(X)\Xi_i$ n_{models} times
For every Ξ_i, to tune λ, integrate validation set and minimize MSE.
Set $\Xi = \text{Median}\{\Xi_i\}$
Integrate test set initial conditions with Ξ and compute MSE
End
MaxAbs Scaled X
Smoothed X

Figure 1. Data flow diagram for model construction.

$$u(t) \in \mathcal{U}, \ \forall \ t \in [t_k, t_{k+N}) \tag{8c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{8d}$$

$$\hat{V}(x(t_k), u) \leq \hat{V}(x(t_k), \Phi_{\text{si}}(x(t_k))), \ \text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{\text{si}}} \tag{8e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{\text{si}}, \ \forall \ t \in [t_k, t_{k+N}), \ \text{if } x(t_k) \in \Omega_{\rho_{\text{si}}} \tag{8f}$$

where $\tilde{x}$, N, and $S(\Delta)$ in eq 8a denote the predicted state trajectory, the number of sampling periods in the prediction horizon, and the set of all piecewise constant functions with a period of Δ, respectively; F_{si} is the sparse-identified process model; Φ_{si} is a stabilizing control law that guarantees exponentially stability of the origin of the closed-loop system of eq 3; $\Omega_{\hat{\rho}}$ and $\Omega_{\rho_{\text{si}}}$ denote the stability region for the closed-loop sparse-identified system and the target region for the final predicted state, respectively; and $\dot{V}$ is the time-derivative of V, the Lyapunov function, and is given by $\frac{\partial V(x)}{\partial x}F_{\text{si}}(x, u)$. The LMPC calculates $u^*(t)$, the optimal input sequence, over the entire prediction horizon, i.e., $t \in [t_k, t_{k+N})$, and transmits the first control action of the sequence $u^*(t_k)$ to the actuator to be implemented during the next sampling period Δ. Subsequently, the horizon of the LMPC is advanced by one sampling period and resolved again at the next sampling period.

The goal of the optimization problem of eq 8 is to minimize the objective function of eq 8a, which is equal to the integral of $L(\tilde{x}(t), u(t))$, over the entire prediction horizon. Equation 8b denotes the sparse-identified model that is utilized to predict the closed-loop states over the prediction horizon while varying the input u within the constraints set by eq 8c. The initial condition of the states required by the SINDy model of eq 8b is given by eq 8d, which is the state measurement at time t_k. For stability considerations, we further add the last two constraints known as Lyapunov constraints. The first constraint, given by eq 8e, guarantees that the state $x(t_k)$, when outside the target region $\Omega_{\rho_{\text{si}}}$, will move toward the origin in every step. The proof for this, which relies on bounded modeling errors or bounded disturbances and Lipschitz properties, can be found in detail in ref 68. The result of the proof is that, as long as the state starts from within the stability region Ω_{ρ}, the state converges to $\Omega_{\rho_{\text{si}}}$ in a finite number of sampling times without leaving the stability region. The second constraint of eq 8f ensures that, once the state enters the invariant set $\Omega_{\rho_{\text{si}}}$, it remains within this region for the entire prediction horizon.

3. ENSEMBLE-BASED SPARSE IDENTIFICATION

Modeling systems using data from first-principles simulations corrupted with sensor noise has been studied in ref 37. However, the combined challenge of noise and industrial process dynamics is found to be too challenging for basic SINDy. Therefore, in this work, to overcome the challenge of dealing with industrial noise, we employ a recent advancement in SINDy, which uses homogeneous ensemble learning. As described in Section 2.5, ensemble learning refers to the identification of multiple models to improve the prediction. However, in the context of sparse identification, the details must be elucidated.

In sparse identification, as either the data set or the function library may be used partially, two types of ensembles exist. Ensemble learning can refer to the development of multiple models using either random subsets of the data set with the complete function library (i.e., subsampling) or random subsets of functions from the candidate library with the complete data set. It was demonstrated in ref 37 that the former mode of ensemble learning does not improve performance under high noise levels. Moreover, it was observed in simulations that the models built using basic SINDy or ensembles utilizing the full function library were often unstable. This may be explained by the presence of too many active nonzero functions in an ODE causing the resulting ODE model to have inherently unstable dynamics. Therefore, in this paper, we investigate the second mode of ensemble learning, where the entire data set is used, but random functions in the library are dropped out to further promote sparsity and stability. Specifically, as shown in Algorithm 1, n_{models} models are identified, each time dropping out n_{dropout}

functions from the candidate library, with each model identification using the entire data set. For the i^{th} iteration, the function library of eq 5 may be of the form

$$\Theta_i(X, U) = \begin{bmatrix} | & | & | & | & | \\ \mathbf{1} & X & \sin X & \cos X & \tan X \\ | & | & | & | & | \end{bmatrix} \tag{9}$$

where quadratic and cubic polynomials, as well as tanh terms have been removed from the basis functions. The corresponding Ξ_i will assign values of zero for the n_{dropout} functions that have been dropped out, and the STLSQ optimizer will find the best ODE model that can be built using this reduced set of basis functions by tuning the sparsification knob λ to minimize the validation error. After constructing all n_{models} models from various subsets of library functions, the final model can be designed by taking the median of all of the values of the model coefficients for each function in the library. While an ensemble may generally and commonly use the mean or median, in this scenario, the median is recommended. If most models assigned a zero value to a certain coefficient, it is highly likely that the corresponding term is insignificant and should be ignored in the model. Therefore, when using the median, if more than half of the identified models have a zero coefficient for a term, it is entirely ignored. In contrast, the mean will always return a nonzero value even if only one of the models has a nonzero coefficient for the term. As a result, using the mean to create an ensemble will very likely reduce sparsity and promote instability. Therefore, the final model is constructed using the median in this work. Once the final model using the median coefficient values is selected, the model is integrated from the test set initial conditions, which is used to compute the test set MSE and also plotted against time to visually confirm stability as well as accuracy. The flow of data and models throughout the algorithm is shown in Figure 1.

```
Algorithm 1: Sparse Identification with Dropout
  Input: $X, U, \Theta, n_{\text{dropout}}, n_{\text{models}}$
  Output: $\Xi$
  center $X$ at the steady-state and scale by the maximum absolute value of each
    column;
  compute estimate of $\dot{X}$ of noisy data using smoothed finite-differences;
  for $i \leftarrow 1$ to $n_{\text{models}}$ do
      randomly remove $n_{\text{dropout}}$ columns from $\Theta(X,U)$ to form $\Theta_i(X,U)$;
      solve $\dot{X} = \Theta_i(X,U)\Xi_i$ using STLSQ with multiple $\lambda$ to get $\Xi_i$ with minimized
        MSE;
      with calculated $\Xi_i$, integrate ODE and compute MSE;
  end
  Let $\Xi = \text{Median}\{\Xi_i\}$;
  Integrate test set initial conditions with $\Xi$;
  Compute MSE and plot predicted test set trajectories using model;
  Verify low MSE and stability.
```

Remark 2: A more accurate yet computationally expensive approach is to record the validation MSE for every submodel during training. Finally, the submodel that yielded the minimum MSE at its optimal value of λ can be selected as the best model. However, in this work, the median was chosen because of its simplicity and functionality.

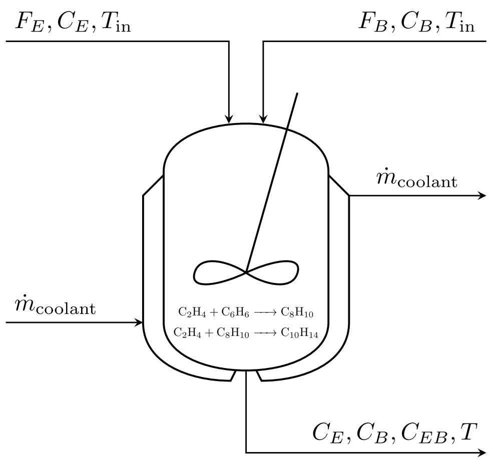
Remark 3: It is important to emphasize that SINDy searches over a "bank" of explicit nonlinearities (basis functions) by solving an optimization problem with suitable penalties on both the error between the values of the actual state and the predicted model state and the number of nonzero premultiplying coefficients of the nonlinear basis functions used to construct the ODE model. If there is any physical insight on the type of nonlinearities that the approach should consider, this physical insight can be incorporated into the optimization search (biasing, for example, the order with which the nonlinearities are considered in the optimization search in an approach similar to the ALAMO modeling technique[69,70]). But if such a physical insight does not exist, then a model will be constructed with the search procedure described above, and then tested to determine if it is a suitable model (i.e., tested for numerical stability, sensitivity to parameters, and predictive ability with respect to validation data). It is important to note here that there is no guarantee that the sparse identification modeling approach will yield a nonlinear model whose nonlinearities provide information about the underlying physicochemical phenomena occurring in the process. If the model is deemed unsatisfactory by the selected accuracy criteria, then the optimization cost parameters should be modified, and another model construction should be done. From a control point of view, only a stable model that accurately predicts the state evolution with time is needed, and there is no need for the model used in the controller to provide any physical insight (this is the case with any system identification technique e.g., N4SID, MOESP, NARMAX). Therefore, there is no requirement for prior knowledge of the nonlinear dynamics in SINDy; if such information is available, it can be used, but it is not needed to apply SINDy, just as it is not needed for any other system identification technique. This point will be further illustrated with an example in Section 4.2.1, where accurate models can be shown to be derived even when not using "physically motivated" basis functions. The advantages of SINDy models in control, such as computational efficiency and explicit modeling of nonlinearities, remain regardless of the availability of prior knowledge of the system dynamics.

Remark 4: The Hammerstein–Wiener modeling framework is another form of modeling that is often used to deal with nonlinearities in process systems. In Hammerstein–Wiener models, a linear dynamic element is followed by a static nonlinear element that can be used to represent nonlinear process behavior. The nonlinear elements considered in these modeling approaches are usually polynomial nonlinearities, and the resulting models are of discrete time. When incorporated into MPC, these nonlinear models may lead to improved closed-loop performance over MPC with linear models.[71,72] However, the polynomial basis functions used in Hammerstein–Wiener models are already possible candidates in the function library of sparse identification. Hence, the SINDy method will yield models that are at least as comprehensive as the Hammerstein–Wiener models, and possibly better when the SINDy basis functions are chosen to be more expansive than only polynomial terms.

4. APPLICATION TO A CHEMICAL PROCESS MODELED IN ASPEN PLUS DYNAMICS

We evaluate the proposed dropout-SINDy algorithm with a large-scale chemical process modeled using Aspen Plus Dynamics V12. First, a dynamic model is constructed, which is then used to generate a time-series data set through extensive open-loop simulations for the purpose of training and testing the SINDy model. The data generation is carried out with a large range of inputs and initial conditions to cover a wide area of the operating region. Subsequently, open- and closed-loop simulations are conducted, and the results presented.

We consider the reaction of ethylene (E) with benzene (B) to form ethylbenzene (EB) in a perfectly mixed, nonisothermal continuous stirred-tank reactor (CSTR) as shown in Figure 2.



F_E, C_E, T_{in}
F_B, C_B, T_{in}
$\dot{m}_{\text{coolant}}$
$\dot{m}_{\text{coolant}}$
$C_2H_4 + C_6H_6 \longrightarrow C_8H_{10}$
$C_2H_4 + C_8H_{10} \longrightarrow C_{10}H_{14}$
C_E, C_B, C_{EB}, T

Figure 2. Continuous stirred-tank reactor with a cooling jacket.

However, a side reaction that consumes ethylene (E) and ethylbenzene (EB) to produce diethylbenzene also occurs simultaneously. Both reactions are exothermic, irreversible, of second-order, and are shown below

$$C_2H_4 + C_6H_6 \overset{k_1}{\rightarrow} C_8H_{10}\ (\text{main}) \tag{10a}$$

$$C_2H_4 + C_8H_{10} \overset{k_2}{\rightarrow} C_{10}H_{14} \tag{10b}$$

where the desired reaction of eq 10a is annotated as "main".

4.1. Dynamic Model Construction. In this paper, we model the CSTR in steady-state and transient modes of operation using Aspen Plus and Aspen Plus Dynamics V12, respectively, both of which are high-fidelity chemical process simulators. The dynamic model's construction begins with the design of the process under steady-state conditions using Aspen Plus, where mass and energy balances are carried out. Once it is ensured that the steady-state simulation converges, Aspen Plus Dynamics is used to investigate the dynamic performance of the model and implement the desired controller during dynamic operation. The end-to-end procedure to construct the steady-state and dynamic models is presented below, with the resulting process flow diagram (PFD) depicted in Figure 3.

1. Feed streams' properties: The raw materials E and B are fed to the CSTR at fixed molar flow rates of F_E and F_B, respectively, with a fixed inlet temperature of T_{in} = 350 K. The feed flow rate of B is chosen to be equal to twice the flow rate of E, despite the 1:1 stoichiometric ratio of the reactants, to minimize the concentration of E in the reactor, thereby suppressing the undesired side reaction. The molar concentrations of ethylene, benzene, and ethylbenzene are represented by C_E, C_B, and C_{EB}, respectively, while the temperature of the reactor is denoted by T. Once converted to scaled deviation variables from their steady-state values, the states C_E, C_B, C_{EB}, and T are denoted by x_1, x_2, x_3, and x_4, respectively. Process parameter values are given in Table 1.

Table 1. Parameter Values for Chemical Process Example

F_E = 0.1 kmol/s	F_B = 0.2 kmol/s
$k_{0,1}$ = 1.528 × 10^6 m^3 kmol^{-1} h^{-1}	$k_{0,2}$ = 27 780 m^3 kmol^{-1} h^{-1}
E_1 = 71 160 kJ/kmol	E_2 = 83 680 kJ/kmol
R = 8.314 kJ kmol^{-1} K^{-1}	V = 60 m^3
T_{in} = 350 K	T_0 = 400 K
$\dot{m}_{\text{coolant,ss}}$ = 77.9869 kg/s	

2. Pressure drop specifications: To create a functional dynamic model, the simulation must allow for pressure drops in the fluid flow throughout the process. This is achieved using valves between every two pieces of equipment, to allow the pressure to vary as the process fluid flows through the equipment. When the pressure drop is selected properly, the model correctly deduces the direction of fluid flow, leading to zero errors during the runtime of the simulation. On the contrary, a pressure drop too low will trigger errors in the simulation. The pressure drops in our model are 5 bar and 2 bar for the feed valves (v_1, v_2) and product valve (v_3), respectively.
3. Reactor specifications: The CSTR is surrounded by a cooling jacket through which liquid water at 298 K flows with a mass flow rate of $\dot{m}_{\text{coolant}}$. The initial temperature and pressure inside the reactor are chosen to be 15 bar and 400 K, respectively, but both values will be altered by the built-in steady-state simulation during runtime. Finally, once reactions in the reactor are specified, the steady-state simulation is run.
4. Thermodynamic package and reactor geometry: We used the predictive Soave–Redlich–Kwong (PSRK) method to estimate the behavior of the phase equilibria. The reactor geometry must also be specified before the steady-state model can be exported to Aspen Plus Dynamics. Therefore, the reactor is specified as a 10 m long, vertical vessel with flat heads.

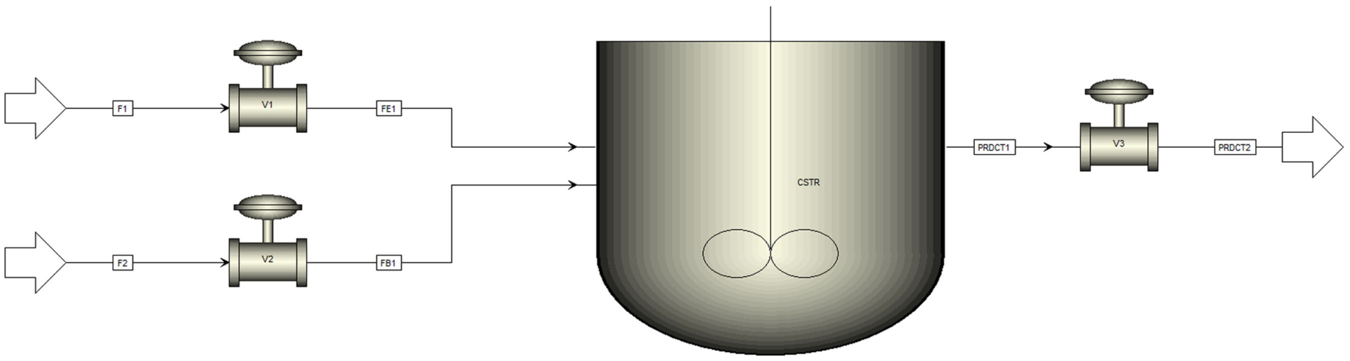


Figure 3. Aspen Plus model process flow diagram of ethylbenzene production process.

5. Pressure checking: Before exporting the steady-state model to Aspen Plus Dynamics, the final step is to run the model once more and perform pressure checking using the built-in pressure checker in the Dynamic tab of Aspen Plus. Once completed without errors, the model is exported to Aspen Plus Dynamics.
6. Controller specifications: The level in the reactor is maintained via a direct-acting level controller developed after exporting the model to Aspen Plus Dynamics. The level controller is set to the auto mode to ensure the level is maintained at the desired setpoint throughout the simulation.
7. Heat transfer option: Since the reactor is surrounded by a jacket with liquid water at 298 K flowing through it, the heating mode of the reactor is selected to be "dynamic", which will allow the user to specify the flow rate of the cooling water to control the reactor temperature. The logarithmic mean temperature difference (LMTD) or temperature approach is calculated and fixed at 77.33 K. The input action affects the temperature in an inverse direction, i.e., increasing the coolant flow rate reduces the temperature and vice versa.
8. Initialization: After specifying the level-controller and cooling jacket settings, a steady-state simulation is completed to obtain the steady-state coolant flow rate of the dynamical model, which is found to be $\dot{m}_{\text{coolant,ss}}$ = 77.9869 kg/s. Following initialization, the model is run in dynamic mode one more time to check that the model does not deviate from the steady state after making the above specifications. This step finalizes the dynamical process model shown in Figure 3.

The addition of noise to the model solutions to represent realistic process state data will be discussed in detail in Section 4.2.

4.2. Data Generation and SINDy Model Development.

Extensive open-loop simulations are carried out with the constructed Aspen Plus Dynamics model to generate numerous trajectories of the states in the reactor, using a wide range of initial conditions and input values. Due to the nature of SINDy models, we generate three types of trajectories to cover the various dynamics that need to be captured by the model. In the first type of runs, a nonzero input u is first applied to the system to drive the states from the origin to a new steady state and subsequently removed ($u = 0$) to let the state return to the origin without any input. Approximately half of these types of runs were conducted very close to the origin by applying small values of the manipulated input to induce smaller deviations from the origin. This was carried out because data-driven modeling is greatly affected by the quality and diversity of the training data, and such models generally underperform near the origin because of the lack of training data at exactly the origin, unless data near the origin are specifically generated. The second category of runs involved applying a nonzero u to the system at the origin to drive it to a new steady state and maintain the system at the new state. The third type of runs involved separately applying two different nonzero values of u to drive the states from the origin to two different steady states successively. Such runs measure the ability of the model to drive the state from an arbitrary state to any desired state using the necessary input u and are therefore critical to improving and evaluating the performance of the model.

The range of inputs considered was between $u = -37.9869$ kg/s and $u = 4.0131$ kg/s. For each pair of initial conditions and input, the Aspen Dynamical model is integrated using an adaptive integration time-step, with the measurements recorded every $\Delta = 0.01$ h. A total of 25 trajectories of the three types are generated. The simulation duration for the runs is not fixed since the data generation is carried out until a steady state has been reached, which varies between runs. The number of data points per trajectory varies between 500 and 1500 points, corresponding to simulation run times of 5 and 15 h. The test run used for further demonstration is generated at the lower limit of the range of u considered, i.e., with $u = -37.9869$ kg/s. This is significantly outside the range of u considered in training and will gauge the ability of the model to capture the inherent dynamics of the system to predict the behavior under a wide range of operating conditions.

As noted in the Section 1, the sampling rate of the data has a significant impact on the accuracy of any system identification method, including sparse identification. This is because discrete sampling of any continuous-time system necessitates loss of information. In general, smaller sampling times lead to smaller loss of information and a more complete history of the state and input trajectories, producing better models when these data are used in a model identification procedure. In the context of sparse identification, the smaller sampling times also directly impact one of the most challenging steps, which is to compute the estimate of the time derivative because a smaller sampling time generally favors the finite-difference method. Case studies and numerical examples in the pioneering literature in the field of sparse identification such as ref 19 reflect the superior performance under such circumstances by producing extremely accurate sparse-identified models using data generated with sampling times of 10^{-4} or even 10^{-6}. However, as remarked in the Introduction section, this is generally infeasible in chemical processes as there is no instrumentation that can provide such high sampling rates in general. Temperature, despite being one of the simplest and fastest variables to measure, is still limited to a sampling period of at least 0.01 s when using a high-end thermocouple. For other variables such as concentrations, it is usually even longer if using chromatography or other similar techniques. Therefore, in this work, we use a sampling period of $\Delta = 0.01$ s for all our data generation. Given this practical sampling time, our simulations demonstrate that the constructed models capture well the process dynamic behavior and lead to very good model predictive controller performance.

Once the data set is generated, data preprocessing is conducted. Specifically, the 25 runs in the data are first split into training and testing sets as follows: 21 trajectories are used as the training set, and 4 runs are used for model testing. The 21 runs in the training set include the 3 validation runs used for hyperparameter tuning. The data split, approximately 80% for training and 20% for testing, is chosen due to the difficulty of training models when using noisy data. Therefore, a larger training/validation set can improve model performance. After data partitioning, data normalization is performed on the data set. The training and testing sets are scaled and normalized only with respect to themselves to prevent data leakage. Three scaling methods were investigated—the z-score scaler, Min-Max scaler, and Max-Abs scaler. The z-score scaler scales the data by subtracting the mean and dividing it by the standard deviation. Mix-Max scalers scale all data points to be between two user-defined limits, usually 0 and 1. Max-Abs scaling refers

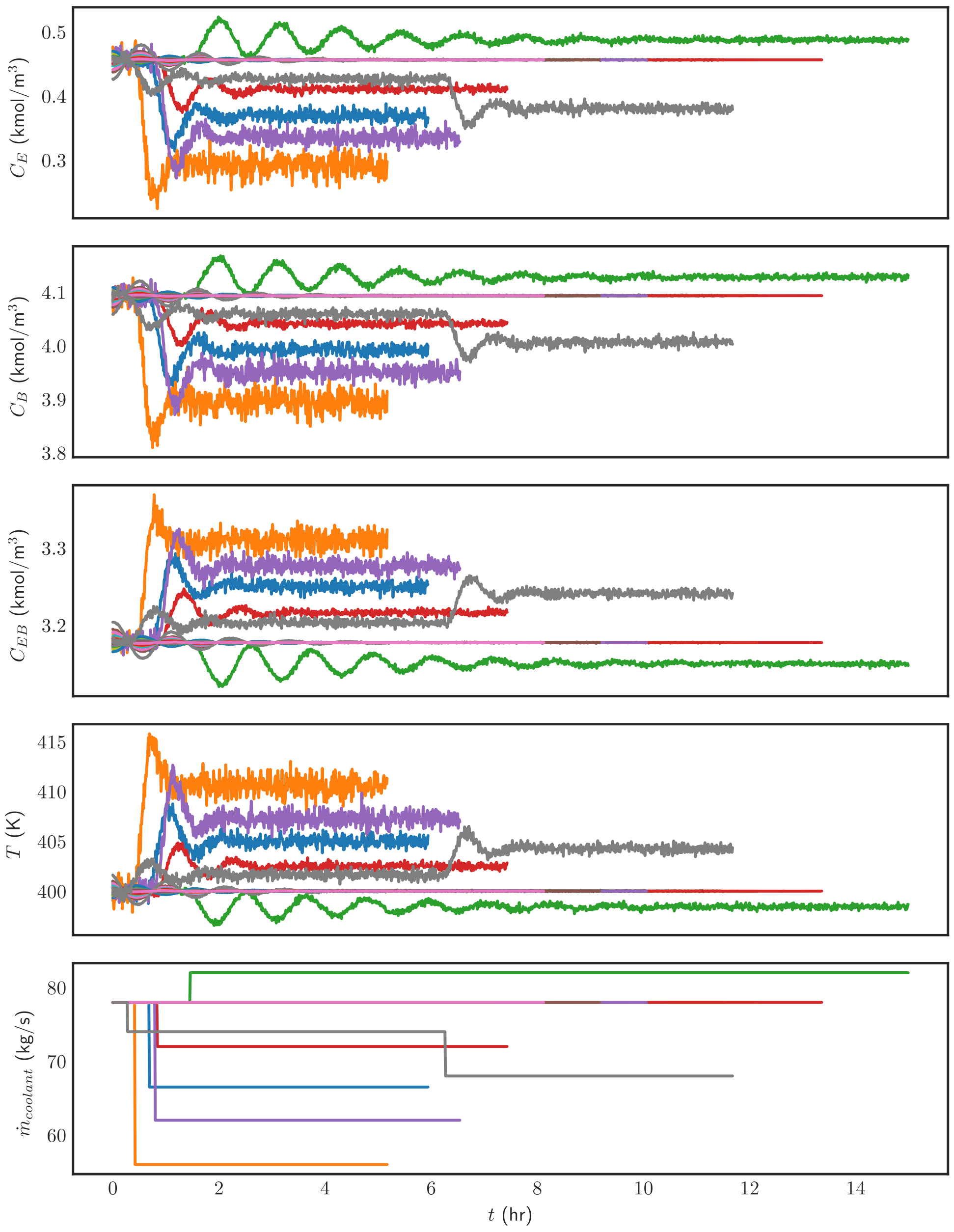


Figure 4. Training data set. Each line is one open-loop trajectory.

to dividing the data by the maximum absolute value of each variable in the data set. After investigating the advantages, disadvantages, and open-loop results of the three scaling methods, the Max-Abs scaling was used because it outperformed the other two methods. One possible reason is that it preserves the sign of the deviation of the states from the equilibrium point, which can affect the performance when using explicit methods such as SINDy. For example, it is known that the manipulated input $\dot{m}_{\text{coolant}}$ has an inverse effect on temperature T. Therefore, in the subspace of scaled variables, the sign of the coefficient associated with u in $\dot{x}_4$ should have a negative sign in the ODE corresponding to the physical system. When the data is scaled using Max-Abs scaler, this property is conserved. However, the z-score and Min-Max scalers do not conserve this property due to the subtraction component of the scaling.

After preprocessing the data, Gaussian noise with the distribution $N \sim (0, \sigma^2)$ is added to corrupt the data and simulate the effect of industrial sensor noise. In this work, a noise level of 8% is used. The relationship between the noise percentage and the standard deviation σ of the added noise is defined as follows

$$\sigma = \text{RMSE} \times \frac{\text{Noise percentage}}{100} \tag{11}$$

where root-mean-square error (RMSE) is the root-mean-square error of the signal given by

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} x_i(t_j)^2}{m \times n}} \tag{12}$$

Since noise is added to the scaled variables, it is not necessary to amplify the noise differently for each variable. The training data set used for the dropout-SINDy procedure is shown in Figure 4.

In this work, the noise was added to the data after preprocessing, i.e., centering and scaling. In an industrial setting, the noise would be in the original states as measured, before any centering or scaling. In this study, however, to define the "noise level" as a fixed percentage as per eqs 11 and 12, it was necessary to appropriately scale either the noise or the data when adding the two due to the orders of magnitude of difference in the scales of the concentrations and temperature. Due to the RMSE (eq 12) being used to define the noise percentage, centering is required regardless of scaling. This ensures x_i in eq 12 is in terms of deviation from the steady state and centered around zero, which ensures that the RMSE is meaningful. With respect to scaling, both methods were attempted: the first being centering and scaling the data and then adding 8% noise, as defined in eqs 11 and 12, to the newly scaled data set. This eliminates the need to scale the noise because all of the x_i values in eq 12 are already of a similar order of magnitude, specifically between +1 and −1. In contrast, the second method is to add scaled noise of 8% to each variable. However, since x_4 in eq 12 is now much larger due to the data not being scaled by the maximum absolute value of the deviation in temperature (14.961 K), the RMSE increases by almost an order of magnitude as well, leading to an extremely large variance from using eq 11. Therefore, the concentrations (x_1, x_2, x_3) are completely masked by noise and become very similar to pure white noise, rendering the data useless for modeling.

To counter this, the variance must be decreased by an order of magnitude (by multiplying by 0.1) to keep the noise amplitude at a realistic level, and each column of the noise data is also multiplied by the maximum absolute value of the respective column, to maintain the scaling between columns. Doing so, the training data using the second method to add the scaled noise into the original data is almost identical to Figure 4 once converted back to nondeviation form. This data set, generated by the second method, after rescaling, when used in the proposed algorithm following the procedure described in the remainder of the manuscript, yielded results almost identical to what will be presented with no visible differences in the plots. Therefore, for the purposes of this work, there is no practical difference whether the data is scaled and then corrupted with a fixed percentage level of noise, or if noise of a fixed percentage level is first generated and then scaled appropriately to be added to each variable. In an industrial setting, the noise would be present in the measured data, but in a manner similar to how it was added in the second method such that the noise is appropriately scaled for each variable. This is because the resolution of the sensor would be designed according to the range of values that will be measured by the sensor. Hence, variables with generally smaller values will likely use sensors that have smaller errors and variances.

Remark 5: The present work focuses on the effect of Gaussian noise in the training data. Typically, non-Gaussian noise is more difficult to mitigate in the modeling step as shown in ref 11, where recurrent neural networks are shown to satisfactorily deal with Gaussian noise but struggle with non-Gaussian noise. In contrast, in ref 73, the author discusses how even an identical, independently distributed or i.i.d. Gaussian noise additive to the states can translate into correlated non-Gaussian effective noise when being used in downstream modeling algorithms, although the specific system studied was different. A detailed study of the effect of additive non-Gaussian noise to the training data is beyond the scope of this work and will be addressed in a future work.

The noisy data is then used to obtain estimates of the derivatives, which is one of the biggest challenges of using SINDy on noisy data. It was demonstrated in ref 37 that the best two methods for estimating the time-derivative in the development of SINDy models are the smoothed finite difference (SFD), where the Savitzky−Golay filter is used to presmooth the data before using finite differences to compute the derivatives, and the total-variation regularized derivative. In this work, SFD was found to be the optimal derivative estimator and was used in all simulations.

Next, the function library is created using the preprocessed, noisy training data. For this system, the set of basis functions chosen was a linear input term, monomials up to second order for the concentrations, the hyperbolic tangent of the temperature, an exponential term in temperature of the form $\frac{1}{1+e^{x_4-1}}$, as well as interactions between the six concentration functions (3 of each order) and the two temperature terms. Therefore, a total of 21 basis functions were considered. The temperature terms were considered in the above forms because of their large effect on the ODEs, the possibility of divergence when added as linear or quadratic terms, and the general understanding of chemical reaction engineering. Explicitly, the terms considered in the function library were x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, $\frac{1}{1+e^{x_4-1}}$, $\tanh x_4$, u, $\frac{x_1}{1+e^{x_4-1}}$, $\frac{x_2}{1+e^{x_4-1}}$, $\frac{x_3}{1+e^{x_4-1}}$, $\frac{x_1^2}{1+e^{x_4-1}}$, $\frac{x_2^2}{1+e^{x_4-1}}$, $\frac{x_3^2}{1+e^{x_4-1}}$, $x_1 \tanh x_4$, $x_2 \tanh x_4$, $x_3 \tanh x_4$, $x_1^2 \tanh x_4$, $x_2^2 \tanh x_4$, and $x_3^2 \tanh x_4$. Using this set of basis functions, Algorithm 1 is implemented in PySINDy,[74,75] a Python Application Programming Interface (API), to build the SINDy model. While identifying each submodel, some of the listed basis functions are dropped out before carrying out the optimization.

Remark 6: It is noted that the form of the temperature terms is highly specific. This was necessary because the choice of basis functions is critical to the performance of sparse identification. A poor selection of basis functions will not yield a sparse representation as such a representation may not exist. In this case, as mentioned, the large effects of linear and/or quadratic temperature terms on the ODE lead to unstable models that could not be integrated without the solution diverging. Hence, the general reaction engineering concept of the absolute temperature appearing as a negative exponential term was utilized to create the temperature basis function (x_4). The remaining six terms in x_4 such as $\frac{x_1}{1+e^{x_4-1}}$ were obtained when computing the interaction terms between concentrations and temperatures since such interactions are found in most material balance equations. This methodology would apply to any general system in consideration. For example, if the system in consideration was a four-tank system involving square root nonlinearities, and we could not obtain satisfactory performance using polynomial and/or trigonometric basis functions, the set of candidate basis functions would be expanded to specifically include square root basis functions in the tank heights since this relationship is well known. Doing so would very likely improve the performance of the sparse identification step by a significant margin.

In dropout-SINDy, three hyperparameters require to be tuned: λ, the sparsification knob, n_{dropout}, the number of library functions to be zeroed for each submodel, and n_{models}, the total number of submodels to be identified. The value of λ was tuned via a coarse search using values between 0.1 and 10 in steps of 0.1. Finer and/or wider ranges of λ did not improve the performance any further. Therefore, a value of 1 was used throughout the simulations. When n_{dropout} is too small, too few functions are dropped from the library to lead to a significant change in the model. However, increasing n_{dropout} too much produces excessive sparsity, leading to poor and/or unstable performance. Hence, a trade-off between model accuracy/complexity and stability exists. The optimal value was found to be $n_{\text{dropout}} = 7$ from extensive simulations in the entire range of $n_{\text{dropout}} \in [1, 21]$. Finally, when n_{models} is small, only a few models are identified, which may not include the optimal model. On the other hand, identifying too many models is computationally expensive and also promotes instability. This is possibly because a larger number of submodels become unstable and/or inaccurate, which affects the median of the coefficients. Therefore, extensive simulations are conducted to obtain a value of $n_{\text{models}} = 10$, which is found to produce the best model with a short processing time. Due to the length of the ODE models produced, a visualization of the coefficients associated with each library function for each ODE model's right-hand side is provided in Figure 5.

Remark 7: If, instead of the median of the submodels, the best of all of the submodels is selected as the final model as mentioned in remark 2, it can be theorized that the overall model accuracy will only improve as n_{models} is increased, although at a higher computational cost.

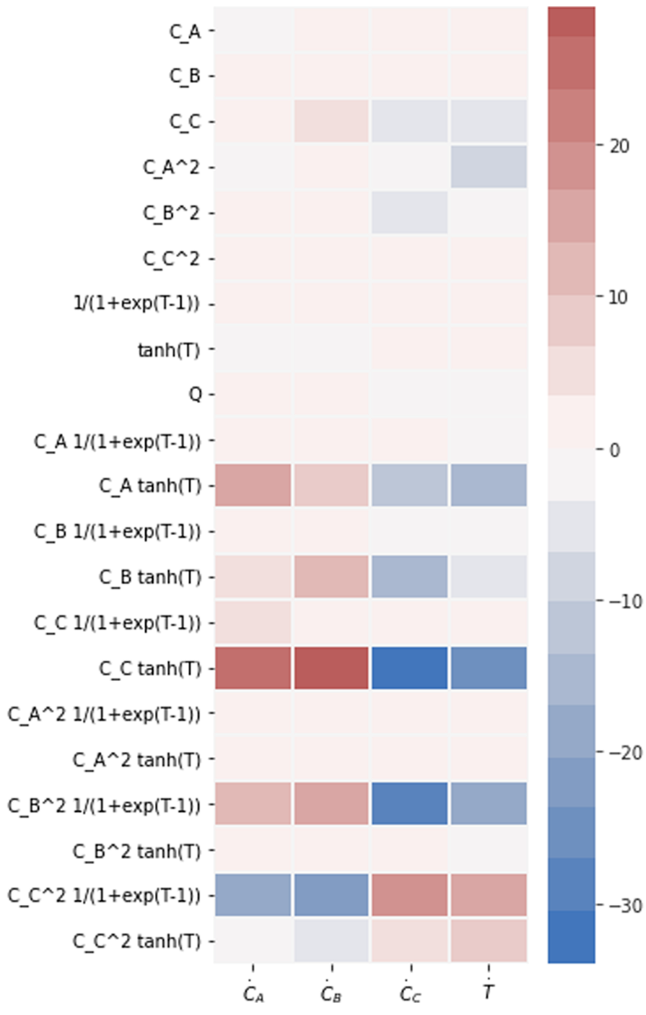


Figure 5. Visualization of dropout-SINDy identified model.

4.2.1. Corroboration between the Sparse-Identified Model and Known CSTR Dynamics. Although the model shown in Figure 5 may appear to contradict prior knowledge on CSTR dynamics, this is a well-known limitation of any system identification technique, not only limited to sparse identification. There are many factors that affect exactly what terms are needed in a system identification method to minimize the chosen error criterion. This is especially true for sparse identification, where there are many basis functions. Based on the comprehensive literature review, except for very simple cases, there will often be a competing basis function (or subset of basis functions) that has (or have) similar dynamics.

As a simple illustrative example, the family of systems, $\dot{x} = -x^n$, where $n = 1, 2, 3$ may be considered. The dynamics of this family of systems are very similar in trend, i.e., they are similar to a decaying exponential, with the speed of decay being the primary difference. As n increases, the dynamics get slower.

If we attempt to identify the system, $\dot{x} = -x^2$, using data in the range of $t \in [0, 1]$, sampled with the same sampling time as used in the paper of $\Delta = 0.01$, we obtain $\dot{x} = -0.999x^2$, which is very accurate. However, if we shorten the data range to $t \in [0, 0.2]$, where the different systems in the family are quite close in terms of dynamics, the identified model degrades to $\dot{x} = -0.111x - 0.783x^2 - 0.107x^3$, which appears to be inaccurate. However, a plot of this "inferior" model indicates it is not as poor as it seems, and the maximum absolute error and MSE of 0.006 and 9×10^{-6} both confirm this (in contrast, x varies between 0.5 and 1 over the entire data set). One way to improve the model, however, was found to be to decrease the sampling time of the data to $\Delta = 0.001$. This led to the x^2 term beginning to dominate the right-hand side of the ODE once more, proving that the loss of information from the discrete sampling procedure also contributes to the model attempting to capture the dynamics using other basis functions in the candidate library.

A more relevant example is the identification of the system, $\dot{x} = -x^3$, in the same family of systems above. In this case, even when using the entire range of $t \in [0, 1]$, the identified model is

$$\dot{x} = -1.556 + 5.976x - 7.484x^2 + 2.065x^3$$

First, once again, this model is very close to $\dot{x} = -x^3$ as seen in the maximum absolute error and MSE of 0.001 and 4×10^{-7}. However, we note the coefficient of x^3 is positive and, in fact, the coefficients associated with x and x^2 are greater in magnitude. A possible explanation can be that, due to the range of x considered, the effect of increasing powers of x is less as n increases. Hence, the negative x^2 term, although not present in the actual system or data used, is sufficient to overpower the positive x and x^3 terms and yield the correct dynamics (because despite the positive x and x^3 terms, the graph is monotonically decreasing). From both of these case studies, it is clear that the exact model obtained is highly dependent on the region of data that is used in the sparse identification procedure, the sampling time, as well as the value of the variables in the basis functions. For the CSTR system, even though the variable C_A should be strongly associated with the first ODE ($\dot{C}_A$), it is possible that the values of C_A in the scaled space are larger than other variables on average, leading to smaller coefficients being associated with it. Alternately, as seen in the example of $\dot{x} = -x^3$ above, it is possible that other

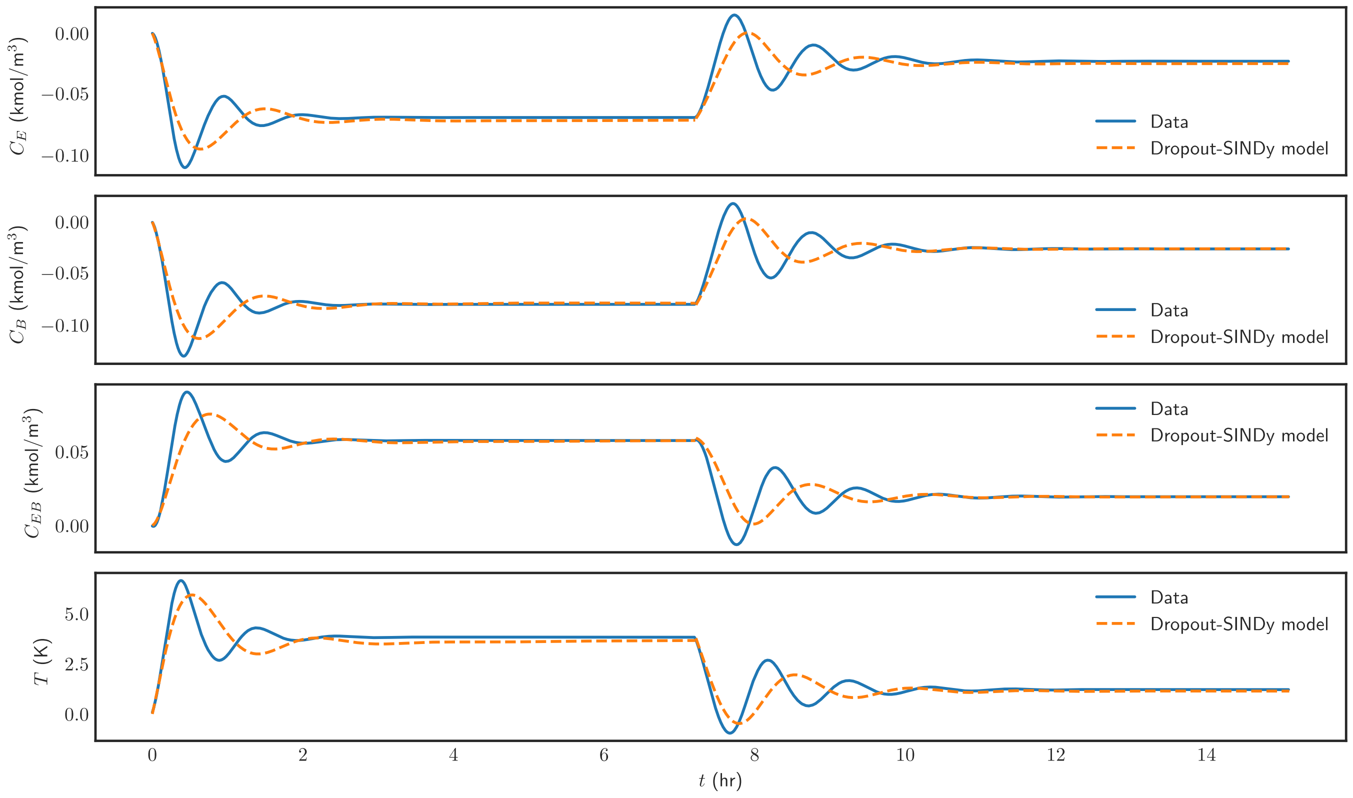


Figure 6. Time-varying profiles of the states for the open-loop simulation using Aspen Plus Dynamics (blue line) and the dropout-SINDy model (orange line) with two nonzero input values of $u_1 = -8.9869$ kg/s and $u_2 = -2.9869$ kg/s applied over the intervals $t_1 \in (0, 7.21]$ and $t_1 \in (7.21, 15.1]$, respectively.

terms, such as $C_C \tanh(T)$, have a dominant effect on the dynamics because of the range of temperatures and C_C in the data set, which leads to it being a major term in every ODE.

The goal of sparse identification is to simply minimize the objective function by optimizing the coefficients associated with the basis functions, whereas the prior process knowledge is incorporated into the choice of the basis functions themselves rather than their coefficients. While placing more constraints on the optimization to obtain a representation closer to the known CSTR dynamics may be possible, that will be a different method rather than sparse identification and is, hence, out of the scope of this article.

Remark 8: The minimization problem of sparse identification can be solved using a number of different algorithms. Besides STLSQ, specifically for minimizing the L_0 norm in sparse problems, greedy algorithms are also popular. Greedy algorithms make the best possible choice at every step but may not yield an overall optimal solution. Two highly popular and established greedy algorithms are the orthogonal matching pursuit (OMP) and its slightly improved yet more computationally costly variation, the orthogonal least squares (OLS).[76] In the PySINDy package, a further improved version of OLS known as Forward Regression Orthogonal Least Squares (FROLS) has been implemented based on ref 77. FROLS iteratively selects the most correlated function in the function library, using as its selection criterion the normalized increase in the explained output variance due to the addition of a given function to the basis. Due to the greedy nature of the algorithms, it is also simpler as there are no hyperparameters to tune in the FROLS algorithm. However, the models obtained for this system using FROLS were not stable and could not be integrated without diverging to infinity. The documentation of the PySINDy package has also demonstrated via extensive simulations with a number of minimizing algorithms that both OMP and FROLS are outperformed by a large margin by STLSQ in the case of noisy data. While OMP and FROLS are standard algorithms for solving the L_0 minimization problem, at least in the implementation available in PySINDy, they perform poorly on noisy data in particular. A possible cause is that greedy algorithms must select new terms by calculating correlations with the target data, which is noisy. One other algorithm available in PySINDy, which is frequently used, is the sparse relaxed regularized regression or SR3 algorithm. While it may be generally superior by formulation, in this case, both STLSQ and SR3 yielded nearly identical models with the same mean-square error values. Therefore, due to both the accuracy and simplicity of STLSQ, it was the only algorithm used in this work.

4.3. Open-Loop Simulation Results. The SINDy model obtained using dropout-SINDy as described in Section 4.2 is tested on the runs in the test set, which corresponds to open-loop tests under a fixed input u. One test run is shown in Figure 6. The SINDy model is observed to be able to correctly predict the evolution of the state from the origin to a new steady state under a nonzero input value of -8.9869 kg/s between $t = 0$ h and $t = 7.21$ h. Once the system reaches the first nonzero steady-state, from $t = 7.21$ h, a new input of -2.9869 kg/s is applied until $t = 15.1$ h. In both halves of the trajectory, it can be seen that the dynamics of the SINDy model are slightly slower and do not reach the correct peak values in deviation form. This is likely due to the denoising/prefiltering step in the estimation of the time derivatives since the Savitzky–Golay filter has a complex mechanism to compute the smoothed derivative including curve fitting and differentiating it. However, the overall dynamics are captured well and, most importantly, the steady states are accurately predicted by the SINDy model. For the other test runs described in Section 4.2, the plots showed similar trends in terms of slower dynamics but correct identification of the final steady states. To quantitatively measure the model accuracy, the MSE of the four states for this run is calculated and shown in Table 2. It is observed that the MSE for the concentrations is on the order of 10^{-4}, while the MSE for the temperature prediction is 0.75. This is similar in magnitude to previous results using more sophisticated neural network models in the presence of industrial noise.[10] As a final test, the SINDy model is initiated from zero initial conditions ($x_0 = 0$) under zero input ($u = 0$) to verify that the state remains at the origin for indefinite time under such conditions, which was found to be the case. This final verification step is important to ensure that, in the subsequent closed-loop implementation, the state can be driven to the origin and maintained there without using any more input u. The choice of data scaler/normalization used also greatly affects the results of this test at the origin.

Table 2. Open-Loop Prediction MSE Results for the Run Shown in Figure 6

State	MSE
C_E	1.9×10^{-4}
C_B	2.7×10^{-4}
C_{EB}	1.6×10^{-4}
T	0.75

4.4. Closed-Loop Simulation Results. After ensuring the quality of the dropout-SINDy model via open-loop testing, we incorporate the model into the LMPC of eq 8 to conduct closed-loop simulations. The control objective is to maintain the state of the reactor at the steady state (C_E, C_B, C_{EB}, T) = (0.456, 4.09, 3.18 kmol/m^3, 400 K) by manipulating the coolant flow rate $\dot{m}_{\text{coolant}}$. The objective function of the LMPC, eq 8a, is considered to be as follows to ensure a value of zero at the steady state itself under no-input conditions

$$L(x, u) = |x|^2_{Q_1} + |u|^2_{Q_2} \tag{13}$$

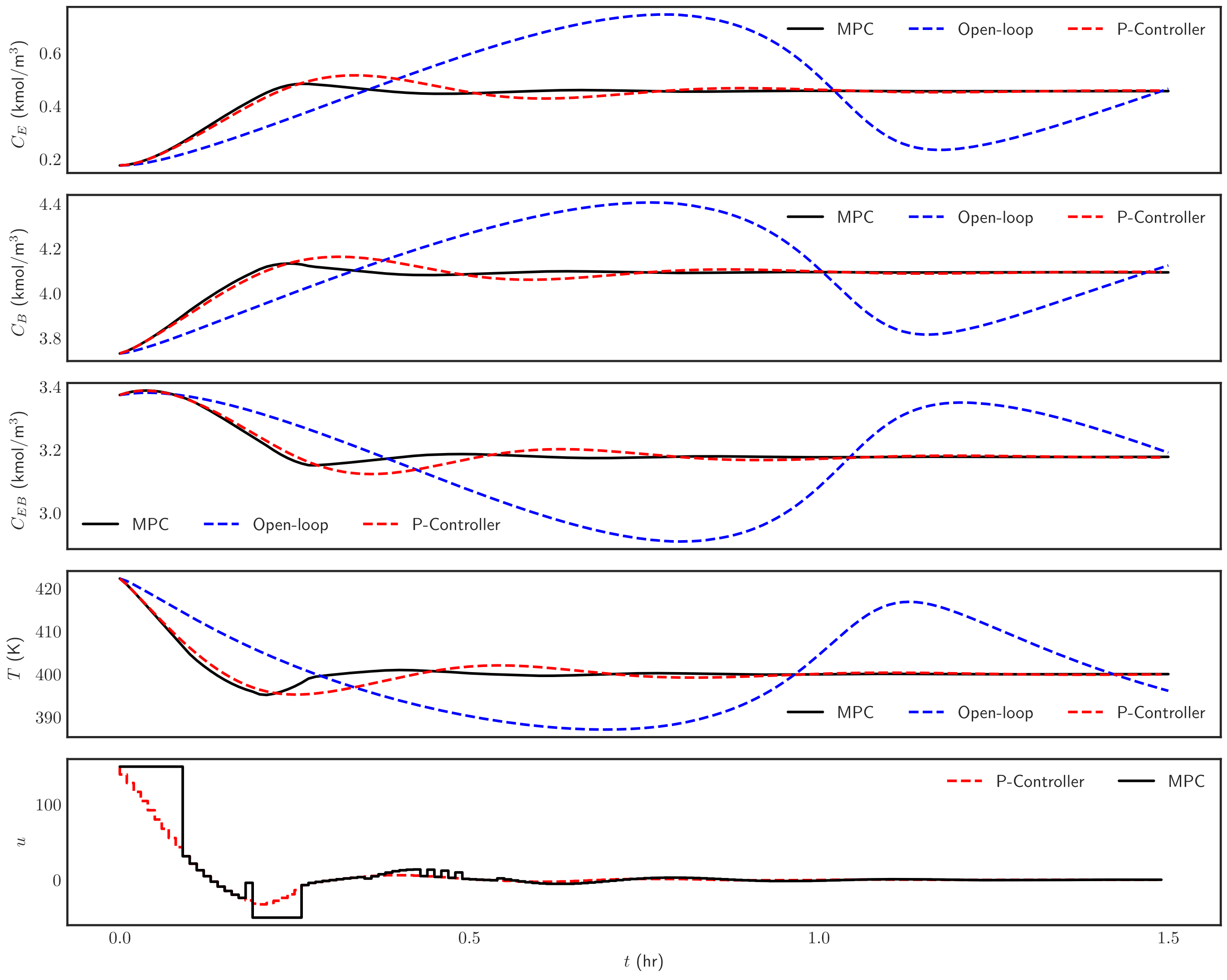


Figure 7. State and input profiles for the CSTR in closed loop under no control (blue line), a P-controller (red line), and the LMPC utilizing the dropout-SINDy model (black line) throughout the simulation period t_p = 1.5 h.

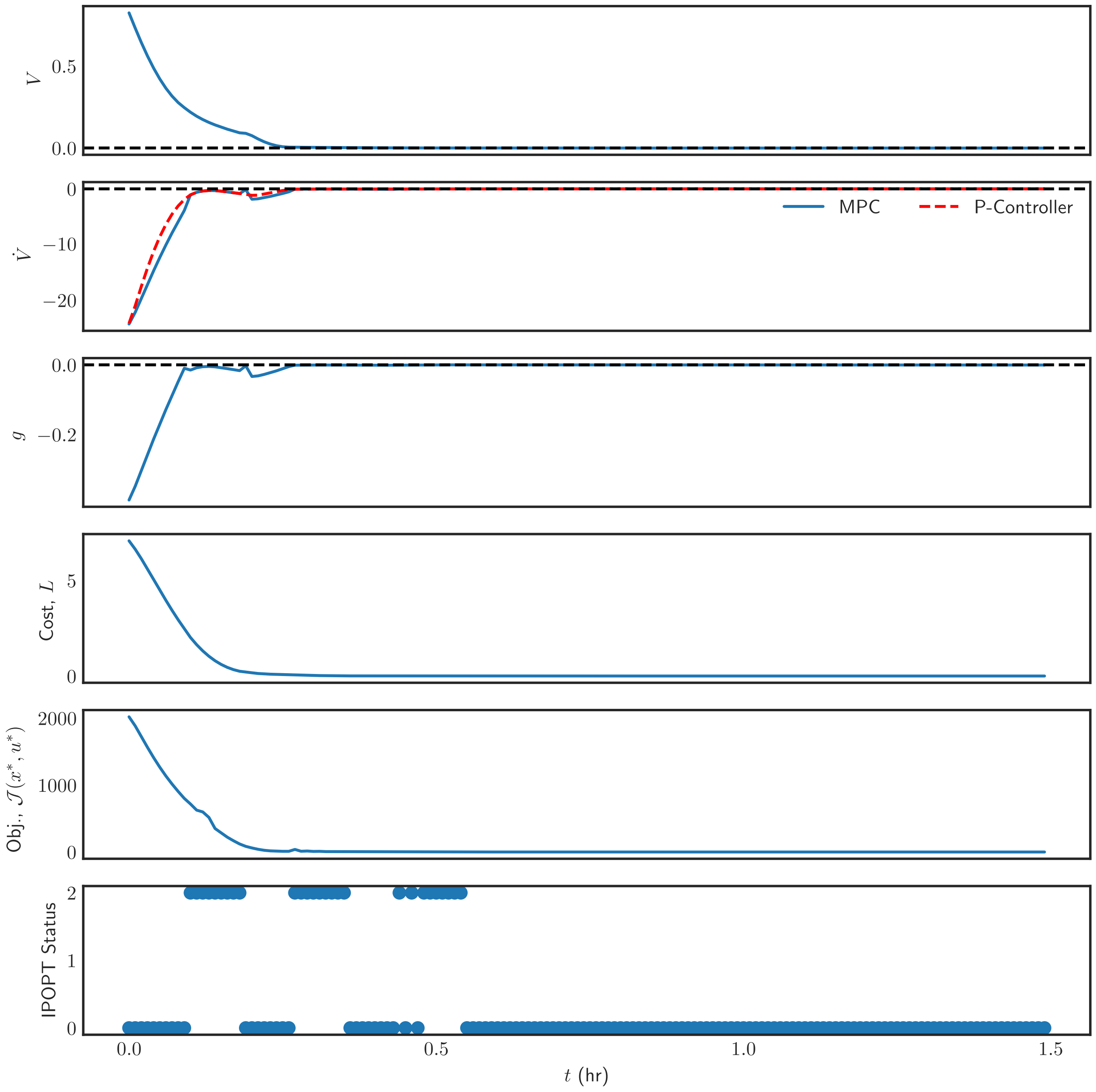


Figure 8. LMPC performance metrics throughout the simulation period t_p = 1.5 h.

where Q_1 and Q_2 are weighting matrices that control the contributions of the state and the input in the LMPC objective function, respectively. Q_1 is chosen to be the 4 × 4 identity matrix, while $Q_2 = 2 \times 10^{-6}$. As the optimization problem of eq 8 is nonlinear and nonconvex, we solve it every Δ = 0.01 h using the numerical solver Ipopt[78] with its Python module PyIpopt. The lower bound for the LMPC is chosen to be a stabilizing proportional or P-controller based on the error in the temperature and with a controller gain of 100.

Figure 7 depicts the closed-loop state and input profiles for the reactor operating without control action as well as under two controllers—the stabilizing P-controller and the LMPC using the dropout-SINDy model as the process model. From the state profiles, it can be observed that the uncontrolled states oscillate and do not reach close to the steady state within the simulation period t_p = 1.5 h. If the simulation is continued, it is observed that the uncontrolled state returns to the steady state after approximately 5 h, which was also observed in Section 4.2. during data generation. In contrast, both controllers are able to reduce the overshoot and, more importantly, rapidly bring the state of the system back to the origin, with the LMPC being significantly faster than the P-controller, especially with respect to the temperature. Specifically, the LMPC brings the states into the $\Omega_{\rho_{si}}$ region at t = 0.5 h in half the time compared to the P-controller, which takes t = 1 h, and a tenth of the time taken in the uncontrolled scenario.

The closed-loop performance of the two controllers is further compared in terms of the convergence of the states to the origin as well as energy consumption over the simulation duration. This is carried out using the LMPC objective function as the metric since it accounts for the deviation in both states and input in its evaluation. Mathematically, it can be observed from eq 13 that a lower value of the objective function indicates both faster convergence and lower energy consumption i.e., lower coolant usage. Therefore, we compute the integral of the objective function of the LMPC over the entire closed-loop simulation period t_p, $\int_{t=0}^{t_p} L(x(\tau), u(\tau))\, d\tau$, for both controllers and also under open loop for comparison purposes. The cost function time integral values are found to be 429.9506 under open loop, 71.7027 under P-control, and 57.5497 under the LMPC. Therefore, it can be concluded from the lower value of L that both controllers greatly improve the convergence of the states, while the LMPC outperforms the P-controller in terms of overall convergence.

Figure 8 shows the various performance metrics of the LMPC throughout the simulation period. It is confirmed that the Lyapunov function V decreases at every sampling time and with a negative value of $\dot{V}$. The constraint functions are observed to be satisfied throughout the simulation duration, implying the LMPC is able to find a value of the input that is at least as effective as that calculated by the P-controller due to the contractive constraint of eq 8e. The cost and objective functions are also monotonously decreasing as expected. The nonlinear optimization solver Ipopt returns a status of 0 corresponding to a successfully solved problem for most ($\sim$70%) of the simulation; however, in some instances (e.g., between $t = 0.1$ h and $t = 0.2$ h), a status of 2, corresponding to an infeasible problem, is returned, in which case the LMPC uses the input calculated by the stabilizing P-controller that is selected as the lower bound in eq 8e, as also evidenced by the input profiles shown in Figure 7.

Remark 9: Although the results of the non-model-based control law Φ_{si} (in this case, P-controller) may be improved by considering integral and derivative control as well, this was found to be unnecessary for this system. Due to the high gain of the P-controller, there was no visible offset in the ultimate values of the states, as seen in the state profiles in Figure 7. Hence, no integral control was used. Since derivative control is typically necessary for excessive oscillations,[79] which were also not observed in this case, a P-controller was deemed sufficient. Most importantly, a stabilizing PID controller, even if found, would be selected to be the lower bound of the control action for the MPC, i.e., Φ_{si} as given in eq 8e. The goal of the MPC is to improve upon this input by solving the optimization problem of eq 8. In the event that such an input is already the optimal input and cannot be improved, the MPC uses this input. Therefore, even if a superior PID controller can be designed for this system, the MPC would improve upon it or perform at least as well as the non-model-based controller. There are other advantages, however, to using MPC, such as accounting for all of the states instead of only the temperature in the case of a MIMO system as the one studied, the MPC's ability to handle constraints, and also its stability guarantees based on converse Lyapunov theorems.

Remark 10: While it is standard practice to compare the performance of an MPC based on the proposed algorithm and a first-principle model-based MPC, in this case study, this was found to be impossible due to the complexity of the nonlinear process model in Aspen Plus. The proposed algorithm is aimed at solving such problems where no first-principles model is readily available due to the extreme nonlinearities and complexities. If such a first-principles model were available, subsampling-based SINDy[37] is a viable and possibly superior algorithm. However, that requires the aid of a first-principles model, which is not possible to be derived manually in this scenario. Hence, such a comparison cannot be made in this case study.

5. CONCLUSIONS

In this paper, sparse identification was combined with ensemble learning to model and control a nonlinear chemical process system using only noisy data from sensor measurements. A high-fidelity chemical process simulator, Aspen Plus Dynamics, was used to simulate a chemical reactor with multiple reactions, which was used for data generation as well as open- and closed-loop control demonstrations. In open-loop, it was found that the dropout-SINDy model could accurately predict the steady state of the system under an arbitrary input starting from any initial condition within the stability region. After confirming this, a dropout-SINDy-based LMPC was applied in closed-loop control to the reactor in Aspen Plus Dynamics. The LMPC depicted superior performance compared to the open-loop performance and a P-controller in terms of faster convergence.

■ AUTHOR INFORMATION

Corresponding Author

Panagiotis D. Christofides − *Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, California 90095, United States; Department of Electrical and Computer Engineering, University of California, Los Angeles, California 90095, United States;* orcid.org/0000-0002-8772-4348; Email: pdc@seas.ucla.edu

Authors

Fahim Abdullah − *Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, California 90095, United States;* orcid.org/0000-0002-5094-788X

Mohammed S. Alhajeri − *Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, California 90095, United States; Department of Chemical Engineering, Kuwait University, Safat 13060, Kuwait*

Complete contact information is available at:
https://pubs.acs.org/10.1021/acs.iecr.2c02639

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

Professor Zhe Wu contributed to the development of ideas and the improvement of the final draft. Financial support from the National Science Foundation and the Department of Energy is gratefully acknowledged.

■ REFERENCES

(1) Aggelogiannaki, E.; Sarimveis, H. Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models. *Comput. Chem. Eng.* **2008**, *32*, 1225−1237.

(2) Al Seyab, R.; Cao, Y. Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation. *J. Process Control* **2008**, *18*, 568−581.
(3) Zeng, J.-s.; Gao, C.; Su, H. Data-driven predictive control for blast furnace ironmaking process. *Comput. Chem. Eng.* **2010**, *34*, 1854−1862.
(4) Aumi, S.; Corbett, B.; Clarke-Pringle, T.; Mhaskar, P. Data-driven model predictive quality control of batch processes. *AIChE J.* **2013**, *59*, 2852−2861.
(5) Xie, W.; Bonis, I.; Theodoropoulos, C. Data-driven model reduction-based nonlinear MPC for large-scale distributed parameter systems. *J. Process Control* **2015**, *35*, 50−58.
(6) Chaffart, D.; Ricardez-Sandoval, L. A. Optimization and control of a thin film growth process: A hybrid first principles/artificial neural network based multiscale modelling approach. *Comput. Chem. Eng.* **2018**, *119*, 465−479.
(7) Garg, A.; Mhaskar, P. Utilizing big data for batch process modeling and control. *Comput. Chem. Eng.* **2018**, *119*, 228−236.
(8) Abdullah, F.; Wu, Z.; Christofides, P. D. Data-based reduced-order modeling of nonlinear two-time-scale processes. *Chem. Eng. Res. Des.* **2021**, *166*, 1−9.
(9) Abdullah, F.; Wu, Z.; Christofides, P. D. Sparse-identification-based model predictive control of nonlinear two-time-scale processes. *Comput. Chem. Eng.* **2021**, *153*, No. 107411.
(10) Wu, Z.; Luo, J.; Rincon, D.; Christofides, P. D. Machine learning-based predictive control using noisy data: evaluating performance and robustness via a large-scale process simulator. *Chem. Eng. Res. Des.* **2021**, *168*, 275−287.
(11) Wu, Z.; Rincon, D.; Luo, J.; Christofides, P. D. Machine learning modeling and predictive control of nonlinear processes using noisy data. *AIChE J.* **2021**, *67*, No. e17164.
(12) Alhajeri, M. S.; Abdullah, F.; Wu, Z.; Christofides, P. D. Physics-informed machine learning modeling for predictive control using noisy data. *Chem. Eng. Res. Des.* **2022**, *186*, 34−49.
(13) Ren, Y. M.; Alhajeri, M. S.; Luo, J.; Chen, S.; Abdullah, F.; Wu, Z.; Christofides, P. D. A tutorial review of neural network modeling approaches for model predictive control. *Comput. Chem. Eng.* **2022**, *165*, No. 107956.
(14) Moore, C.Application of Singular Value Decomposition to the Design, Analysis, and Control of Industrial Processes. In *Proceedings of the American of Industrial Processes Conference*; IEEE: Seattle, WA, 1986; pp 643−650.
(15) Van Overschee, P.; De Moor, B. N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica* **1994**, *30*, 75−93.
(16) Siegelmann, H.; Horne, B.; Giles, C. Computational capabilities of recurrent NARX neural networks. *IEEE Trans. Syst. Man Cybern., Part B* **1997**, *27*, 208−215.
(17) Menezes, J. M. P.; Barreto, G. A. Long-term time series prediction with the NARX network: An empirical evaluation. *Neurocomputing* **2008**, *71*, 3335−3343.
(18) Huusom, J. K.; Poulsen, N.; Jørgensen, S.; Jørgensen, J. Tuning SISO offset-free Model Predictive Control based on ARX models. *J. Process Control* **2012**, *22*, 1997−2007.
(19) Brunton, S. L.; Proctor, J. L.; Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci. U.S.A.* **2016**, *113*, 3932−3937.
(20) de Silva, B. M.; Higdon, D. M.; Brunton, S. L.; Kutz, J. N. Discovery of Physics From Data: Universal Laws and Discrepancies. *Front. Artif. Intell.* **2020**, *3*, 25.
(21) Mangan, N. M.; Brunton, S. L.; Proctor, J. L.; Kutz, J. N. Inferring Biological Networks by Sparse Identification of Nonlinear Dynamics. *IEEE Trans. Mol., Biol., Multi-Scale Commun.* **2016**, *2*, 52−63.
(22) Dam, M.; Brøns, M.; Juul Rasmussen, J.; Naulin, V.; Hesthaven, J. S. Sparse identification of a predator-prey system from simulation data of a convection model. *Phys. Plasmas* **2017**, *24*, No. 022310.
(23) Tran, G.; Ward, R. Exact Recovery of Chaotic Systems from Highly Corrupted Data. *Multiscale Model. Simul.* **2017**, *15*, 1108−1129.
(24) Boninsegna, L.; Nüske, F.; Clementi, C. Sparse learning of stochastic dynamical equations. *J. Chem. Phys.* **2018**, *148*, No. 241723.
(25) Schaeffer, H.; Tran, G.; Ward, R. Extracting Sparse High-Dimensional Dynamics from Limited Data. *SIAM J. Appl. Math.* **2018**, *78*, 3279−3295.
(26) Loiseau, J.-C.; Brunton, S. L. Constrained sparse Galerkin regression. *J. Fluid Mech.* **2018**, *838*, 42−67.
(27) Kaiser, E.; Kutz, J. N.; Brunton, S. L. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proc. R. Soc. A* **2018**, *474*, No. 20180335.
(28) Mangan, N. M.; Askham, T.; Brunton, S. L.; Kutz, J. N.; Proctor, J. L. Model selection for hybrid dynamical systems via sparse regression. *Proc. R. Soc. A* **2019**, *475*, No. 20180534.
(29) Zhang, L.; Schaeffer, H. On the Convergence of the SINDy Algorithm. *Multiscale Model. Simul.* **2019**, *17*, 948−972.
(30) Schaeffer, H.; Tran, G.; Ward, R.; Zhang, L. Extracting Structured Dynamical Systems Using Sparse Optimization With Very Few Samples. *Multiscale Model. Simul.* **2020**, *18*, 1435−1461.
(31) Shah, D.; Wang, J.; He, Q. P. Feature engineering in big data analytics for IoT-enabled smart manufacturing − Comparison between deep learning and statistical learning. *Comput. Chem. Eng.* **2020**, *141*, No. 106970.
(32) Diversi, R.; Guidorzi, R.; Soverini, U. Identification of ARX and ARARX Models in the Presence of Input and Output Noises. *Eur. J. Control* **2010**, *16*, 242−255.
(33) Wu, P.; Pan, H.; Ren, J.; Yang, C. A New Subspace Identification Approach Based on Principal Component Analysis and Noise Estimation. *Ind. Eng. Chem. Res.* **2015**, *54*, 5106−5114.
(34) Juricek, B. C.; Larimore, W. E.; Seborg, W. E. Reduced-Rank ARX and Subspace System Identification for Process Control. *IFAC Proc. Vol.* **1998**, *31*, 247−252.
(35) Patwardhan, S. C.; Narasimhan, S.; Jagadeesan, P.; Gopaluni, B.; L Shah, S. Nonlinear Bayesian state estimation: A review of recent developments. *Control Eng. Pract.* **2012**, *20*, 933−953.
(36) Yeo, K.; Melnyk, I. Deep learning algorithm for data-driven simulation of noisy dynamical system. *J. Comput. Phys.* **2019**, *376*, 1212−1231.
(37) Abdullah, F.; Wu, Z.; Christofides, P. D. Handling noisy data in sparse model identification using subsampling and co-teaching. *Comput. Chem. Eng.* **2022**, *157*, No. 107628.
(38) Nguyen, D.; Ouala, S.; Drumetz, L.; Fablet, R.Assimilation-based Learning of Chaotic Dynamical Systems from Noisy and Partial Data. Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, IEEE: Barcelona, Spain, 2020; pp 3862−3866.
(39) de Silva, B. M.; Higdon, D. M.; Brunton, S. L.; Kutz, J. N. Discovery of Physics From Data: Universal Laws and Discrepancies. *Front. Artif. Intell.* **2020**, *3*, 25.
(40) Quade, M.; Abel, M.; Nathan Kutz, J.; Brunton, S. L. Sparse identification of nonlinear dynamics for rapid model recovery. *Chaos: Interdiscip. J. Nonlinear Sci.* **2018**, *28*, No. 063116.
(41) Leylaz, G.; Wang, S.; Sun, J.-Q. Identification of nonlinear dynamical systems with time delay. *Int. J. Dyn. Control* **2021**, *10*, 13−24.
(42) Ramsay, J. O.; Hooker, G.; Campbell, D.; Cao, J. Parameter estimation for differential equations: a generalized smoothing approach. *J. R. Stat. Soc. B* **2007**, *69*, 741−796.
(43) Lin, M.; Cheng, C.; Peng, Z.; Dong, X.; Qu, Y.; Meng, G. Nonlinear dynamical system identification using the sparse regression and separable least squares methods. *J. Sound Vib.* **2021**, *505*, No. 116141.
(44) Hesthaven, J. S.; Gottlieb, S.; Gottlieb, D.*Spectral Methods for Time-Dependent Problems*; Cambridge Monographs on Applied and Computational Mathematics; Cambridge University Press: Cambridge, 2007; pp 117−134.

(45) Schaeffer, H. Learning partial differential equations via data discovery and sparse optimization. *Proc. R. Soc. A* **2017**, *473*, No. 20160446.
(46) Cortiella, A.; Park, K.-C.; Doostan, A. Sparse identification of nonlinear dynamical systems via reweighted l1-regularized least squares. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, No. 113620.
(47) Zhang, S.; Lin, G. SubTSBR to tackle high noise and outliers for data-driven discovery of differential equations. *J. Comput. Phys.* **2021**, *428*, No. 109962.
(48) Didonna, M.; Stender, M.; Papangelo, A.; Fontanela, F.; Ciavarella, M.; Hoffmann, N. Reconstruction of Governing Equations from Vibration Measurements for Geometrically Nonlinear Systems. *Lubricants* **2019**, *7*, 64.
(49) Feng, Z.; Shen, W.; Rangaiah, G.; Dong, L. Closed-loop identification and model predictive control of extractive dividing-wall column. *Chem. Eng. Process.: Process Intensif.* **2019**, *142*, No. 107552.
(50) Mendoza, D. F.; Palacio, L. M.; Graciano, J. E.; Riascos, C. A.; Vianna, A. S.; Le Roux, G. A. Real-time optimization of an industrial-scale vapor recompression distillation process. model validation and analysis. *Ind. Eng. Chem. Res.* **2013**, *52*, 5735−5746.
(51) Sharifzadeh, M. Integration of process design and control: A review. *Chem. Eng. Res. Des.* **2013**, *91*, 2515−2549.
(52) Lin, Y.; Sontag, E.; Wang, Y. A smooth converse Lyapunov theorem for robust stability. *SIAM J. Control Optim.* **1996**, *34*, 124−160.
(53) Massera, J. L. Contributions to stability theory. *Ann. Math.* **1956**, *64*, 182−206.
(54) Christofides, P. D.; El-Farra, N. H.*Control of Nonlinear and Hybrid Process Systems: Designs for Uncertainty, Constraints and Time-Delays*; Springer-Verlag: Berlin, Germany, 2005; p 18.
(55) Lin, Y.; Sontag, E. D. A universal formula for stabilization with bounded controls. *Syst. Control Lett.* **1991**, *16*, 393−397.
(56) Khalil, H. K.*Nonlinear Systems*, 3rd ed.; Prentice Hall: Upper Saddle River, New Jersey, 2002; p 115.
(57) Wang, W.-X.; Yang, R.; Lai, Y.-C.; Kovanis, V.; Grebogi, C. Predicting Catastrophes in Nonlinear Dynamical Systems by Compressive Sensing. *Phys. Rev. Lett.* **2011**, *106*, No. 154101.
(58) Schaeffer, H.; Caflisch, R.; Hauck, C. D.; Osher, S. Sparse dynamics for partial differential equations. *Proc. Natl. Acad. Sci. U.S.A.* **2013**, *110*, 6634−6639.
(59) Ozoliņš, V.; Lai, R.; Caflisch, R.; Osher, S. Compressed modes for variational problems in mathematics and physics. *Proc. Natl. Acad. Sci. U.S.A.* **2013**, *110*, 18368−18373.
(60) Mackey, A.; Schaeffer, H.; Osher, S. On the Compressive Spectral Method. *Multiscale Model. Simul.* **2014**, *12*, 1800−1827.
(61) Brunton, S. L.; Tu, J. H.; Bright, I.; Kutz, J. N. Compressive Sensing and Low-Rank Libraries for Classification of Bifurcation Regimes in Nonlinear Dynamical Systems. *SIAM J. Appl. Dyn. Syst.* **2014**, *13*, 1716−1732.
(62) Proctor, J. L.; Brunton, S. L.; Brunton, B. W.; Kutz, J. N. Exploiting sparsity and equation-free architectures in complex systems. *Eur. Phys. J.: Spec. Top.* **2014**, *223*, 2665−2684.
(63) Bai, Z.; Wimalajeewa, T.; Berger, Z.; Wang, G.; Glauser, M.; Varshney, P. K. Low-Dimensional Approach for Reconstruction of Airfoil Data via Compressive Sensing. *AIAA J.* **2015**, *53*, 920−933.
(64) Zheng, P.; Askham, T.; Brunton, S. L.; Kutz, J. N.; Aravkin, A. Y. A Unified Framework for Sparse Relaxed Regularized Regression: SR3. *IEEE Access* **2019**, *7*, 1404−1423.
(65) Alhajeri, M. S.; Luo, J.; Wu, Z.; Albalawi, F.; Christofides, P. D. Process structure-based recurrent neural network modeling for predictive control: A comparative study. *Chem. Eng. Res. Des.* **2022**, *179*, 77−89.
(66) Polikar, R.*Ensemble Machine Learning*; Springer, 2012; pp 1−34.
(67) Mendes-Moreira, J.; Soares, C.; Jorge, A. M.; Sousa, J. F. D. Ensemble approaches for regression: A survey. *ACM Comput. Surv.* **2012**, *45*, 1−40.
(68) Heidarinejad, M.; Liu, J.; Christofides, P. D. Economic model predictive control of nonlinear process systems using Lyapunov techniques. *AIChE J.* **2012**, *58*, 855−870.
(69) Wilson, Z. T.; Sahinidis, N. V. The ALAMO approach to machine learning. *Comput. Chem. Eng.* **2017**, *106*, 785−795.
(70) Wilson, Z. T.; Sahinidis, N. V.Data Driven Modeling in Alamo: Feature Selection and Non-Parametric Modeling Applications. AIChE Annual Meeting, Pittsburgh, PA, 2018.
(71) Norquay, S. J.; Palazoglu, A.; Romagnoli, J. Model predictive control based on Wiener models. *Chem. Eng. Sci.* **1998**, *53*, 75−84.
(72) Fruzzetti, K.; Palazoğlu, A.; McDonald, K. Nolinear model predictive control using Hammerstein models. *J. Process Control* **1997**, *7*, 31−41.
(73) AlMomani, A. A. R.; Sun, J.; Bollt, E. How entropic regression beats the outliers problem in nonlinear system identification. *Chaos: Interdiscip. J. Nonlinear Sci.* **2020**, *30*, No. 013107.
(74) de Silva, B.; Champion, K.; Quade, M.; Loiseau, J.-C.; Kutz, J.; Brunton, S. PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *J. Open Source Software* **2020**, *5*, 2104.
(75) Kaptanoglu, A. A.; de Silva, B. M.; Fasel, U.; Kaheman, K.; Goldschmidt, A. J.; Callaham, J.; Delahunt, C. B.; Nicolaou, Z. G.; Champion, K.; Loiseau, J.-C.; Kutz, J. N.; Brunton, S. L. PySINDy: A comprehensive Python package for robust sparse system identification. *J. Open Source Software* **2022**, *7*, 3994.
(76) Leibovitz, G.; Giryes, R. Efficient Least Residual Greedy Algorithms for Sparse Recovery. *IEEE Trans. Signal Process.* **2020**, *68*, 3707−3722.
(77) Billings, S. A.*Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-temporal Domains*; John Wiley & Sons, 2013; pp 70−102.
(78) Wächter, A.; Biegler, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* **2006**, *106*, 25−57.
(79) Coughanowr, D.*Process Systems Analysis and Control*, McGraw-Hill International Editions; McGraw-Hill, 1991; p 196.