UNIVERSITY OF CALIFORNIA

Los Angeles

Sparse Identification Modeling and Predictive Control of Nonlinear Processes

A dissertation submitted in partial satisfaction of the

requirement for the degree Doctor of Philosophy

in Chemical Engineering

by

Fahim Abdullah

2023

ABSTRACT OF THE DISSERTATION

Sparse Identification Modeling and Predictive Control of Nonlinear Processes

by

Fahim Abdullah

Doctor of Philosophy in Chemical Engineering

University of California, Los Angeles, 2023

Professor Panagiotis D. Christofides, Chair

Data is widely recognized as a crucial player in the fourth industrial revolution, in which engineers and computers must harness data to enhance the efficiency of industrial processes and their associated control systems. Traditional industrial process control systems rely on linear data-driven models, with parameters fitted to experimental or simulated data. In specific control loops, such as those critical for profit optimization, they may employ first-principles models describing the underlying physico-chemical phenomena but with a few data-derived parameters. Nevertheless, modeling complex, nonlinear processes on a large scale remains an open challenge in process systems engineering. The quality of these models depends on various factors, including model parameter estimation, model uncertainty, the number of assumptions made during model development, model dimensionality, structure, and the computational demands for real-time model solutions [1, 2]. This is especially pertinent as process models are integral to advanced model-based control systems, such as model predictive control (MPC) and economic MPC (EMPC). Designing MPC systems that utilize data-driven modeling techniques to account in real-time for large data sets is a new frontier that will impact the next generation of industrial control systems. While a significant body of research has been dedicated to the use of neural networks for nonlinear process modeling and control, in both the theoretical [3] and practical [4] domains, more computationally efficient models that can directly be used in MPC rather than their linearized counterparts, are still an growing area of research that can lead to the design of more robust and efficient control systems.

Motivated by the above considerations, this dissertation presents the use of a computationally efficient data-driven technique known as sparse identification in model predictive control for chemical processes described by nonlinear dynamic models. The motivation and organization of this dissertation are first presented. Then, the use of sparse identification to develop nonlinear dynamic process models to be used in model predictive controllers is presented, specifically addressing the challenges of two-time-scale systems, sensor noise, industrial nonlinearities, and process shifts. The MPC and economic MPC schemes that use sparse identified models are presented in detail with rigorous analysis provided on their closed-loop stability and recursive feasibility properties. Finally, the dissertation closes with an overview of the novelties introduced to overcome the aforementioned challenges and a detailed guide to developing nonlinear process models for complex chemical processes using sparse identification. Throughout the dissertation, the proposed methods are applied to numerical simulations of nonlinear chemical process examples and Aspen Plus simulations of large-scale chemical process networks to demonstrate their effectiveness.

The dissertation of Fahim Abduullah is approved.

University of California, Los Angeles

2023

Contents

List of Figures

List of Tables

Acknowledgements

I would like to deliver my deepest gratitude to my advisor, Professor Panagiotis D. Christofides, for his support, encouragement, and guidance throughout my academic journey over the past four years. Professor Christofides exemplifies excellence as a researcher, mentor, instructor, and role model. I consider myself fortunate to be his student, as this Ph.D. experience has provided me with a solid foundation for my future career. I would also like to extend my sincere gratitude to my doctoral committee: Professor Morales-Guio, Professor Srivastava, and Professor Tsao for their time, comments, and valuable advice.

In addition, I would like to thank all of my colleagues in the Christofides research group, including Dr. David Rincon, Dr. Zhihao Zhang, Dr. Scarlett Chen, Professor Mohammed Alhajeri, Dr. Sungil Yun, Aisha Alnajdi, Matthew Tom, Feiyang Ou, Atharva Suryavanshi, and Yash Kadakia. I would like to particularly thank Professor Zhe Wu, Dr. Yi Ming Ren, Junwei Luo, and Berkay Citmaci who worked closely with me to tackle significant challenges in my Ph.D. journey.

Last but not least, I would like to deliver my thanks to my parents for their endless love and support and especially my sister, Anika, for her company and encouragement in the most difficult times of my degree.

Curriculum Vitae

Education

University of Alberta Sep 2014 - Jun 2018
B. Sc., Chemical Engineering **Edmonton, Canada**

Publications

1. Kadakia, Y., A. Alnajdi, **F. Abdullah**, and P. D. Christofides, "Encrypted Distributed Model Predictive Control with State Estimation for Nonlinear Processes," *Dig. Chem. Eng.*, **9**, 100133, 2023.

2. Kadakia, Y., A. Alnajdi, **F. Abdullah**, and P. D. Christofides, "Encrypted Decentralized Model Predictive Control of Nonlinear Processes with Delays," *Chem. Eng. Res. & Des.*, **200**, 312–324, 2023.

3. Kadakia, Y., A. Suryavanshi, A. Alnajdi, **F. Abdullah**, and P. D. Christofides, "Integrating Machine Learning Detection and Encrypted Control for Enhanced Cybersecurity of Nonlinear Processes," *Comp. & Chem. Eng.*, **180**, 108498, 2024.

4. Luo, J., B. Çıtmacı, J. B. Jang, **F. Abdullah**, C. G. Morales-Guio, and P. D. Christofides, "Machine Learning-Based Predictive Control Using On-line Model Linearization: Application to an Experimental Electrochemical Reactor," *Chem. Eng. Res. & Des.*, **197**, 721–737, 2023.

5. **Abdullah, F.** and P. D. Christofides, "Real-Time Adaptive Sparse-Identification-Based Predictive Control of Nonlinear Processes," *Chem. Eng. Res. & Des.*, **196**, 750–769, 2023.

6. Kadakia, Y., A. Suryavanshi, A. Alnajdi, **F. Abdullah**, and P. D. Christofides, "Encrypted Model Predictive Control of a Nonlinear Chemical Process Network," *Processes*, **11 (8)**, 2501, 2023.

7. Luo, J., **F. Abdullah**, and P. D. Christofides, "Model Predictive Control of Nonlinear Processes Using Neural Ordinary Differential Equation Models," *Comp. & Chem. Eng.*, **178**, 108367, 2023.

8. Alnajdi, A., **F. Abdullah**, A. Suryavanshi and P. D. Christofides, "Machine Learning-Based Model Predictive Control of Two-Time-Scale Systems," *Mathematics*, **11 (18)**, 3827, 2023.

9. **Abdullah, F.** and P. D. Christofides, "Data-based Modeling and Control of Nonlinear Process Systems Using Sparse Identification: An Overview of Recent Results," *Comp. & Chem. Eng.*, **174**, 108247, 2023.

10. Suryavanshi, A., A. Alnajdi, M. S. Alhajeri, **F. Abdullah**, and P. D. Christofides, "Encrypted Model Predictive Control Design for Security to Cyber-Attacks," *AIChE J.*, **69**, e18104, 2023.

11. Dale P., O. AlShareedah, M. Sarkar, J. Arabit, I. Mehdipour, S. Afzal, J. Luo, **F. Abdullah**, S. Yun, P. D. Christofides, D. Simonetti, and G. Sant, "Process Modeling Guides Operational Variables that affect CO_2 Utilization During the Accelerated Carbonation of Concrete," *AIChE J.*, accepted.

12. Alnajdi, A., A. Suryavanshi, M. S. Alhajeri, **F. Abdullah**, and P. D. Christofides, "Machine Learning-Based Predictive Control of Nonlinear Time-Delay Systems: Closed-loop Stability and Delay Compensation," *Dig. Chem. Eng.*, **7**, 100084, 2023.

13. Alhajeri, M. S., A. Alnajdi, **F. Abdullah**, and P. D. Christofides, "On Generalization Error of Neural Network Models and its Application to Predictive Control of Nonlinear Processes," *Chem. Eng. Res. & Des.*, **189**, 664–679, 2023.

14. **Abdullah, F.**, M. S. Alhajeri, and P. D. Christofides, "Modeling and Control of Nonlinear Processes Using Sparse Identification: Using Dropout to Handle Noisy Data," *Ind. & Eng. Chem. Res.*, **61**, 17976–17992, 2022.

15. Ren, Y. M., M. S. Alhajeri, J. Luo, S. Chen, **F. Abdullah**, Z. Wu, and P. D. Christofides, "A Tutorial Review of Neural Network Modeling Tools for Model Predictive Control," *Comp. & Chem. Eng.*, **165**, 107956, 2022.

16. Alhajeri, M., **F. Abdullah**, Z. Wu, and P. D. Christofides, "Physics-informed Machine Learning Modeling for Predictive Control Using Noisy Data," *Chem. Eng. Res. & Des.*, **186**, 34–49, 2022.

17. **Abdullah, F.**, Z. Wu, and P. D. Christofides, "Handling Noisy Data in Sparse Model Identification Using Subsampling and Co-teaching," *Comp. & Chem. Eng.*, **157**, 107628, 2022.

18. **Abdullah, F.**, Z. Wu, and P. D. Christofides, "Sparse Identification-Based Model Predictive Control of Two-Time-Scale Processes," *Comp. & Chem. Eng.*, **153**, 107411, 2021.

19. **Abdullah, F.**, Z. Wu, and P. D. Christofides, "Data-Based Reduced-Order Modeling of Nonlinear Two-Time-Scale Processes," *Chem. Eng. Res. & Des.*, **166**, 1–9, 2021.

20. Binyaminov, H., **F. Abdullah**, L. Zargarzadeh, and J. A. W. Elliott, "Thermodynamic Investigation of Droplet–Droplet and Bubble–Droplet Equilibrium in an Immiscible Medium," *J. Phys. Chem. B*, **125 (30)**, 8636–8651, 2021.

Chapter 1

Introduction

1.1 Motivation

In numerous industries, predictive maintenance, production management/scheduling, optimization of operating conditions, and advanced process control play central roles in the daily as well as long-term functioning and economics of the plant. However, these technologies, among others, require knowledge of the behavior of the process, specifically in the form of dynamic process models that describe time-varying operation. Traditionally, industrial process control systems rely on linear data-driven models with parameters derived from industrial or simulation data [5, 6]. In some cases, particularly for critical control loops that impact profitability, first-principles models are employed. These first-principles models are constructed either entirely based on knowledge of the physico-chemical phenomena or proposed in a general form and (some of) its parameters are then fitted to data from experiments or simulations. The resulting model usually describes the underlying physico-chemical phenomena satisfactorily provided that the *a priori* knowledge was accurate. Nonetheless, most engineering systems are highly complex, typically due to severe nonlinearities, time-scale multiplicities, high dimensionality, and/or interactions between variables, leading to a poor understanding of the internal mechanisms of the system which, in turn, complicates the derivation of first-principle models following the aforementioned methodology. Furthermore, model quality depends on various factors, including model parameter estimation, model uncertainty, the number of assumptions made during model development, dimensionality, model structure, and the computational burden of real-time model solving [1, 2]. While systems have increased in scale and complexity over time, the availability of data due to sensors as well as computational power of modern devices have increased exponentially over the last two decades, resulting in a shift towards empirically constructing models for such complex systems from data rather than first-principles.

The mathematical form of models for many physical systems, i.e., the laws governing their dynamics, can be written as ordinary differential equations (ODE) or partial differential equations (PDE) in terms of time and/or space as independent variables. Prominent examples within chemical engineering include the Boltzmann equation in thermodynamics and the Navier–Stokes equations in fluid dynamics [7]. Practically, since PDEs can also be spatially discretized into a finite set of ODEs, the dissertation focuses on ODEs rather than PDEs. For dynamic model building, data-based system identification techniques have been investigated since the twentieth century. Input/output (I/O) system identification methods such as autoregressive moving average with exogenous input (ARMAX) that identify a system based on only input/output data were introduced relatively early [8] and matured significantly over time. I/O methods were also readily extended to nonlinear systems such as methods that identify Volterra series [9], polynomial models [10, 11], and nonlinear autoregressive moving average with exogenous input (NARMAX) models [12], possibly due to their specific structures/forms allowing for easier solutions with respect to finding the model parameters. Other I/O methods such as Wiener models [13], Hammerstein models [14], Hammerstein–Wiener models [15, 16], polynomial ARX models [17], multiple local linear models [18], and neural wiener models [19, 20] have also been developed and are established system identification methods that are widely used for modeling dynamical systems in applications such as model predictive control (MPC).

While I/O methods provide information on the input/output behavior of a dynamical system, the states themselves may not necessarily be recoverable from the identified I/O model, possibly providing only a minimal state-space realization. This limitation can be crucial in certain settings such as economic MPC, wherin the MPC cost function depends on most, if not all, of the states. The lack of state information in I/O methods led to the proposal and maturation of empirical state-space methods that provide information on the states in addition to the inputs and outputs. State-space methods mostly result in linear models but are widely used for dynamic systems due to several advantages they provide. For example, they have been studied in-depth in various literature, and a robust framework for applying state-space methods to data exists. Furthermore, once a state-space model is obtained, the breadth of functionality of such a model continues to expand until present [21]. Additionally, for modeling multi-input-multi-output (MIMO) systems, state-space methods have gained traction due to their ability to account for interactions between process states and inputs [22–29]. Some subspace identification methods include the numerical method for subspace state-space system identification (N4SID) based on Hankel singular values [30], the multivariable output error state-space algorithm (MOESP) [31–34], and the canonical variate algorithm (CVA) [35]. The incorporation of empirical state-space models into MPC has also been investigated [e.g., 36]. Nevertheless, the key limitation of linear state-space models is the fact that most chemical processes are highly nonlinear, and the degree of accuracy and applicability of a linear model in modeling a nonlinear system cannot be generally known. In theory, a nonlinear system can be represented by an extensive array of linear equations across various regions within the phase space. This can be accomplished by constructing Jacobian matrices over a suitable grid that encompasses the phase-space area of interest [37, 38]. Nevertheless, ensuring the accuracy and robustness of such an approach poses significant challenges, primarily due to the computational complexities involved. To effectively predict the dynamics of highly nonlinear systems, merely performing local reconstruction of multiple linearized dynamics is often insufficient and, instead, an accurate nonlinear equation is required. Furthermore, under certain conditions, such as when operating at an unstable steady state, the region of validity for the linear model can be extremely small, which can lead to poor performance of an MPC incorporating the linear model once the state exits the region of model validity. Consequently, the need for nonlinear state-space models was realized.

While I/O methods were already extended to their nonlinear counterparts unlike state-space methods, they were not without limitations in terms of capturing nonlinear dynamics. Most nonlinear I/O methods were easily tractable using classical computers and used widely due to the specific forms they were restricted to, facilitating computations and optimization of model parameters. For example, Hammerstein models assume the dynamics of a nonlinear system can be decomposed into a dynamic linear component and a static nonlinear component, with the nonlinear steady-state gain being applied to the linear dynamics. This assumption greatly simplifies the problem but also limits the applicability of models of this structure (e.g., Wiener models, which are of the same structure but with the components reversed) to processes where the nonlinearities cannot be decoupled from the dynamics. Other polynomial forms that have been used in nonlinear I/O modeling include Chebyshev polynomials [39], Volterra polynomials [10, 40], polynomial ARX models [17], Laguerre polynomials [41], and polynomial neural networks [10]. However, similar restrictions apply to the above methods, with some being derivatives of or closely interconnected to others (e.g., the Hammerstein model can be shown to be described fully by a second-order Volterra kernel).

To overcome the challenges of a presumed model structure that is encountered in both nonlinear I/O methods as well as linear state-space methods, research in process systems engineering and data-driven modeling has continued to progress, with significant advances in the last two decades, with the ultimate goal being to impose as few assumptions and restrictions on the model form as possible. In this vein, a nonlinear state-space method, polynomial nonlinear state-space (PNLSS), was recently proposed for MIMO systems as a system identification method that yields multivariable nonlinear models based on input/output data [42–47]. As a nonlinear state-space technique, PNLSS encompasses a significantly wider range of nonlinear systems than those accommodated by bilinear models, Hammerstein-Wiener models, or models featuring nonlinearities in either the states, inputs, or both. Due to the model itself being both nonlinear and dynamic, PNLSS models enabled operation across a broader state-space region. Since the PNLSS algorithm can produce ill-conditioned models that need an extremely small time step to be numerically integrated using explicit methods, additional constraints have been incorporated into the algorithm to enforce the identification of well-conditioned models [48]. While the bank of possible nonlinear functions for a dynamic model was significantly expanded via PNLSS to include monomials of any order, the overall model was restricted to be a polynomial, barring the possibility of other nonlinear functions such as trigonometric or exponential functions, which are ubiquitous in engineering systems.

However, due to the remarkable surge in data production within modern industries in the past ten years, with an estimated annual generation of over 1000 Exabytes by machines and devices [49], the explosive growth of computing resources, the widespread availability of cloud platforms/resources, the development of numerous open-source software libraries tailored for machine learning applications [e.g., 50–52], the paradigm of data-driven modeling has experienced a significant shift, with machine learning techniques, especially neural network (NN) models, being used to solve longstanding problems in classical engineering fields [53–59]. This is primarily due to their large number of tunable hyperparameters (leading to an even larger set of trainable parameters) leading to their ability to capture complex relationships between variables in a "black-box" manner, i.e. the model parameters are known and stored, but the exact input-output mapping found by the model cannot be expressed explicitly. In essence, NN models impose no restrictions on model form at the expense of significantly higher computational burden in both the training and testing phases of the model, compared to classical methods such as I/O models or state-space models, whether linear or nonlinear. When the required amount and quality of data is provided, a feedforward NN can capture any static nonlinearities extremely accurately if trained appropriately due to its universal approximation property [60, 61]. Recurrent NNs are another type of NN models that can accurately capture dynamics of a system from time-series data and have been proposed and used widely in process systems engineering in the last decade [3, 62]. The incorporation of both types of NNs into model-based controllers has also been investigated in-depth [63, 64] and reviewed at large [65, 66]. However, two limitations associated with NN models are their higher computational burden and black-box nature, which constrain their utility and acceptance within the field of process systems engineering and, especially, control loops with shorter sampling times. In particular, NN models typically require an exorbitant amount of data to be trained. While 1000 Exabytes of data may be generated annually, this applies neither to every industrial system nor to every quantity within a system. For example, a pilot plant based on a novel process such as an electrochemical reactor may not produce enough data until sufficient time passes, but a model-based controller may be required prior to that. Additionally, even if high-frequency measurements are available in a plant for a certain state such as a temperature, which can be easily and rapidly measured using a thermocouple, another quantity, such as gas-phase concentrations/compositions may require slower methodologies like gas chromatography, greatly reducing the sampling rate of certain variables. Due to NN models being, at their core, interpolators, NN models are also practically never extrapolated beyond the training data. Hence, an NN model trained on a certain region of the state-space or one steady state will likely not be accurate in another region or at another steady state and, even if it is accurate, this is not generally the case and is inadvisable to do so. Hence, despite the advancements in nonlinear dynamical modeling via the introduction of recurrent neural networks, other challenges were coupled with NN models that have not yet been overcome. As a result, the field of nonlinear dynamical modeling remains an open area of research. A more general nonlinear method that produces an explicit and well-conditioned model as classical I/O and state-space methods such as N4SID or PNLSS do is highly desirable from a control-centric perspective due to the computational efficiency and similarity to the governing physical laws in terms of model form, i.e., both the physical laws and state-space models are ordinary differential equations.

Regardless of the methodology used to obtain a data-driven model, in scenarios where the process is exposed to substantial plant disturbances, variations, actuator faults, or when there is a need to adapt to a different operating region, the initial data-driven model(s) may not adequately capture the dynamics that emerge under these new conditions. Consequently, depending on the model's structure and training requirements, it becomes essential to perform re-identification or update of the process model in real-time, especially when significant prediction errors arise. This re-identification/update is necessary to accommodate changes in the plant's dynamic behavior. Hence, when operational conditions undergo alterations, employing on-line model re-identification or updates within an MPC design is essential to compensate for variations in the plant model, which can prevent oscillations in the case of tracking MPC [67] or improve economic yield for economic MPC.

1.2 Background

In the realm of chemical process systems, model predictive control (MPC) stands as an advanced control system widely adopted in the industry [68]. As its name implies, MPC employs a dynamic model, often in the form of an ODE, to forecast process states (outputs) over a user-defined prediction horizon. This predictive capability enables the system to take optimal control actions based on anticipated future trajectories. A detailed review of classical methods as well as recent advancements on data-driven modeling within the context of model-based control can be found in [69], including entirely model-free controllers such as explicit MPC, where the control law itself is replaced by a function [70].

In order to relax constraints on model structure, maintain explicitness, and better capture the physics of systems, several studies concentrated on symbolic regression. Despite being computationally intractable for large-scale systems, symbolic regression was a successful direction in dynamic modeling. Concurrently, the notion of compressive sensing was used widely in the modeling of various dynamic systems including catastrophe prediction [71], classification of bifurcation regimes to derive low-dimensional Galerkin models [72], reconstruction of airfoil data [73], and various PDE systems such as convection equations and diffusion equations [74], multiscale PDEs [75], and elliptic PDEs [76]. Concepts from the simultaneous developments in symbolic regression and compressive sensing led to the development of a relatively modern technique known as Sparse Identification for Nonlinear Dynamics (SINDy). SINDy uses only sensor measurements to identify a dynamic model in the form of a system of first-order nonlinear ODEs expressed as a linear combination of general nonlinear functions whose associated coefficients can be calculated using established methods such as least-squares or LASSO (least absolute shrinkage and selection operator). The nonlinear functions considered are not restricted to any particular form, greatly increasing the range of systems that the method can be applied to. PNLSS, for example, can be considered a special case of SINDy when using only monomial candidate functions (and their interactions) for the nonlinear dynamics. It is important to note, while the initial aim of the SINDy method was to discover interpretable ODEs/PDEs that capture the underlying physics using data [e.g., 77, 78], SINDy's utility extends beyond this objective. It can be employed to construct computationally efficient, closed-form models for model-based control purposes. Moreover, the ODE models derived via the SINDy method may or may not adhere to the physical laws, in general. This is because the search across the set of nonlinear basis functions does not guarantee that the resulting dynamic model will provide such valuable physical insights. In the realm of process systems engineering, the objective of employing SINDy for constructing process models is to facilitate the direct identification of explicit and closed-form nonlinear first-order ODEs from data. These identified equations can be easily integrated into optimization problems, including MPC. The computational burden associated with incorporating these explicit ODE models is generally minimal, especially when the models are well-conditioned, due to the existence of efficient differential equation solvers that make use of established integration techniques, such as 4th/5th order Runge-Kutta methods. Since its inception, SINDy has found applications across a wide spectrum of problems in the natural sciences including power plants [79] and robotics [80].

A diverse array of literature can be found on advancements of SINDy to model chemical processes as well. SINDy has been expanded to model stochastic dynamical systems, which are commonly encountered in biophysical processes, in [81], and key challenges in its implementation were discussed along with data requirements and the necessity for cross validation. [82] used SINDy to develop reduced-order models for controlling hydraulic fracturing processes via SINDy-based MPC. The use of SINDy to identify chemical reaction networks have been investigated in several works [83–85]. SINDy was used to model the black-box component of a gray-box model developed for nonlinear MPC in [86], and the resulting MPC was applied to the Chylla-Haase benchmark reactor, whose temperature must be maintained in a narrow range of ± 0.5 K from its setpoint. In this case, interestingly, reaction kinetics, thermodynamic assumptions, and other *a priori* knowledge of the chemical process were included as part of the white-box model of the overall gray-box model, further emphasizing that SINDy is not restricted to the case of obtaining the underlying physical laws of a system but can be used for optimization and model-based control strategies. Dynamic models were constructed for a distillation column, which is the most common unit found in chemical processes, using both SINDy and symbolic regression in [87] and compared with each other, where SINDy identified more accurate models (although not in the form of the underlying equations) within the perturbation region. For the purpose of modeling time-series data gathered from multiple air quality stations in Spain, a novel optimization approach for solving the sparse identification problem was introduced in [88] and subsequently applied to model ozone and nitrogen dioxide concentrations in the atmosphere as functions of time, taking into account the issue of data corruption due to noise, which is prevalent in real-world datasets. SINDy has also been attempted to be extended to large-scale systems such to model an overall plant's dynamics in [89], with varying degrees of success. The sparse identification problem was modified to use constrained nonlinear optimization to improve the coefficient identification and the resulting problem was solved using MATLAB's *fmincon*, a sequential quadratic programming (SQP) solver. However, since a model of the entire plant's dynamics was required to account for both short-term and long-term water availability arising from rain and drought, considering streamflow dynamics was necessary. This complication introduced hysteresis in the SINDy model, necessitating the inclusion of time-derivatives of the input terms in the SINDy function library and subsequently deteriorating model performance when the hysteresis was significant. Most recently, [90] used SINDy to model Li-ion batteries, which are characterized by not only highly nonlinear dynamics but also time-scale separation due to the slow inter-cycle battery dynamics that determine the remaining battery lifespan and fast intra-cyclic dynamics that dictate the level of charge/power left within the current cycle. Two SINDy models were used to model the slow and fast dynamics separately, and the proposed framework was successfully demonstrated on a $LiFePO_4$/graphite battery system. While sparse identification has been developing on several fronts as described, other challenges when modeling chemical processes remain to be addressed.

Numerous chemical processes exhibit dynamics with multiple time scales, including systems like catalytic continuous stirred-tank reactors (CSTRs) [91], biochemical reactors [92–94], distillation columns [95], and fluidized catalytic crackers [91, 96, 97]. Nonlinear process models derived from SINDy are represented as ODEs in time and require integration for state and/or output predictions. However, such processes with fast dynamics result in stiff ODEs that require very small integration time steps to maintain numerical stability and accuracy when using explicit integration methods such as forward Euler or Runge-Kutta methods. From a purely modeling perspective, this may not necessarily be a limitation as models for both the slow and fast time scales were already remarked to have been constructed successfully in [90]. Earlier, as well, [98] has addressed the issue of modeling two-time-scale systems with SINDy, specifically focusing on data availability, sampling, and periodic systems that evolve on an attractor. When full state measurements were accessible, it was demonstrated that, while sparse identification could accurately identify single-time-scale periodic systems with as little as 5% of data covering one period, the data requirements increased linearly with the ratio of slow to fast dynamics in the case of two-time-scale systems, especially with naive simple uniform sampling. This increase in data requirements stemmed from the necessity for a shorter sampling period for fast trajectories. In fact, even in single-time-scale systems, where 5% of data sufficed for system reconstruction, it had to be collected from the relatively fast segments of the period at a notably high sampling rate because these fast regions contained considerably more information than the slower segments. To address this challenge of escalating data requirements with increasing time-scale multiplicities, an alternative sampling strategy was introduced in [98], known as burst sampling. Burst sampling enabled sparse identification to identify both the slow and fast subsystems using a significantly smaller dataset collected over a single period. A key limitation of [98], however, was the assumption of linear coupling across the time scales. While the advances and sampling strategies of [98] and [90] are significant and may be used to build stiff yet accurate SINDy models, the incorporation of stiff systems of ODEs directly into conventional nonlinear feedback control techniques can lead to issues such as controller ill-conditioning or loss of closed-loop stability [99]. Therefore, from a control perspective, an alternative approach is needed to handle the temporal evolution of fast states without relying on numerical integration of stiff ODEs, even if they can be identified using SINDy with careful tuning. Therefore, the construction of SINDy models for two-time-scale systems which can then be incorporated into model-based controllers is a crucial area of research yet to be explored.

Another challenge of implementing SINDy to model chemical processes is the ubiquitous presence of sensor noise in measurement data. This is due to the sparse identification algorithm requiring time-derivatives of the states, which are typically unavailable in chemical processes. The derivatives must then be numerically estimated, often by finite differences, which are unstable and yield highly random estimates even at low levels of noise. While the initial paper [77] suggested that the total-variation regularized derivative is robust to noise, a more detailed examination of noise levels, data generation, and sampling intricacies reveals that this claim cannot be universally applied to every case, especially in practical scenarios. When confronted with high levels of noise, the calculation of model parameters in sparse identification is significantly affected, with various factors hypothesized in the literature, such as the violation of the incoherence property [100–102] or the restricted isometry property [103–105]. Consequently, numerous investigations have been conducted to explicitly outline the influence of noise on sparse identification, resulting in several extensions to the original sparse identification algorithm to accommodate noise and other practical considerations. In [106], both noise and irregularly sampled data issues were addressed by introducing an assimilation step using an autoencoder or ensemble Kalman filter before the model-identification step. Partially available data and noise were also explored in [107], albeit at relatively low noise levels, with a signal-to-noise ratio (SNR) of 22.33. In [78], the discrepancy between the actual governing equations and the sparse-identified models, due to noise, was tackled through the utilization of group sparsity, which involved grouping terms based on partial knowledge of the underlying physics, ensuring that all or none of the terms appear in the model. It was observed that smoothing the data and using centered finite-differences generally improved performance. Furthermore, [108] utilized “algebraic operations”, specifically Laplace transforms and inverse Laplace transforms, to reformulate the ODE model using integral terms, mitigating noise in subsequent operations. In [109], the computation of the second derivative in mass-spring systems was avoided by using Duhamel’s integral, while further de-noising was proposed through the RKHS (Reproducing Kernel Hilbert Space)-based non-parametric de-noise method. In addition, while not discussed here in detail, various other methods have also been proposed to be combined with SINDy throughout the last few years to combat the effects of noise, including but not limited to the manifold boundary approximation method [110], spectral methods/filtering such as those in [111] to estimate derivatives [112], and implicit modeling [113, 114]. However, two papers pioneered the application of sparse identification to noisy data, [115], based on the second-order Tikhonov regularization, and [116], which involved randomly subsampling a fraction of the entire dataset multiple times, and then selecting the best model using a model-selection criterion. A key limitation in most of the papers mentioned, except for two, was that derivatives were calculated either using the ODE in the data generation step, with noise subsequently added, or the derivatives were estimated using numerical methods from the exact data generated, and noise was added to both the data and computed derivatives afterward. Based on the varying results in the two exceptions, [112] and [115], it can be concluded that the use of sparse identification to model highly noisy chemical processes where no clean data is available is still a significant challenge that requires further investigation. The challenge of handling noisy measurements, however, is further complicated by its close link to the data sampling period or rate. Specifically, data in industrial process systems must be acquired at finite, discrete time intervals, otherwise referred to as the sampling time. This limitation is inherent to the specific sensors and can vary depending on the type of variable being measured (e.g., a thermocouple can sample temperature data much faster than a gas chromatograph can provide compositional data). The sampling time plays a pivotal role in influencing the accuracy of SINDy, as the numerical estimation of derivatives relies on the temporal spacing between data points. This point is often unaccounted for in papers developing SINDy to handle noise since the data used may have been sampled on the order of microseconds, which is extremely rare in chemical processes. Therefore, the challenge of noisy data must be addressed together with the additional complication of data sampling rate, i.e., a robust algorithm to identify SINDy models from noisy data measured at a reasonable sampling frequency is a highly relevant challenge that has yet to be addressed in the current literature.

Finally, even when accurate SINDy models are obtained for a process from a large offline data set, possibly using advanced novel algorithms to handle time-scale multiplicities and noise, in real-world scenarios, process models are typically subject to changing dynamics over time due to a multitude of factors. These include external influences such as the aging of equipment, unexpected disturbances, and the implementation of new operational technologies, as well as internal factors like equipment fouling or catalyst deactivation. Consequently, the SINDy model, initially trained on historical data under normal operating conditions, may struggle to accurately predict process states in the presence of disturbances. The concept of updating SINDy models in real-time was initially introduced in [117], where the proposed methodology allowed for the adaptation of model coefficients, the addition/removal of terms, or a combination of these adjustments as required. The re-identification process was triggered by a noticeable deviation between the local Lyapunov exponent and the prediction horizon estimate, although it is worth noting that their definition of "prediction horizon" differs from its usage in this dissertation. While the findings of [117] were primarily limited to the modeling context and did not involve the use of closed-loop data in model updates, this methodology was recently adapted in [118] for the modeling of ducted fan aerial vehicles (DFAVs) with closed-loop control under MPC. Initially, an offline model was constructed using extensive knowledge of DFAVs, and then the model parameters were updated based on the approach proposed in [117], using the same Lyapunov exponent-based trigger for model updates. This methodology was, hence, demonstrated to be able to control DFAVs effectively, despite the complexity of modeling their airflow distribution under the disturbances considered, such as wind turbulence, which is a practical concern in the context of DFAVs. Most recently, within the paradigm of chemical process systems, an MPC framework using SINDy models with updates to accommodate changing process conditions was investigated in [119]. The focus of this work was on updating the model when faced with entirely unknown or newly encountered process conditions, and it considered processes operating under multiple operating conditions or steady states. It employed the discrete-time formulation of SINDy with actuation and used the popular greedy algorithm, orthogonal matching pursuit, to efficiently calculate changes to the model coefficients. During the model transition/update period, an elastic feedback correction method served as a temporary solution. The proposed error-triggered adaptive sparse identification for predictive control (ETASI4PC) method exhibited significant improvements over state-of-the-art methods, including SINDy without model updates and the well-developed OASIS (operable adaptive sparse identification of systems) approach, especially in the presence of significant disturbances in the feed flow rate of a chemical reactor system. It's important to note that the ETASI4PC method holds promise for updating SINDy models in real-time for a basic tracking MPC. However, as of our literature review, the operation of Lyapunov-based MPC, both Lyapunov-based tracking MPC (LMPC) and Lyapunov-based economic MPC (LEMPC), under SINDy models updated in real-time has not been thoroughly explored.

1.3 Dissertation objectives and structure

This dissertation presents numerous advancements to the sparse identification algorithm and its overall methodology, with the aim of expanding and facilitating its use in process systems engineering, along with control theoretic analyses and demonstration of the proposed algorithms through their application to various nonlinear chemical process examples. The primary objectives of this thesis can be outlined as follows:

1. To present a SINDy-based reduced-order modeling framework for two-time-scale systems and the integration of such models into Lyapunov-based MPC.

2. To design a framework to mitigate the numerical and practical challenges arising from the

use of noisy sensor data to develop nonlinear SINDy models.

3. To propose a SINDy-based online economic MPC methodology that updates its process model as required online using an error-based detector.

4. To create SINDy-based MPC schemes, accompanied by comprehensive theoretical analysis of their closed-loop stability properties.

5. To demonstrate how the proposed modeling and control frameworks can be applied to practical nonlinear chemical process scenarios.

The remainder of this dissertation is organized as follows: **Chapter 2** focuses on data-based reduced-order modeling of nonlinear processes that exhibit time-scale multiplicity. Using time-series data from all the state variables of a nonlinear process, an approach that involves nonlinear principal component analysis and neural network function approximators is employed to identify the fast and slow process state variables as well as to compute a nonlinear slow manifold function approximation in which the fast variables are “slaved” in terms of the slow variables. Subsequently, a nonlinear sparse identification approach is employed to calculate a dynamic model of nonlinear first-order ordinary differential equations that describe the temporal evolution of the slow process states. The method is applied to two chemical process examples, and the advantages of the proposed method over using only sparse identification for the original two-time-scale process are discussed. The approach can be thought of as a data-based analogue of the classical singular perturbation modeling of two-time-scale processes where explicit time-scale separation is assumed (and expressed in terms of a small positive parameter ϵ multiplying the time derivative of the fast states), and the reduced-order slow subsystem is calculated analytically.

Chapter 3 focuses on the design of model predictive controllers for nonlinear two-time-scale processes using only process measurement data. By first identifying and isolating the slow and fast variables in a two-time-scale process, the model predictive controller is designed based on the reduced slow subsystem consisting of only the slow variables, since the fast states can deteriorate controller performance when directly included in the model used in the controller. In contrast to earlier works, in the present work, the reduced slow subsystem is constructed from process data using sparse identification, which identifies nonlinear dynamical systems as first-order ordinary differential equations using an efficient, convex algorithm that is highly optimized and scalable. Results from the mathematical framework of singular perturbations are combined with standard assumptions to derive sufficient conditions for closed-loop stability of the full singularly perturbed closed-loop system. The effectiveness of the proposed controller design is illustrated via its application to a non-isothermal reactor with the concentration and temperature profiles evolving in different time-scales, where it is found that the controller based on the sparse identified slow subsystem can achieve superior closed-loop performance versus existing approaches for the same controller parameters.

In **Chapter 4**, a novel algorithm based on sparse identification, subsampling and co-teaching is developed to mitigate the problems of highly noisy data from sensor measurements in modeling of nonlinear systems. Specifically, sparse identification is combined with subsampling, a method where a fraction of the data set is randomly sampled and used for model identification, as well as co-teaching, a method that mixes noise-free data from first-principles simulations with the noisy measurements to provide a mixed data set that is less corrupted with noise for model training. The proposed method is bench-marked against sparse identification without subsampling as well as subsampling but without co-teaching using two examples, a predator-prey system and a chemical process, both of which are modeled as nonlinear systems of ordinary differential equations. It was shown that the proposed method yields better models in terms of prediction accuracy in the presence of high noise levels.

Chapter 5 studies the impact of noise on the SINDy algorithm, especially when applied to industrial processes with severe nonlinearities that cannot be adequately captured with an easily derivable first-principles model, hindering the use of co-teaching methods. 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 sub-model 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 hyper-parameters 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 requirements and performance specifications.

Chapter 6 introduces a sparse identification-based model predictive control (MPC) framework that incorporates on-line updates of the sparse-identified model to account for nonlinear dynamics and model uncertainty in process systems. The methodology involves obtaining a nonlinear first-order ordinary differential equation model using sparse identification for nonlinear dynamics (SINDy), which is integrated into two control schemes: Lyapunov-based MPC (LMPC) for achieving steady-state operation and Lyapunov-based economic MPC (LEMPC) for achieving both closed-loop stability and optimal economic performance. To improve prediction accuracy, an on-line model update scheme is proposed for the SINDy models. Specifically, an error-trigger mechanism that utilizes prediction errors and then uses the most recent process data to update the parameters of the SINDy model in real-time is designed. By incorporating the error-triggered on-line model updates in the SINDy-based LMPC and LEMPC, the dynamic performance of the process is enhanced, ensuring closed-loop stability, optimality, and smooth control actions. Following theoretical results on the boundedness of the closed-loop states and detailed discussions on the selection criteria for parameters of the error-triggered SINDy update scheme, the effectiveness of the proposed methodology is demonstrated through a chemical process example with time-varying disturbances under the LEMPC framework.

Chapter 7 discusses recent developments in the data-based modeling and control of nonlinear chemical process systems using sparse identification of nonlinear dynamics (SINDy). SINDy is a recent nonlinear system identification technique that uses only measurement data to identify model dynamical systems in the form of first-order nonlinear differential equations. In this work, the challenges of handling time-scale multiplicities and noisy sensor data when using SINDy are addressed. Specifically, a brief overview of novel methods devised to overcome these challenges are described, along with modeling guidelines for using the proposed techniques for process systems. When applied to two-time-scale systems, to overcome model stiffness, which leads to ill-conditioned controllers, a reduced-order modeling approach is proposed where SINDy is used to model the slow dynamics, and nonlinear principal component analysis is used to algebraically "slave" the fast states to the slow states. The resulting model can then be used in a Lyapunov-based model predictive controller with guaranteed closed-loop stability provided the separation of fast and slow dynamics is sufficiently large. To handle high levels of sensor noise, SINDy is combined with subsampling and co-teaching to improve modeling accuracy. The challenges of modeling and controlling large-scale systems using noisy industrial data are then addressed by using ensemble learning with SINDy. Finally, several future research directions for the incorporation of SINDy into process systems engineering are proposed.

Chapter 8 summarizes the main results of the dissertation.

Chapter 2

Data-based reduced-order modeling of nonlinear two-time-scale processes

2.1 Introduction

Advanced control methods such as model-predictive control (MPC) require a process model for predicting the states and/or outputs for the control and optimization of chemical processes. However, most chemical processes exhibit nonlinear behavior and, often, also multiple-time-scale dynamics. A number of such processes that have been studied include biochemical reactors [92–94], catalytic continuous stirred-tank reactors (CSTRs) [91], fluidized catalytic crackers [91, 96, 97], and distillation columns [95]. If the time-scale separation in such systems is neglected, directly implementing standard nonlinear feedback control methods may lead to controller ill-conditioning or loss of closed-loop stability [99].

As nonlinear process models are in the form of ordinary differential equations (ODEs) in time, they need to be integrated to predict the states and/or outputs. However the presence of fast dynamics in such processes result in stiff ODEs. Such stiff ODEs require a very small integration step size when using explicit integration methods to avoid numerical instability and to produce accurate solutions. An alternative is to compute the temporal evolution of the fast states using a different method than numerical integration of a stiff ODE.

Specifically, utilizing the mathematical framework of singular perturbations [99], the stiff ODE system is written in the standard singularly perturbed form, where a small positive parameter, ϵ, is used to multiply the time-derivative of the fast states. Using singular perturbations, the original system is decomposed into two lower-order subsystems by utilizing the inherent two-time-scale property of the system. Each lower-order subsystem separately represents the slow and fast dynamics of the original stiff ODE model and may be simpler to study individually. The asymptotic behavior (for small ϵ) of the original system can then be inferred based on the behavior of the lower-order subsystems. In particular, once the fast states converge to the slow manifold after a short transient period, there exists a nonlinear relationship between the slow and fast states, which can be taken advantage of.

To capture nonlinear relationships in data, Ref. [120] suggested a nonlinear generalization of principal component analysis, termed nonlinear principal component analysis (NLPCA). This was developed further in Ref. [121], where it was proved that this new method could accurately capture significant nonlinear relationships between the variables in a time-series data set. Furthermore, for the reconstruction of ODEs from data, a procedure known as sparse identification was devised in Ref. [77].

In a two-time-scale system, if the nonlinear relationships between the slow and fast states can be derived using NLPCA, then the fast subsystem may be predicted using a deterministic neural network (NN) model obtained from NLPCA with the slow subsystem as its input, with the results being valid after a short transient period. The evolution of the system on the slow manifold, on the other hand, can be predicted by integrating the sparse identified model with a larger integration step size compared to the one needed to integrate the original system of stiff ODEs. The objective of this work is to investigate an approach of combining NLPCA with sparse identification to efficiently predict the full state of the original stiff ODE system with minimal loss of accuracy except for a short transient period. Neural networks have already been used effectively in another two-time-scale process, chemical vapor deposition, for parameter estimation under uncertainty [122], dynamic optimization [123], and nonlinear control using MPC [124].

2.2 Preliminaries

2.2.1 Class of systems

The general class of continuous-time nonlinear systems with m states considered in this work has the following general form:

$$\dot{\bar{x}} = f(\bar{x}) \tag{2.1}$$

where $\bar{x} \in \mathbb{R}^m$ is the state vector, and the vector function $f(\bar{x})$ contains the dynamic modeling constraints that are inherently present in the system due to its physics. We assume that the system of Eq. (2.1) exhibits time-scale separation in the sense that it involves coupled slow and fast states where $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^p$ are the slow and fast state vectors, respectively, with $n + p = m$.

Traditionally, the system of Eq. (2.1) has been studied within the framework of singular perturbations where a small positive parameter ϵ representing the speed ratio of the slow to the fast dynamics of the system is used to write the system of Eq. (2.1) in the following standard singularly perturbed form:

$$\begin{aligned} \dot{x} &= f_1(x, z) \\ \epsilon \dot{z} &= f_2(x, z) \end{aligned} \tag{2.2}$$

where $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^p$ are the slow and fast state vectors, respectively, with $n + p = m$. The vector functions $f_1(x, z)$ and $f_2(x, z)$ are sufficiently smooth vector functions in $\mathbb{R}^n$ and $\mathbb{R}^p$, respectively. In the system of Eq. (2.2), after a short transient period, the fast states, z, converge to a slow manifold (provided such an isolated manifold exists) and can be expressed by a nonlinear algebraic expression in x, the slow states. In this work, nonlinear principal component analysis (NLPCA) is utilized to capture this nonlinear manifold relationship between the slow and fast states, while sparse identification is used to reconstruct the slow dynamic model for the slow states.

Remark 2.1. *Traditionally, two-time-scale systems of the form of Eq.* (2.2) *have been analyzed by setting ϵ to zero in Eq.* (2.2) *and calculating the slow manifold and the slow subsystem analytically. This requires knowing the vector functions $f_1(x, z)$ and $f_2(x, z)$. The objective of this work is to calculate the slow manifold and the slow subsystem using only time series data of $\bar{x}(t)$.*

2.2.2 Nonlinear principal component analysis

Principal component analysis (PCA) is a well-established dimensionality reduction technique for linearly mapping higher-dimensional data onto a lower-dimensional space with marginal loss of information by minimizing the sum of orthogonal deviations from a line. It is applicable to chemical processes as they contain voluminous data but much less information. However, as chemical processes are usually nonlinear, this can introduce inaccuracies [125] such as the minor components containing more than just noise or insignificant variance [126]. Therefore, for nonlinear dynamical systems such as chemical processes, NLPCA was developed.

2.2.2.1 Principal curve

In NLPCA as defined in Ref. [121], a "principal curve" [127] passing through the "middle" of the data set is first fitted to the data. This curve captures the nonlinear relationship between the variables by minimizing the sum of orthogonal deviations between the data and the curve. If not, analogously to PCA, the procedure can be carried out successively on the residuals to compute more components as required to capture the desired variance.

The principal curve is a 1-D curve in the m-D space of the states. It is represented by a vector function, $f(\mu)$, consisting of m functions of only the ordered arc-length along the curve, μ. If $\bar{x} \in \mathbb{R}^m$ is a continuous random vector with probability distribution, h, and $\|\cdot\|$ denotes the Euclidean norm of a vector, then the projection index, $\mu_f : \mathbb{R}^m \to \mathbb{R}$, can be defined as

$$\mu_f(\bar{x}) = \sup_{\mu \in \mathbb{R}} \{\mu : \big\|\bar{x} - f(\mu)\big\| = \inf_{\nu \in \mathbb{R}} \big\|\bar{x} - f(\nu)\big\|\} \tag{2.3}$$

where ν is a substitute for μ, and also represents the arc length. Hence, μ_f is the value of μ for which the curve, $f(\mu)$, is nearest to $\bar{x}$. In the case of multiple such values, the largest is used. A principal curve with distribution h can now be defined as the curve, f, such that

$$\mathbb{E}\big(\bar{x} | \mu_f(\bar{x}) = \mu\big) = f(\mu) \tag{2.4}$$

where $\mathbb{E}$ is the expectation operator.

The procedure to estimate the principal curve represented by Eq. (2.4) is described next. The challenge is that the expectation operator $\mathbb{E}$ in Eq. (2.4) can be computed only if the distribution of $\bar{x}$ is known. However, in most engineering applications, including the systems of interest in this work, $\bar{x}$ is a discrete, multivariate data set with unknown distribution sampled from a process. Hence, as suggested in Ref. [127], we estimate the principal curve, given by the left-hand side of Eq. (2.4), $\mathbb{E}\big(\bar{x}|\mu_f(\bar{x}) = \mu\big)$, using scatter-plot smoothing with locally weighted regression. This is implemented through a two-step iteration procedure. To initialize the iterations, the first linear PCA line, f^0, is used as the initial guess. Then, in each iteration, first, the data points are orthogonally projected onto the current estimate of the curve, f^i, and their ordered arc lengths are calculated from the "first" projected point on the curve. Since the points are joined by line segments in this work, this is equivalent to a cumulative sum of successive Euclidean distances. The "first" data point is taken as the data point with the lowest value of $\bar{x}_1$ (first dimension of the data set) after projection onto f^i. In the second step, the new curve, $f^{i+1} = \mathbb{E}\big(\bar{x}|\mu_{f^i}(\bar{x}) = \mu\big)$, is estimated by using locally weighted regression [128] on the data set where the neighborhood of each point is based on the ordering from the first step. The number of points used for smoothing is based on the "span" parameter, $0 < s \leq 1$, representing the percentage of points to be used. Specifically, for r data points, sr rounded to the nearest integer would be the number of points used for smoothing. The weights are assigned to the neighboring points using the tri-cube weight function,

$$w_{ij}(\mu) = \begin{cases} \left(1 - \left|\frac{\mu_j - \mu_i}{d_i}\right|^3\right)^3 & \text{if } |\mu_j - \mu_i| < d_i \\ 0 & \text{if } |\mu_j - \mu_i| \geq d_i \end{cases} \tag{2.5}$$

where w_{ij} is the weight assigned to point j when computing the new point i with $i, j = 1, \ldots, r$, and d_i is the distance from point i to the furthest neighboring point considered. For further details regarding locally weighted regression, the reader is referred to Ref. [128]. This second step yields the new curve, f^{i+1}. The two steps described are then repeated with the new curve in the next iteration until a stopping criterion is met. The criterion used in this work is the relative change in the Euclidean distance from a point $\bar{x}$ to its projection on the principal curve, f. Specifically, if ε is a small, positive number, we iterate until

$$\left| \frac{\left\|\bar{x} - f^i(\mu_f)\right\| - \left\|\bar{x} - f^{i+1}(\mu_f)\right\|}{\left\|\bar{x} - f^i(\mu_f)\right\|} \right| < \varepsilon$$

The principal curve, however, is not a model and cannot generate an output from a new input sample. Therefore, the principal curve is approximated using a feedforward neural network, which is then used as the model to query new data.

2.2.2.2 Neural network

A two-layer feedforward neural network model that approximates a nonlinear function of the form, $y = f(x)$, can be written in the general form,

$$y = \sigma^{(3)}(W^{(3)}h^{(2)} + b^{(3)}) \tag{2.6}$$

$$h^{(2)} = \sigma^{(2)}(W^{(2)}h^{(1)} + b^{(2)}) \tag{2.7}$$

$$h^{(1)} = \sigma^{(1)}(W^{(1)}x + b^{(1)}) \tag{2.8}$$

where $x \in \mathbb{R}^n$ is the NN input vector, $y \in \mathbb{R}^p$ is the NN output vector. Each layer has a bias vector $b^{(i)}$ with $i = 1, 2, 3$, and an activation function $\sigma^{(i)}$ with $i = 1, 2, 3$, which is a nonlinear activation function such as the sigmoid function, $\sigma(x) = 1/(1 + \mathrm{e}^{-x})$. The hidden layers have output vectors $h^{(i)}$ with $i = 1, 2$. Every pair of units in two consecutive layers has an associated weight, which is stored in the weight matrix $W^{(i)}$.

2.2.3 Sparse identification

Sparse identification is a technique developed to identify dynamical systems made possible due to recent advances in sparsity algorithms. Sparsity methods have been applied to dynamic system modeling in recent works [71–76, 129, 130]. Sparse identification attempts to reconstruct the original ODE using only measured data from the system and identifies dynamical systems expressed in the form given in Eq. (2.1).

Sparse identification takes advantage of the fact that $f(\bar{x})$ in Eq. (2.1) typically contains very few nonzero terms, which makes it sparse in a higher-dimensional space of many candidate nonlinear functions. This sparsity allows calculation of the nonzero terms using scalable convex methods while avoiding a prohibitively expensive brute-force search.

In sparse identification, time-series data of the state x is first collected by sampling the process with sampling period, Δ, over a range of initial and operating conditions. The concatenated data matrix, X, is then, of the form,

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix} \tag{2.9}$$

where each x_i is a column of time-series data for state i for $i = 1, \ldots, m$. The matrix of the derivative of X,

$$\dot{X} = \begin{bmatrix} \dot{x}_1 & \dot{x}_2 & \cdots & \dot{x}_m \end{bmatrix} \tag{2.10}$$

is estimated in this work using second-order central finite differences (except the first and last points, which use second-order forward and backward finite differences, respectively). Next, a library, $\Theta(X)$, of q nonlinear functions of the columns of X is created. These functions are candidate terms for f, the right-hand side of Eq. (2.1). The objective is to find which of these terms are active by leveraging sparsity. An example of an augmented library or $\Theta(X)$ is

$$\Theta(X) = \begin{bmatrix} | & | & | & | & | & | \\ \mathbf{1} & X & X^{P_2} & \cdots & \sin X & \tanh X \\ | & | & | & | & | & | \end{bmatrix} \tag{2.11}$$

where, for example, X^{P_2} denotes all quadratic nonlinearities, given by

$$X^{P_2} = \begin{bmatrix} x_1^2 & x_1 x_2 & \cdots & x_2^2 & x_2 x_3 & \cdots & x_n^2 \end{bmatrix} \tag{2.12}$$

The sparse identification problem is to determine the q coefficients associated with the q candidate nonlinear functions considered in $\Theta(X)$ for each of the m states. The m coefficient vectors, each denoted by ξ, can be compactly written as a matrix,

$$\Xi = \begin{bmatrix} \xi_1 & \xi_2 & \cdots & \xi_m \end{bmatrix} \tag{2.13}$$

Each $\xi_i \in \mathbb{R}^q$ is a sparse vector of coefficients indicating the active terms in the dynamical equation of the corresponding row, $\dot{x}_i = f_i(x)$. The equation that must now be solved to determine Ξ can be set up as

$$\dot{X} = \Theta(X)\Xi \tag{2.14}$$

The above equation is solved using least-squares after zeroing all coefficients in Ξ that are smaller than a threshold, λ, known as the sparsification knob since this single parameter controls the sparsity of Ξ. This is repeated until the non-zero coefficients converge, which is very rapid in practice.

As expected, the effectiveness of sparse identification is intrinsically related to the candidate nonlinear functions used to compute the sparse dynamics. However, as discussed in Ref. [77], polynomials and trigonometric functions are a natural basis for many systems and may be used as an initial bank of basis functions. More nonlinear functions may be added to the bank based on physical knowledge of the specific system such as exponential terms for systems with chemical reactions.

2.3 NLPCA-SI implementation

In this work, the two methods described—NLPCA and sparse identification—are used together to identify the original two-time-scale system dynamics. The objective is to use NLPCA for deriving nonlinear relationships between the slow and fast dynamics and use sparse identification to reconstruct the slow dynamics. Thus, the final system is in the form of ODEs for the slow states and a neural network for the fast states. The method is abbreviated as NLPCA-SI.

The NLPCA-SI procedure is as follows: 1) a large data set is generated through open-loop simulations of the original nonlinear system in Eq. (2.1) using different initial conditions in a reasonable operating region, 2) a time-series plot of the states is used to separate the slow and fast states, 3) the collected data is then auto-scaled by subtracting the mean and dividing by the standard deviation since NNs have activation functions with relatively small ranges, (often 0 to 1), making them inaccurate or difficult to train if the data varies too widely, 4) the principal curve is computed using the entire m-dimensional data set, 5) and a feedforward NN as described in Section 2.2.2.2 is built using only the slow states as the inputs and the fast states as the outputs. Two remarks are made regarding the procedure. In the second step, another, possibly more robust, method to identify the slow and fast states *a priori* is to plot the time-series gradients of the data. For the fast states, due to the stiffness, the magnitudes of the gradients are extremely large initially. For every simulation of the ODE systems for data generation, the initial gradients of the fast states were at least one order of magnitude greater than that of the slow states. In most runs, the differences were two orders of magnitudes. Therefore, the ratio of initial gradients of the states can be used as a criterion to identify the slow and fast states *a priori*. This idea can be further developed as well. For example, norms of the ratios at multiple time instances can be used, where several orders of magnitudes separate the slow and fast states. In the final step of building the NN, an open-source platform for machine learning, Tensorflow, is used to solve the required optimization problems and obtain the optimal weight matrices, $W^{(i)}$, which minimize the loss function used: the mean-squared error between the predicted and actual outputs.

The NN used in the NLPCA is a two-hidden-layer network with a linear output layer. Ref. [121] used an overall five-layer network with the principal scores (defined as the arc length, μ, along the principal curve) as a middle layer, based on the initial structure proposed by Ref. [120]. However, it treated the network as two three-layer networks, with the third layer of the first network being the first layer of the second network. This was done because five-layer networks were considered difficult to train. However, with the recent advances in data science and machine learning, this is not intractable anymore. Hence, a four or even five-layer network may be trained without splitting it into two networks. However, the number of hidden layers is not arbitrary. Ref. [60, 61] proved that at least one sigmoidal hidden layer is required for a feed-forward neural network to have the universal approximation property. However, this is in theory and the parameters may be extremely specific to achieve the required accuracy. In addition, for nonlinear control systems, two layers may be required for closed-loop stabilization [131]. Finally, the function to be approximated by the NN may have regions where it behaves differently from its behavior in most of its domain. This can lead to the failure of a one-hidden-layer network and might even require a separate NN for distinct regions. Due to these considerations, two hidden layers are used in this NN with the number of hidden neurons in each hidden layer being determined by a grid search and cross-validation as in Ref. [121]. However, it is noted that these two parameters—the number of hidden layers and the number of neurons in each hidden layer—which define the structure of the NN should be based on the complexity of the nonlinear relationship between the inputs and outputs of the NN. Typically, as seen in the examples considered in this work, a shallow neural network (*i.e.*, an NN with only one or two hidden layers) is sufficient for most chemical process modeling problems. However, a deep neural network can be used for a large-scale, complex system with a large number of process states when a shallow neural network cannot achieve the desired model accuracy. In practical implementation, to obtain a well-conditioned neural network model with sufficiently high model accuracy on both training and testing data sets, we will start with a one-hidden-layer neural network, and gradually increase the number of layers and of neurons until no further improvement is noticed. Following these steps, a well-conditioned NN model is obtained to predict the fast states from the slow states. Once the optimal structure of the NN is determined, the NN is trained 10 times and the most accurate model is chosen based on the test set accuracy. The structure of the NN used in NLPCA-SI is shown in Fig. 2.1.

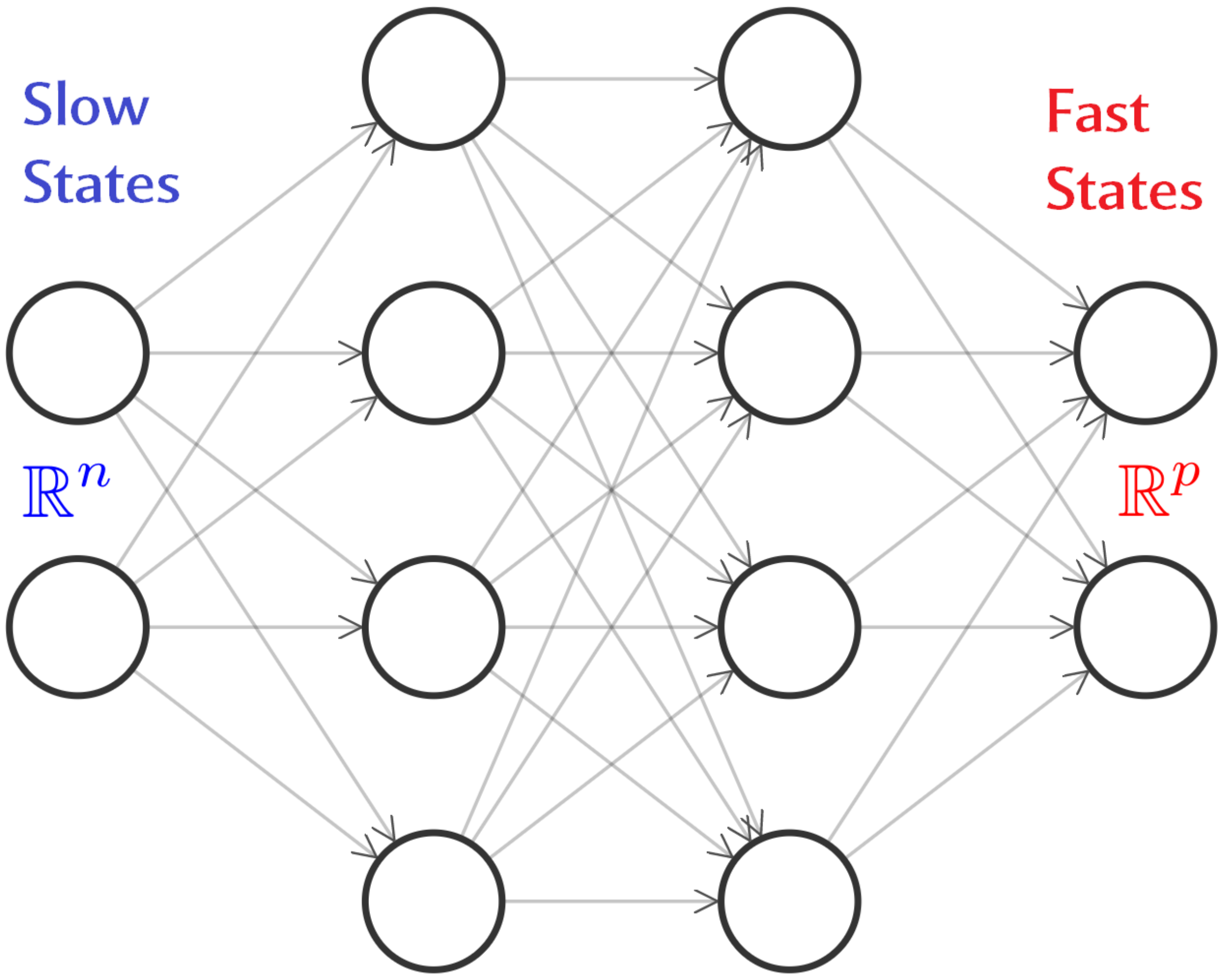


Figure 2.1: Structure of the neural network used for NLPCA-SI

For the activation functions of the two hidden layers, the first hidden layer uses a sigmoidal activation since, as mentioned, this is required for the universal approximation theorem to hold. For the second activation function, several common activation functions are tested, and the model with the minimal test set error is selected. The most important metric used in the NN training, however, was the learning rate, which typically requires significant tuning. In this method, the learning rate can significantly affect the results for some systems and is tuned until satisfactory performance is observed.

Finally, sparse identification is used on the data containing only the slow states. This identifies the slow dynamics in the form of ODEs containing only the slow states in the vector function f in Eq. (2.1). For predicting the states of the original system, new initial conditions are first used to integrate the sparse identified ODEs in time to yield the slow manifold. It is then used as input for the NN built during NLPCA to predict the fast states, neglecting a short transient period. In this way, the full state of the original system is recovered for any initial condition without integrating the fast dynamics.

The alternative is to simply reconstruct the entire system using only sparse identification and integrate it to predict the full state of the system at any time. The results from the NLPCA-SI method are compared with the results from this aforementioned brute-force approach using only sparse identification, abbreviated SI.

Remark 2.2. *It is important to note that, as opposed to classical singular perturbation modeling of two-time-scale processes where the dimension of the slow state vector and of the fast state vector are fixed once the singular perturbation parameter is specified and the process model is written in the standard singularly perturbed form, the proposed approach uses process data to determine the number of fast and slow states and, therefore, leads to a more accurate representation of the dimensions of the slow and fast states. Furthermore, it directly accounts for "hidden" time-scale multiplicity in the right hand-side of the process dynamic model owing to the presence of large or* *small parameters there and yields a slow subsystem of the lowest order as a result of the application of NLPCA to the overall process state data set. Therefore, this approach can be viewed as a way of carrying out index reduction in high-order (higher than 1) differential-algebraic equation systems (which arise as slow subsystems of two-time-scale systems) to yield a slow subsystem of the lowest order.*

2.4 Application of NLPCA-SI

The method explained in Section 2.3 is applied to two reactor systems. In the first example, a two-time-scale system due to multiple reactions with different rate constants is studied. Next, a CSTR with a single reaction but fast temperature dynamics is considered. In this section, a "proper" solution refers to a sparse identified system that can be integrated until the end of the time span without early termination due to divergence to infinity during numerical integration.

2.4.1 Example 1: Isothermal batch reactor with multiple reactions

An isothermal, constant-volume batch reactor with the following reaction scheme is considered:

$$\mathrm{A} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \mathrm{B} \xrightarrow{k_2} \mathrm{C} \tag{2.15}$$

where k_i is the rate constant associated with the corresponding reaction. The forward reaction in A $\rightleftharpoons$ B is much faster than both the backward reaction and the B $\longrightarrow$ C reaction such that $k_1 \gg k_{-1} > k_2$. This causes a two-time-scale separation in this system since species A is consumed much faster than the rate of consumption of species B or production of species C.

If the reactor has volume, V, and the concentration of species i is denoted by C_i, the material balances that govern the dynamical behavior of the batch reactor take the following form:

$$\dot{C}_{\mathrm{A}} = -k_1 C_{\mathrm{A}} + k_{-1} C_{\mathrm{B}} \tag{2.16}$$

$$\dot{C}_{\mathrm{B}} = k_1 C_{\mathrm{A}} - k_{-1} C_{\mathrm{B}} - k_2 C_{\mathrm{B}} \tag{2.17}$$

$$\dot{C}_{\mathrm{C}} = k_2 C_{\mathrm{B}} \tag{2.18}$$

$$C_{\mathrm{A}}(0) + C_{\mathrm{B}}(0) = \text{constant} \tag{2.19}$$

$$C_{\mathrm{C}}(0) = 0 \tag{2.20}$$

The initial concentration of species C is assumed to be zero, and Eq. (2.19) ensures that the final, steady-state concentration of species C is constant for each initial condition chosen. Therefore, only one species' initial concentration can be arbitrarily chosen. Note that an explicit ϵ is not necessary in the left-hand side of the fast ODE as seen in this example. Therefore, for this system, standard singularly perturbed methods for analytically separating the slow and fast states would not be adequate.

The rate constants in this system are chosen to be $k_1 = 5.0\,\mathrm{h}^{-1}, k_{-1} = 1.0\,\mathrm{h}^{-1}, k_2 = 0.5\,\mathrm{h}^{-1}$. The constant in Eq. (2.19), which is the sum of initial concentrations of species A and B, is $9.0\,\mathrm{mol\,m}^{-3}$. Ten initial conditions are chosen with $C_{\mathrm{A}}(0)$ increasing from $0\,\mathrm{mol\,m}^{-3}$ to $9.0\,\mathrm{mol\,m}^{-3}$ in increments of $1.0\,\mathrm{mol\,m}^{-3}$, while $C_{\mathrm{B}}(0)$ decreases to satisfy Eq. (2.19). Using each of these initial conditions, the ODE system given by Eqs. (2.16) to (2.18) is numerically integrated from $0.0\,\mathrm{hr}$ to $10.0\,\mathrm{hr}$ with a step size of 10^{-6} hr. The resulting data is sampled every $0.1\,\mathrm{hr}$ and concatenated from the 10 runs into the X data matrix, which is used for NLPCA.

In NLPCA, a span of 0.25 is used, meaning 25% of the data points are used in the smoothing step of each iteration when computing the principal curve. This value yielded the optimal principal curve. The NN built to approximate this curve used a sigmoidal activation function for the first hidden layer and the rectified linear unit (ReLU) for the second as this was sufficient to accurately capture the curve. For both hidden layers, by testing 1 to 15 neurons using the simplified cross-validation scheme, 13 neurons were found to be optimal. With these values of the hyper-parameters, the learning rate did not significantly affect the results and 0.01 was used.

For sparse identification, the augmented library of functions consisted of the constant/bias term (column of 1), polynomial terms up to fourth order, $\sin(X)$, $\cos(X)$, $\tan(X)$, $\tanh(X)$, $\exp(X)$, and $\exp(-X)$. The exponential terms are added since the reaction terms in reactor mass and energy balances often contain such terms. The optimal λ for the sparse identification algorithm was found to be 0.5 for the NLPCA-SI and 0.05 for SI.

Figure 2.2 shows the results for this system. The reconstructed equations from the NLPCA-SI and SI are not included due to their length. Instead, for brevity, only the reconstructed states are plotted and compared. Both NLPCA-SI and SI reproduce the dynamics very accurately. As seen earlier, the NLPCA-SI does not model the transient regime well.

The differences between NLPCA-SI and only SI becomes apparent when the value of the sparsification knob, λ, in sparse identification and the sampling period in the data generation step, Δ, are investigated.

The sparsification knob, λ, is tuned through extensive simulations to achieve the desired balance between sparsity and approximation accuracy for nonlinear dynamic systems—a larger λ leads to more sparse but less accurate dynamics. However, the range of values of λ that yielded

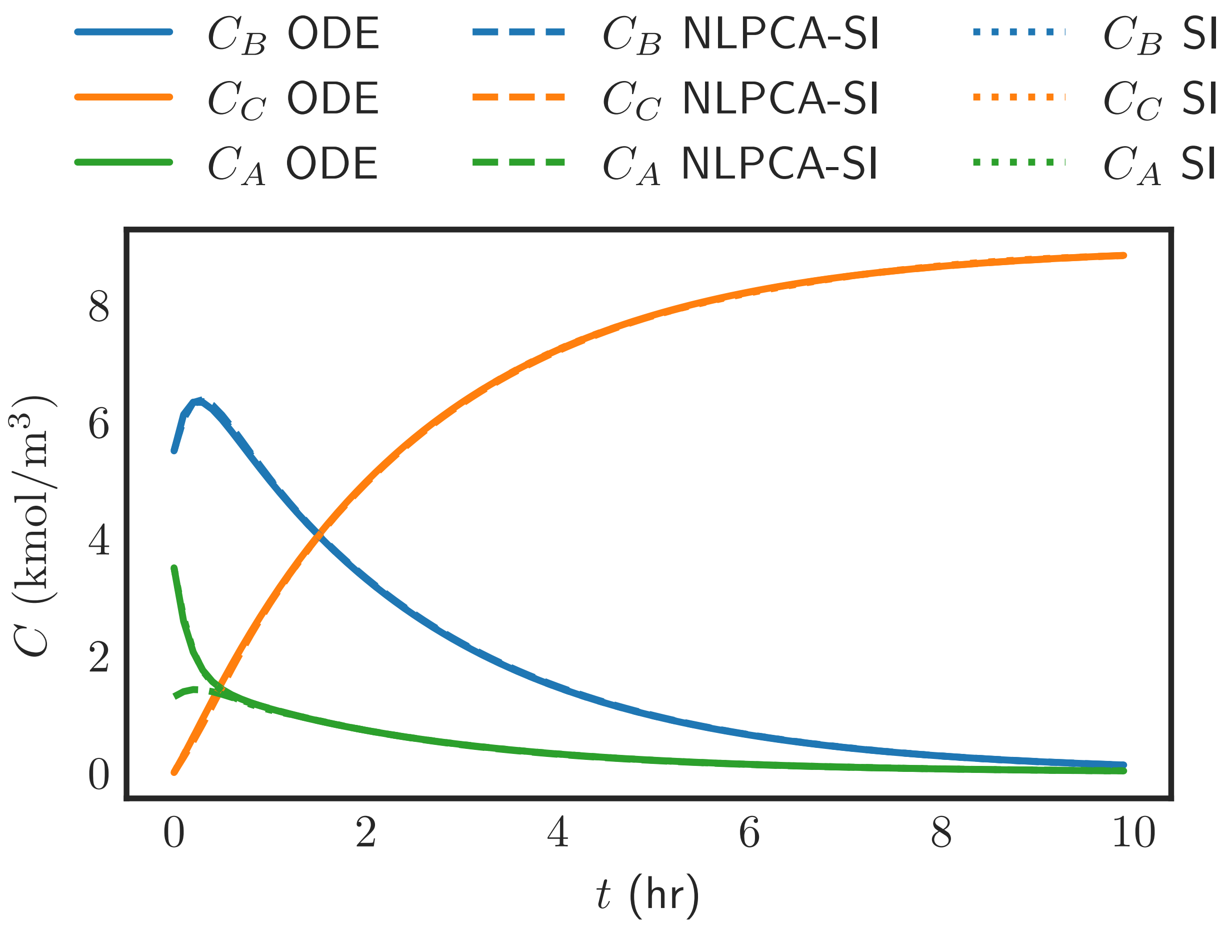


Figure 2.2: Comparison of original data (solid lines) with results from NLPCA-SI (dashed lines) and only sparse identification (dotted lines) for Example 1. Slow states: C_{B} (blue lines), and C_{C} (orange lines). Fast state: C_{A} (green lines).

the optimal results shown in Fig. 2.2 were not similar in the two cases. For the NLPCA-SI, any value of λ below the optimal 0.5 produced identical trajectories for the slow states, C_{B} and C_{C}. This was tested until $\lambda = 10^{-6}$. However, for SI, the value of λ that produced the ideal curves for all three states was only 0.05. Several values did not yield a proper solution, while most of them resulted in significantly inferior performance than shown in Fig. 2.2. It could not be determined that the optimal solution has been found without extensive testing of a large number of values for λ, as there was no range of values that consistently produced optimal or near-optimal solutions.

For the sampling period, Δ, as expected, reducing the sampling period generally yields significantly more accurate sparse identified models due to the greater number of data points and more accurate gradient calculations from more closely spaced data points. However, it might not be practically feasible to sample data at such short intervals due to, for example, limitations of the measurement device, such as for concentration. Furthermore, this can cause chattering behavior and numerical instabilities in the gradient estimation for the fast states, especially if noise is present. If the sampling period is not small enough, and the fast dynamics are extremely fast (ϵ in Eq. 2 is extremely small), it is possible that the fast transient takes place in a time shorter than the sampling period. In this case, the sparse identification algorithm cannot, in general, capture the fast dynamics and may predict a wrong steady-state value entirely. Therefore, the sampling period, Δ, is also carefully chosen based on extensive simulations in this work. Conversely, since NLPCA-SI only uses the slow manifold to predict the fast states, it is affected by neither the sampling period nor the ϵ. However, the key advantage of NLPCA-SI over SI, as seen clearly in this example, is that NLPCA-SI predicts the fast state at least as accurately as SI (after the short transient) without requiring any integration of stiff ODEs with extremely small time steps.

2.4.2 Example 2: Non-isothermal CSTR with jacket

A single, endothermic, irreversible reaction of the form,

$$\mathrm{A} \xrightarrow{k} \mathrm{B} \tag{2.21}$$

takes place in a perfectly mixed non-isothermal CSTR as shown in Fig. 2.3.

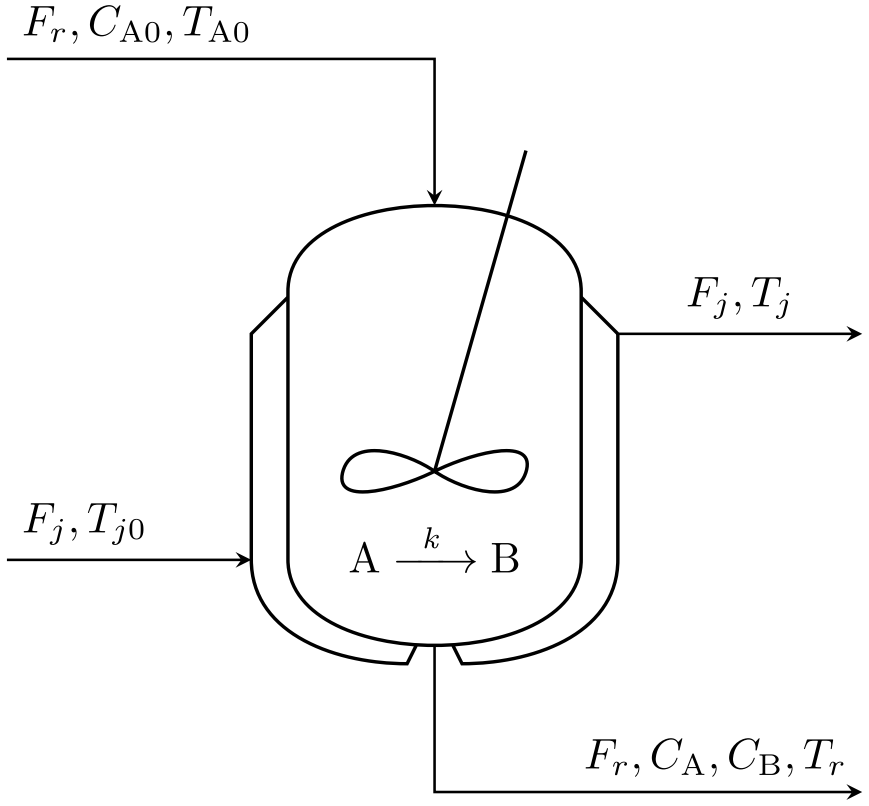


Figure 2.3: A continuous-stirred tank reactor with jacket.

In this process, C_A is the concentration of reactant A in the reactor, V_r is the volume of the reacting liquid in the reactor (assuming the vessel has constant holdup), and T_r is the temperature of the reactor. The inlet stream contains pure species A at a volumetric flow rate F, concentration C_{A0}, and temperature T_{A0}. A heating jacket of volume, V_j, heats the reactor. The heat transfer fluid is added to the jacket at a volumetric flow rate F_j and an inlet temperature T_{j0}. The reacting liquid has a constant density of ρ_m and a heat capacity of $c_{p,m}$, while the heat transfer fluid has a constant density of ρ_j and a heat capacity of $c_{p,j}$. The enthalpy of the reaction is ΔH_r. The heat transfer coefficient is U, and the area of heat transfer between the jacket and the reactor is A_r. The rate of the reaction in Eq. (2.21) is assumed to be of the form,

$$r = -k_0 \exp\left(\frac{-E}{RT_r}\right) C_{\mathrm{A}} \tag{2.22}$$

where k_0, R, and E represent the pre-exponential constant, ideal gas constant, and activation energy, respectively. Under these assumptions, a material balance for species A, a reactor energy balance, and a jacket energy balance can be formulated, respectively, as:

$$\begin{aligned} V_r\dot{C}_{\mathrm{A}} &= F_r\left(C_{\mathrm{A0}} - C_{\mathrm{A}}\right) - k_0 \mathrm{e}^{-E/RT_r} C_{\mathrm{A}} V_r && (2.23)\\ V_r\dot{T}_r &= F_r\left(T_{\mathrm{A0}} - T_r\right) + \frac{(-\Delta H_r)}{\rho_m c_{p,m}} k_0 \mathrm{e}^{-E/RT_r} C_{\mathrm{A}} V_r \\ &\quad + \frac{UA_r}{\rho_m c_{p,m}}\left(T_j - T_r\right) && (2.24)\\ V_j\dot{T}_j &= F_j T_{j0} - F_j T_j - \frac{UA_r}{\rho_j c_{p,j}}\left(T_j - T_r\right) && (2.25) \end{aligned}$$

If ϵ is defined as the ratio of the jacket volume to the reactor volume ($\epsilon = V_j/V_r$), Eq. (2.25) can be rewritten explicitly in ϵ and V_r only without a V_j term as follows:

$$\epsilon\dot{T}_j = \frac{1}{V_r}\left(F_j T_{j0} - F_j T_j - \frac{UA_r}{\rho_j c_{p,j}}\left(T_j - T_r\right)\right)$$

However, the right-hand sides of the equations are also relevant in determining the slow-fast dynamics of a process. Hence, writing the system in terms of ϵ is not crucial in our approach and can also be misleading. In this system both temperatures have fast dynamics in comparison to the concentration, which has much slower dynamics. In standard singularly perturbed methods, only T_j would be considered the fast state, while, in fact, T_r is also a fast state in this system.

The parameter values used for this system are given in Table 2.1. Ten initial conditions are chosen with $C_\text{A}(0)$ increasing from $0\,\text{mol}\,\text{m}^{-3}$ to $9.0\,\text{mol}\,\text{m}^{-3}$ in increments of $1.0\,\text{mol}\,\text{m}^{-3}$, $T_r(0)$ increasing from $280\,\text{K}$ to $370\,\text{K}$ in increments of $10\,\text{K}$, and $T_j(0)$ increasing from $300\,\text{K}$ to $390\,\text{K}$ in increments of $10\,\text{K}$. Using each initial condition, the ODE system given by Eqs. (2.23) to (2.25) is numerically integrated from $0.0\,\text{hr}$ to $1.0\,\text{hr}$ with a step size of 10^{-6} hr. The resulting data is sampled every $0.005\,\text{hr}$ and concatenated from the 10 runs into the data matrix, X. A time-series plot of X and/or gradient of X shows that only C_A is the slow state in this system, while both temperatures are fast. The NLPCA is carried out accordingly with C_A being the input to the NN and T_r and T_j being the outputs of the NN.

Table 2.1: Parameter values for Example 2

$V_r = 1.0\,\text{m}^3$	$k_0 = 3.36 \times 10^6\,\text{h}^{-1}$
$V_j = 0.08\,\text{m}^3$	$E = 8.0 \times 10^3\,\text{kcal}\,\text{kg}^{-1}$
$A_r = 6.0\,\text{m}^3$	$T_{\text{A}0} = 310.0\,\text{K}$
$U = 1000.0\,\text{kcal}\,\text{h}^{-1}\,\text{m}^{-2}\,\text{K}^{-1}$	$T_{j0} = 357.5\,\text{K}$
$R = 1.987\,\text{kcal}\,\text{kmol}^{-1}\,\text{K}^{-1}$	$\rho_m = 900.0\,\text{kg}\,\text{m}^{-3}$
$C_{\text{A}0} = 3.75\,\text{kmol}\,\text{m}^{-3}$	$\rho_j = 800.0\,\text{kg}\,\text{m}^{-3}$
$c_{p,m} = 0.231\,\text{kcal}\,\text{kg}^{-1}\,\text{K}^{-1}$	$F_r = 3.0\,\text{m}^3\,\text{h}^{-1}$
$c_{p,j} = 0.200\,\text{kcal}\,\text{kg}^{-1}\,\text{K}^{-1}$	$F_j = 20.0\,\text{m}^3\,\text{h}^{-1}$
$\Delta H_r = 5.4 \times 10^4\,\text{kcal}\,\text{mol}^{-1}$	

The optimal span for NLPCA is 0.2 *i.e.* 20% of the data points are used for the smoothing step when calculating the principal curve. The activation function for the second hidden layer in the NN required tanh in this example. For the first hidden layer, only 5 neurons are required, while 13 is optimal for the second hidden layer. With these values of the hyper-parameters, the learning rate does not significantly affect the results and 0.01 is used.

For sparse identification, the augmented library of functions consisted of the constant term, polynomials up to third order, $\sin(X)$, $\cos(X)$, $\tan(X)$, and $\tanh(X)$. The exponential terms are omitted in this example because, if they are included, while NLPCA-SI results improve, SI does not yield a proper solution. The optimal λ for the sparse identification algorithm was found to be 2.0 for the NLPCA-SI and 1.0 for SI. The sampling period in this example was chosen to be $0.005\,\mathrm{hr}$ because any value larger than this caused SI to not produce a proper solution.

Figure 2.4 shows the results for this system. The reconstructed equations are omitted for brevity. The prediction of the slow state, C_{A}, is similar across both methods. The differences in Fig. 2.4(a) appear more significant due to the scale of the figure. The fast states, both the reactor and jacket temperatures, are reconstructed very accurately in the post-transient regime using both methods.

Table 2.2: Neural network hyperparameters used for NLPCA in example systems of Section 2.4

Neural Network	Learning	First Hidden Layer		Second Hidden Layer	
in example	rate	Neurons	Activation	Neurons	Activation
1	0.01	13	Sigmoid	13	ReLU
2	0.01	5	Sigmoid	13	tanh

Remark 2.3. *It is noted that the computational time needed to solve the original two-time-scale ODEs and the reduced-order slow model obtained with the proposed NLPCA-SI approach depends on the numerical integration method and time step of integration adopted. However, the lack of stiffness of the reduced-order slow model allows the use of larger time-steps leading to a reduced solution time for the calculation of the system solutions at the expense of missing the initial sharp transient of the fast states of the two-time-scale ODE system. Therefore, a fair comparison cannot be made when using different integration time steps for the different ODE systems identified, while*

the advantage of the NLPCA-SI method is nullified if an identical, very small integration time step is used for both systems.

2.5 Conclusion

In this work, reduced-order models were developed for nonlinear two-time-scale systems using measurement data. Nonlinear principal component analysis, a technique to extract nonlinear relationships between variables, was combined with sparse identification, an algorithm to reconstruct the underlying ODEs governing the system, to build a nonlinear model that can be used to predict the full state of the original system. The effectiveness of this method was demonstrated using two reactor-based examples and comparing the results with the original system and a purely sparse identified system. In both examples, the results were in close agreement with the original system and also the purely sparse identified system. It was observed, however, that the accuracy of the sparse identification method for the full state was strongly linked to the sampling period at which the original data was sampled and also, in some cases, required a very specific degree of sparsity in the algorithm. Furthermore, the sparse identified stiff ODEs required a very short integration time step of 10^{-6}, while the NN model in NLPCA-SI could predict the fast state (after a short transient) to at least a similar degree of accuracy without any integration. In the future, we will focus on the designs of controllers for nonlinear two-time-scale systems using data-based reduced-order models. The predicted states using NLPCA-SI can be used as the process model in model-based control methods such as MPC.

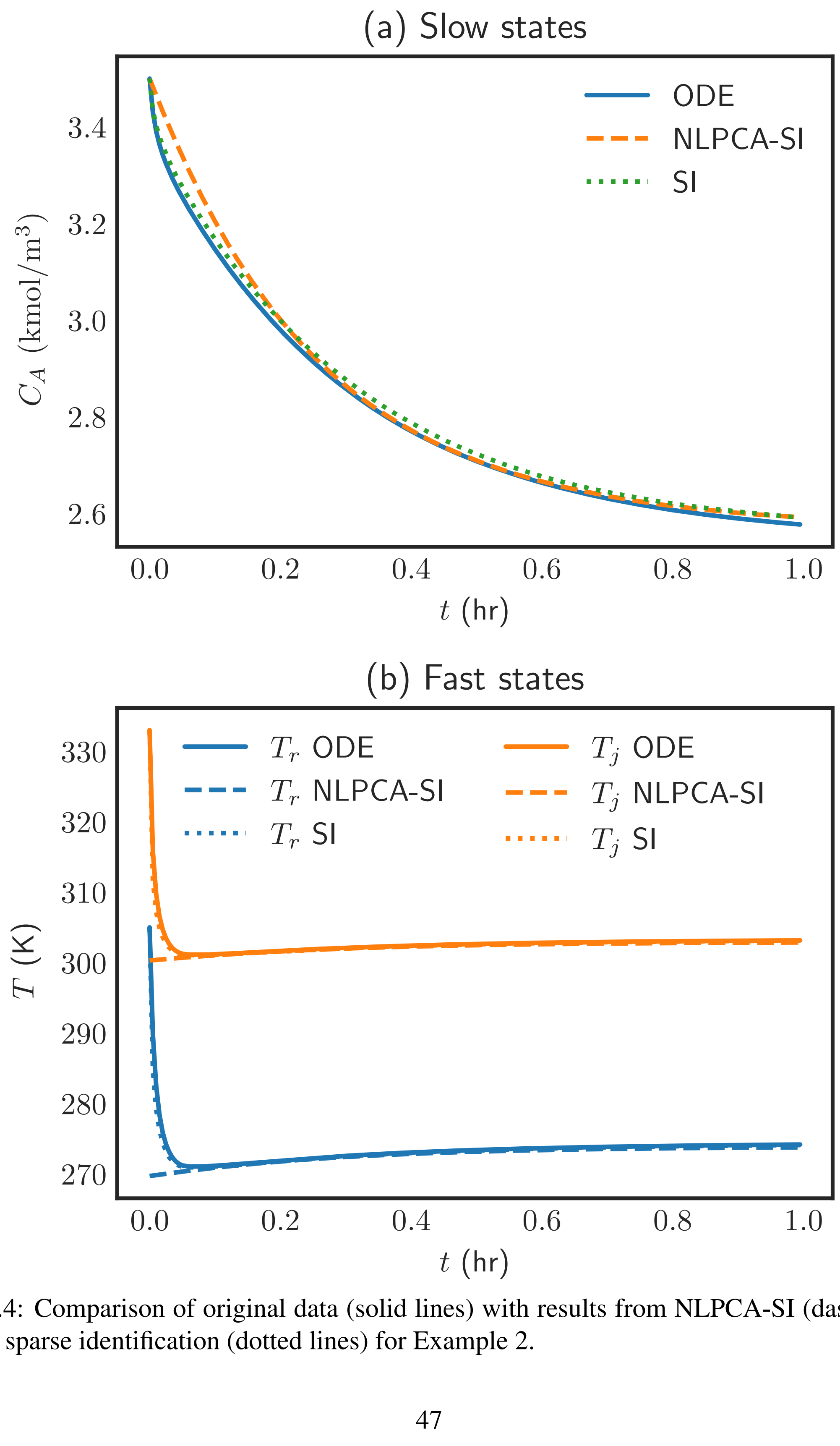


Figure 2.4: Comparison of original data (solid lines) with results from NLPCA-SI (dashed lines) and only sparse identification (dotted lines) for Example 2.

Chapter 3

Sparse-identification-based predictive control of nonlinear multiple time-scale processes

3.1 Introduction

Chemical processes such as fluidized catalytic crackers, biochemical reactors and distillation columns, often involve physico-chemical phenomena occurring in vastly-different time scales. In chemical process modeling, time-scale separation has been handled via singular perturbation techniques [e.g., 132] in the context of first-principles modeling with the goal of building well-conditioned ordinary differential equation (ODE) models that can be used for process analysis and controller design. More recently, in [48], a constraint was incorporated in the optimization problem of fitting a polynomial nonlinear state-space model to nonlinear process data to avoid identified model ill-conditioning (stiffness) in the construction of nonlinear ODE models. To control a two-time-scale system, a composite control system using multi-rate sampling was proposed in [133], where a “fast” feedback controller was designed to stabilize the fast dynamics subsystem, and a model predictive controller (MPC) was developed to stabilize the slow dynamic and optimize closed-loop performance. Additionally, in [134], a composite controller which consisted of a Lyapunov-based MPC for the fast subsystem, and a Lyapunov-based Economic MPC for the slow subsystem, was developed to operate the system in a time-varying fashion in order to improve process economics while achieving desired closed-loop stability properties. In all the above efforts, an explicit nonlinear process model, typically obtained from first-principles, was assumed to be available.

Model predictive control (MPC) is an optimization-based process control scheme that can optimize process performance and account for state and control input constraints. Since the MPC optimization problem relies on a process model for predicting the future state evolution, the performance of MPC relies heavily on its model quality. In recent years, machine learning techniques have been applied to model chemical processes from data when first-principle models are unavailable. For example, in [3, 62], recurrent neural networks were used to build the data-driven model for a general class of nonlinear processes, and were then incorporated in Lyapunov-based MPC to ensure desired closed-loop stability and performance. In [135], reduced-order data-driven models were developed for nonlinear two-time-scale systems using nonlinear principal component analysis and sparse identification methods. The simultaneous design and control of nonlinear dynamical systems via the use of robust tools based on Lyapunov theory has been investigated generally in [136, 137] and also under uncertainty in the context of MPC in [138].

In the field of nonlinear dynamical modeling, the technique of sparse identification developed in [77] has successfully been applied to a diverse array of nonlinear regression problems, including chaotic systems and nonlinear partial differential equations. However, its application to two-time-scale systems has only recently been investigated. In [98], the authors extended the sparse identification procedure to the discovery of two-time-scale systems when full or even partial measurements of the variables are available under the assumption of linear coupling across the time-scales. In particular, when full state measurements were available, it was shown that, although sparse identification could correctly identify uni-time-scale systems with as little data as 5% of a period, for two-time-scale systems, the data requirement increased linearly with the ratio of slow to fast dynamics under naive uniform sampling. This was due to the fast trajectories requiring a smaller sampling period. In fact, even in uni-time-scale systems, when the aforementioned 5% of data was sufficient to rebuild the system, the data had to be captured at the relatively fast segments of the period at a very high sampling rate, as the information contained in the faster regions was much more than the information captured in the slower regions of the period. An alternative to overcome the linear increase in data requirement with diverging time-scale-multiplicities was proposed and devised in [98] known as burst sampling, which allowed sparse identification to identify both the slow and fast subsystems with a significantly smaller fragment of the data over one period. However, since the present work aims to utilize sparse identification for the purpose of control, identifying the fast subsystem as a stiff differential equation via a more complicated composite multirate sampling is not advantageous. Instead, sparse identification is only used to identify the slow subsystem, which is incorporated into the controller. The application of sparse identification in closed-loop control is an area that has not been investigated in-depth yet.

Inspired by these results, in this work, we use time-series data from the slow process variables (subset of process state variables in some coordinate system) to construct well-conditioned data-driven models for a general class of two-time-scale nonlinear processes, and evaluate their performance in the context of MPC of two-time-scale processes. The rest of this article is organized as follows: in Section 3.2, the notations, and the class of nonlinear two-time-scale systems considered are given. In Section 3.3, the sparse identification (SI) model for the slow subsystem is introduced. In Section 3.4, the formulation of LMPC using SI models is presented, while the closed-loop singularly perturbed system stability analysis showing the boundedness of closed-loop states in a small neighborhood around the origin is given in Section 3.5. In Section 3.6, a chemical reactor example which exhibits two-time-scale behavior is simulated to demonstrate the effectiveness of the proposed modeling and control approaches.

3.2 Preliminaries

3.2.1 Notation

The notation $|\cdot|$ denotes the Euclidean norm of a vector. $x^\top$ is used to denote the transpose of x. The notation $L_fV(x)$ denotes the standard Lie derivative $L_fV(x) := \frac{\partial V(x)}{\partial x}f(x)$. Set subtraction is denoted by "$\backslash$", *i.e.*, $A\backslash B := \{x \in \mathbb{R}^n \mid x \in A, x \notin B\}$. The function $f(\cdot)$ is of class $\mathcal{C}^1$ if it is continuously differentiable in its domain. A continuous function $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and is zero only when evaluated at zero. A continuous function $\beta : [0, a) \times [0, \infty) \to [0, \infty)$ is said to belong to class $\mathcal{KL}$ if, for each fixed t, the function $\beta(\cdot, t)$ is of class $\mathcal{K}$, while, for each fixed s, the function $\beta(s, \cdot)$ is decreasing and tends to zero as $s \to \infty$.

3.2.2 Class of systems

The general class of two-time-scale continuous-time nonlinear systems with m states considered in this work has the following general form:

$$\dot{x} = f_1(x, z, u, \epsilon) \tag{3.1a}$$

$$\epsilon \dot{z} = f_2(x, z, \epsilon) \tag{3.1b}$$

where $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^p$ are the slow and fast state vectors, respectively, with $n+p = m$. $u \in \mathbb{R}^q$ is the manipulated input vector with constraints defined by $u \in U := \{u_i^{\min} \leq u_i \leq u_i^{\max}, i = 1, ..., q\} \subset \mathbb{R}^q$. ϵ is a small positive parameter representing the speed ratio of the slow to the fast dynamics of the system. The vector functions $f_1(x, z, u, \epsilon)$ and $f_2(x, z, \epsilon)$ are sufficiently smooth vector functions in $\mathbb{R}^n$ and $\mathbb{R}^p$, respectively. In the system of Eq. (3.1), after a short transient period, the fast states, z, converge to a slow manifold (provided such an isolated manifold exists) and can be expressed by a nonlinear algebraic expression in x, the slow states. Therefore, following the standard two-time scale decomposition procedure in [139], we can set $\epsilon = 0$ in Eq. (3.1) and obtain:

$$\dot{x} = f_1(x, z, u, 0) \tag{3.2a}$$

$$0 = f_2(x, z, 0) \tag{3.2b}$$

Remark 3.1. *The class of singularly perturbed system of Eq. 3.1 are presented for analysis purposes, and will not be used to derive models that will be used in the controllers. Furthermore,* *to simplify the development, we focus on two-time-scale processes with stable fast dynamic, and for this reason, we take $f_2(x, z, \epsilon)$ to be independent of u. However, our analysis can be readily adapted to deal with systems in which $f_2(\cdot)$ is a function of u.*

Assumption 3.1. *Equation* (3.2b) *possesses a unique root given by*

$$z_s = \hat{f}_2(x) \tag{3.3}$$

where z_s is a quasi-steady state for the fast state z, and $\hat{f}_2 : \mathbb{R}^n \to \mathbb{R}^p$ and its derivative are locally Lipschitz continuous.

Assumption 3.1 is standard in the singular perturbation framework as it ensures that the system has an isolated equilibrium manifold for the fast dynamics on which z can be written as an algebraic function of x. Substituting Eq. (3.3) into Eq. (3.2a) gives the reduced slow subsystem,

$$\dot{x} = f_1(x, \hat{f}_2(x), u, 0) \tag{3.4}$$

For the fast state, we define a fast time scale $\tau = t/\epsilon$ and a new coordinate $y := z - \hat{f}_2(x)$. Rewriting Eq. (3.1b) as a derivative with respect to τ rather than t and setting $\epsilon = 0$ yields the fast subsystem,

$$\frac{\mathrm{d}y}{\mathrm{d}\tau} = f_2(x, \hat{f}_2(x) + y, 0) \tag{3.5}$$

We assume the input appears linearly in Eq. (3.4) and, therefore, rewrite the slow subsystem of

Eq. (3.4) in the following form throughout the manuscript:

$$\dot{x} = F(x,u) := f(x) + g(x)u, \qquad x(t_0) = x_0 \tag{3.6}$$

where $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$ and $n \times q$, respectively. Without loss of generality, throughout the manuscript, the initial time t_0 is taken to be zero ($t_0 = 0$), and it is assumed that $f(0) = 0$, and thus, the origin is a steady-state of the nonlinear system of Eq. (3.6), *i.e.*, $(x_s^*, u_s^*) = (0,0)$, where x_s^* and u_s^* represent the steady-state slow state and input vectors, respectively.

Assumption 3.2. *The origin of the closed-loop fast subsystem of Eq.* (3.5) *is globally asymptotically stable, uniformly in x in the sense that there exists a function β_y of class $\mathcal{KL}$ such that for any $y(0) \in \mathbb{R}^p$,*

$$|y(t)| \leq \beta_y\left(|y(0)|, \frac{t}{\epsilon}\right) \qquad \forall\, t\ \geq 0 \tag{3.7}$$

3.2.3 Stabilizability assumption via control Lyapunov function

With respect to the stabilizability requirement for the slow dynamics, it is assumed that there exists a stabilizing control law $u = \Phi(x) \in U$ (*e.g.*, the universal Sontag control law [140]) such that the origin of the nominal slow subsystem of Eq. (3.6) is rendered asymptotically stable in an open neighborhood D around the origin in the sense that there exist a $\mathcal{C}^1$ control Lyapunov function, $V(x)$, and four functions, a_1, a_2, a_3, a_4 of class $\mathcal{K}$ such that $\forall\, x \in \mathbb{R}^n$:

$$a_1(|x|) \leq V(x) \leq a_2(|x|), \tag{3.8a}$$

$$\dot{V}(x) = \frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \leq -a_3(|x|), \tag{3.8b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq a_4(|x|) \tag{3.8c}$$

The Sontag law, a candidate controller for $\Phi(x)$, is given in the following form:

$$\varphi_i(x) = \begin{cases} -\dfrac{p + \sqrt{p^2 + q^4}}{q^\top q} q & \text{if} \quad q \neq 0 \\ 0 & \text{if} \quad q = 0 \end{cases} \tag{3.9a}$$

$$\Phi_i(x) = \begin{cases} u_i^{\min} & \text{if} \quad \varphi_i(x) < u_i^{\min} \\ \varphi_i(x) & \text{if} \quad u_i^{\min} \leq \varphi_i(x) \leq u_i^{\max} \\ u_i^{\max} & \text{if} \quad \varphi_i(x) > u_i^{\max} \end{cases} \tag{3.9b}$$

where p denotes $L_f V(x)$ and q denotes $L_{g_i} V(x)$, $f = [f_1 \cdots f_n]^\top$, $g_i = [g_{1i}, ..., g_{ni}]^\top$, $i = 1, 2, ..., q$. $\varphi_i(x)$ of Eq. (3.9a) represents the i_{th} component of the control law $\varphi(x)$. $\Phi_i(x)$ of Eq. (3.9) represents the i_{th} component of the saturated control law $\Phi(x)$ that accounts for the input constraint $u \in U$.

First, a region where the conditions of Eq. (3.8) are satisfied under the controller $u = \Phi(x) \in U$ as $\phi_u = \{x \in \mathbb{R}^n \mid \dot{V}(x) = L_f V + L_g V u < -kV(x), u = \Phi(x) \in U\} \cup \{0\}$, where $k > 0$, is characterized. Then the closed-loop stability region Ω_ρ for the nonlinear slow subsystem of Eq. (3.6) is defined as a level set of the Lyapunov function, which is inside ϕ_u: $\Omega_\rho := \{x \in \phi_u \mid V(x) \leq \rho\}$, where $\rho > 0$ and $\Omega_\rho \subset \phi_u$. Furthermore, the Lipschitz property of $F(x, u)$ combined with the bound on u implies that there exist positive constants M, L, L' such that the following inequalities hold $\forall\, x, x' \in D, \forall\, u \in U$:

$$|F(x,u)| \leq M \tag{3.10a}$$

$$|F(x,u) - F(x',u)| \leq L|x - x'| \tag{3.10b}$$

$$\left| \frac{\partial V(x)}{\partial x} F(x,u) - \frac{\partial V(x')}{\partial x} F(x',u) \right| \leq L'|x - x'| \tag{3.10c}$$

3.3 Sparse identification model for the slow subsystem

3.3.1 Sparse identification

Sparse identification is a nonlinear system identification method used to model dynamical systems. The application of sparsity methods to dynamic system modeling can be found in the recent literature [71–76, 129, 130]. Sparse identification is used to approximately reconstruct a continuous-time ODE of the form,

$$\dot{\hat{x}} = \hat{f}(\hat{x}) \tag{3.11}$$

using only numerical data from the system without requiring knowledge of the underlying physics of the process.

Sparse identification exploits the sparsity of the right-hand side of Eq. (3.11) since $\hat{f}(\hat{x})$ contains very few nonzero terms in practical systems, rendering it sparse in a higher-dimensional space of candidate nonlinear functions. The nonzero terms can then be calculated using scalable convex methods. To carry out sparse identification, the open-loop process is simulated over a wide range of initial conditions. From the resulting data, the slow state measurements are sampled with a sufficiently small sampling period and concatenated into a data matrix, X, of the form,

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \tag{3.12}$$

where each x_i is a column of time-series data for state i for $i = 1, \ldots, n$. The time-derivative of X, denoted by $\dot{X}$, is estimated in this work using second-order central finite differences since the noise-free case is considered and since the sampling period is sufficiently small. Subsequently, a function library, $\Theta(X)$, is created consisting of r nonlinear functions of the columns of X. The r functions represent possible terms for f, the right-hand side of Eq. (3.6). The goal of the sparse identification algorithm is to identify the active terms in this library by taking advantage of sparsity. The augmented library, $\Theta(X)$, considered in this work is of the form,

$$\Theta(X) = \begin{bmatrix} | & | & | & | & | & | & | & | \\ \mathbf{1} & X & X^{P_2} & X^{P_3} & \sin X & \cos X & \tan X & \tanh X \\ | & | & | & | & | & | & | & | \end{bmatrix} \tag{3.13}$$

where, for example, X^{P_2} denotes all quadratic nonlinearities, given by

$$X^{P_2} = \begin{bmatrix} x_1^2 & x_1x_2 & \cdots & x_2^2 & x_2x_3 & \cdots & x_n^2 \end{bmatrix} \tag{3.14}$$

The above choice of candidate nonlinear functions was motivated by the fact that polynomials and trigonometric functions form a basis for many practical systems as mentioned in [77].

Remark 3.2. *As this is a novel method, only the noise-free case is considered in this work. However, if the method is applied to noisy data, the time-derivative $\dot{X}$ cannot be computed using regular* *finite difference methods as such methods will amplify the noise present and not yield meaningful values. Instead, the derivative approximation will require the application of more advanced techniques such as the total-variation regularized derivative [141] or smoothed finite difference, where the noisy data is first smoothed out using a filter (such as the Savitzky–Golay filter) before calculating the finite differences [142].*

In sparse identification, for each of the n slow states, we determine the r coefficients that premultiply the r candidate nonlinear functions considered in the function library, $\Theta(X)$. Denoting each corresponding coefficient vector by ξ, the n coefficient vectors can be compactly written in matrix form as

$$\Xi = \begin{bmatrix} \xi_1 & \xi_2 & \cdots & \xi_n \end{bmatrix} \tag{3.15}$$

where each $\xi_i \in \mathbb{R}^r$ is a sparse column vector of coefficients identifying the nonzero terms in the dynamic model of the corresponding slow state, $\dot{x}_i = f_i(x)$. Therefore, to determine Ξ, we need to solve the following equation:

$$\dot{X} = \Theta(X)\Xi \tag{3.16}$$

In brief, Eq. (3.16) can be solved using a straightforward least-squares routine after neglecting and zeroing all coefficients in Ξ that are smaller than a threshold, λ. The least-squares problem associated with Eq. (3.16) can most generally be formulated as follows:

$$\Xi = \arg\min_{\Xi'} \left\| \dot{X} - \Theta(X)\Xi' \right\|_2 + \lambda \left\| \Xi' \right\|_1 \tag{3.17}$$

where Ξ' is a notational substitute for Ξ, and the second term enforces sparsity of Ξ. Practically, we first define the matrix Ξ'' to be the matrix Ξ' with all coefficients with magnitudes below λ set to zero, which is the practical implementation of the L_1 regularization term in Eq. (3.17). Subsequently, we solve the following least-squares problem

$$\Xi = \underset{\Xi''}{\arg\min} \left\| \dot{X} - \Theta(X)\Xi'' \right\|_2 \tag{3.18}$$

in each iteration using MATLAB's built-in linear solver called with $A \backslash b$ where $A = \dot{X}$ and $b = \Theta(X)$ until the big/nonzero coefficients (larger than λ in each iteration) converge.

To find the Pareto optimal value of the parameter λ that balances model complexity with accuracy, methods such as cross-validation from machine learning may be utilized [77]. In this work, a broad sweep of λ is first used to identify the order of magnitude above which the model has too few nonzero terms to capture the dynamics, resulting in large error values. Subsequently, a narrower sweep in the relevant order of magnitude is used to refine this value of λ. Further refinement was not found to be necessary as a wide range of values from the first refinement yielded identical and optimal models. Once Ξ is found, the overall model is written as the continuous-time differential equation,

$$\dot{x} = \Xi^\top (\Theta(x^\top))^\top$$

where $\Theta(x^\top)$ is not a data matrix but a column vector of symbolic functions of elements of x corresponding to the functions considered in the function library. It is noted that the sampling period of the discrete-time data used to carry out sparse identification affects the accuracy of the finite-difference estimate of the time-derivative, which significantly affects the accuracy of the continuous-time model identified by sparse identification.

3.3.2 Identifying a model for the slow subsystem

In this work, sparse identification (SI) is used to reconstruct the slow dynamic model for the slow states. The following slow dynamic model is developed to approximate the slow subsystem of Eq. (3.5).

$$\dot{\hat{x}} = F_{si}(\hat{x}, u) := \hat{f}(\hat{x}) + \hat{g}(\hat{x})u, \qquad \hat{x}(t_0) = x_0 \tag{3.19}$$

where $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$ and $n \times q$ that approximately capture the functions $f(\cdot)$ and $g(\cdot)$, respectively. Specifically, we first construct the following slow dynamic model using the data set generated with various initial states and $u = 0$ (*i.e.*, steady-state input value):

$$\dot{\hat{x}} = \hat{f}(\hat{x}) \tag{3.11}$$

Subsequently, we use the data set generated with various inputs $u \in U$ and initial states to approximate the function $g(x)$ associated with the input u in the right-hand side of Eq. (3.6) as $\hat{g}(\hat{x}) = \frac{\dot{\hat{x}} - \hat{f}(\hat{x})}{u}$ where $u \neq 0$. Finally, the model performance is evaluated using unseen data in the testing data set, and is shown to achieve a good representation of $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ with a desired accuracy.

Remark 3.3. *An alternative method to find $\hat{g}(\cdot)$ is to use a more advanced sparse identification algorithm that expands the function library $\Theta(X)$ to include input terms (i.e., $\Theta(X, u)$) [143]. However, due to the presence of feedback control in this work, this method would require further development before implementation, and the method described above was considered sufficiently* *accurate.*

3.4 Lyapunov-based MPC using SI Models

This section proposes the design of a Lyapunov-based MPC (LMPC) that incorporates the SI model to predict future slow states, followed by a stability analysis of the closed-loop system of Eq. (3.6) in the succeeding section. Specifically, the stability of the nonlinear system of Eq. (3.6) under a Lyapunov-based controller associated with the SI model of Eq. (3.19) is first analyzed. Subsequently, the SI model of Eq. (3.19) is incorporated into the design of the LMPC under sample-and-hold implementation of the control action to drive the state of the closed-loop system to a small neighborhood around the origin.

3.4.1 Lyapunov-based control using SI models

For the slow dynamics, it is assumed that there exists a stabilizing control law $u = \Phi_{si}(x) \in U$ such that the origin of the SI slow subsystem of Eq. (3.19) is rendered asymptotically stable in an open neighborhood $\hat{\phi}_u$ around the origin in the sense that there exist a $\mathcal{C}^1$ control Lyapunov function $\hat{V}(x)$ and four functions, $\hat{a}_1, \hat{a}_2, \hat{a}_3, \hat{a}_4$ of class $\mathcal{K}$ such that $\forall\, x \in \mathbb{R}^n$:

$$\hat{a}_1(|x|) \leq \hat{V}(x) \leq \hat{a}_2(|x|), \tag{3.20a}$$

$$\dot{\hat{V}}(x) = \frac{\partial \hat{V}(x)}{\partial x} F_{si}(x, \Phi_{si}(x)) \leq -\hat{a}_3(|x|), \tag{3.20b}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} \right| \leq \hat{a}_4(|x|) \tag{3.20c}$$

We first characterize the region $\hat{\phi}_u \subset \mathbb{R}^n$ in which the conditions of Eq. (3.20) are satisfied under the controller $u = \Phi_{si}(x) \in U$. Therefore, starting from inside $\hat{\phi}_u$, the sparse identified slow subsystem of Eq. (3.19) can be rendered asymptotically stable under the controller $u = \Phi_{si}(x) \in U$. The closed-loop stability region of the sparse identified slow subsystem of Eq. (3.19) is defined as a level set of the Lyapunov function inside $\hat{\phi}_u$: $\Omega_{\hat{\rho}} := \{x \in \hat{\phi}_u \mid \hat{V}(x) \leq \hat{\rho}\}, \hat{\rho} > 0$. The assumptions of Eq. (3.8) and Eq. (3.20) are essentially the stabilizability requirements of the first-principles slow model of Eq. (3.6) and the sparse identified slow model of Eq. (3.19), respectively.

Since the dataset for developing the sparse identified slow model of Eq. (3.6) is obtained from open-loop simulations for $x \in \Omega_\rho$ and $u \in U$, we have $\Omega_{\hat{\rho}} \subseteq \Omega_\rho$. Additionally, there exist positive constants M_{si} and L_{si} such that the following inequalities hold for all $x, x' \in \Omega_{\hat{\rho}}$ and $u \in U$:

$$|F_{si}(x,u)| \leq M_{si} \tag{3.21a}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} F_{si}(x,u) - \frac{\partial \hat{V}(x')}{\partial x} F_{si}(x',u) \right| \leq L_{si}|x - x'| \tag{3.21b}$$

The following proposition demonstrates that the feedback controller $u = \Phi_{si}(x) \in U$ is able to stabilize the nominal slow subsystem of Eq. (3.6) in the presence of model mismatch between the nominal subsystem of Eq. (3.6) and the sparse identified slow model of Eq. (3.19), provided that the modeling error is sufficiently small.

Proposition 3.1. (c.f. proposition 2 in [3]) *Under the assumption that the closed-loop sparse identified slow subsystem of Eq.* (3.19) *is rendered asymptotically stable under the controller* $u = \Phi_{si}(x) \in U\ \forall x \in \Omega_{\hat{\rho}}$, *if there exists a positive real number* $\nu_m < \hat{a}_3(|x|)/\hat{a}_4(|x|)$ *that constrains the modeling error* $|\nu| = |F(x,u) - F_{si}(x,u)| \leq \nu_m,\ \forall u \in U$ *and* $\forall x \in \Omega_{\hat{\rho}}$, *then the nominal* *closed-loop system of Eq.* (3.6) *under* $u = \Phi_{si}(x) \in U$ *is also asymptotically stable* $\forall\, x \in \Omega_{\hat{\rho}}$.

Proof. To prove that the nominal slow subsystem of Eq. (3.6) can be rendered asymptotically stable $\forall\, x \in \Omega_{\hat{\rho}}$ under the controller based on the sparse identified model of Eq. (3.19), we show that $\dot{\hat{V}}$ based on the state of the nominal slow subsystem of Eq. (3.6) can still be rendered negative under $u = \Phi_{si}(x) \in U$, $\forall\, x \in \Omega_{\hat{\rho}}$. Based on Eq. (3.20b) and Eq. (3.20c), the time-derivative of $\hat{V}$ is computed as follows:

$$\begin{aligned}\dot{\hat{V}} &= \frac{\partial \hat{V}(x)}{\partial x} F(x, \Phi_{si}(x)) \\ &= \frac{\partial \hat{V}(x)}{\partial x} \big(F_{si}(x, \Phi_{si}(x)) + F(x, \Phi_{si}(x)) - F_{si}(x, \Phi_{si}(x))\big) \\ &\leq -\hat{a}_3(|x|) + \hat{a}_4(|x|)\big(F(x, \Phi_{si}(x)) - F_{si}(x, \Phi_{si}(x))\big) \\ &\leq -\hat{a}_3(|x|) + \hat{a}_4(|x|)\nu_m\end{aligned} \tag{3.22}$$

If ν_m is chosen to satisfy $\nu_m < \hat{a}_3(|x|)/\hat{a}_4(|x|)$, then it holds that $\dot{\hat{V}} \leq -\tilde{a}_3(|x|) \leq 0$ where $\tilde{a}_3(|x|) = -\hat{a}_3(|x|) + \hat{a}_4(|x|)\nu_m > 0$. This is possible because $\hat{a}_3$ and $\hat{a}_4$ are known functions. For example, if we choose $\hat{a}_3 = a_3|x|$ and $\hat{a}_4 = a_4|x|$, then $\nu_m = a_3/a_4$. As a result, the closed-loop state of the nominal slow subsystem of Eq. (3.6) converges to the origin under $u = \Phi_{si}(x) \in U$ for all $x_0 \in \Omega_{\hat{\rho}}$. $\square$

After incorporating the SI model of Eq. (3.19) in the LMPC design, the control actions of the LMPC will be implemented in a sample-and-hold fashion. Hence, the next two propositions demonstrate the sample-and-hold properties of the Lyapunov-based controller $u = \Phi_{si}(x)$. In particular, the following proposition derives an upper bound for the error between the slow states calculated by the nominal slow subsystem of Eq. (3.6) and the slow states predicted by the SI model of Eq. (3.19).

Proposition 3.2. (c.f. proposition 3 in [3]) *For the nonlinear system $\dot{x} = F(x,u)$ of Eq.* (3.6) *and the SI model $\dot{\hat{x}} = F_{si}(\hat{x},u)$ of Eq.* (3.19) *with the same initial condition $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$, there exist a class $\mathcal{K}$ function $f_w(\cdot)$ and a positive constant κ such that the following inequalities hold $\forall\, x, \hat{x} \in \Omega_{\hat{\rho}}$:*

$$|x(t) - \hat{x}(t)| \leq f_w(t) := \frac{\nu_m}{L}(\mathrm{e}^{Lt} - 1) \tag{3.23a}$$

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \hat{a}_4\big(\hat{a}_1^{-1}(\hat{\rho})\big)|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{3.23b}$$

Proof. Denoting the error vector between the solutions of the system $\dot{x} = F(x,u)$ and the SI model $\dot{\hat{x}} = F_{si}(\hat{x},u)$ by $e(t) = x(t) - \hat{x}(t)$, the time-derivative of $e(t)$ is obtained as follows:

$$\begin{aligned} |\dot{e}(t)| &= |F(x,u) - F_{si}(\hat{x},u)| \\ &\leq |F(x,u) - F(\hat{x},u)| + |F(\hat{x},u) - F_{si}(\hat{x},u)| \end{aligned} \tag{3.24}$$

From Eq. (3.10b), $\forall\, x, \hat{x} \in \Omega_{\hat{\rho}}$, it is derived that

$$|F(x,u) - F(\hat{x},u)| \leq L|x(t) - \hat{x}(t)| \tag{3.25}$$

Since the second term $|F(\hat{x},u) - F_{si}(\hat{x},u)|$ in Eq. (3.24) represents the modeling error, it is bounded by $|\nu| \leq \nu_m$ for all $\hat{x} \in \Omega_{\hat{\rho}}$. Hence, based on Eq. (3.25) and the bound on the modeling error, $\dot{e}(t)$ is bounded as follows:

$$\begin{aligned} |\dot{e}(t)| &\leq L|x(t) - \hat{x}(t)| + \nu_m \\ &\leq L|e(t)| + \nu_m \end{aligned} \tag{3.26}$$

Based on the zero initial condition (*i.e.*, $e(0) = 0$), the norm of the error vector can be bounded as follows $\forall\, x(t), \hat{x}(t) \in \Omega_{\hat{\rho}}$:

$$|e(t)| = |x(t) - \hat{x}(t)| \leq \frac{\nu_m}{L}(\mathrm{e}^{Lt} - 1) \tag{3.27}$$

Subsequently, to derive Eq. (3.23b) for all $x, \hat{x} \in \Omega_{\hat{\rho}}$, we derive the Taylor series expansion of $\hat{V}(x)$ around $\hat{x}$ as follows:

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\partial \hat{V}(\hat{x})}{\partial x}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{3.28}$$

where κ is a positive real number. Using Eq. (3.20a) and Eq. (3.20c), it follows that

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \hat{a}_4\big(\hat{a}_1^{-1}(\hat{\rho})\big)|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{3.29}$$

This completes the proof of Proposition 3.2. □

The final proposition below proves that the closed-loop state of the nominal slow subsystem of Eq. (3.6) remains bounded in $\Omega_{\hat{\rho}}$ for all times, and can be ultimately bounded in a small subset $\Omega_{\rho_{\min}}$ containing the origin under the sample-and-hold implementation of the Lyapunov-based controller $u = \Phi_{si}(x) \in U$.

Proposition 3.3. *Consider the nominal slow subsystem of Eq.* (3.6) *under the controller* $u = \Phi_{si}(\hat{x}) \in U$ *that is designed to stabilize the SI system of Eq.* (3.19) *and meets the conditions of Eq.* (3.20)*. The controller is implemented in a sample-and-hold fashion, i.e.,* $u(t) = \Phi_{si}(\hat{x}(t_k))$,

$\forall\, t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$. Let $\epsilon_s,\ \epsilon_w > 0$, $\Delta > 0$ and $\hat{\rho} > \rho_{\min} > \rho_{si} > \rho_s$ satisfy

$$-\hat{a}_3\big(\hat{a}_2^{-1}(\rho_s)\big) + L_{si}M_{si}\Delta \leq -\epsilon_s \tag{3.30a}$$

$$-\tilde{a}_3\big(\hat{a}_2^{-1}(\rho_s)\big) + L'M\Delta \leq -\epsilon_w \tag{3.30b}$$

and

$$\rho_{si} := \max\{\hat{V}(\hat{x}(t+\Delta)) \mid \hat{x}(t) \in \Omega_{\rho_s}, u \in U\} \tag{3.31a}$$

$$\rho_{\min} \geq \rho_{si} + \hat{a}_4\big(\hat{a}_1^{-1}(\hat{\rho})\big)f_w(\Delta) + \kappa(f_w(\Delta))^2 \tag{3.31b}$$

Then, for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, there exists a class $\mathcal{KL}$ function β_x and a class $\mathcal{K}$ function $\bar{\gamma}$ such that the following inequality holds:

$$|x(t)| \leq \beta_x(|x(0)|, t) + \bar{\gamma}(\rho_{\min}) \tag{3.32}$$

and the slow states $x(t)$ of the nominal subsystem of Eq. (3.6) *is bounded in $\Omega_{\hat{\rho}}$ for all times and ultimately bounded in $\Omega_{\rho_{\min}}$.*

Proof. Part 1: Assuming $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, we first show that $\hat{V}(\hat{x})$ is decreasing under the controller $u(t) = \Phi_{si}(x(t_k)) \in U$ for $t \in [t_k, t_{k+1})$, where $x(t)$ and $\hat{x}(t)$ denote the solutions of the nominal slow subsystem of Eq. (3.6) and the SI subsystem of Eq. (3.19), respectively. The time-derivative of $\hat{V}(\hat{x})$ along the trajectory $\hat{x}(t)$ of the SI model of Eq. (3.19) for $t \in [t_k, t_{k+1})$ is computed as follows:

$$
\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) &= \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{si}(\hat{x}(t), \Phi_{si}(\hat{x}(t_k))) \\
&= \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{si}(\hat{x}(t_k), \Phi_{si}(\hat{x}(t_k))) + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{si}(\hat{x}(t), \Phi_{si}(\hat{x}(t_k))) \\
&\quad - \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{si}(\hat{x}(t_k), \Phi_{si}(\hat{x}(t_k)))
\end{aligned}
\tag{3.33}
$$

Combining Eq. (3.20a) and Eq. (3.20b), the following inequality is derived:

$$
\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) \leq & -\hat{a}_3\big(\hat{a}_2^{-1}(\rho_s)\big) + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{si}(\hat{x}(t), \Phi_{si}(\hat{x}(t_k))) \\
& - \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{si}(\hat{x}(t_k), \Phi_{si}(\hat{x}(t_k)))
\end{aligned}
\tag{3.34}
$$

Using the Lipschitz condition of Eq. (3.21) and the fact that $\hat{x} \in \Omega_{\hat{\rho}}$, $u \in U$, the upper bound of $\dot{\hat{V}}(\hat{x}(t))$ is obtained $\forall\, t \in [t_k, t_{k+1})$:

$$
\begin{aligned}
\dot{\hat{V}}(\hat{x}(t)) \leq & -\hat{a}_3\big(\hat{a}_2^{-1}(\rho_s)\big) + L_{si}|\hat{x}(t) - \hat{x}(t_k)| \\
\leq & -\hat{a}_3\big(\hat{a}_2^{-1}(\rho_s)\big) + L_{si} M_{si} \Delta
\end{aligned}
\tag{3.35}
$$

Hence, if Eq. (3.30a) is satisfied, the following inequality holds $\forall\, \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, $\forall\, t \in [t_k, t_{k+1})$:

$$
\dot{\hat{V}}(\hat{x}(t)) \leq -\epsilon_s \tag{3.36}
$$

Integrating the above differential equation with respect to time over $t \in [t_k, t_{k+1})$, it is derived that $V(\hat{x}(t_{k+1})) \leq V(\hat{x}(t_k)) - \epsilon_s \Delta$. We have shown thus far that for all $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, the closed-loop state of the SI slow subsystem of Eq. (3.19) is bounded inside the closed-loop stability region $\Omega_{\hat{\rho}}$ for all times and moves towards the origin under the controller $u = \Phi_{si}(\hat{x}) \in U$ when implemented in a sample-and-hold fashion.

We note, however, that Eq. (3.36) may fail to hold when $x(t_k) = \hat{x}(t_k) \in \Omega_{\rho_s}$, which would imply that the state may exit Ω_{ρ_s} within one sampling period. Hence, we design another region

$\Omega_{\rho_{si}}$ according to Eq. (3.31a), which ensures that the closed-loop state $\hat{x}(t)$ of the SI model does not leave $\Omega_{\rho_{si}}$ for all $t \in [t_k, t_{k+1})$, $u \in U$ and $\hat{x}(t_k) \in \Omega_{\rho_s}$ within one sampling period. If the state $\hat{x}(t_{k+1})$ leaves Ω_{ρ_s}, Eq. (3.36) is satisfied again at $t = t_{k+1}$, reactivating the controller $u = \Phi_{si}(x(t_{k+1}))$ and driving the state towards Ω_{ρ_s} over the next sampling period. As a result, it is shown that the state converges to $\Omega_{\rho_{si}}$ for the closed-loop SI subsystem of Eq. (3.19) for all $\hat{x}_0 \in \Omega_{\hat{\rho}}$. In Part 2, we show that the closed-loop state of the nominal slow subsystem of Eq. (3.6) can also be bounded in $\Omega_{\hat{\rho}}$ for all times and ultimately bounded in a small neighborhood around the origin under the sample-and-hold implementation of the controller $u = \Phi_{si}(x) \in U$.

Part 2: Repeating the analysis performed for the SI subsystem of Eq. (3.19), we first assume $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and compute the time-derivative of $\hat{V}(x)$ for the nominal slow subsystem of Eq. (3.6) (*i.e.*, $\dot{x} = F(x, u)$) as follows:

$$\begin{aligned}\dot{\hat{V}}(x(t)) &= \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{si}(x(t_k))) \\ &= \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k))) + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{si}(x(t_k))) \\ &\quad - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k)))\end{aligned} \tag{3.37}$$

From Eq. (3.22), $\frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k))) \leq -\tilde{a}_3\big(x(t_k)\big)$ holds for all $x \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ where $\tilde{a}_3(\cdot)$ was defined at the end of Proposition 3.1. Using Eq. (3.8a) and the Lipschitz definition in Eq. (3.10), the following inequality is obtained for $\dot{\hat{V}}(x(t))$, $\forall\, t \in [t_k, t_{k+1})$ and $\forall\, x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$:

$$
\begin{aligned}
\dot{\hat{V}}(x(t)) \leq & -\tilde{a}_3\big(\hat{a}_2^{-1}(\rho_s)\big) + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{si}(x(t_k))) - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k))) \\
\leq & -\tilde{a}_3\big(\hat{a}_2^{-1}(\rho_s)\big) + L^{'}|x(t) - x(t_k)| \\
\leq & -\tilde{a}_3\big(\hat{a}_2^{-1}(\rho_s)\big) + L^{'} M \Delta
\end{aligned}
\tag{3.38}
$$

Consequently, if Eq. (3.30b) is satisfied, the following inequality holds $\forall\, x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s},\ \forall\, t \in [t_k, t_{k+1})$:

$$
\dot{\hat{V}}(x(t)) \leq -\epsilon_w \tag{3.39}
$$

Integrating the above differential equation with respect to time between any two points in $[t_k, t_{k+1})$, it is derived for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$:

$$
\hat{V}(x(t_{k+1})) \leq V(x(t_k)) - \epsilon_w \Delta \tag{3.40}
$$

$$
\hat{V}(x(t)) \leq \hat{V}(x(t_k)), \forall\, t \in [t_k, t_{k+1}) \tag{3.41}
$$

Therefore, the state of the closed-loop system of Eq. (3.6) remains in $\Omega_{\hat{\rho}}$ for all times. Furthermore, it follows that the controller $u = \Phi_{si}(x)$ is still able to drive the state of the nominal slow subsystem of Eq. (3.6) towards the origin within every sampling period. Moreover, if $x(t_k) \in \Omega_{\rho_s}$, it was already shown in Part 1 that the state of the SI model of Eq. (3.19) is maintained in $\Omega_{\rho_{si}}$ for one sampling period. Considering the bounded modeling error between the state of the SI model of Eq. (3.19) and the state of the nominal slow subsystem of Eq. (3.6) given by Eq. (3.30a), there exists a compact set $\Omega_{\rho_{\min}} \supset \Omega_{\rho_{si}}$ that satisfies Eq. (3.31b) such that the state of the nominal slow subsystem of Eq. (3.6) remains within $\Omega_{\rho_{\min}}$ during one sampling period if the state of the SI model of Eq. (3.19) is bounded in $\Omega_{\rho_{si}}$. If the state $x(t)$ enters $\Omega_{\rho_{\min}} \backslash \Omega_{\rho_s}$, we have shown that Eq. (3.41) holds, and thus, the state will be driven towards the origin again under $u = \Phi_{si}(x)$ during the next sampling period, ultimately bounding the closed-loop slow subsystem in $\Omega_{\rho_{\min}}$. Therefore, based on the continuity of the Lyapunov function $\hat{V}$, there exist a class $\mathcal{KL}$ function β_x and a class $\mathcal{K}$ function $\bar{\gamma}$ such that if $x_0 \in \Omega_{\hat{\rho}}$, then $x(t) \in \Omega_{\hat{\rho}}$, $\forall\, t \geq t_0$ and

$$|x(t)| \leq \beta_x(|x(0)|, t) + \bar{\gamma}(\rho_{\min}) \tag{3.42}$$

□

3.4.2 Lyapunov-based MPC (LMPC) formulation

The LMPC is based on the Lyapunov-based controller $\Phi_{si}(x)$. The controller $\Phi_{si}(x)$ is used to define a stability constraint for the LMPC controller. This ensures that the LMPC controller inherits the stability and robustness properties of the Lyapunov-based controller $\Phi_{si}(x)$. The LMPC controller is given by the following optimization problem:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u(t)) \,\mathrm{d}t \tag{3.43a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{si}(\tilde{x}(t), u(t)) \tag{3.43b}$$

$$u(t) \in U, \ \forall\, t \in [t_k, t_{k+N}) \tag{3.43c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{3.43d}$$

$$\dot{\hat{V}}(x(t_k), u) \leq \dot{\hat{V}}(x(t_k), \Phi_{si}(x(t_k))), \ \text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{si}} \tag{3.43e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{si}, \ \forall\, t \in [t_k, t_{k+N}), \ \text{if } x(t_k) \in \Omega_{\rho_{si}} \tag{3.43f}$$

where $\tilde{x}$ denotes the predicted trajectory of the slow states, $S(\Delta)$ is the set of piece-wise constant functions with period Δ, and N is the number of sampling periods within the prediction horizon. $\dot{\hat{V}}(x,u)$ denotes $\frac{\partial \hat{V}(x)}{\partial x}(F_{si}(x,u))$. Let $u = u^*(t)$, $t \in [t_k,\, t_{k+N})$ denote the optimal input trajectory calculated by the LMPC over the entire prediction horizon. It is noted that only the first control action of the computed sequence, $u^*(t_k)$, which corresponds to the first sampling period of the prediction horizon, is applied over the first sampling period, and the LMPC is resolved at the next sampling time.

In the optimization problem of Eq. (3.43), the objective function of Eq. (3.43a) is equal to the integral of $L(\tilde{x}(t), u(t))$ over the entire prediction horizon. The constraint of Eq. (3.43b) is the approximate slow model of Eq. (3.19) that is used to predict the slow states of the closed-loop slow subsystem. Eq. (3.43c) specifies the input constraints to be applied over the entire prediction horizon. Eq. (3.43d) defines the initial condition $\tilde{x}(t_k)$ of Eq. (3.43b), which is the slow state measurement at $t = t_k$. The first Lyapunov constraint of Eq. (3.43e) guarantees that the closed-loop state moves towards the origin if $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{si}}$. However, if $x(t_k)$ enters $\Omega_{\rho_{si}}$, the states predicted by the approximate slow model of Eq. (3.43b) will be maintained in $\Omega_{\rho_{si}}$ over the entire prediction horizon.

3.5 Closed-loop stability analysis

The closed-loop stability of the singularly perturbed system of Eq. (3.1) under the LMPC of Eq. (3.43) is established in the following theorem under appropriate conditions.

Theorem 3.1. *Consider the system of Eq.* (3.1) *in closed-loop with u^* computed by the LMPC of*

Eq. (3.43) *based on the Lyapunov-based controller* $\Phi_{si}(x)$ *that satisfies the conditions of Eq.* (3.20). *Let Assumptions 3.1 and 3.2 and the conditions of Propositions 3.1–3.3 hold. Then there exist functions* β_x, β_y *of class* $\mathcal{KL}$*, a pair of positive real numbers* (δ, d) *and* $\exists\ \epsilon^* > 0$ *such that if* $\max\{|x(0)|, |y(0)|\} \leq \delta$ *and* $\epsilon \in (0, \epsilon^*]$*, then* $\forall\, t \geq 0$,

$$|x(t)| \leq \beta_x(|x(0)|, t) + \bar{\gamma}(\rho_{\min}) + d \tag{3.44}$$

$$|y(t)| \leq \beta_y\left(|y(0)|, \frac{t}{\epsilon}\right) + d \tag{3.45}$$

Proof. The closed-loop system takes the following form after substituting the optimal control action u^* into Eq. (3.1).

$$\dot{x} = f_1(x, z, u^*, \epsilon) \tag{3.46a}$$

$$\epsilon\dot{z} = f_2(x, z, \epsilon) \tag{3.46b}$$

By setting $\epsilon = 0$, we have

$$\dot{x} = f_1(x, z, u^*, 0) \tag{3.47a}$$

$$0 = f_2(x, z, 0) \tag{3.47b}$$

Since Eq. (3.47b) has a unique, isolated solution $z_s = \hat{f}_2(x)$, following Eq. (3.4), Eq. (3.47) can be written in the following form:

$$\dot{x} = f_1(x, \hat{f}_2(x), u^*, 0) \tag{3.48}$$

When $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{si}}$, the constraint of Eq. (3.43e) requires that the Lyapunov function value decrease at least at the rate under $u = \Phi_{si}(x)$. As a result, the time-derivative of Lyapunov function $\dot{V}$ under $u = u^*$ is rendered negative. Based on the results in Proposition 3, the state of the slow subsystem of Eq. (3.46a) will approach the origin and enter $\Omega_{\rho_{si}}$ within finite sampling steps provided that the modeling error is sufficiently small. After $x(t_k)$ enters $\Omega_{\rho_{si}}$, the constraint of Eq. (3.43f) maintains the predicted state within $\Omega_{\rho_{si}}$ afterwards. Since the modeling error and sampling period are sufficiently small, we have shown in Proposition 3 that the true state of Eq. (3.46a) can be bounded in $\Omega_{\rho_{\min}}$, which is a slightly larger set containing $\Omega_{\rho_{si}}$. Therefore, for any initial state $x_0 \in \Omega_{\hat{\rho}}$, LMPC ensures that the state $x(t)$ of the closed-loop slow subsystem of Eq. (3.46a) is bounded in $\Omega_{\hat{\rho}}$ for all times, and satisfies the bound of Eq. (3.32) in Proposition 3.

Subsequently, by letting $\tau = \frac{t}{\epsilon}$, $y = z - \hat{f}_2(x)$ and $\epsilon = 0$, we obtain the closed-loop fast subsystem:

$$\frac{\mathrm{d}y}{\mathrm{d}\tau} = f_2(x, \hat{f}_2(x) + y, 0) \tag{3.49}$$

Note that the origin of the closed-loop system of Eq. (3.49) is assumed to be globally asymptotically stable such that Eq. (3.7) holds for any $y(0) \in \mathbb{R}^p$ in Assumption 2. Therefore, the closed-loop system of Eq. (3.46) satisfies all the assumptions for Theorem 1 in [144], which implies that there exist class $\mathcal{KL}$ functions $\beta_x(\cdot)$ and $\beta_y(\cdot)$, positive real numbers (δ, d), and $\epsilon^* > 0$ such that if $\max\{|x(0)|, |y(0)|\} \leq \delta$ and $\epsilon \in (0, \epsilon^*]$, then, the slow and fast system states are bounded by Eqs. (3.44) and (3.45). □

3.6 Application to a chemical process example

3.6.1 Process description

We demonstrate the application of the LMPC of Eq. (3.43) based on the sparse-identified slow subsystem using a chemical process example. We consider a perfectly-mixed, non-isothermal CSTR as shown in Fig. 3.1.

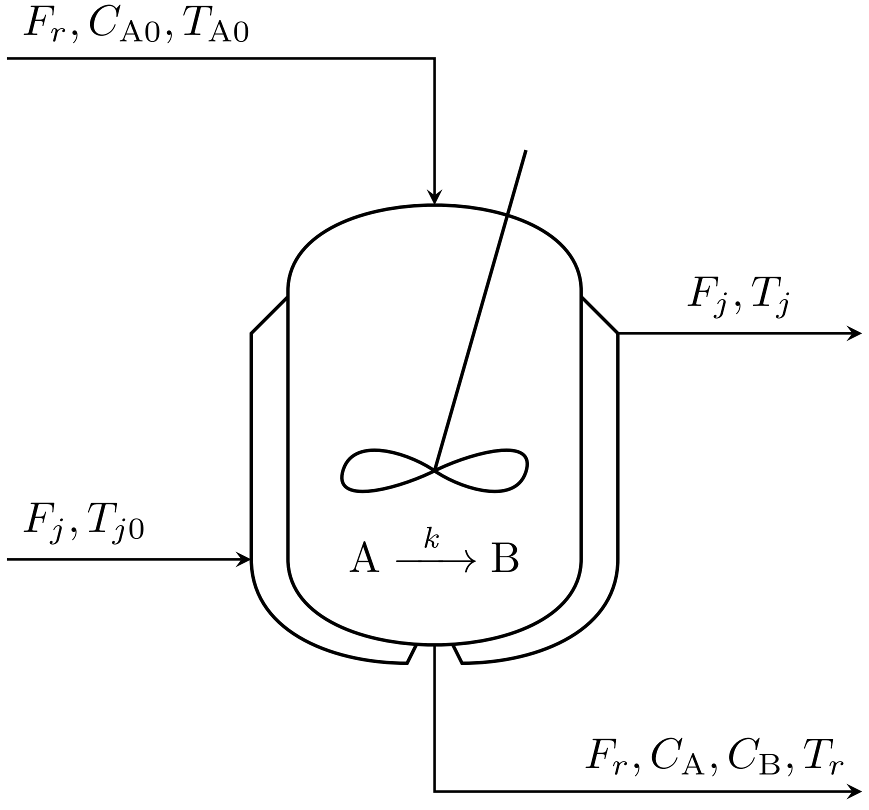


Figure 3.1: A continuous-stirred tank reactor with jacket.

A single, endothermic, irreversible reaction of the form,

$$\mathrm{A} \xrightarrow{k} \mathrm{B} \tag{3.50}$$

occurs in the CSTR. The concentration of reactant A in the reactor is denoted by C_A. Assuming the vessel has a constant holdup, the volume of the liquid in the reactor is represented by V_r. The temperature of the reactor contents is denoted by T_r. The feed to the reactor contains pure species A with molar concentration $C_{\mathrm{A}0}$, at a flow rate F, and temperature $T_{\mathrm{A}0}$. Owing to the endothermic reaction taking place in the reactor, energy must be provided to the reactor via a jacket. The heating jacket has a volume V_j with heat transfer fluid at an inlet temperature of T_{j0} being added to it at a flow rate F_j. The reactor contents and the heat transfer fluid have constant densities of ρ_m and ρ_j, respectively, and have constant heat capacities of $c_{p,m}$ and $c_{p,j}$, respectively. ΔH_r denotes the enthalpy of the reaction, U represents the heat transfer coefficient, and A_r is the heat transfer contact area between the reactor and the jacket. The rate constant k in Eq. (3.50) is assumed to be of the form,

$$k = k_0 \exp\left(\frac{-E}{RT_r}\right) \tag{3.51}$$

where k_0, R, and E denote the pre-exponential constant, ideal gas constant, and activation energy of the reaction, respectively. Under these modeling assumptions, the dynamic equations governing the behavior of the reactor are given by the following material and energy balances:

$$V_r \frac{\mathrm{d}C_\mathrm{A}}{\mathrm{d}t} = F_r\left(C_{\mathrm{A}0} - C_\mathrm{A}\right) - k_0 \mathrm{e}^{-E/RT_r} C_\mathrm{A} V_r \tag{3.52}$$

$$V_r \frac{\mathrm{d}T_r}{\mathrm{d}t} = F_r\left(T_{\mathrm{A}0} - T_r\right) + \frac{(-\Delta H_r)}{\rho_m c_{p,m}} k_0 \mathrm{e}^{-E/RT_r} C_\mathrm{A} V_r + \frac{UA_r}{\rho_m c_{p,m}}\left(T_j - T_r\right) \tag{3.53}$$

$$V_j \frac{\mathrm{d}T_j}{\mathrm{d}t} = F_j(T_{j0} - T_j) - \frac{UA_r}{\rho_j c_{p,j}}\left(T_j - T_r\right) \tag{3.54}$$

with the process parameters' values given in Table 3.1.

While we may define $\epsilon = V_r/V_j$ to rewrite Eq. (3.54) in the standard singularly perturbed

Table 3.1: Parameter values for chemical process example.

$V_r = 1.0\,\mathrm{m}^3$	$k_0 = 3.36 \times 10^6\,\mathrm{h}^{-1}$
$V_j = 0.08\,\mathrm{m}^3$	$E = 8.0 \times 10^3\,\mathrm{kcal\,kg}^{-1}$
$A_r = 6.0\,\mathrm{m}^3$	$T_{\mathrm{A0}} = 310.0\,\mathrm{K}$
$U = 1000.0\,\mathrm{kcal\,h}^{-1}\,\mathrm{m}^{-2}\,\mathrm{K}^{-1}$	$T_{j0} = 357.5\,\mathrm{K}$
$R = 1.987\,\mathrm{kcal\,kmol}^{-1}\,\mathrm{K}^{-1}$	$\rho_m = 900.0\,\mathrm{kg\,m}^{-3}$
$\Delta H_r = 5.4 \times 10^4\,\mathrm{kcal\,mol}^{-1}$	$\rho_j = 800.0\,\mathrm{kg\,m}^{-3}$
$c_{p,m} = 0.231\,\mathrm{kcal\,kg}^{-1}\,\mathrm{K}^{-1}$	$F_r = 3.0\,\mathrm{m}^3\,\mathrm{h}^{-1}$
$c_{p,j} = 0.200\,\mathrm{kcal\,kg}^{-1}\,\mathrm{K}^{-1}$	$F_j = 20.0\,\mathrm{m}^3\,\mathrm{h}^{-1}$
$C_{\mathrm{A0},s} = 3.75\,\mathrm{kmol\,m}^{-3}$	$C_{\mathrm{A},s} = 2.54\,\mathrm{kmol\,m}^{-3}$
$T_{r,s} = 274.4\,\mathrm{K}$	$T_{j,s} = 303.3\,\mathrm{K}$

form, in which case both temperatures have fast dynamics relative to the concentration dynamics, this can also be deduced from the data generation outlined in the next subsection.

The initial operating point of the CSTR is its steady-state $(C_{\mathrm{A},s}, T_{r,s}, T_{j,s}) = (2.54\,\mathrm{kmol\,m}^{-3}, 274.4\,\mathrm{K}, 303.3\,\mathrm{K})$ and $C_{\mathrm{A0},s} = 3.75\,\mathrm{kmol\,m}^{-3}$. The manipulated input variable is the feed concentration of reactant A, represented by the deviation variable $\Delta C_{\mathrm{A0}} = C_{\mathrm{A0}} - C_{\mathrm{A0},s}$ and bounded as per $|\Delta C_{\mathrm{A0}}| \leq 3.5\,\mathrm{kmol\,m}^{-3}$. Consequently, the states and manipulated input of the closed-loop system in deviation form are represented by

$$x = C_{\mathrm{A}} - C_{\mathrm{A},s} \tag{3.55}$$

and

$$u = \Delta C_{\mathrm{A0}} \tag{3.56}$$

respectively. Hence, the origin of the state-space, given by $(x_s^*, u_s^*) = (0, 0)$, is the equilibrium point of the system.

It is desired to apply the SI-based LMPC of Eq. (3.43) to maintain the operation of the CSTR at the equilibrium point $(C_{\mathrm{A},s}, T_{r,s}, T_{j,s})$ by manipulating the feed concentration ΔC_{A0}. For the simulation of the ODE system of Eqs. (3.52) to (3.54), the explicit Euler method is used to numerically integrate the equations with an integration time step of $h_c = 0.1\,\mathrm{seconds}$. Since the optimization problem in the LMPC of Eq. (3.43) is nonlinear, it is solved using PyIpopt, the python interface of the IPOPT software package [145], with a sampling period of $\Delta = 10\,\mathrm{seconds}$.

3.6.2 Data generation and model development

To design the LMPC of Eq. (3.43), the slow subsystem must be first reconstructed as an ODE to be incorporated as the process model. To identify the slow subsystem using data, a range of 10 different initial conditions are chosen with $C_{\mathrm{A}}(0)$ taking values between $0\,\mathrm{mol\,m^{-3}}$ and $9.0\,\mathrm{mol\,m^{-3}}$ in intervals of $1.0\,\mathrm{mol\,m^{-3}}$, $T_r(0)$ taking values between $280\,\mathrm{K}$ and $370\,\mathrm{K}$ in intervals of $10\,\mathrm{K}$, and $T_j(0)$ varying between $300\,\mathrm{K}$ and $390\,\mathrm{K}$ in intervals of $10\,\mathrm{K}$. The system of differential equations described by Eqs. (3.52) to (3.54) is numerically integrated for each set of initial conditions from the initial time of $0.0\,\mathrm{hr}$ to $1.0\,\mathrm{hr}$ with a step size of 10^{-6} hr to simulate the experimental/industrial process. The generated data is then sampled with a sampling period larger than the step size to simulate data collection via sensors and measuring instruments. This sampling is conducted with a sampling period of $0.005\,\mathrm{hr}$ since any larger sampling period leads to inaccurate estimates of the time-derivative, causing the sparse identification algorithm to fail for this system. The sampled data from the 10 runs is finally combined into the data matrix, X. A plot of X and/or the gradient of X versus time clearly indicates that both temperature variables, T_r and T_j, are the fast states, while C_{A} is the only slow state in this system. Examples of such plots can be found in Figures 3.6, 3.7 and 3.8 in the simulation results (Section 3.6.3). Therefore, we only construct a model for the concentration C_{A} using sparse identification, as it is the slow subsystem.

The sparse identification algorithm was applied to the concentration data from the open-loop simulations. Ten iterations were completed, and the nonzero coefficients were confirmed to have converged within five iterations. The coefficient threshold λ was fine-tuned to a value of 2.0 to yield the following slow model,

$$\frac{\mathrm{d}C_{\mathrm{A}}}{\mathrm{d}t} = 9.93101 - 3.85957C_{\mathrm{A}} \tag{3.57}$$

However, Eq. (3.57) needed to be converted to deviation variables and the input accounted for. Hence, using Eq. (3.55), Eq. (3.57) was written as

$$\frac{\mathrm{d}x}{\mathrm{d}t} = 9.93101 - 3.85957(x + C_{\mathrm{A},s}) = 0.11901 - 3.85957x \tag{3.58}$$

In Eq. (3.58), the constant term, 0.11901, was due to numerical inaccuracies resulting from modeling via sparse identification and the lack of data near the origin of the system in deviation variables (which corresponds to data near the steady-state of the original system). The data generation was carried out over a 1-hour simulation duration, within which the system did not reach very close to its steady-state. However, the simulation length can be increased beyond 1 hour to increase the amount of data near the steady-state, which would improve model performance near the origin. The value of this constant residual term in Eq. (3.58) was found to monotonically decrease with increasing simulation length for the data generation step as shown in Figure 3.2. If the data generation is carried out over a simulation length of 20 hours, the value of the residual term reduces to 0.00562783. The optimal sparse identification model built from data generated over an infinitely long simulation would not contain the constant term. Furthermore, we know that the origin is an equilibrium point for the system in deviation variables. Therefore, the constant term in Eq. (3.58) was neglected. Additionally, the input term must be present in the closed-loop model. The procedure outlined in Section 3.3.2 was used to calculate $\hat{g}(x)$, which was approximately between 3 and 4. Comparing this result to the first-principles model, where the term corresponding to the manipulated input was known to be $3u$, $\hat{g}(x)$ is taken to be equal to 3. Hence, the sparse-identified slow subsystem was identified as

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -3.85957x + 3u \tag{3.59}$$

While the above procedure required partial knowledge of the first-principles model of the process, the stability results of Section 3.5 merely assumed an input-affine model as given in Eq. (3.19). Hence, we could alternatively add a manipulated input term of u rather than $3u$ to Eq. (3.59), and the stability results would hold, while the simulation results would scale accordingly. Furthermore, the coefficient associated with u could also be found more accurately using steady-state data for different input sequences and initial conditions. Henceforth, Eq. (3.59) will be referred to as the SI slow subsystem or SI slow model for the CSTR system.

After the model identification, the Lyapunov functions for the nominal slow subsystem and the SI slow subsystem are chosen to be identical *i.e.,* $V(x) = \hat{V}(x) = 0.01x^2$. The region $\hat{\phi}_u$ where $\dot{\hat{V}} < -k\hat{V}$ under the controller $u = \Phi_{si}(x) \in U$ is analyzed in Figure 3.3. The range $x \in [-2, 2]$ is considered because the steady-state concentration is $2.54\,\mathrm{kmol\,m^{-3}}$, and concentrations may not be negative. As the Lyapunov function in this case is a quadratic function, its level sets are represented

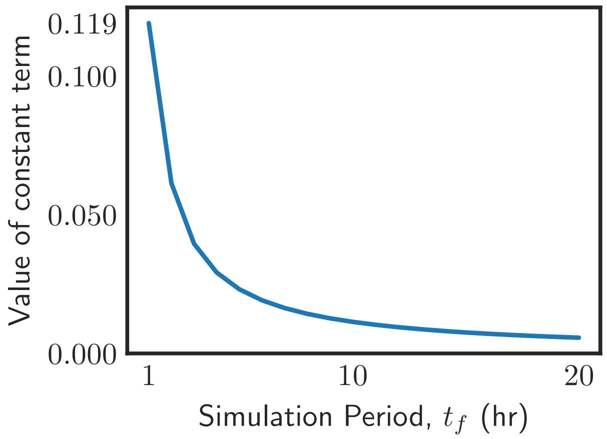


Figure 3.2: Plot of the constant residual term in Eq. (3.58) as a function of the total simulation duration t_f used for data generation.

by intervals in x. Hence, the closed-loop stability region $\Omega_{\hat{\rho}}$ for the reactor system described by the SI slow subsystem is represented by the largest level set of $\hat{V}$ in $\hat{\phi}_u$. From Figure 3.3, it is observed that $\dot{\hat{V}} < -k\hat{V}$ is satisfied for the entire range of x. Therefore, the closed-loop stability region $\Omega_{\hat{\rho}}$ is also characterized as the entire range of $x \in [-2, 2]$. Considering $|x| < 0.05$ to be sufficiently close to the origin, a value of 2.5×10^{-5} was calculated for ρ_{si}. The cost function in Eq. (3.43a) is chosen to be $L(x, u) = |x|^2_{Q_1} + |u|^2_{Q_2}$ where $Q_1 = 10$ and $Q_2 = 1$, such that L achieves its minimum at the origin of the closed-loop system.

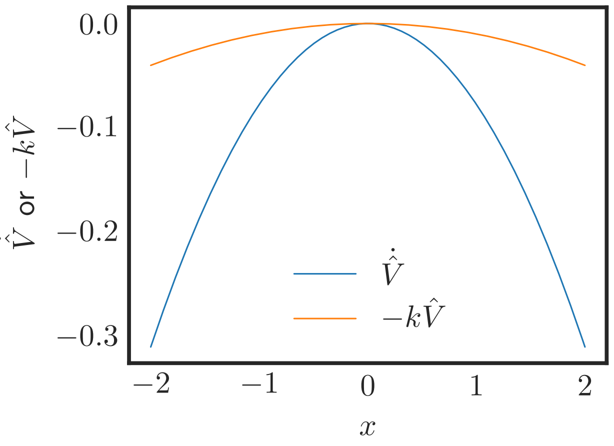


Figure 3.3: Plot of $\dot{\hat{V}}$ and $-k\hat{V}$.

3.6.3 Simulation results

First, simulations are conducted for the closed-loop system under the MPC using the nominal slow subsystem and the SI slow subsystem as the process model of Eq. (3.43b), separately. While, in practice, if solely data is available, only the sparse identification approach can be used, the nominal slow subsystem is used to represent the experimental process. Hence, the LMPC utilizing the nominal slow subsystem yields the ideal performance that an LMPC with a data-driven process model can achieve and be compared against. Figure 3.4 shows that the open-loop state trajectory of the slow state predicted by the sparse-identified concentration model of Eq. (3.59) with $u = 0$ is in close agreement with the first-principles concentration model of Eq. (3.52), for a fixed time interval and the same initial condition of $x_0 = 1 \in \Omega_{\hat{\rho}}$. As a result, it can be concluded that the sparse-identified model of Eq. (3.59) can be considered a satisfactory approximation of the first-principles process model of Eq. (3.52). Consequently, the sparse-identified slow model of Eq. (3.59) is used as the process model in the LMPC of Eq. (3.43).

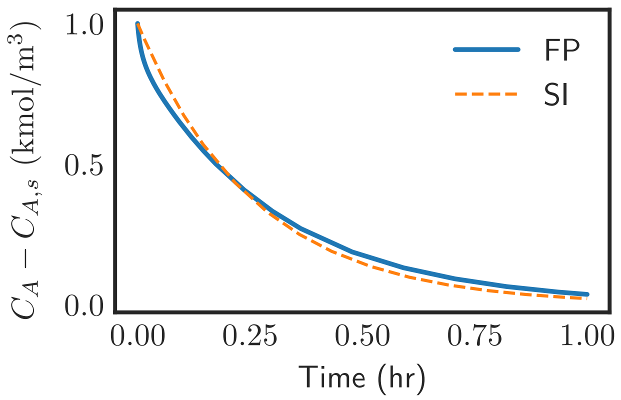


Figure 3.4: Comparison of the slow state as computed by the first-principles slow subsystem (solid line) of Eq. (3.52) and predicted by the SI slow subsystem (dashed line) of Eq. (3.59) with $u = 0$.

Figures 3.5 and 3.6 depict the closed-loop states of the CSTR for a range of initial conditions

$x_0 \in \Omega_{\hat{\rho}}$ under the LMPC utilizing the SI slow model. It is observed that the states converge to a small neighborhood containing the origin, $\Omega_{\rho_{\min}}$, for all initial conditions studied. Therefore, the LMPC with the SI slow model can be considered adequate to stabilize the CSTR system due to the sufficiently small modeling error.

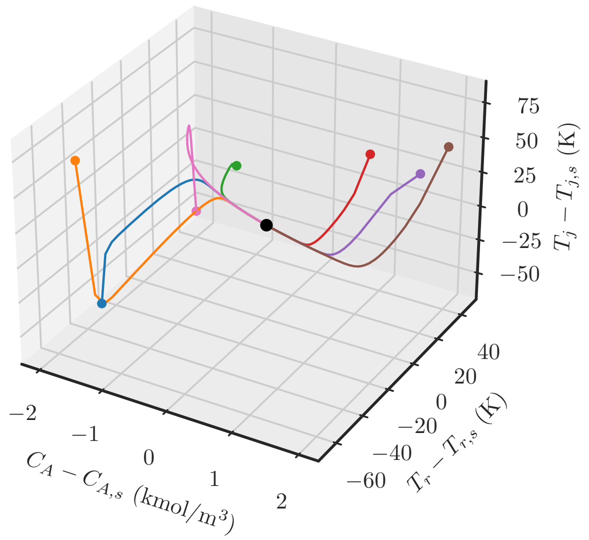


Figure 3.5: State-space trajectories for the CSTR in closed-loop under the LMPC utilizing the SI slow model for a range of initial conditions. Each line represents a trajectory from an initial condition marked by a colored dot of the corresponding color. The black dot is the origin.

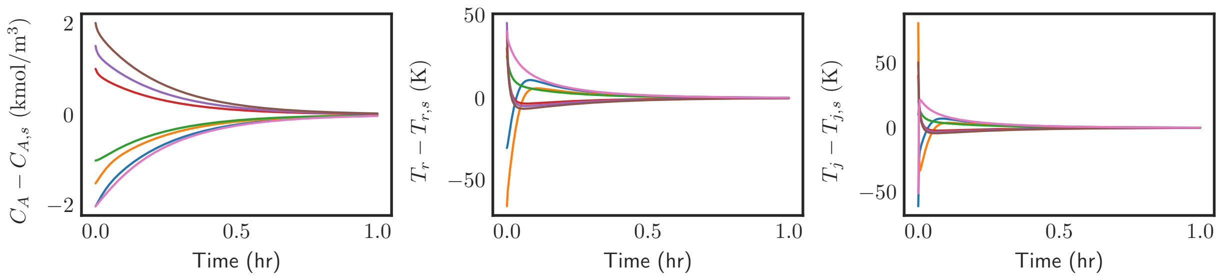


Figure 3.6: State profiles for the CSTR in closed-loop under the LMPC utilizing the SI slow model for a range of initial conditions, each line representing a trajectory from a different initial condition.

The closed-loop states and manipulated input profiles of the CSTR system of Eqs. (3.52) to (3.54) under the LMPC are shown in Figures 3.7, 3.8 and 3.9. Figure 3.7 compares the state trajectories for the closed-loop system from an initial condition of $(C_{\mathrm{A}} - C_{\mathrm{A},s}, T_r - T_{r,s}, T_j - T_{j,s}) = (1\,\mathrm{kmol\,m^{-3}}, 30\,\mathrm{K}, 40\,\mathrm{K})$ when the different process models are used in the controller. An MPC prediction horizon length of $N = 5$ is chosen for both cases. In both simulations, it is observed that the state trajectory is driven to $\Omega_{\rho_{\min}}$ under the controller faster than in the open-loop scenario without a controller. Analyzing the time-varying states more closely as shown in Figure 3.8, it is demonstrated that the evolution of the states is nearly identical for both the nominal slow model and the SI slow model. Figure 3.9 shows the manipulated input profiles for the closed-loop system, which are observed to be within the range of permissible u. The results of Figures 3.7, 3.8 and 3.9 were also reproduced for other initial conditions, implying the LMPC incorporating the SI slow model is able to stabilize the CSTR system and drive the states to the origin nearly as efficiently as the LMPC utilizing the nominal slow model. Furthermore, when the integral of the cost function of the LMPC $\int_{t=0}^{t_p} L(x(\tau), u(\tau))\,\mathrm{d}\tau$ is calculated over the simulation period $t_p = 1\,\mathrm{hr}$, it is obtained that $L = 330.2$ and $L = 330.0$ for the LMPC utilizing the first-principles model and the SI slow model, respectively. The negligible difference indicates that the closed-loop performance under both models is similar with respect to energy as well as speed of convergence to the origin. The value of the cost $\mathcal{J}$ over the simulation period of 1 hr is shown in Figure 3.10, indicating that there is no significant difference in the states and manipulated inputs computed by the LMPC using either process model.

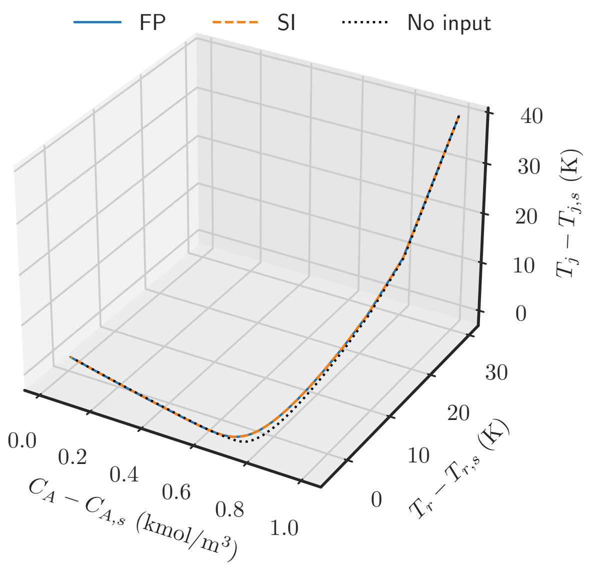


Figure 3.7: State-space profiles for the CSTR in closed-loop under the LMPC utilizing the first-principles model (blue line) and the SI slow model (orange line) and in open-loop with $u = 0$ (black line).

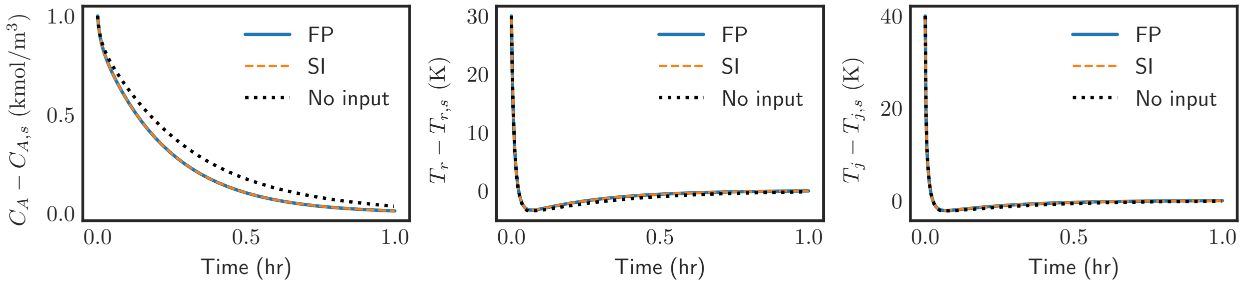


Figure 3.8: State profiles for the CSTR in closed-loop under the LMPC utilizing the first-principles model (blue line) and the SI slow model (orange line) and in open-loop with $u = 0$ (black line).

3.6.4 Effect of process model selection on computational time and maximum allowable prediction horizon length

The choice of the process model for the LMPC has a significant effect on the total as well as per iteration computational time of the MPC, directly limiting the maximum prediction horizon length that can be implemented. Due to the increased complexity of the first-principles process model, the computational time required to solve the LMPC optimization problem for the LMPC

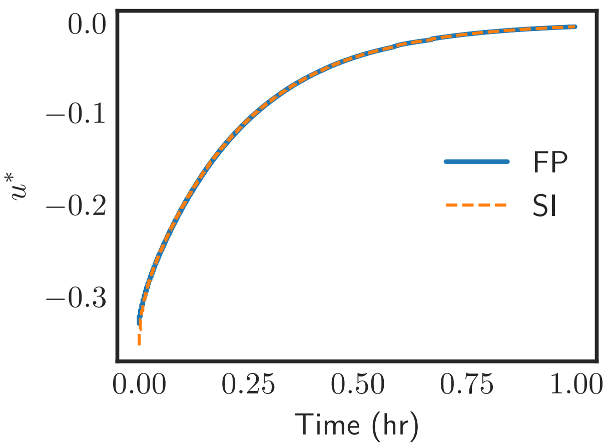


Figure 3.9: Input profiles for the CSTR in closed-loop under the LMPC utilizing the first-principles model (blue line) and the SI slow model (orange line).

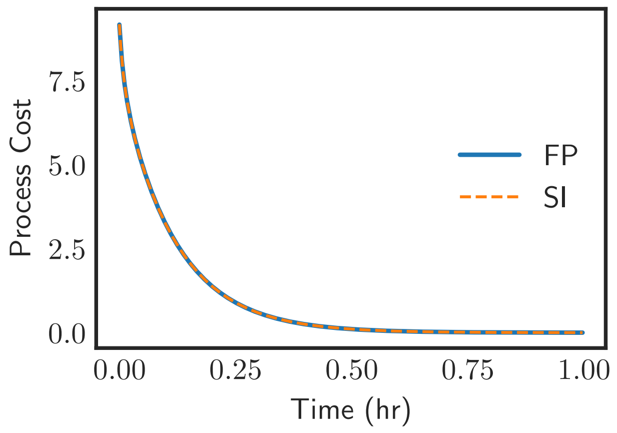


Figure 3.10: Cost function of LMPC for the CSTR in closed-loop under the LMPC utilizing the first-principles model (blue line) and the SI slow model (orange line).

incorporating the first-principles slow model is higher for any given prediction horizon length. Consequently, the longest prediction horizon that can be implemented in the controller is lower for the LMPC with the first-principles model, leading to a higher total of the integral of the LMPC cost function over the simulation period.

Figures 3.11 and 3.12 depict the iteration time required for each of the 360 LMPC steps overs the entire simulation period of 3600 s with a sampling period Δ of 10 s for the LMPC under the first-principles model and the SI slow model, respectively, while using the initial conditions,

$(C_{\mathrm{A}} - C_{\mathrm{A},s}, T_r - T_{r,s}, T_j - T_{j,s}) = (1\,\mathrm{kmol\,m^{-3}}, 30\,\mathrm{K}, 40\,\mathrm{K})$. Practically, since the optimization problem is solved in every sampling period Δ, the computational time for a single iteration cannot exceed Δ. As Δ is chosen to be 10 s in this application, for a feasible N, no iteration time can exceed 10 s. From Figure 3.11, it is observed that, for the LMPC utilizing the first-principles slow model, the maximum allowable prediction horizon length is $N = 16$. Numerically, it is confirmed that, in this case, the longest iteration times when $N = 16$ and $N = 17$ are 9.59 s and 10.85 s, respectively. In contrast, for the LMPC with the SI slow model, the maximum allowable prediction horizon length is $N = 24$ as seen in Figure 3.12. For this controller, the longest iteration times for prediction horizon lengths of $N = 24$ and $N = 25$ were calculated to be 9.83 s and 13.6 s, respectively. Finally, the time-integral of the LMPC cost function $\int_{t=0}^{t_p} L(x(\tau), u(\tau))\, \mathrm{d}\tau$ is calculated over the simulation period $t_p = 1\,\mathrm{hr}$ for the controller utilizing the first-principles model with $N = 16$ and the the controller incorporating the SI slow model with $N = 24$. It is found that the costs are $L = 243.8$ and $L = 225.9$ for the first-principles model-based controller and SI-based controller, respectively. This implies that the LMPC based on the SI slow model, when maximizing the prediction horizon length to $N = 24$, outperforms the LMPC based on the first-principles slow model with its prediction horizon length maximized at $N = 16$, in terms of lower energy and faster convergence to the origin. This fact can also be observed from the state and input profiles, shown in Figures 3.13 and 3.14, respectively. The LMPC based on the SI slow model takes more aggressive control action in the earlier MPC steps as seen in Figure 3.13, causing the state to converge slightly faster to the origin as depicted in Figure 3.14. The maximum allowable prediction horizon lengths and their corresponding costs are summarized in Table 3.2.

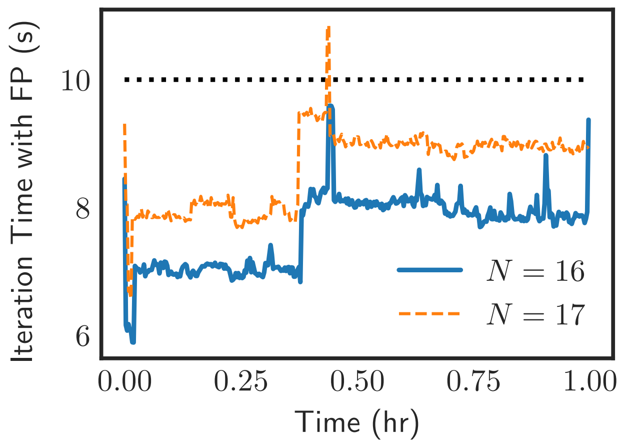


Figure 3.11: Iteration time of each MPC step for prediction horizon lengths of $N = 16$ (blue line) and $N = 17$ (orange line) when the LMPC utilizes the first-principles process model, where the black dotted line represents the sampling time $\Delta = 10$ s.

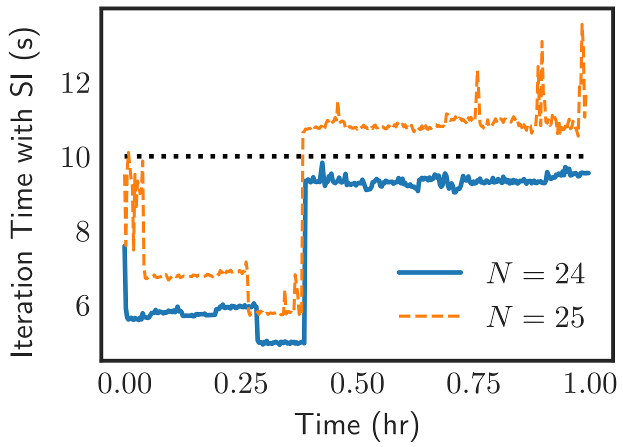


Figure 3.12: Iteration time of each MPC step for prediction horizon lengths of $N = 23$ (blue line) and $N = 24$ (orange line) when the LMPC utilizes the sparse-identified slow model, where the black dotted line represents the sampling time $\Delta = 10$ s.

Table 3.2: Maximum allowable prediction horizon length and corresponding integral of cost function over simulation period t_p.

LMPC Process Model	Max allowable N	Integral of Cost Function
First-principles slow model	16	243.8
Sparse-identified slow model	24	225.9

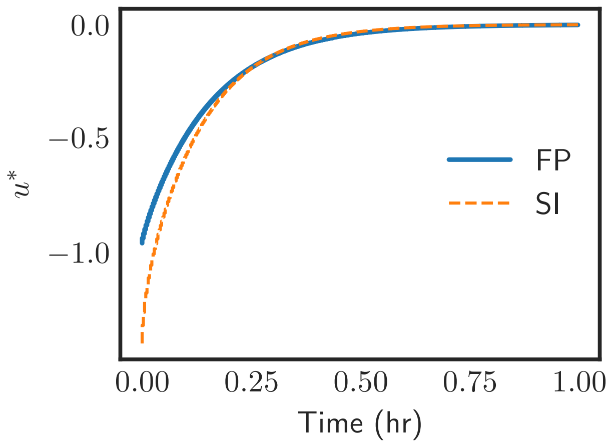


Figure 3.13: Input profiles for the CSTR in closed-loop under the LMPC utilizing the first-principles model with $N = 16$ (blue line) and the SI slow model with $N = 24$ (orange line).

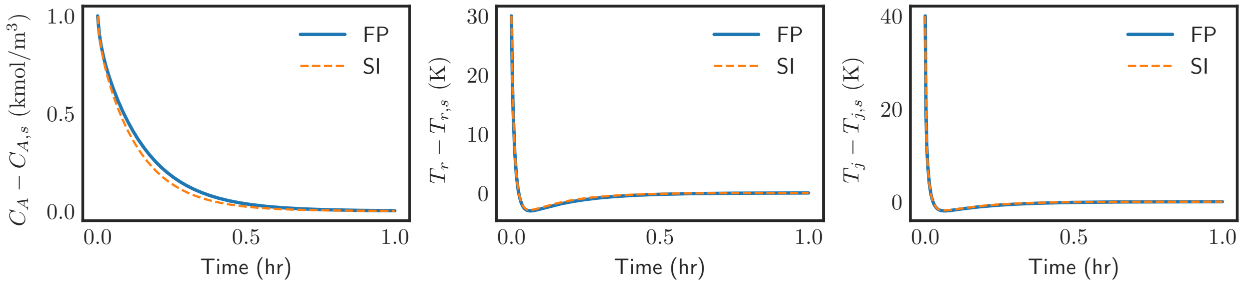


Figure 3.14: State profiles for the CSTR in closed-loop under the LMPC utilizing the first-principles model with $N = 16$ (blue line) and the SI slow model with $N = 24$ (orange line).

3.7 Conclusion

This article focused on the design of a Lyapunov-based MPC for a class of nonlinear singularly perturbed systems using only measurement data from processes. In singularly perturbed systems, due to the presence of time-scale multiplicities, a direct application of MPC without accounting for the evolution of the states in different time scales can lead to closed-loop performance deterioration or even closed-loop instability due to controller ill-conditioning. Hence, we proposed a method to first separate the slow and fast variables in the system and then design the MPC based on the reduced-order slow subsystem. Furthermore, due to the lack of a first-principles model in most practical applications, our method used only sampled experimental/industrial simulation data to reconstruct the reduced slow subsystem via a machine-learning method known as sparse identification. Subsequently, the theory was developed by deriving sufficient conditions for closed-loop stability under sample-and-hold implementation. Finally, the proposed LMPC design was applied to a non-isothermal reactor that exhibited time-scale separation. It was observed that the controllers yielded nearly identical performance for the same controller parameters. However, the LMPC based on the sparse-identified slow subsystem could implement superior controller parameters, such as a longer prediction horizon, due to its reduced complexity and, hence, lower computational time. As a result, the SI based LMPC outperformed the LMPC utilizing the first-principles model when the superior parameters were used for the former controller, demonstrating the practicality and benefits of designing MPC by reconstructing the reduced slow subsystem from measurement data.

Chapter 4

Handling noisy data in sparse model identification using subsampling

4.1 Introduction

Historically, a key focus of research in the fields of science and engineering has been the discovery of physical laws in the form of governing equations. In recent years, however, the focus has shifted to data-driven discovery of these laws. These fundamental laws are often in the form of dynamical models *i.e.,* ordinary differential equations (ODE) or partial differential equations (PDE) in time. Examples include the Maxwell equations from electromagnetism, Boltzmann equation from thermodynamics, Navier-Stokes equations from momentum transfer, Black-Scholes equation from finance, and predator-prey equations from biology [7]. In other words, the laws are discovered as time-series predictive models, which are a necessary building block in many engineering applications, from predictive maintenance in industrial engineering to advanced control systems such as model predictive control (MPC) that are widely found in chemical process systems. MPC requires a dynamical model to predict the states and/or outputs of the process over a prediction horizon. As a result, considerable work on data-driven modeling can be found in the context of MPC [135, 146–155]. Some of the data-driven system identification methods developed and investigated in the recent literature include singular value decomposition [156], Numerical algorithms for Subspace State Space System Identification (N4SID) [30], and auto-regressive models with exogenous inputs (ARX/ARARX) [157–159]. Due to the exponential increase in computational power over the past decade, machine learning methods, a subset of data-driven modeling methods, have produced remarkable results when utilized to their fullest extents due to their ability to capture complex, interacting nonlinearities by tuning numerous hyper-parameters [53, 160–162]. Machine learning methods are a broad class of data-driven modeling methods including linear regression, support vector machines, deep neural networks, sparse identification, *etc.* [77]. The primary method investigated and improved in this article is sparse identification. While sparse identification has already been developed in many facets [81, 113, 163–170], the aspect of noisy data remains largely a challenge for the method.

Since machine learning methods have mostly been developed in the domain of computer science, they often either assume the availability of high-fidelity data or use the term "noisy data" to refer to mislabels in classification problems rather than numerical inaccuracies in regression problems [171]. Hence, the reported accuracy and success of these methods stem from application of the algorithms to standard, clean data sets. However, in the field of engineering, particularly chemical engineering, the availability of noise-free data remains a challenge. Most engineering systems contain at least sensor and measurement noise, if not disturbances. Consequently, when machine learning methods are applied to noisy data from process engineering systems, the results may be unexpectedly inferior. To mitigate the subpar performance of traditional system identification methods when using noisy data, a significant amount of effort surfaces when the recent literature is reviewed. Examples include the extension of the ARX and ARARX methods to input/output data with additive white noise [172], estimation of the noise term using methods involving principal component analysis [173], applying subspace identification methods to closed-loop operation data [174], and the Kalman filter for linear dynamical systems with Gaussian white noise [175, 176]. For nonlinear systems, methods investigated in the literature include the extended Kalman filter and moving horizon estimation [175]. However, the methods described incorporate numerous assumptions regarding the structure of the system and noise distribution, limiting their applicability to industry [175]. As a result, the field of data-driven process modeling for dynamical systems using noisy sensor data requires further innovation and improvement to be implemented in practice.

Although the initial paper, [77], remarked that the total-variation regularized derivative is robust to noise, a deeper investigation into the levels of noise as well as the data generation and sampling details reveals that the claim cannot be broadened to every case, or even many practical cases. Under high levels of noise, the calculation of the model parameters in sparse identification suffers greatly. This is due to the dynamics possibly being sparse only in a non-orthonormal basis, such as monomials, or due to the sampling of the variables following the system dynamics instead of being random or experimentally designed, which can lead to ill-conditioning measurement data matrices [177]. Due to these two issues in the measurement data matrix, the matrix may generally violate the incoherence property [100–102] or the restricted isometry property [103–105] in the underdetermined problem, or not satisfy the incoherence property [178, 179] in the overdetermined problem. Therefore, a number of investigations have been carried out to explicitly detail the impact of noise on sparse identification, and several extensions have been proposed to the original sparse identification algorithm to account for noise and other practical concerns. In [106], the issues of both noise and irregularly sampled data are addressed by using an assimilation step with an autoencoder or ensemble Kalman filter before the model-identification step to use recovered states in the identification step. Partially available data and noise have also been investigated in [107], although the noise levels considered are relatively low with a signal-to-noise ratio (SNR) of 22.33. In [78], the discrepancy between the true governing equations and the sparse-identified models due to noise is addressed by exploiting group sparsity, where partial knowledge of the underlying physics is used to group terms together and ensure all or none of the terms appear in the model. [78] used a smoothed finite-difference approach using the Savitzky-Golay filter, as will be used in this paper, and deeply studied the effect of most of the parameters and possible variables in the derivative estimation step. [78] provides the highest level of detail regarding the numerical differentiation procedure used in this paper. In summary, it was shown that smoothing the data generally improves performance, changing the type of finite-difference or smoothing weakly affect the results at sufficiently high noise levels, and that the default window used in the filter is appropriate for highly noisy data. These results further demonstrate the necessity to develop novel methods to extend sparse identification to the case of highly noisy data. [117] proves that when sparse identification is used to update an existing model in real-time by using only new data as it becomes available, the algorithm is less susceptible to noise than re-identifying a new model. The focus of [117] was on real-time model updates, which are highly applicable to many practical problems. Hence, the first and perhaps most important step of identifying a model from the bulk of the raw data collected did not include any noise. Hence, the derivative estimation was carried out using a simple first-order forward finite-difference routine, which causes no significant inaccuracies or even numerical instabilities. However, the effect of noise was studied in the sense that noise was added to the new data as it becomes available, which were used to update the model in real-time. However, as shown in their results, the amount of new data collected to update a model is much less than the amount of total data used to initially identify an accurate model. Furthermore, in this step, the noise was added directly to the derivative rather than using the forward finite-difference scheme used in the main data set. Due to these two factors, it is ambiguous what the effect of the new noisy data would be if a finite-difference method was used to compute the derivatives, or if the original data itself was corrupted with noise. [108] uses an "algebraic operation", specifically Laplace transforms and inverse Laplace transforms, to reformulate the ODE model using integral terms, which mitigates noise in successive operations. In [109], the computation of the second derivative in mass-spring systems is avoided by using the Duhamel's integral, while further de-noising is proposed by means of the RKHS (Reproducing Kernel Hilbert Space)-based non-parametric de-noise method. Sparse identification has also been assisted by the manifold boundary approximation method in [110] to extend it to power systems, which involve differential algebraic equations (DAE), stiff ODEs, and low measurement noise. Although the application was to PDEs rather than ODEs, [112] studied a large number of example systems using sparse identification and employed spectral methods/filtering [111] to estimate the derivatives.

To deal with implicit ODEs as well as noise, sparse identification was extended in [113]. However, the method suffered from an explosive increase in the sensitivity to noise. This was subsequently addressed and improved in [114] via parallel processing and multiple optimization algorithms, making the new method, SINDy-PI, orders of magnitude more robust to noise than before. However, from the case studies in [114], the improved method still failed at noise levels above variances of 10^{-4}, which is very low for practical purposes. Furthermore, as the method is implicit, the derivative terms and their interactions must all be included in the function library for this method, which can greatly expand the function space. Hence, even more caution is required in building the library when using SINDy-PI. While the increase in computational expense is offset by parallel processing, the additional infrastructure required for parallelization might not be available in many applications.

Two papers, [115] and [116], have pioneered the application of sparse identification to noisy data. In the first, [115] modified the original L_1-regularized sequential least squares algorithm from [77] by instead using a reweighted L_1-regularized sequential least squares algorithm for the optimization and the second-order Tikhonov regularization for the derivative approximation. The regularization parameter was determined using a Pareto curve. The other paper, [116], proposed a subsampling-based threshold sparse Bayesian regression or SubTSBR, where a fraction of the entire data set is randomly subsampled a number of times and the best model selected using a model-selection criterion.

In all of the papers mentioned above, with the exception of two, the derivatives were calculated either using the ODE in the data generation step with noise subsequently added to it, or the derivatives were estimated using numerical methods from the exact data generated and noise was added to both the data and computed derivatives afterwards. The only exceptions were [112] and [115]. In [112], except the final example, the remaining examples were also carried out with noise being added directly to the derivatives rather than the data sets themselves. However, in the final example, it was noted that the underlying assumption of such an estimation methodology is that the derivatives themselves are more corrupted by the noise than the original data, which restricted the scope of application of the method. Therefore, attempting to identify the governing PDEs with only noisy data and estimating derivatives from the data was attempted. It was demonstrated that, even with presmoothing and spectral filtering, the method worked until a noise level of 50$\%$ and failed at 100$\%$, the latter of which is the low level of noise considered in this work. In [115], despite using Tikhonov regularization and reweighted L_1-regularized sequential least squares, the authors showed that the errors in both the derivative approximation and the solution increase rapidly in orders of magnitude when the noise level increases beyond $\sigma = 10^{-2}$, which is also very small for all practical purposes. Moreover, their main results for all their studies are based on data with a SNR of approximately 60. Furthermore, all the papers considered only zero-mean Gaussian white noise. To the best of our knowledge, in the context of sparse identification, the direct impact of noisy industrial data on the derivative computation as well as non-Gaussian noise have not been studied in detail. The method used to approximate the derivatives, as well as the optimizer that is used with the derivative approximator are crucial hyperparameters that require in-depth analysis as will be seen in Section 4.4.

Beyond the domain of sparse identification, another recent method to prevent overfitting in the presence of high noise in measured data that has received significant attention, developed primarily in the context of neural networks, is co-teaching, where noise-free data from first-principles simulations is used to assist the training process [153, 154]. The method proposed in this work combines both subsampling [116] and co-teaching [153, 154] with sparse identification to deal with even higher levels of noise than was previously possible to handle.

The rest of this manuscript is organized as follows: in Section 4.2, the class of nonlinear systems considered and the basic methods combined for the implementation of the proposed novel method are reviewed in brief. In Section 4.3, the novel method combining sparse identification with subsampling and co-teaching is introduced and described in detail. In Section 4.4, the proposed method is applied to a predator-prey example and a chemical reactor example to demonstrate its effectiveness, and the conclusions are summarized in Section 4.5.

4.2 Preliminaries

4.2.1 Class of systems

The class of continuous-time nonlinear systems considered in this work can be written in the form,

$$\dot{x}(t) = f(x(t)), \qquad x(t_0) = x_0 \tag{4.1a}$$

$$y = x + w \tag{4.1b}$$

where $x \in \mathbb{R}^n$ is the state vector, $y \in \mathbb{R}^n$ is the vector of sampled measurements of the states, $w \in \mathbb{R}^n$ is the sensor noise, and the unknown vector function $f(\cdot)$ is the process model representing the inherent physical laws constraining the system. Without loss of generality, the initial time t_0 is taken to be 0 throughout the article.

4.2.2 Sparse identification

Sparse identification is a relatively new method for identifying nonlinear systems based on data. It has been effectively deployed on a wide range of engineering systems of relevance [71–

76, 129]. Provided with only sensor measurements from a system of the form of Eq. (4.1), sparse identification aims to reconstruct the system as a first-order differential equation of the form,

$$\dot{\hat{x}} = \hat{f}(\hat{x}) \tag{4.2}$$

where $\hat{x} \in \mathbb{R}^n$ is the vector of state of the sparse-identified model $\hat{f}(\cdot)$.

The central idea of sparse identification is to consider many possible nonlinear terms for the right-hand side of Eq. (4.2), $\hat{f}$, and subsequently identify the few active terms in $\hat{f}$. This apparent simplification stems from the fact that physical systems in practice only contain a small number of nonzero terms when considering a large set of terms *i.e.,* candidate basis functions. Hence, the space of all nonlinear basis functions considered is sparse, and efficient algorithms may be used to compute the pre-multiplying coefficients. To carry out sparse identification we first sample and collect a set of measurements of the states $x_1, x_2, \ldots, x_n$ at times $t_1, t_2, \ldots, t_m$ from open-loop simulations and concatenate them into the data matrix X,

$$X = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix} \tag{4.3}$$

where $x_i(t_j)$ denotes the measurement of state i at the j-th sampling time with $i = 1, \ldots, n$ and $j = 1, \ldots, m$. The time-derivative of X is represented by $\dot{X}$ and is a required quantity in the sparse identification procedure. However, in the presence of measurement noise, the estimation of the derivative is a challenge. Although some methods to robustly approximate the derivative are detailed in [77], particularly the total-variation regularized derivative from [141, 180], the results in this paper disagree in terms of its robustness and show that a smoothed finite difference can often yield better results. After acquiring X and $\dot{X}$, we build a function library, $\Theta(X)$, containing p candidate nonlinear functions of X corresponding to the p basis functions that may be active or inactive in the right-hand side of the ODE, $\hat{f}$. The sparse identification technique leverages sparsity to identify the active terms in this library, Θ. The augmented library, $\Theta(X)$, is optimized like a hyperparameter in this paper. Specifically, it is constructed using either only monomials up to third-order including interaction terms or a combination of monomials, their interactions, and the four common trigonometric functions: *sin*, *cos*, *tan*, and *tanh*. Hence, the latter library containing all possible terms is of the form,

$$\Theta(X) = \begin{bmatrix} | & | & | & | & | & | & | & | \\ \mathbf{1} & X & X^{P_2} & X^{P_3} & \sin X & \cos X & \tan X & \tanh X \\ | & | & | & | & | & | & | & | \end{bmatrix} \tag{4.4}$$

In Eq. (4.4), X^{P_2} represents all quadratic nonlinearities:

$$X^{P_2} = \begin{bmatrix} x_1^2 & x_1x_2 & \cdots & x_2^2 & x_2x_3 & \cdots & x_n^2 \end{bmatrix} \tag{4.5}$$

The aforementioned choice of candidate basis functions is rooted in the observation that polynomials and trigonometric functions can be found in the natural laws governing many known physical systems [77].

The objective of the sparse identification algorithm is to find the p coefficients that pre-multiply the p nonlinear basis functions considered in $\Theta(X)$ for each state x_i. Each x_i is associated with a corresponding sparse vector of coefficients, $\xi_i \in \mathbb{R}^p$, that characterize the nonzero terms in the respective ODE, $\dot{x}_i = f_i(x)$. Hence, n such coefficient vectors must be computed. In matrix notation, the quantity to be found is

$$\Xi = \begin{bmatrix} \xi_1 & \xi_2 & \cdots & \xi_n \end{bmatrix} \tag{4.6}$$

Therefore, to determine the matrix Ξ, the following equation needs to be solved:

$$\dot{X} = \Theta(X)\Xi \tag{4.7}$$

Equation (4.7) is solved using sequential least-squares by zeroing all coefficients in Ξ that are smaller than a threshold λ, known as the sparsification knob, and repeatedly solving the resulting equation with zeroed terms until the non-zero coefficients converge. Due to the sparse structure of Ξ, convergence of the iterative step is rapid. Once Ξ is calculated, the overall model can be written as the continuous-time differential equation,

$$\dot{x} = \Xi^\top (\Theta(x^\top))^\top$$

where $\Theta(x^\top)$ is a column vector of symbolic functions of x from the function library, and $x^\top$ denotes the transpose of x.

4.2.3 Subsampling

Subsampling is a statistical technique where a subset of the entire data set is randomly selected for analysis instead of the entire data set. While the method is typically employed to estimate statistical metrics [181] or speed up an algorithm [182], the objective of subsampling in this paper is to increase the modeling accuracy in the presence of high levels of sensor noise. Specifically, in regression, when the number of data points, m, is more than the number of unknown weights, as is the case in most practical problems, the regression might be carried out with only a subset of the entire data set to estimate the regression parameters. This may be considered because a fraction of the data points are corrupted by high noise levels or are outliers. Traditional regression methods like least squares use the entire data set by making the smoothing assumption, which states that the larger proportion of low-noise or "good" values will sufficiently smooth out any large noise in the data set. However, in practice, this assumption may not hold in the presence of either very high noise or consistently high noise throughout the data set as the fraction of "good" data points is inadequate to compensate the noise in these cases. The idea behind subsampling is to mitigate the problem of data points with large noise by randomly subsampling to eliminate them before carrying out the regression or model identification step.

4.2.4 Co-teaching

Co-teaching is a technique that improves the accuracy of a machine learning method by utilizing noise-free data sets from first-principles models and simulations of a physical process. Co-teaching originated in the field of classification problems, particularly image classification, where neural networks are abundantly used to classify images into user-defined categories. However, a fraction of the images can be miscategorized, which drastically reduces the data quality and, consequently, neural network accuracy if used without accounting for the mislabeled data samples. Such mislabels are termed "noisy labels" in machine learning.

Although the co-teaching method has primarily been developed in the context of classification problems and deep neural networks, there is nascent research into extending this method to regression problems using long short-term memory (LSTM) networks [153, 154]. Specifically, in [154], co-teaching was compared to a Monte-Carlo dropout LSTM network, which is an advanced form of the standard dropout neural network where weights are randomly omitted during the training process to improve generalization and minimize overfitting. Dropout layers may be considered as a regularization term in the context of neural networks. Carrying out regression on a subsampled fraction of a data set repeatedly is analogous to a standard dropout neural network where the omission of the weights occurs only in the training and not the testing phase. In contrast to dropout neural networks and subsampling, the key proposition of co-teaching is that the loss function value is significantly lower for noise-free data as compared to noisy data. Therefore, mixing any proportion of noise-free data with measured noisy data can guide the model training process to be more robust to noise and overfitting. This assertion was proven in [154], where the co-teaching LSTM outperformed even the Monte-Carlo dropout LSTM in terms of lower open-loop mean-square error as well as computational time (without parallel processing in either case).

4.3 Subsampling-based sparse identification with co-teaching

In this paper, the original sparse identification method is improved by combining it with subsampling and co-teaching as described in Section 4.2. The proposed method, Sparse Identification with Subsampling and Co-teaching (SISC), aims to mitigate the problem of highly noisy measurement data. When the measured data is either too noisy or consistently noisy, subsampling alone may not be sufficient as it will merely randomly choose the least noisy data in the subsamples, which can still be heavily corrupted by noise in this case, or a very large number of subsamples may be required to successfully isolate a relatively low-noise subsample, which can lead to an excessive computational expense. Therefore, in this paper, we also add noise-free data into each subsample to further improve the identification procedure.

In SISC (Algorithm 1), a subset of the total data set is randomly selected and mixed with noise-free data from first-principles simulations. The mixed data subset is used to estimate the model parameters. The resulting model is then evaluated using a model-selection criterion. After repeating the above steps for each subsample, the best model is selected to be the model with the lowest value of the model-selection criterion. Specifically, the algorithm is initiated with three user-defined parameters: the subsampling fraction $p \in (0, 1)$, the noise-free subsampling fraction $q \in (0, 1)$, and the number of times $L \geq 1$ to subsample and identify a model. For each of the L subsamples, a fraction containing $p \times m$ randomly selected data points from the original data set is mixed with $q \times m$ randomly selected data points from the noise-free data set. The resulting data submatrix for subsample i, where $i = 1, 2, \ldots, L$, is of the form

$$X_i = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1\left(t_{p\times m+q\times m}\right) & x_2\left(t_{p\times m+q\times m}\right) & \cdots & x_n\left(t_{p\times m+q\times m}\right) \end{bmatrix} \tag{4.8}$$

The equation to be solved for each subsample is, instead of Equation (4.7),

$$\dot{X}_i = \Theta(X_i)\Xi_i \tag{4.9}$$

where Ξ_i is the coefficient matrix that needs to be computed. Each computed Ξ_i yields an ODE model,

$$\dot{x} = \Xi_i^\top (\Theta(x^\top))^\top \tag{4.10}$$

which is simulated and evaluated using the model-selection criterion against a portion of the data that is reserved for validation. In this work, the Akaike Information Criterion (AIC) is used as the model-selection criterion since it is calculated using not only the mean-squared error (MSE) but also the number of terms in the model, promoting sparsity by penalizing models with a higher number of terms. The AIC is defined in [183] as

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}\big(x(t_i) - \hat{x}(t_i)\big)^2 \tag{4.11}$$

$$\mathrm{AIC} = m \log \mathrm{MSE} + 2L_0 \tag{4.12}$$

where L_0 is the 0-th norm, which is equal to the number of non-zero terms in the sparse-identified model.

The hyperparameters to be tuned in the algorithm include the function library to be constructed, the method chosen to compute the derivative from noisy data, the optimizer, and the value of the sparsification knob λ. The function library contains either only polynomial functions or both polynomial and trignonometric functions. The derivatives are approximated using either the total variation regularized derivative (TVRD) or smoothed finite-difference (SFD) after presmoothing with the Savitzky-Golay filter. The optimization is carried out using sequential thresholded least squares (STLSQ) or SR3, which is an enhancement of the least absolute shrinkage and selection operator (LASSO) [184]. To find a suitable value of λ, values of 0.1, 0.2, and 0.3 were tested. A detailed justification for the choices of λ is given in Section 4.4.2.1. Once all the hyperparameters have been tuned, a model is selected based on the AIC value of the validation set. Finally, out of the L models obtained from L subsamples after tuning all hyperparameters, the sparse-identified model with the lowest value of the AIC is taken to be the optimal model. The flow of the data through the algorithm is depicted in Fig. 4.1.

Algorithm 1: Sparse Identification with Subsampling and Co-teaching (SISC)

```
Input: $X, \Theta, p, q, L$
Output: $\Xi$
compute exact $\dot{X}$ of noise-free data using first-principles ODE model;
compute estimate of $\dot{X}$ of noisy data using SFD or TVRD;
for $i \leftarrow 1$ to $L$ do
    randomly sample $p$ fraction of industrial (noisy) data and its estimated derivative;
    randomly sample $q$ fraction of first-principles simulation (noise-free) data and its exact
      derivative;
    mix and concatenate the data into $X_i$ and $\dot{X}_i$;
    calculate function library $\Theta(X_i)$ using only polynomial functions or a combination of
      polynomial and trignonometric functions;
    solve $\dot{X}_i = \Theta(X_i)\Xi_i$ using STLSQ or SR3 with different values of $\lambda$ to get $\Xi_i$;
    with calculated $\Xi_i$, integrate ODE and compute AIC;
end
Let final $R = \arg\min_i\{\text{AIC}_i\}$;
Let $\Xi = \Xi_R$.
```

Remark 4.1. *Both of the derivative approximation algorithms utilize certain parameters that may be tuned as well. For the Savitzky-Golay filter in SFD, the default values of a window length of 11 and cubic polynomials for curve-fitting were retained as they were satisfactory. For the TVRD, the regularization term was set to 0.001 instead of the default 0.01 and the maximum number of iterations with increased to 1000 to ensure convergence. Further tuning was infeasible due to the large number of hyperparameters to be trained in the main algorithm.*

Remark 4.2. *It is noted that, due to the issues of noise and sparse identification, some works have*

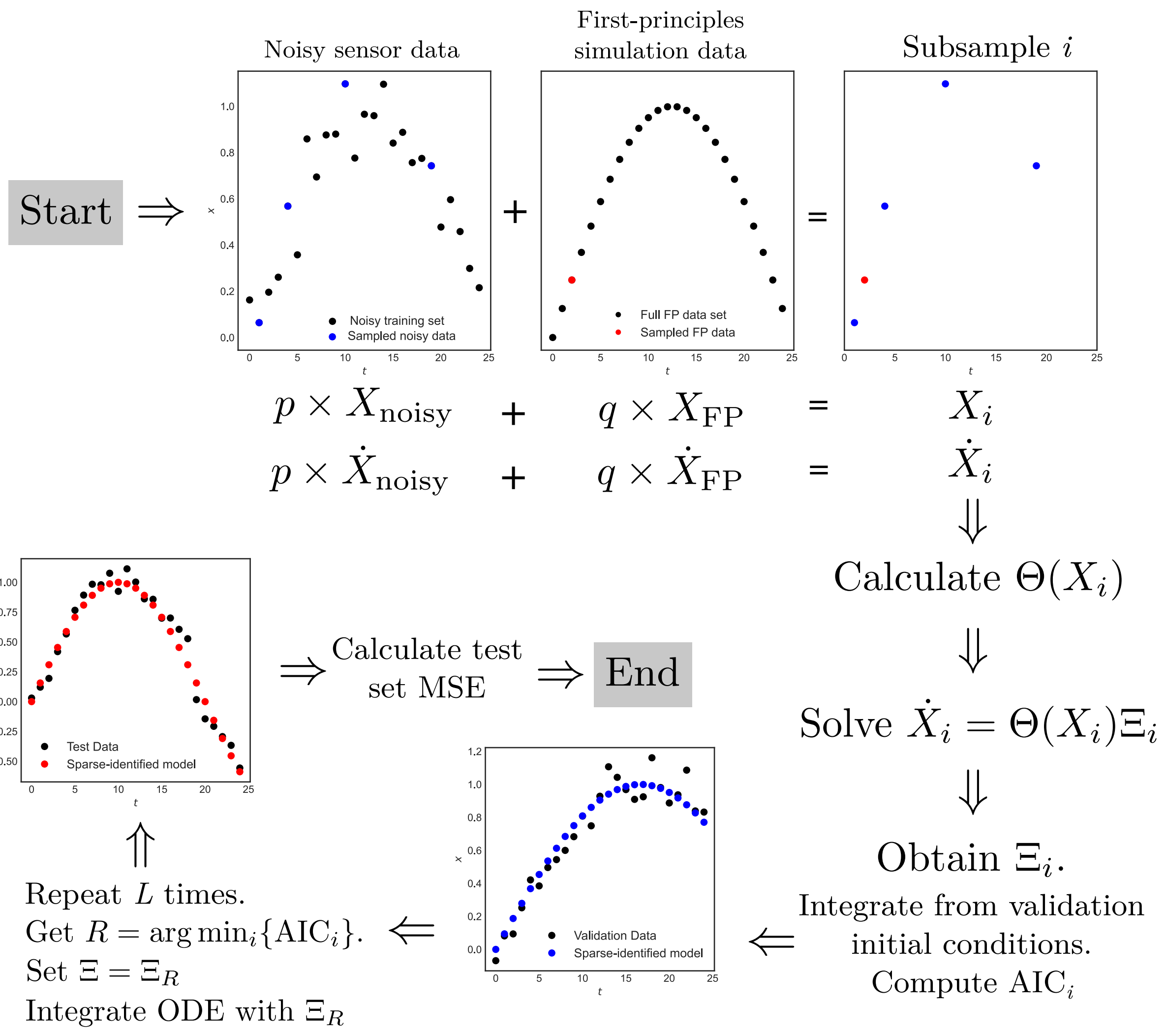

Start
Noisy sensor data
Noisy training set
Sampled noisy data
First-principles simulation data
Full FP data set
Sampled FP data
Subsample i
$p \times X_{\text{noisy}} + q \times X_{\text{FP}} = X_i$
$p \times \dot{X}_{\text{noisy}} + q \times \dot{X}_{\text{FP}} = \dot{X}_i$
Calculate $\Theta(X_i)$
Solve $\dot{X}_i = \Theta(X_i)\Xi_i$
Obtain Ξ_i.
Integrate from validation initial conditions.
Compute AIC_i
Validation Data
Sparse-identified model
Repeat L times.
Get $R = \arg\min_i\{\text{AIC}_i\}$.
Set $\Xi = \Xi_R$
Integrate ODE with Ξ_R
Test Data
Sparse-identified model
Calculate test set MSE
End

Figure 4.1: Data flow diagram.

attempted to find alternatives to explanatory dynamical modeling instead. Such methods include black-box modeling using Runge-Kutta time-steppers embedding neural networks to account for nonlinearities [185–188]. It is noted that these papers propose alternative methods to nonlinear dynamical modeling rather than developing sparse identification further. Consequently, their techniques of dealing with noisy data as well as the results obtained in this context cannot be directly inferred to sparse identification, where noisy data directly and heavily affects the derivative estimation. For example, in [187], neural network function approximators are used in combination with classical linear multi-step time-stepping schemes (such as Runge-Kutta) to identify the nonlinear dynamic constraints in the right-hand-side of an ODE model, similar to the problem to be solved in sparse identification. However, when the effect of specifically the noise on the results is analyzed, the pattern is initially unexpected. As the noise level in the input data is increased, the results become more accurate and then less accurate. This was explained by neural networks, in particular, benefiting from small noise levels in the input, which work as regularization. At even higher noise levels, however, the results deteriorate due to the model not being able to disambiguate the noisy dynamics from the ground truth. Clearly, this is not the case for sparse identification and most system identification methods, where overfitting is not a common issue to the extent that data needs to be intentionally corrupted with noise to increase a model's generalization capabilities. [188] presents a similar but slightly more advanced work, where the data is decomposed into separate true dynamics and noisy dynamics, each of which are separately modeled and predicted when forecasting. The modeling is similar to [187] in terms of combining neural networks with time-stepping schemes, although constraints are added, making the overall problem more complex. The effect of noise and general pitfalls of neural networks are discussed *near the end of their work, where, for example, the point is made that a neural network is, at its core, an interpolation technique. Hence, their proposed methods work best with systems where the dynamics evolve on an attractor, especially if it can be sampled with a small sampling time. The generated models also predict the trajectories to remain on the attractor for long simulation durations. If their method is used to integrate from new initial conditions or further from the attractor, the models may be insufficient. This is particularly important in the context of control, where they suggest collecting dynamic, transient data further from the attractor as perturbations and actuators must be considered. [189] reformulates the ODE model using integral terms, which can be approximated using piecewise constant quadrature, and the resulting equations efficiently solved using the Douglas-Rachford algorithm. However, [189] mostly restricts its candidate functions for the model to polynomial terms and further notes that their method works better on fact and/or chaotic manifolds rather than slower, equilibrium behavior, which is more relevant in chemical process systems. A recent, novel method that has shown promise, especially when the data is corrupted by high noise or outliers, is entropic regression [190]. We remark that these methods may be used instead of or in combination with our method.*

Remark 4.3. *While the focus of this work is the modeling of nonlinear dynamical systems, the ultimate objective is to incorporate the developed models into model predictive control (MPC) as the process model, which will be explored in a future work. Real-time optimization has been studied in several recent papers. [191] incorporated feedforward neural networks and first-principles process models into real-time optimization RTO and MPC. In the event of an increase in energy prices or in the presence of a large error in the feed concentration, it was shown that the perfor-*

mance of the controller with RTO outperforms the one without RTO. In [67], a recurrent neural network (RNN) was used as the process model for a Lyapunov-based model predictive controller (LMPC) and a Lyapunov-based economic model predictive controller (LEMPC). The RNN model was updated in real-time based on event-triggers and/or error-triggers, which improved closed-loop performance in terms of stability, optimality, and smoothness of control actions. Hence, the extension of our proposed modeling approach to include both real-time optimization and model predictive control can be investigated in a future work after first extending it to solely model predictive control.

Remark 4.4. *The proposed method is applicable to any dynamic system where data is obtained as time-series measurements. This includes most chemical processes and plant data. The method would not be directly applicable to data that are in other formats and structures such as images. An example is the sequence of moving digits from the Modified National Institute of Standards and Technology (MNIST) database, also known as the moving MNIST data set, where digits are moving. However, such data sets are mostly relevant in other disciplines of engineering and computer science rather than chemical engineering.*

4.4 Applications

In this work, the effect of Gaussian white noise $w \sim \mathcal{N}(0, \sigma^2)$ in the sensor measurement y in Eq. (4.1) is analyzed. Specifically, the predictive ability of the model developed using SISC is benchmarked against the original sparse identification (SI) method as well as sparse identification with subsampling without co-teaching (SIS). First, the details of the demonstration procedure are outlined. Subsequently, the two examples, a predator-prey system and a chemical reactor example, are individually studied.

4.4.1 Data generation, sampling, and noise generation

The data generation procedure for both examples consists of first integrating the corresponding system of ODEs using 50 different initial conditions over a wide range of values to maximize coverage of the operating region of interest. The integration is carried out using the explicit Euler method with an integration time step of $h_c = 10^{-4}$, while the sampling time Δ varies between the examples. This yields the noise-free or clean data set.

For both examples, four levels of noise are considered in this study. Noise levels 1 (very low), 2 (low), 3 (medium), and 4 (high) refer to white Gaussian noise with standard deviations $\sigma_1 = 0.02,\ \sigma_2 = 0.1,\ \sigma_3 = 0.2$ and $\sigma_4 = 0.3$, respectively. For each level, white Gaussian noise is generated and amplified as necessary to account for any disparate scales of variables, and then added to the clean data set to generate the noisy data set.

Once the 50 different state trajectories are obtained from the open-loop simulations described above, the data set is split into 60% training, 20% validation, and 20% testing data, respectively. The train-validation-test split of 60-20-20 used is highly subjective, and many variations exist. Other common choices include 50-25-25, 70-15-15, and 80-10-10. There is no “right ratio” and tradeoffs exist. If the training proportion is increased, the model is able to use more data and possibly generalize better. A larger validation set provides greater confidence in the model selection process, while a larger test set produces a better assessment of the ability of the model to generalize to new, unseen data. The 4 combinations suggested above attempt to balance these three objectives. However, as mentioned, these are not rigid rules and should be adjusted as required. For example, if data is scarce due to expensive experimental trials required to collect data, as is the case in several chemical processes, the smaller data set may warrant a larger training set to maximize the model performance. In this scenario, the validation data set can be reduced and a k-fold cross-validation technique may be used instead of information criteria. In fact, due to the lower number of data, the most expensive yet accurate validation technique, leave-one-out-cross-validation (LOOCV) can be used even though it is rarely used in practice for larger data sets due to the computational expense. For this scenario, an 80-10-10 scheme may be preferable. If the data set is extremely large, however, the preferred scheme may be 50-25-25 since there is sufficient data for training even with only 50% of the data being used. Moreover, as the training process is likely to be the most computationally expensive step, a smaller data set for training may also produce models faster. This strategy is common in training neural networks, where "minibatches" are used rather than the entire training data set.

For the base case with no subsampling, the entire training set is used for sparse identification to generate a single model following hyperparameters tuning. For both subsampling methods, the number of subsamples L is taken to be 5 in order to keep the computational burden reasonable. Furthermore, the total subsampling fraction, which is equal to p for the SIS case and $p + q$ for the SISC case, remains the same for both the SIS and SISC cases when compared. This is to ensure that both methods use the same total number of data points for training but only differ in the nature of the subsample used. The total subsampling fraction is assigned with the values of 0.2, 0.4, 0.6, and 0.8. The $p : q$ ratio is taken to be 4:1 throughout this paper as the goal is to demonstrate that only a small fraction of noise-free data is sufficient to improve the performance when the SI and SIS methods fail. For example, a total subsampling fraction of 0.2 indicates $p = 0.2$ for SIS and $p = 0.16, q = 0.04$ for SISC. Hence, for the SIS case, 20% of the training data set will be subsampled $L = 5$ times and a sparse-identified model will be trained for each subsample. For the SISC case, each of $L = 5$ times, 16% of the noisy training data set will be subsampled and then mixed with 4% of the noise-free training data set to yield a subsample that contains the same number of data points as the SIS method. Once the subsamples are extracted, a model is identified with sparse identification for each by tuning the hyperparameters and the best model based on the AIC is isolated.

Remark 4.5. *The sampling time in the data generation step greatly affects the results of any model identification method because information is inevitably lost in the sampling step. A smaller sampling time will generally yield better results due to the more accurate derivative computations and a more comprehensive history of the trajectory of the state over the simulation duration. However, a very small sampling time such as 10^{-4} or 10^{-6} may be infeasible in practice. Hence, even if such a sampling time produces extremely accurate models, the variables, especially in a chemical process, can rarely be measured with such short sampling periods between each measurement. Hence, the values of the sampling period in this paper are of the order of 10^{-2}, which balances practicality with accuracy. This constitutes another limitation of past studies that primarily focus on high-fidelity data with impractically small sampling times to identify models with a very high accuracy that may not be immediately transferable to process engineering.*

Remark 4.6. *Apart from the sampling time and range of initial conditions chosen, the amount of data generated is also dependent on the simulation duration t_f. In case of a system with a* *steady-state solution, the simulation duration should be long enough for the system to reach the steady-state, ensuring that the same steady-state is reached for every initial condition. However, once the steady-state is reached, continuing the simulation for much longer is redundant as there are no new dynamics to be captured. Instead, focus can be shifted to regions of the trajectories with higher information density such as those with higher gradients and faster dynamics. For example, in the case of multiscale systems, [98] suggests burst sampling—a sampling technique where the sampling time is as short as possible in the region of change of the fast subsystem before it converges to a slow manifold, and is much larger for the rest of the simulation duration since the dynamics of the slow subsystem can be captured with significantly fewer data points. Advanced sampling techniques such as burst sampling reduce the amount of data storage as well as computational burden and must be considered when generating data from simulations instead of following a simple, iterative procedure for data collection.*

Remark 4.7. *It is difficult to quantify the amount of data collection required to obtain an accurate model in a general manner that is applicable to all systems. However, when investigating the data structure required and attempting to train models for various patterns of generated data, it was observed from extensive simulations and trials that the dynamics covered over the duration of the simulation were key to identifying an accurate model as opposed to merely the amount of data itself or even the sampling time. Hence, a very short period of data collection with an extremely small, possibly physically infeasible, sampling time can yield a very high number of data-points but will likely yield a poor model. Instead, a larger simulation period with a larger sampling time where a fewer total number of data points are collected will yield a superior model. Therefore, the* *emphasis should be not on the amount of data but rather the dynamics captured over the range of time of data collection. Once sufficient dynamic information is captured in the entire data set, the aforementioned rules of determining the train-validation-test split can be used.*

Remark 4.8. *The primary reason that some parameters such as the $p : q$ ratio were fixed as constants instead of tuning by treating as a hyperparameter was due to the large number of hyperparameters to be tuned in this system, and because it was sufficient to demonstrate the results for a low noise-free fraction as this implies that the results will generally improve when the noise-free fraction is increased.*

4.4.2 Example 1: Predator-prey model

The predator-prey system consists of two first-order differential equations describing the dynamics of two species: a predator and a prey. The equations take the form,

$$\dot{x}_1 = a_1 x_1 - a_2 x_1 x_2 \tag{4.13a}$$

$$\dot{x}_2 = a_3 x_1 x_2 - a_4 x_2 \tag{4.13b}$$

where x_1 and x_2 are the population (numbers) of the prey and the predator, respectively, with parameters $a_i \in \mathbb{R}, a_i > 0$ for $i = 1, 2, 3, 4$. In this paper, the specific system considered is

$$\dot{x}_1 = 0.5 x_1 - 1.5 x_1 x_2 \tag{4.14a}$$

$$\dot{x}_2 = x_1 x_2 - 0.5 x_2 \tag{4.14b}$$

The objective is to reconstruct Eq. (4.14) from noisy data using the methods described in the previous section.

For data generation, using 40 out of 50 initial conditions between 0.01 and 2.0 for each variable, the system of Eq. (4.14) is integrated using a time step of $h_c = 10^{-4}$ from $t_0 = 0$ to $t_f = 20$ and sampled with $\Delta = 0.2$ for a total of 100 data points per trajectory. This yields a total of 4000 data points to be split into training and validation sets. For the testing set, the remaining 10 initial conditions are integrated identically as for the training/validation sets except t_f is increased to 30. The goal of extending the simulation period for the testing set beyond the training/validation sets is to gauge the extrapolating capacity of the model since one supposed advantage of reconstructing dynamical models as opposed to black-box models is the ability to extrapolate beyond the training data.

Figure 4.2 shows the results of this system when the various model identification methods are employed. The various identified models are used to predict the states forward in time using the 10 initial conditions from the testing set and compared to the original testing data generated in the data generation step. The results were similar for both variables but more pronounced in x_1. To evaluate the results numerically, the mean-squared error (MSE) is also calculated between the original data and each predicted trajectory over the entire testing period. The MSE are summarized in Table 4.1. At the lowest noise level ($\sigma = 0.02$), the base case achieves a low test set MSE, and accurately models the system as seen in Fig. 4.2. However, further improvement is observed when subsampling with or without co-teaching, with the order of magnitude of the MSE being one order lower than the base case. Although it appears that the SISC method performed slightly inferior to SIS, both MSE are extremely small, and the difference between the two MSEs, 0.00018 and

0.00021, is negligible in this context, especially as also observed in Figs. 4.2 and 4.3. As the noise level is increased to low ($\sigma = 0.1$) and even medium ($\sigma = 0.2$) levels, the results do not change significantly, except all the errors are now higher. At the highest noise level ($\sigma = 0.3$), the base case deteriorates significantly, overpredicting during most of the simulation period. Subsampling without co-teaching is not sufficient, and yields even poorer results than the base case in this example. In contrast, mixing in only 20% of noise-free data to each subsample was sufficient to improve the model with a much lower MSE than the runs with no subsampling or co-teaching. Although specific time instances might have poor results in the case of SISC as well, this is not across most of the simulation period, and the results are generally much more accurate, especially as also proven by the lower MSE. It is noted that the test set MSE values are generally small, and hence, even the percent decreases in the test set MSE between the SIS and SISC cases are more significant than it appears, which confirms the large improvement in accuracy.

When the total subsampling fraction is increased from 0.2 to 0.4, 0.6, and 0.8, it is observed that the test set MSE usually decreases in this example. However, the difference between the MSE from the SIS and SISC methods also begins to decrease. This indicates that subsampling alone without co-teaching can be a large source of improvement in some (but not all) examples. However, there are two concerns to address if such a strategy is adopted. Firstly, the SISC method, when using only 40% of the data ($p+q = 0.4$), surpasses the accuracy of the SIS method under all values of p that are used, even $p = 0.8$ where double the amount of data is being used to identify a model. Secondly, as the proportion used for subsampling increases, the computational expense increases. Hence, the SISC method outperforms the SIS method in the sense that it uses smaller subsamples to identify a model with a lower (or even equal) MSE.

Table 4.1: Test set MSE for the predator-prey system

σ	No subsampling	SIS	SISC
0.02	0.00596	0.00018	0.00021
0.1	0.15925	0.05550	0.03851
0.2	0.34151	0.08472	0.07614
0.3	0.95057	1.70773	0.35808

4.4.2.1 Justification of coarse search for λ

The most important hyperparameter in the sparse identification algorithm is the sparsification knob λ due to its impact on the model as well as the continuous nature of the parameter. It may be considered analogous to the learning rate in neural networks in these aspects. Therefore, it is typically tuned using either a fine search or a coarse-to-fine search. In this work, however, only 3 values were tested. To ensure the coarse search for the optimal λ did not invalidate the results, an investigation into the effect of λ on the results was carried out.

To study the effect of λ only, since the best optimizer, derivative approximator, and function library have already been determined, these parameters can be fixed at their optimal selections and only λ can be varied. This yields the best possible model that can be achieved by finely tuning λ while all other parameters have already been optimized. The results for $\sigma = 0.1$ are shown in Fig. 4.4. A number of observations can be made from Fig. 4.4.

1. Values of λ above 1.0 zero too many terms, possibly all terms, leading to very high errors, with the error remaining constant until $\lambda = 4.0$. Therefore, it is sufficient to only consider values of λ between 0 and 1.

2. For values of λ below 1.0, smaller values generally yield lower values for both the AIC and

the MSE. However, large differences, particular in orders of magnitude, occur at intervals of approximately 0.1. Hence, a few small values of λ space 0.1 units apart can yield a model very close to the best model possible and is sufficient for demonstration and comparison purposes.

3. At the noise level considered in this investigation of $\sigma = 0.1$, our optimal model was found to use the SR3 optimizer with the TVRD for derivative approximation and only a polynomial library. The best λ from our coarse search was 0.2. The final training AIC and test set MSE were -2617.451 and 0.15925, respectively.

4. In contrast, the best λ from the fine search is 0.03, which gives training AIC and test set MSE of -2973.28 and 0.09942, respectively. The decrease in the metrics from the optimal model from the coarse search is not very large. Moreover, the best λ from the fine search was only 0.03, requiring a grid spacing of 0.01 or smaller when searching for λ. The optimal model from the coarse search, however, has a relatively wide range of values over which the same model is obtained. Hence, a larger step can be used when searching for λ.

5. Due to the small differences in the metrics between the two optimal models in Fig. 4.4, it is possible that the optimal models would be reversed if the penalty on the number of terms were to be increased because the model from the coarse search only has 6 terms, while the model from the fine search consists of 15 terms. Specifically, in Eq. (4.12), the first term accounts for the model-fit or accuracy by using the MSE while the second term penalizes the number of terms in the sparse-identified model. The pre-multiplier of “2” taken in Eq. (4.12) can be increased to penalize the number of terms more heavily and compromise on the

accuracy. In fact, if the pre-multiplier on the number of terms is larger than 41.54, the model identified from the coarse search remains as the optimal model. Hence, a more elaborate scheme to select a search method for λ can take into consideration the balance between sparsity and accuracy. If a more sparse and less complex model is desired at the expense of some loss of accuracy, a coarse search may be preferred. If accuracy is crucial, a coarse-to-fine search may be the fastest method since a coarse search will determine the region for the fine search and reduce the number of computations required. In the case study of Fig. 4.4, a coarse search can reveal that values of λ below 0.5 require a finer search.

6. Finally, even the optimal model from the fine search for the base case is inferior in terms of the test set MSE to both the subsampling cases. Hence, even if the subsampling cases used a finer search for λ to yield a superior model, it would only increase the differences in performance. The key findings of this work are the clear, qualitative improvements between the three different methods rather than the exact quantitative improvement from one model to another.

7. As seen in Table 4.2, at the lowest noise level, the models produced by the two searches were identical. This is likely due to the optimal model being almost the exact model required to be found in both cases. Hence, a finer search is not required.

8. From Table 4.2, at the higher noise levels, which are the highlight of this work, it is observed that there is either an insignificant improvement in performance or no improvement at all. Therefore, in this work, a finer search for λ was omitted due to both the extremely heavy computational expense of optimizing such a continuous hyperparameter as well as the

insignificant improvements that can be achieved from such an effort.

Table 4.2: Test set MSE for the predator-prey system using coarse and fine searches for λ in the *No subsampling* case

σ	No subsampling (coarse)	No subsampling (fine)	SIS	SISC
0.02	0.00596	0.00596	0.00018	0.00021
0.1	0.15925	0.09942	0.05550	0.03851
0.2	0.34151	0.61213	0.08472	0.07614
0.3	0.95057	0.92768	1.70773	0.35808

4.4.3 Example 2: CSTR

A chemical process example with noisy sensor data is considered. In particular, a single irreversible second-order exothermic reaction that converts a reactant A to a product B (A $\rightarrow$ B) takes place in a perfectly mixed non-isothermal continuous stirred tank reactor (CSTR) as shown in Fig. 4.5 described by the following set of ODEs:

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = \frac{F}{V}(C_{A0} - C_A) - k_0 \mathrm{e}^{-\frac{E}{RT}} C_A^2 \tag{4.15a}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 \mathrm{e}^{-\frac{E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \tag{4.15b}$$

The states are the reactant concentration C_A and temperature T inside the reactor. The inlet contains pure reactant A with concentration C_{A0} and temperature T_0 at a flow rate F. A heating jacket surrounding the CSTR provides/removes energy at a rate Q to adjust the temperature. The fluid in the reactor is assumed to have a constant density of ρ_L with heat capacity C_p. The enthalpy of reaction, Arrhenius constant, activation energy of reaction, and the ideal gas constant in appropriate units are denoted by $\Delta H, k_0, E$, and R, respectively. Process parameter values are given in

Table 4.3. The total subsampling fractions considered in this example were 0.3, 0.4, 0.6, and 0.8 since 0.2 caused numerical issues.

Table 4.3: Parameter values for chemical process example

$F = 5.0\,\mathrm{m^3/h}$	$V = 1.0\,\mathrm{m^3}$
$k_0 = 8.46 \times 10^6\,\mathrm{m^3\,kmol^{-1}\,h^{-1}}$	$E = 5.0 \times 10^4\,\mathrm{kJ/kmol}$
$R = 8.314\,\mathrm{kJ\,kmol^{-1}\,K^{-1}}$	$\rho_L = 1000.0\,\mathrm{kg/m^3}$
$\Delta H_r = -1.15 \times 10^4\,\mathrm{kJ/kmol}$	$T_0 = 300.0\,\mathrm{K}$
$Q = 0\,\mathrm{kW}$	$C_{A0} = 4\,\mathrm{kmol/m^3}$
$C_p = 0.231\,\mathrm{kJ\,kg^{-1}\,K^{-1}}$	

The data generation for this process is carried out in a manner to ensure all 50 trajectories converge to the same steady-state. In particular, the initial concentration $C_A(0)$ is assigned values between $0.01\,\mathrm{mol/m^3}$ and $10\,\mathrm{mol/m^3}$, while the initial temperature $T(0)$ takes values between $290\,\mathrm{K}$ to $320\,\mathrm{K}$. With each initial condition, the system of Eq. (4.15) is numerically integrated with explicit Euler with a step size of 10^{-4} hr from $0.0\,\mathrm{hr}$ to $2.0\,\mathrm{hr}$ and then sampled every $0.05\,\mathrm{hr}$ to generate the clean data set for this process. In this example, all data generation was carried out until $t_f = 2.0\,\mathrm{hr}$. After extensively tuning the hyperparameters, it was found that the sparse relaxed regularized regression (SR3) optimizer [184] was consistently the superior optimizer for this system. The function library that yielded the optimal results was the library with only polynomial terms. In general, when selecting the function library, it is advisable to start with a small, less complex library, such as a polynomial library, and including trigonometric or other terms only when necessary.

The results for this system are plotted in Figs. 4.6 and 4.7 while the MSE are given in Table 4.4. It is observed that, at the lowest two noise levels of $\sigma = 0.02$ or 0.1, both methods satisfactorily capture the dynamics of the CSTR system, with small improvements from the base case to the SIS case and from the SIS case to the SISC case. However, as these differences are only noticeable when analyzing either Fig. 4.6 or Table 4.4 very closely, they are not significant. At the medium noise level of $\sigma = 0.2$, the differences become noticeable in some of the test trajectories. It is observed in Fig. 4.6 that the SIS prediction marginally outperforms the base case, but the SISC model significantly performs better than the SIS model, especially in the intervals $t \in (6, 10) \cup (12, 14)$. The MSE also decreases by roughly 20% from the base case to the SIS case and a further 10% between the SIS case and the SISC case. The largest differences, however, are seen at the highest level of noise *i.e.,* $\sigma = 0.3$. In this case, the SIS model actually fails to capture the dynamics completely and performs worse than the base case with almost twice the value of the MSE. In contrast, the SISC model outperforms the base case even using only 20% of the data in each subsample. There is a significant 21% drop in the MSE as seen from Table 4.4. However, the difference can also be observed in Fig. 4.7, particularly in x_1, which is the plotted variable, most strongly between $t = 8$ and $t = 10$.

In this example, as the total subsampling fraction is increased, there does not appear to be a consistent trend in the results. In fact, the results remain very similar for all the values of p and q tested. Therefore, it may be desirable to use lower values of the subsampling fraction and only increase it as required.

Table 4.4: Test set MSE for the CSTR system for four noise levels

σ	No subsampling	SIS	SISC
0.02	0.01113	0.01059	0.01102
0.1	0.10510	0.09922	0.10370
0.2	0.49837	0.40037	0.36283
0.3	0.98210	1.89607	0.77613

4.4.4 Advantages of explicit methods

The data generation in this paper is done via integrating the ordinary differential equation systems of the case studies considered in time using the Runge-Kutta numerical integration method with an integration time-step of $h_c = 10^{-4}$. As this is an explicit numerical integration method, the calculation time is on the order of milliseconds. To be precise, for the 50 total trajectories that the system is integrated along from the 50 initial conditions, the total time is around 0.1 seconds and 0.13 seconds for the predator-prey and CSTR systems, respectively.

The integration of the identified models is also on the same order: 0.1/50 or 0.13/50 seconds for a trajectory. The computational time for the 10 test trajectories was calculated and multiplied by 5 to make a fairer comparison to the times required for the first-principles system. The results had a wider range from 0.170 seconds to 0.482 seconds, with an average of 0.266 seconds. Two points must be emphasized here. Firstly, these times are slightly higher than those of the first-principles model data generation. This can be due to the higher complexity of the models. However, some of the models had the same number of terms as the first-principles model, with coefficients also very close to the original system. This was usually seen in the lower noise levels where a near-exact reconstruction was possible. These models should have required the same time to integrate as the first-principles models. Therefore, the reason for the higher times can be attributed to (a) the integration for the sparse-identified models being carried out by the PySINDy package's internal ODE integration function rather than Python's ODE solvers, and (b) some of the models in some of the subsamples might not be numerically stable, contributing to a large increase in the maximum and average values reported above. The most important fact to note, however, is that these times, especially for one trajectory, are still all below one second, and this rapid prediction is a key advantage of using ODE models with explicit nonlinearities in system identification as opposed to ODEs with neural network function approximators for the nonlinear basis functions, which is as a possibility if the identified model with the explicit functions is not accurate enough (not the case in our studies) at the expense of increasing computational burden.

Lastly, although the above range and average of integration times for the sparse-identified models are calculated from only the base (no subsampling) case of the predator-prey system, the results generalize to both of the other cases and also the CSTR system. This is because, once the ODE is identified, the data or procedure used to identify the ODE is irrelevant. Hence, even though the model identification step is more complex and time-consuming for the subsampling scenarios, once identified, the model is integrated using the same Runge-Kutta methods with the same step sizes. Moreover, there was no clear correlation between the number of terms in the identified ODE model and the integration time required, as long as the system was not unstable and did not diverge. Hence, the times reported above can be extended to all the sparse-identified models considered in this paper.

4.5 Conclusion

In this work, a novel algorithm was devised to build dynamical models that capture nonlinear process dynamics given only highly noisy sensor data. The noise was assumed to follow a white Gaussian distribution with different variances. A predator-prey model and a chemical process were used to demonstrate the performance and applicability of the new algorithm. It was shown that the basic sparse identification algorithm was inadequate in identifying the model in the presence of high noise in the data, particularly above a variance of 0.01 for normalized data. However, when the subsampling technique was introduced, without co-teaching, by randomly subsampling to leave out the more noisy data in some iterations, it could identify the dynamics satisfactorily up to a noise variance of 0.04. Finally, the proposed algorithm combining subsampling with co-teaching, where the original data is subsampled but also mixed with some noise-free data from first-principles model simulations was used. Using the third algorithm, the performance improved slightly in the presence of noise with variance up to 0.04. However, at the highest noise level studied, which was characterized by a variance of 0.09, both the base case and the subsampling without co-teaching failed and could not identify the models using the extremely noisy data. The subsampling with co-teaching could accurately identify the models in this case, even when only 20% of the subsamples consisted of noise-free data generated from first-principles model simulations. The performance was evaluated based on plots of the outputs as well as the mean squared error (MSE) on the testing data sets. The results were qualitatively similar in both systems investigated, with more accurate models predicting the testing data set more accurately and yielding lower MSE values.

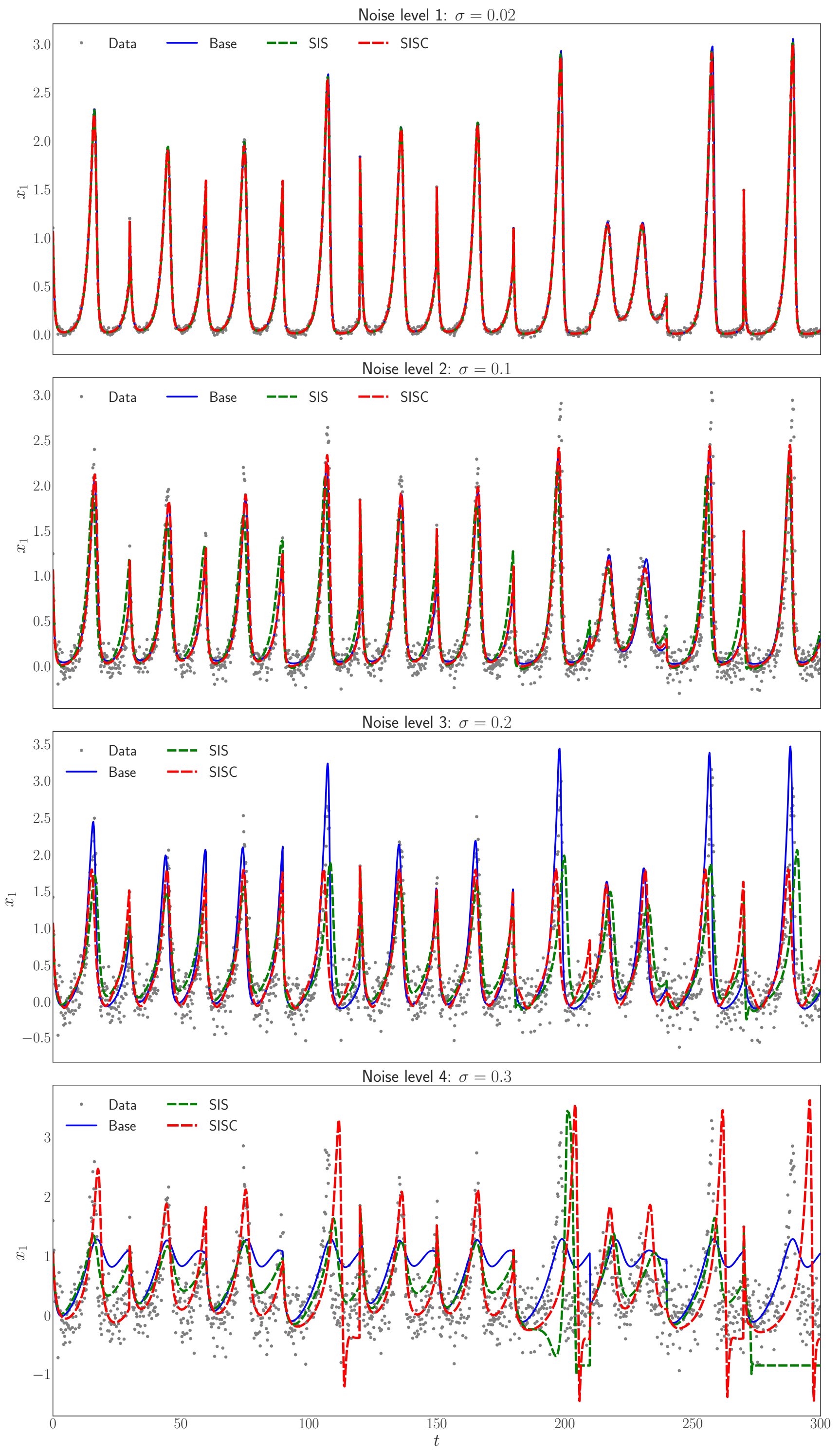


Figure 4.2: Comparison of original noisy data (grey dots) with results from sparse identification without any subsampling (blue line), subsampling without co-teaching with $p = 0.2$ (green line), and subsampling with co-teaching and $p = 0.16, q = 0.04$ (red line) for the quantity x_1 in the predator-prey system.

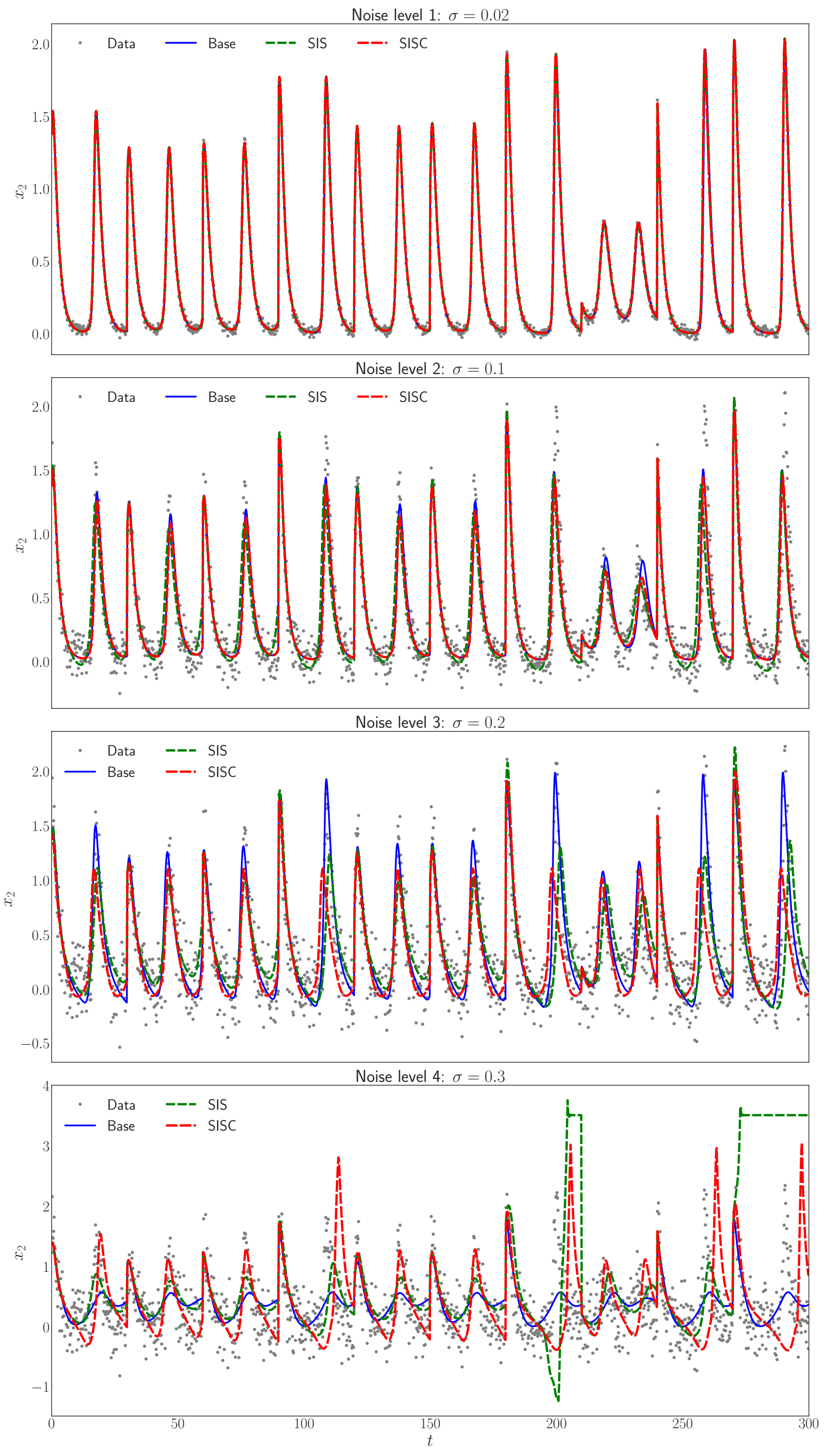


Figure 4.3: Comparison of original noisy data (grey dots) with results from sparse identification without any subsampling (blue line), subsampling without co-teaching with $p = 0.2$ (green line), and subsampling with co-teaching and $p = 0.16, q = 0.04$ (red line) for the quantity x_2 in the predator-prey system.

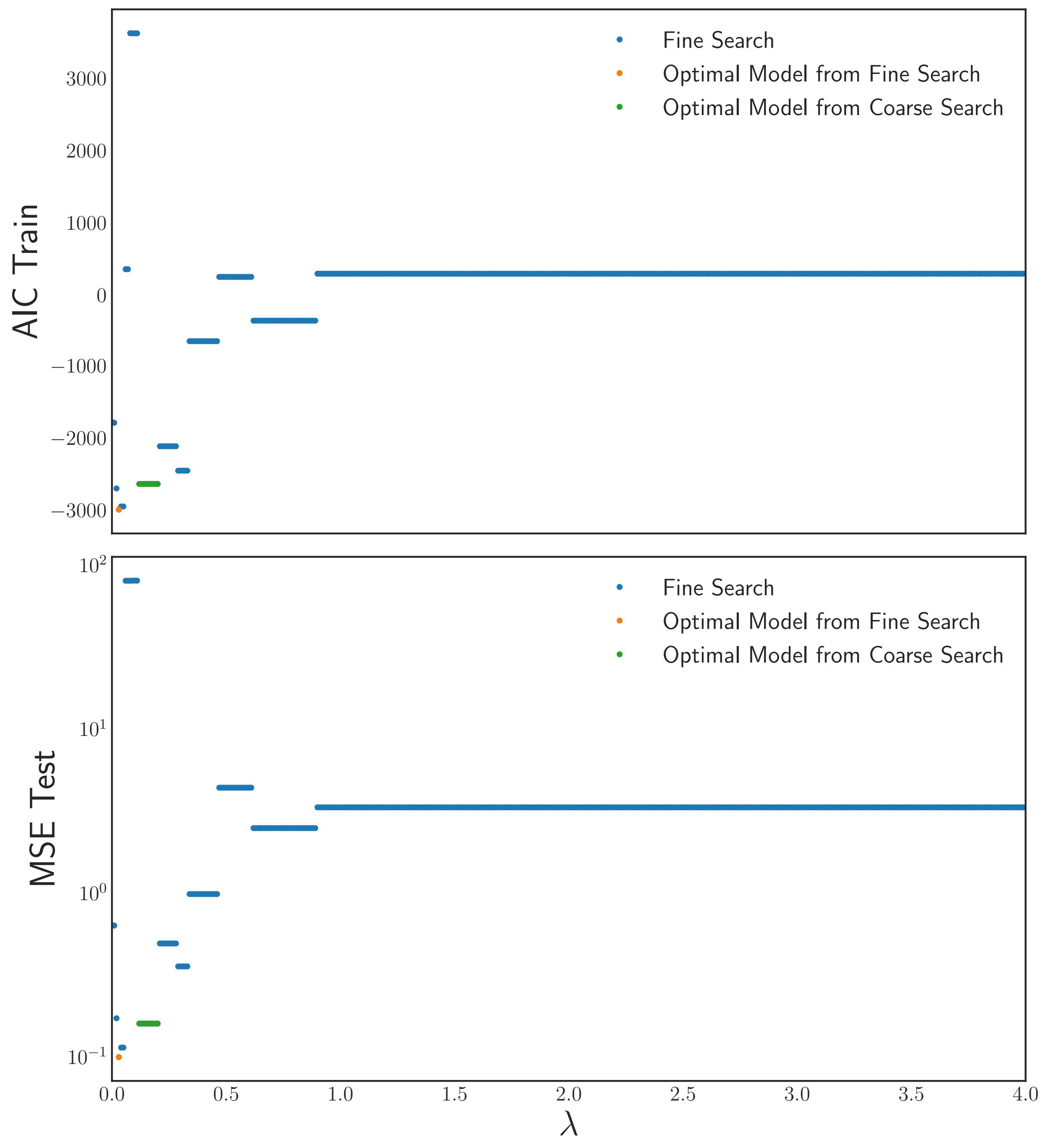


Figure 4.4: Training set AIC and test set MSE for various $\lambda \in [0, 4]$ for the base case with no subsampling and $\sigma = 0.1$.

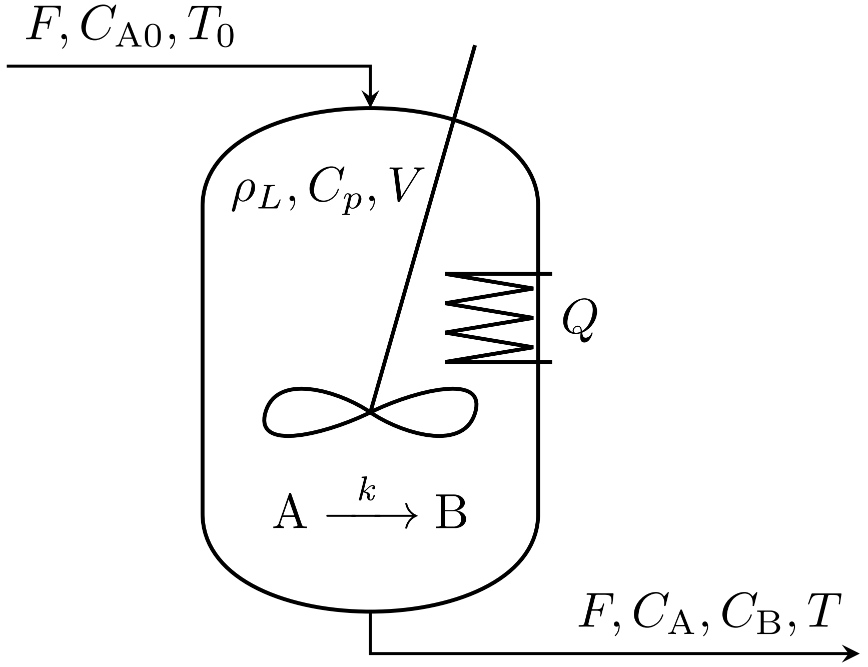


Figure 4.5: A continuous-stirred tank reactor with a heating coil

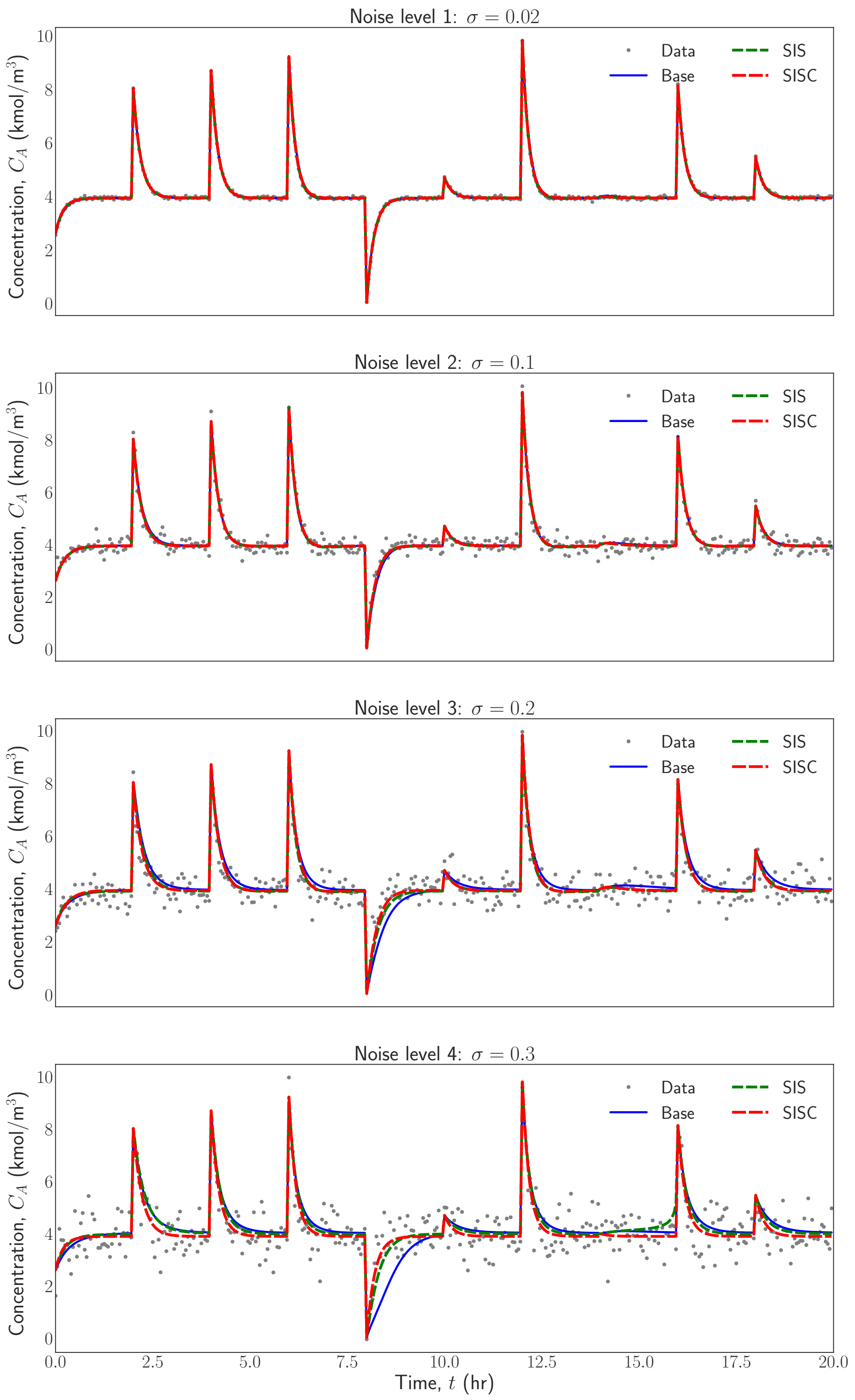


Figure 4.6: Comparison of original noisy data (grey dots) with results from sparse identification without any subsampling (blue line), subsampling without co-teaching with $p = 0.2$ (green line), and subsampling with co-teaching and $p = 0.16, q = 0.04$ (red line) for the concentration C_A of the CSTR system.

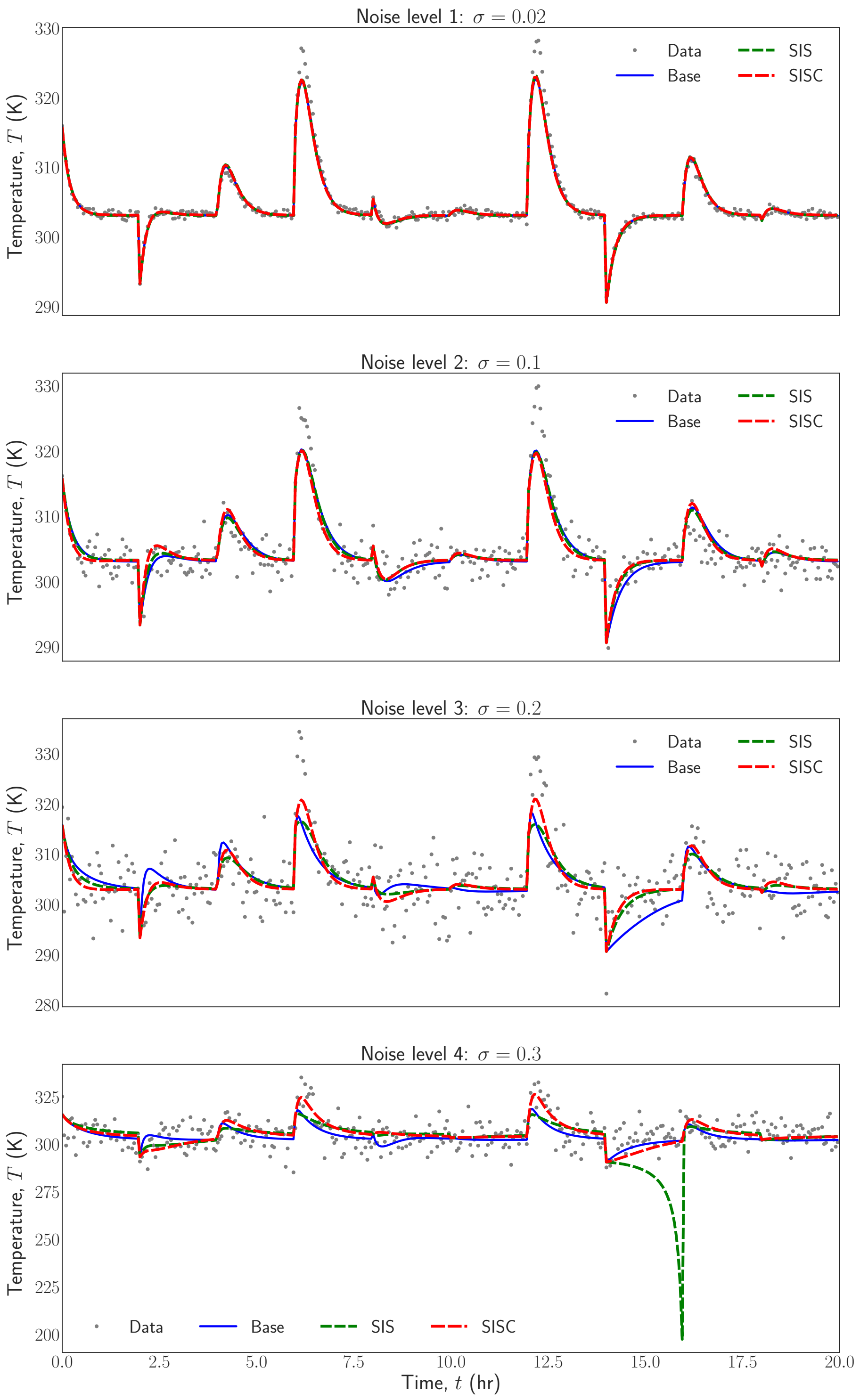


Figure 4.7: Comparison of original noisy data (grey dots) with results from sparse identification without any subsampling (blue line), subsampling without co-teaching with $p = 0.2$ (green line), and subsampling with co-teaching and $p = 0.16, q = 0.04$ (red line) for the temperature T of the CSTR system.

Chapter 5

Modeling and control of nonlinear processes using sparse identification Using dropout to handle noisy data

5.1 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., 135, 146–155, 192, 193]. The bank of data-driven system identification algorithms is expansive, ranging from traditional methods such as singular value decomposition [156] and numerical algorithms for Subspace State Space System Identification (N4SID) [30] to more recent advances such as auto-regressive models with exogenous inputs (ARX/ARARX) [157–159]. 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 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 by [77], 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., 77, 78], 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 the work of [81, 113, 163–170], 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 [194], 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, [172] extended ARX/ARARX methods to the case of additive white noise in input/output data, [173] used principal component analysis (PCA) to estimate the noise term, [174] applied subspace identification methods to closed-loop operational data, and [175, 176] handled

Gaussian white noise in linear dynamical systems with the Kalman filter. However, as linear approximations may not perform adequately for nonlinear systems, [175] proposed methods such as the extended Kalman filter and moving-horizon estimation for the case of nonlinear systems. 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 described in detail in [195]. This includes the use of an autoencoder and Kalman filter to recover the denoised states [106], the use of the total-variation regularized derivative, the application of a smoother such as the Savitzky-Golay filter to prefilter the data [78], updating rather than re-identifying new models upon the availability of new data [117], the use of Laplace transforms to rewrite ODE models using integral terms [108], the use of splines as basis functions to estimate the parameters in the already known dynamic model structure [196], the use of Duhamel's integral for mass-spring systems [109], exploiting spectral methods [111, 112] or Tikhonov regularization [115] to estimate the derivatives, the incorporation of weights in the least squares algorithm [115], subsampling, and Bayesian regression [116], among others. A common inadequacy identified in many of the articles [e.g., 106, 107] 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 seconds, while a gas chromatograph can only yield interpretable results on the composition of the substance every 15 minutes. 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, [195] studied in depth the dual effect of Gaussian noise and large sampling times on sparse identified models and proposed a novel algorithm that combined subsampling from statistics with co-teaching from the neural network context. 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 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 [197]. Chemical process simulators can be broadly divided into equation-oriented and sequential modular approaches such as EMSO software and Aspen Plus, respectively [198]. Moreover, [199] investigated the integration of process control systems within process simulators since the two units share decisions and information. 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 “Preliminaries”, 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 “Ensemble-based sparse identification”, 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 “Application to a chemical process modeled in Aspen Plus Dynamics”.

5.2 Preliminaries

5.2.1 Notation

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

5.2.2 Class of systems

We consider the general class of continuous-time nonlinear systems of the form,

$$\dot{x} = F(x, u) := f(x) + g(x)u, \qquad x(t_0) = x_0 \tag{5.1a}$$

$$y = x + w \tag{5.1b}$$

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 Eq. (5.1), i.e., $(x_s^*, u_s^*) = (0, 0)$, where x_s^* and u_s^* represent the steady-state state and input vectors, respectively.

5.2.3 Stability assumption

For the nominal system of Eq. (5.1) 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 Eq. (5.1) exponentially stable. Converse Lyapunov theorems [200–202] then imply that there exist a $\mathcal{C}^1$ 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:

$$c_1|x|^2 \leq V(x) \leq c_2|x|^2, \tag{5.2a}$$

$$\dot{V}(x) = \frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \leq -c_3|x|^2, \tag{5.2b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq c_4|x| \tag{5.2c}$$

where $\dot{V}$ represents the time-derivative of the Lyapunov function, and $F(x, \Phi(x))$ represents the nominal system of Eq. (5.1) under a candidate controller $\Phi(x)$ such as the universal Sontag control law [140]. We first characterize a set of states $\phi_u = \{x \in \mathbb{R}^n \mid \dot{V}(x) = L_fV + L_gVu < -kV(x), u = \Phi(x) \in \mathcal{U}, k > 0\} \cup \{0\}$ under the controller $u = \Phi(x) \in \mathcal{U}$ that satisfies the conditions of Eq. (5.2). Subsequently, we define the closed-loop stability region Ω_ρ [203] for the nominal system of Eq. (5.1) to be a sublevel set of V inside ϕ_u i.e., $\Omega_\rho := \{x \in \phi_u \mid V(x) \leq \rho\}$, where $\rho > 0$ and $\Omega_\rho \subset \phi_u$.

5.2.4 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 [71–76, 129]. The goal of SINDy is to use only discrete measurement data from a physical system to identify a first-order ODE of the form,

$$\dot{\hat{x}} = \hat{f}(\hat{x}) + \hat{g}(\hat{x})u \tag{5.3}$$

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. (5.3) 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 non-zero 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,

$$X = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix} \tag{5.4a}$$

$$U = \begin{bmatrix} u_1(t_1) & u_2(t_1) & \cdots & u_r(t_1) \\ u_1(t_2) & u_2(t_2) & \cdots & u_r(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_m) & u_2(t_m) & \cdots & u_r(t_m) \end{bmatrix} \tag{5.4b}$$

where $x_i(t_\ell)$ and $u_j(t_\ell)$ represent the i^{th} state measurement and j^{th} input measurement at the ℓ^{th} sampling time, respectively, with $i = 1, \ldots, n$, $j = 1, \ldots, r$, and $\ell = 1, \ldots, m$. $\dot{X}$ 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:

$$\Theta(X, U) = \begin{bmatrix} | & | & | & | & | & | & | \\ \mathbf{1} & X & X^{P_2} & \sin(X) & \mathrm{e}^X & U & UX^2 \\ | & | & | & | & | & | & | \end{bmatrix} \tag{5.5}$$

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

$$X^{P_2} = \begin{bmatrix} x_1^2 & x_1x_2 & \cdots & x_2^2 & x_2x_3 & \cdots & x_n^2 \end{bmatrix} \tag{5.6}$$

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. (5.5) associated with each variable or row of Eq. (5.3) is assigned a coefficient, with all 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:

$$\dot{X} = \Theta(X, U)\Xi \tag{5.7}$$

Popular methods for solving Eq. (5.7) 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) [184]. 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 non-zero 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,

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

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

Remark 5.1. *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 dynamical models from data. The sparse identification modeling technique's central advantage 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 [204], 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 minutes (127 seconds), while the proposed dropout-SINDy-based MPC took an average of 42 seconds. While the system of [204] 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.*

5.2.5 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 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 variance of the algorithm without increasing the bias for individual models [205] and also enables the accounting of uncertainty during the model selection process [206]. 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 the models in the ensemble or selected by cross-validation. Common central tendencies include the mean (bagging) or median (bragging) of all the model parameters/outputs.

5.2.6 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:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u(t))\, \mathrm{d}t \tag{5.8a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{si}(\tilde{x}(t), u(t)) \tag{5.8b}$$

$$u(t) \in \mathcal{U}, \ \forall\, t \in [t_k, t_{k+N}) \tag{5.8c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{5.8d}$$

$$\dot{\hat{V}}(x(t_k), u) \leq \dot{\hat{V}}(x(t_k), \Phi_{si}(x(t_k))), \ \text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{si}} \tag{5.8e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{si}, \ \forall\, t \in [t_k, t_{k+N}), \ \text{if } x(t_k) \in \Omega_{\rho_{si}} \tag{5.8f}$$

where $\tilde{x}$, N, and $S(\Delta)$ in Eq. (5.8a) 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. (5.3). $\Omega_{\hat{\rho}}$ and $\Omega_{\rho_{si}}$ denote the stability region for the closed-loop sparse identified system and the target region for the final predicted state, respectively. $\dot{V}$ is the time-derivative of V, the Lyapunov function, and is given by $\frac{\partial V(x)}{\partial x} F_{si}(x, u)$. 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 the optimization problem of Eq. (5.8) is to minimize the objective function of Eq. (5.8a), which is equal to the integral of $L(\tilde{x}(t), u(t))$, over the entire prediction horizon. Equation (5.8b) 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. (5.8c). The initial condition of the states required by the SINDy model of Eq. (5.8b) is given by Eq. (5.8d), 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. (5.8e), guarantees that the state $x(t_k)$, when outside the target region $\Omega_{\rho_{si}}$, 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 [207]. The result of the proof is that, as long as the state starts from within the stability region Ω_ρ, the state converges to $\Omega_{\rho_{si}}$ in a finite number of sampling times without leaving the stability region. The second constraint of Eq. (5.8f) ensures that, once the state enters the invariant set $\Omega_{\rho_{si}}$, it remains within this region for the entire prediction horizon.

5.3 Ensemble-based sparse identification

Modeling systems using data from first-principles simulations corrupted with sensor noise has been studied in [195]. 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 the "Ensemble learning" Subsection, 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 [195] that the former mode of ensemble learning does not improve the 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. (5.5) may be of the form,

$$\Theta_i(X, U) = \begin{bmatrix} | & | & | & | & | \\ \mathbf{1} & X & \sin X & \cos X & \tan X \\ | & | & | & | & | \end{bmatrix} \tag{5.9}$$

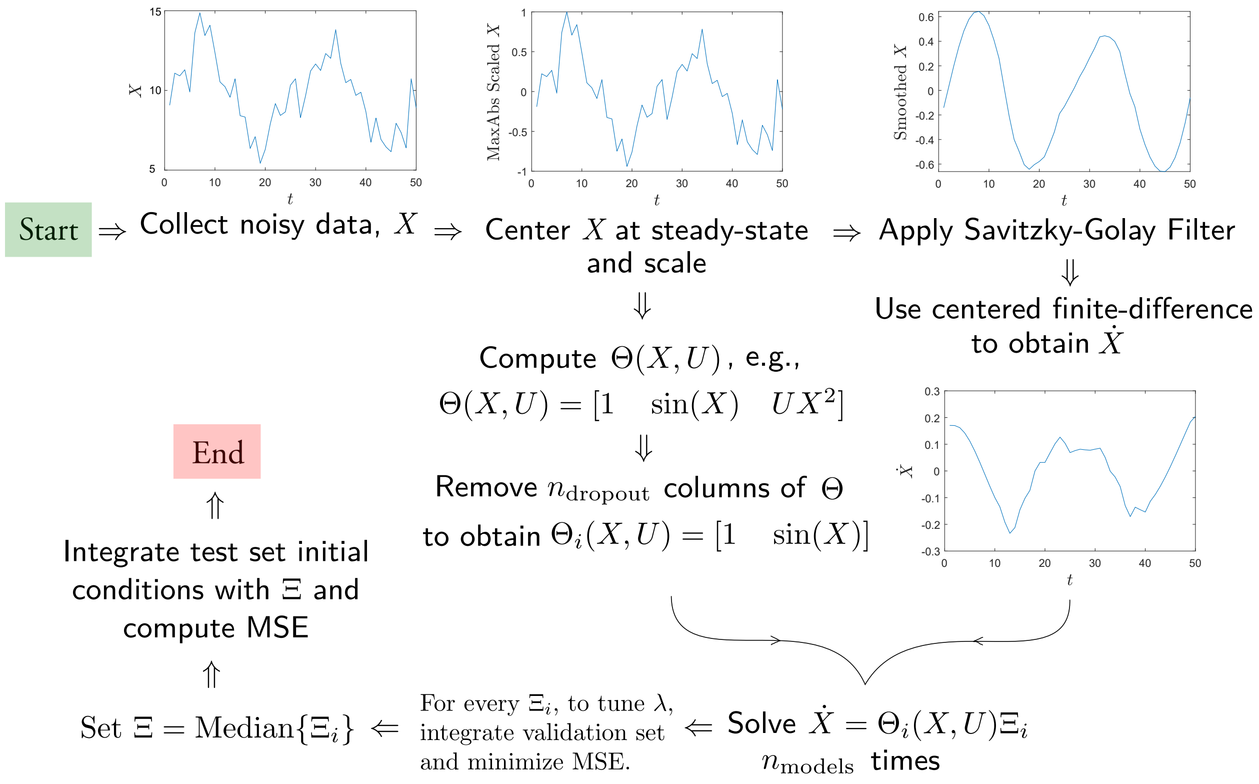
where quadratic and cubic polynomials, as well as *tanh* terms have been removed from the basis functions. The corresponding Ξ_i will assign values of zero for the n_{dropout} functions that have been dropped out, and the STLSQ optimizer will find the best ODE model that can be built using this reduced 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 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 Fig. 5.1.

Remark 5.2. *A more accurate yet computationally expensive approach is to record the validation MSE for every sub-model during training. Finally, the sub-model 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 5.3. *It is important to emphasize that SINDy searches over a "bank" of explicit non-linearities (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*

Algorithm 2: 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.
```



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 $\Theta(X,U)$, e.g., $\Theta(X,U) = [1 \quad \sin(X) \quad UX^2]$
Remove n_{dropout} columns of Θ to obtain $\Theta_i(X,U) = [1 \quad \sin(X)]$
Solve $\dot{X} = \Theta_i(X,U)\Xi_i$ 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
X
MaxAbs Scaled X
Smoothed X
$\dot{X}$
t

Figure 5.1: Data flow diagram for model construction.

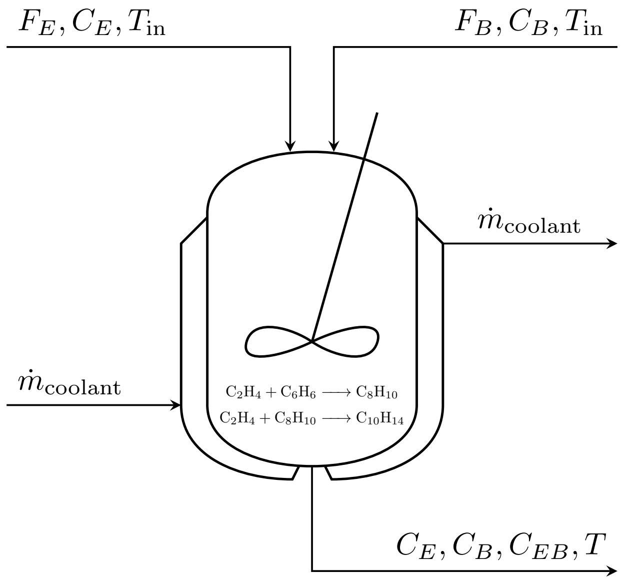
nonzero pre-multipying coefficients of the nonlinear basis functions used to construct the ODE model. If there is any physical insight on the type of nonlinearities that the approach should consider, this physical insight can be incorporated into the optimization search (biasing, for example, the order with which the nonlinearities are considered in the optimization search in an approach similar to the ALAMO modeling technique [6, 208]). 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 physico-chemical 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 5.4.2.1, 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 5.4. *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 [13, 14]. 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.*

5.4 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 in order 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



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 5.2: A continuous-stirred tank reactor with a cooling jacket.

a perfectly mixed, non-isothermal continuous stirred tank reactor (CSTR) as shown in Fig. 5.2. 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:

$$C_2H_4 + C_6H_6 \xrightarrow{k_1} C_8H_{10} \qquad \text{(main)} \tag{5.10a}$$

$$C_2H_4 + C_8H_{10} \xrightarrow{k_2} C_{10}H_{14} \tag{5.10b}$$

where the desired reaction of Eq. (5.10a) is annotated as “main”.

5.4.1 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 Fig. 5.3.

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 $T_{\text{in}} = 350\,\text{K}$. 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, in order 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 5.1.

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 by 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 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-meter long, vertical vessel with flat heads.

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.

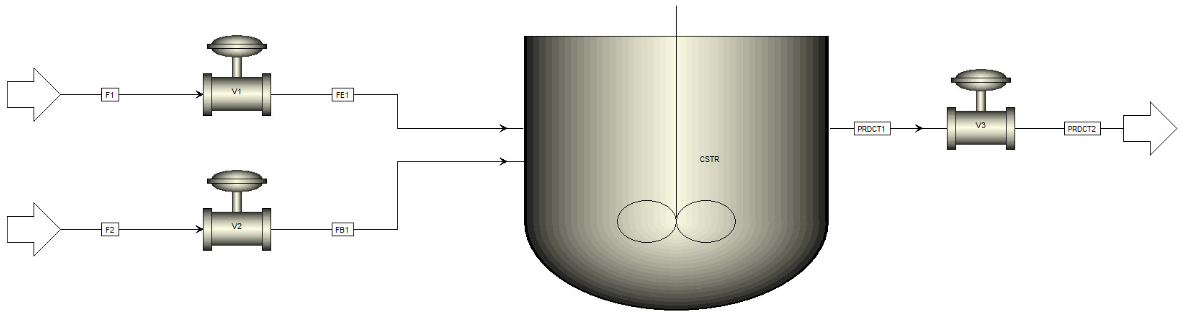


Figure 5.3: Aspen Plus model process flow diagram of ethylbenzene production process.

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 $\dot{m}_{\text{coolant,ss}} = 77.9869\,\text{kg/s}$. 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 Fig. 5.3.

The addition of noise to the model solutions to represent realistic process state data will be discussed in detail in the “Data generation and SINDy model development” Subsection.

Table 5.1: Parameter values for chemical process example.

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

5.4.2 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 $u = -37.9869\,\mathrm{kg/s}$ and $u = 4.0131\,\mathrm{kg/s}$. 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 $\Delta = 0.01$ hr. A total of 25 trajectories of the three types are generated. The simulation duration for the runs are 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 hr and 15 hr. The test run used for further demonstration is generated at the lower limit of the range of u considered i.e., with $u = -37.9869\,\mathrm{kg/s}$. 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 introduction, 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 [77] 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, 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 seconds 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 $\Delta = 0.01$ seconds 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 pre-processing is conducted. Specifically, the 25 runs in the data are first split into the 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 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 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 the 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:

$$\sigma = \text{RMSE} \times \frac{\text{Noise percentage}}{100} \tag{5.11}$$

where RMSE is the root mean squared error of the signal given by

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} x_i(t_j)^2}{m \times n}} \tag{5.12}$$

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 Fig. 5.4.

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. (5.11)

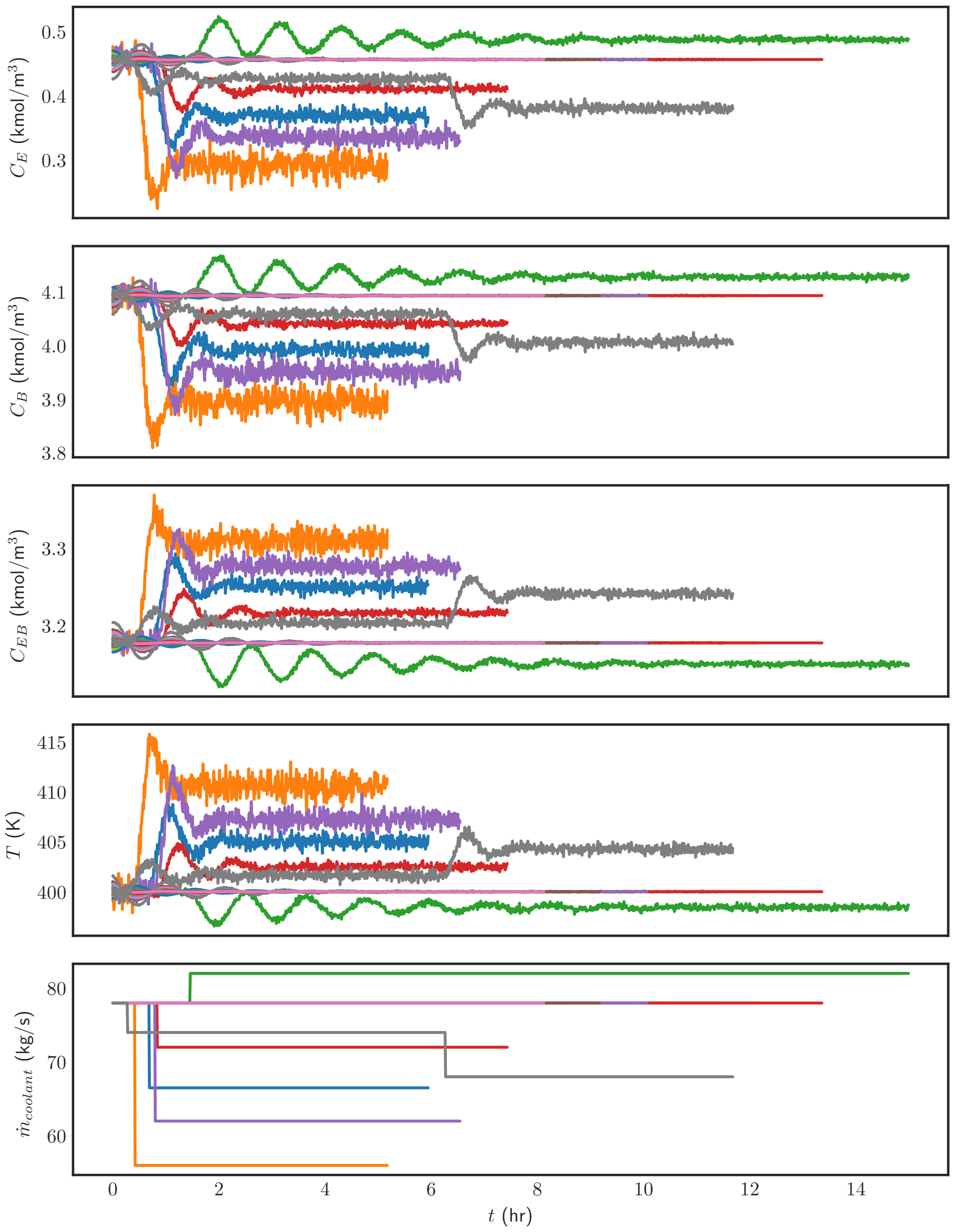


Figure 5.4: Training data set. Each line is one open-loop trajectory.

and (5.12), 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. (5.12)) being used to define the noise percentage, centering is required regardless of scaling. This ensures x_i in Eq. (5.12) 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. (5.11) and (5.12), to the newly scaled data set. This eliminates the need to scale the noise because all the x_i in Eq. (5.12) are already of a similar order of magnitude, specifically between ± 1. In contrast, the second method is to add scaled noise of 8% to each variable. However, since x_4 in Eq. (5.12) 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. (5.11). 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.

In order 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 Fig. 5.4 once converted back to non-deviation 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 fashion 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 5.5. *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 [154], where recurrent neural networks are shown to satisfactorily deal with Gaussian noise but struggle with non-Gaussian noise. In contrast, [190] 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 [195] 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 were 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 6 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, $\frac{x_1}{1+e^{x_4-1}}$, $\frac{x_2}{1+e^{x_4-1}}$, $\frac{x_3}{1+e^{x_4-1}}$, $\frac{x_1^2}{1+e^{x_4-1}}$, $\frac{x_2^2}{1+e^{x_4-1}}$, $\frac{x_3^2}{1+e^{x_4-1}}$, $x_1 \tanh x_4$, $x_2 \tanh x_4$, $x_3 \tanh x_4$, $x_1^2 \tanh x_4$, $x_2^2 \tanh x_4$, $x_3^2 \tanh x_4$. Using this set of basis functions, Algorithm 1 is implemented in PySINDy [209, 210], a Python Application Programming Interface (API), to build the SINDy model. While identifying each sub-model, some of the listed basis functions are dropped out before carrying out the optimization.

Remark 5.6. *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 hyper-parameters require to be tuned: λ, the sparsification knob, n_{dropout}, the number of library functions to be zeroed for each sub-model, and n_{models}, the total number of sub-models 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 sub-models 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 Fig. 5.5.

Remark 5.7. *If, instead of the median of the sub-models, the best of all the sub-models is selected as the final model as mentioned in Remark 5.2, it can be theorized that the overall model accuracy will only improve as* n_{models} *is increased, although at a higher computational cost.*

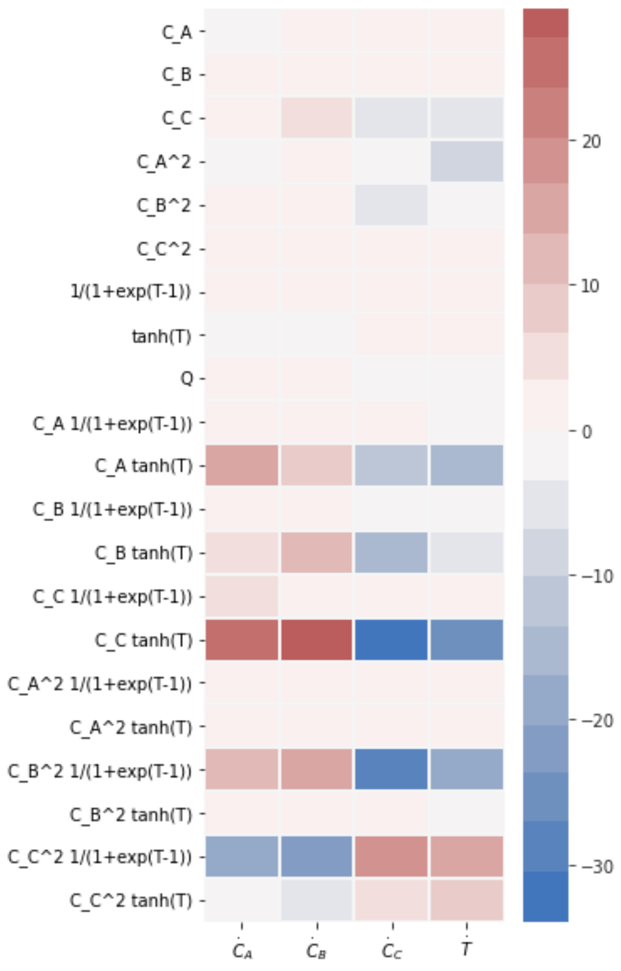


Figure 5.5: Visualization of dropout-SINDy identified model.

5.4.2.1 Corroboration between the sparse-identified model and known CSTR dynamics

Although the model shown in Fig. 5.5 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 lead 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

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

Firstly, 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} both confirm this. However, we notice 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. 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 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 5.8. *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)[211]. In the PySINDy package, a further improved version of OLS known as Forward Regression Orthogonal Least Squares (FROLS) has been implemented based on [12]. 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 hyper-parameters 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 have 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.

5.4.3 Open-loop simulation results

The SINDy model obtained using dropout-SINDy as described in the "Data generation and SINDy model development" Subsection 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 Fig. 5.6. 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 $-8.9869\,\mathrm{kg/s}$ between $t = 0\,\mathrm{hr}$ and $t = 7.21\,\mathrm{hr}$. Once the system reaches the first nonzero steady-state, from $t = 7.21\,\mathrm{hr}$, a new input of $-2.9869\,\mathrm{kg/s}$ is applied until $t = 15.1\,\mathrm{hr}$. 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 the "Data generation and SINDy model development" Subsection, 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 4 states for this run are calculated and shown in Table 5.2. It is observed that the MSE for the concentrations are 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 [153]. 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 5.2: Open-loop prediction MSE results for the run shown in Fig. 5.6.

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

5.4.4 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. (5.8) 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) = ($0.456\,\mathrm{kmol/m^3}$,

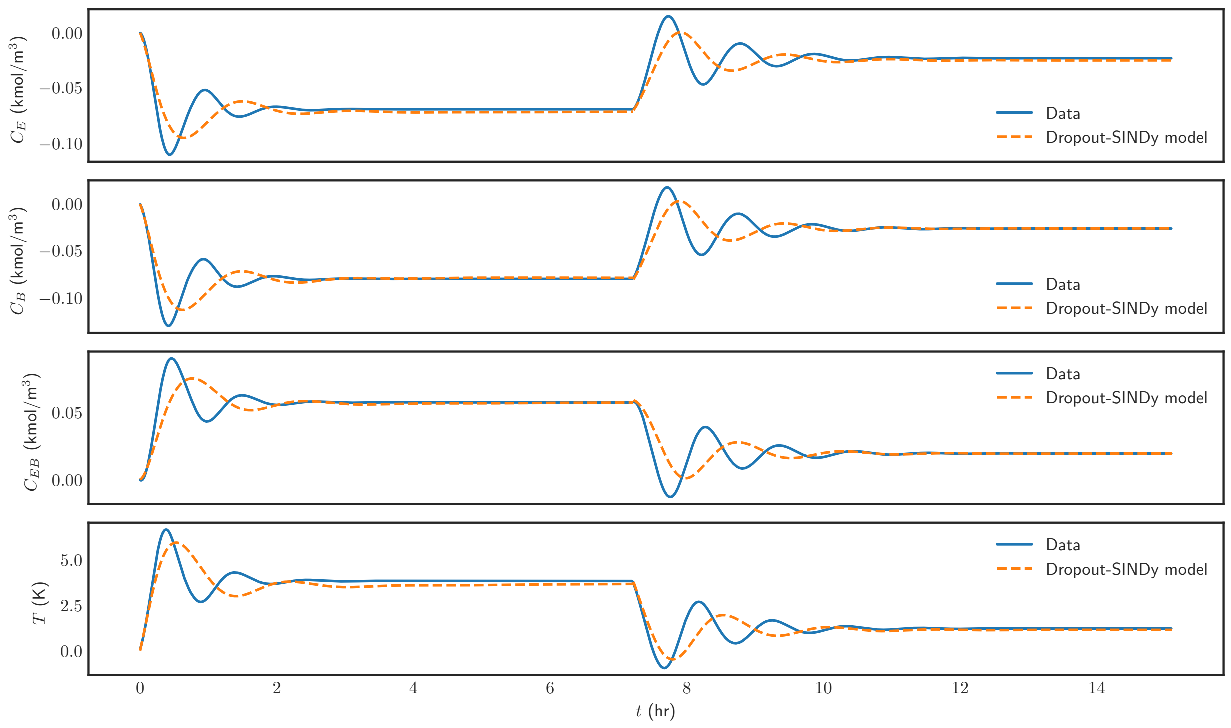


Figure 5.6: Time-varying profiles of the states for the open-loop simulation using Aspen Plus Dynamics (blue line) and the dropout-SINDy model (orange line) with two nonzero input values of $u_1 = -8.9869\,\mathrm{kg/s}$ and $u_2 = -2.9869\,\mathrm{kg/s}$ applied over the intervals $t_1 \in (0, 7.21]$ and $t_1 \in (7.21, 15.1]$, respectively.

$4.09\,\mathrm{kmol/m^3}$, $3.18\,\mathrm{kmol/m^3}$, $400\,\mathrm{K}$) by manipulating the coolant flow rate $\dot{m}_{\text{coolant}}$. The objective function of the LMPC, Eq. (5.8a), is considered to be as follows to ensure a value of zero at the steady-state itself under no-input conditions:

$$L(x, u) = |x|^2_{Q_1} + |u|^2_{Q_2} \tag{5.13}$$

where Q_1 and Q_2 are weighting matrices that control the contributions of the state and the input in the LMPC objective function. Q_1 is chosen to be the 4×4 identity matrix, while $Q_2 = 2 \times 10^{-6}$. As the optimization problem of Eq. (5.8) is nonlinear and non-convex, we solve it every $\Delta =$

$0.01\,\mathrm{hr}$ using the numerical solver Ipopt [145] 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 5.7 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 $t_p = 1.5\,\mathrm{hr}$. If the simulation is continued, it is observed that the uncontrolled state returns to the steady-state after approximately 5 hours, which was also observed in the "Data generation and SINDy model development" Subsection. 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 $t = 0.5\,\mathrm{hr}$ in half the time compared to the P-controller, which takes $t = 1\,\mathrm{hr}$, and a tenth of the time taken in the uncontrolled scenario.

The closed-loop performance of the two controllers are 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. (5.13) that a lower value of the objective function indicates both faster convergence as well as 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, $\int_{t=0}^{t_p} L(x(\tau), u(\tau))\ \mathrm{d}\tau$,

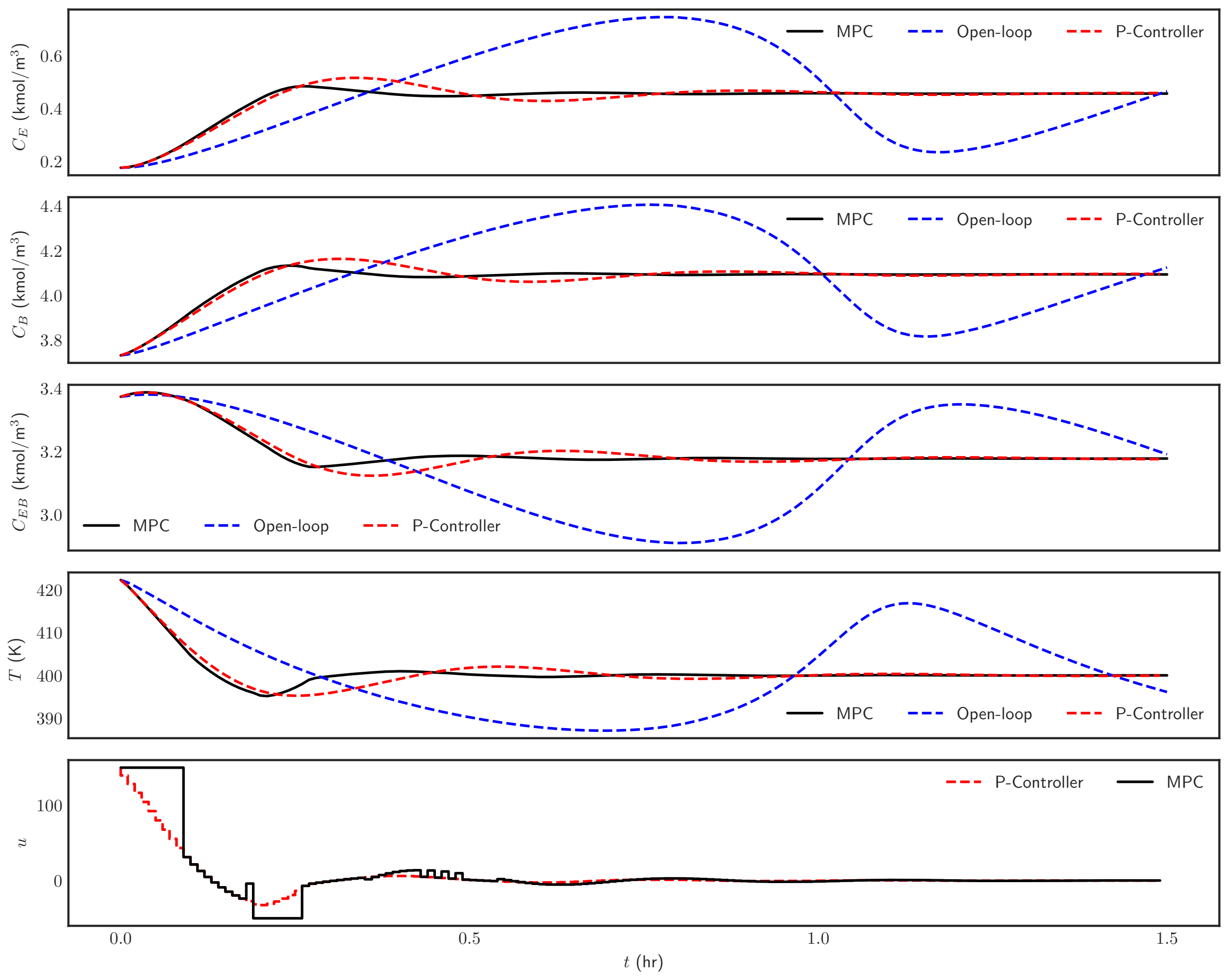


Figure 5.7: State and input profiles for the CSTR in closed-loop under no control (blue line), a P-controller (red line), and the LMPC utilizing the dropout-SINDy model (black line) throughout the simulation period $t_p = 1.5\,\text{hr}$.

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 5.8 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. (5.8e). 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 $t = 0.1\,\mathrm{hr}$ and $t = 0.2\,\mathrm{hr}$), 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. (5.8e), as also evidenced by the input profiles shown in Fig. 5.7.

Remark 5.9. *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 Fig. 5.7. Hence, no integral control was used. Since derivative control is typically necessary for excessive oscillations [212], 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.* (5.8e). *The goal of the MPC is to improve upon this input by solving the optimization problem of Eq.* (5.8). *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*

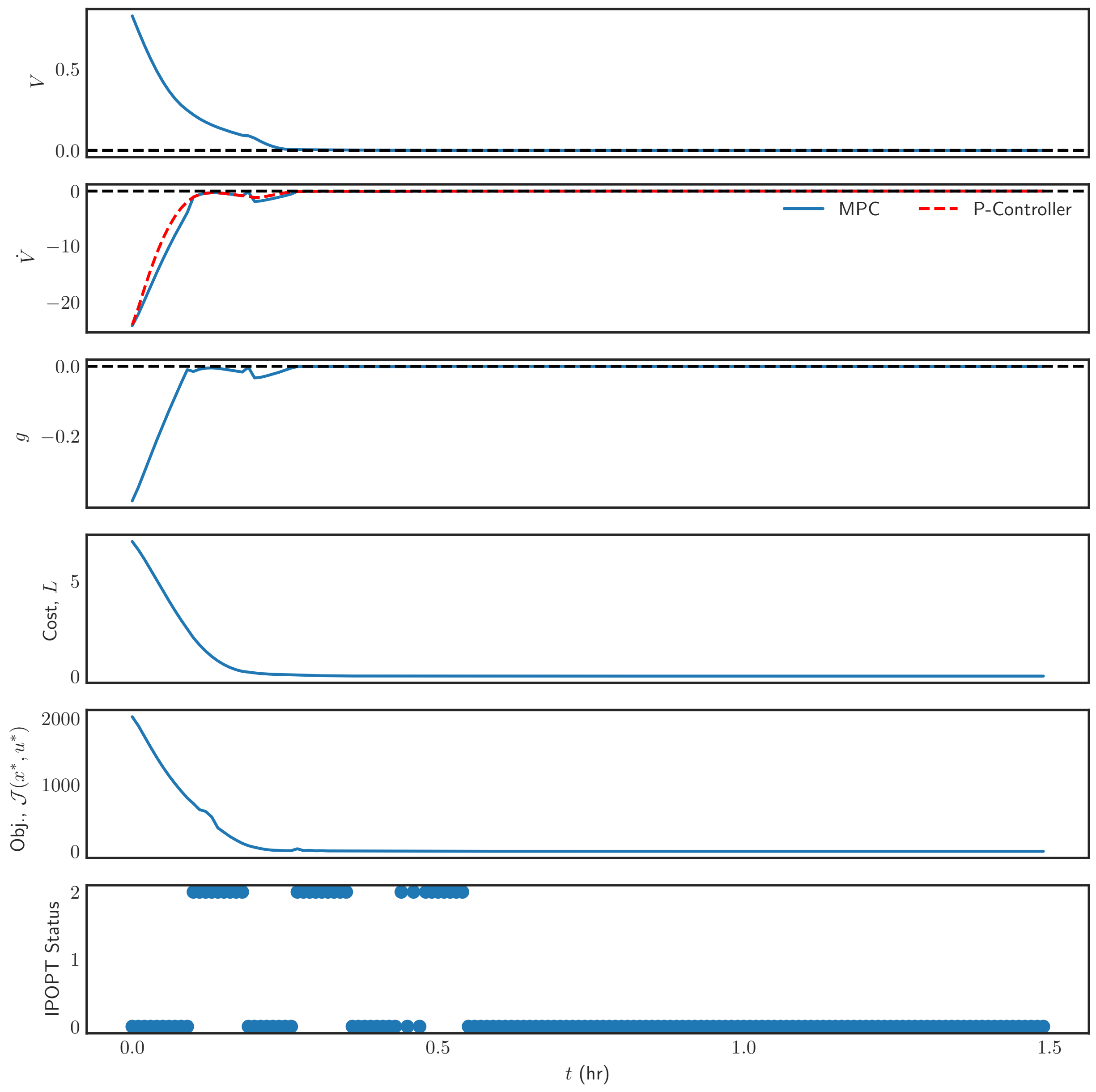


Figure 5.8: LMPC performance metrics throughout the simulation period $t_p = 1.5\,\mathrm{hr}$.

as well as the non-model-based controller. There are other advantages, however, to using MPC, such as accounting for all the states instead of only the temperature in 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 5.10. *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-principle model is readily available due to the extreme nonlinearities and complexities. If such a first-principle model were available, subsampling-based SINDy[195] is a viable and possibly superior algorithm. However, that requires the aid of a first-principle model, which is not possible to be derived manually in this scenario. Hence, such a comparison cannot be made in this case study.*

5.5 Conclusion

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 as compared to the open-loop performance and a P-controller in terms of faster convergence.

Chapter 6

Real-time adaptive sparse-identification-based predictive control of nonlinear processes

6.1 Introduction

Advanced process control techniques, such as model predictive control (MPC) play a crucial role in industrial applications and can leverage the recent and ongoing revolution in data-driven approaches in the science and engineering ecosystem. MPC is widely used due to its ability to handle strongly nonlinear processes with constraints, which are challenging for traditional linear control methods [213, 214]. MPC offers advantages such as straightforward tuning, control of systems with time delays and instability, incorporation of known constraints and multiple operating conditions, compensation for dead time, and flexibility in defining control objectives. However, a major drawback is the requirement for a suitable model to predict the future states in the real-time calculations, which can be costly and time-consuming to develop for large-scale, complex nonlinear processes using existing system identification or model reduction techniques [215]. The quality of process models is influenced by various factors such as parameter estimation, model uncertainty, assumptions made during model development, dimensionality, model structure, and computational complexity for real-time implementation [1, 2].

To combat the difficulty of developing process models for large-scale or poorly understood processes, over the last decade, there has been a paradigm shift from first-principles modeling to data-driven modeling. Machine learning techniques are a subset of data-driven modeling techniques that have seen increasing application in modeling chemical processes when traditional first-principle models are not available. For instance, in previous works [3, 62], recurrent neural networks were utilized to construct data-driven models for nonlinear processes, which were subsequently integrated into Lyapunov-based model predictive control (MPC) to ensure stability and performance. Although machine learning algorithms have demonstrated continuous success in modeling complex systems in the large-data limit due to their large number of hyperparameters and degrees of freedom of the model, their black-box nature can hinder their advancement to deployment in practical engineering systems, even more so in safety-sensitive fields such as chemical plants [216]. Such models are also usually strongly restricted to the domain of training data, and it is highly inadvisable to use such models for extrapolating the dynamics of the remainder of the state space. To better capture the physics of systems, several works focused on symbolic regression, which was a successful direction but computationally intractable for large-scale systems. Hence, this idea was developed further with the concept of compressive sensing [72, 75] into a relatively modern technique known as sparse identification for nonlinear dynamics (SINDy). Since its inception, SINDy has been applied to a broad range of systems [73, 129]. In the field of process systems engineering, the goal of using SINDy for building process models is that SINDy enables the direct identification of models of explicit and closed-form nonlinear first-order ordinary differential equations (ODEs) from data. These identified equations can be readily incorporated into optimization problems, including MPC. The computational cost of integrating these explicit ODE models is typically low, particularly when the models are well-conditioned, thanks to the availability of efficient differential equation solvers that use well-established integration algorithms such as 4th/5th order Runge-Kutta methods.

In the recent literature, SINDy has been implemented successfully to develop models in chemical engineering, such as the identification of reaction networks [84] and the development of reduced-order models for controlling hydraulic fracturing processes [82] and nonlinear reactors [135, 155]. The practical challenge of handling noise in sensor measurements when using SINDy was also addressed in [195, 217], where subsampling, co-teaching, and ensemble learning were used to build accurate SINDy models that captured the original nonlinear system from noisy data sets. Despite the successes in initial model building using SINDy, in practical applications, process models undergo changes over time due to various factors, including external influences (such as aging equipment, disturbances, and deployment of new operational technology) and internal factors (such as equipment fouling or catalyst deactivation). As a result, the SINDy model trained on past normal operations may not accurately predict process states in the presence of disturbances. To address this challenge, researchers have explored adaptive, robust, and event-triggered control approaches within both classical (first-principles) modeling and data-driven modeling techniques like SINDy to mitigate the impact of model uncertainty.

[218] proposed the operable adaptive sparse identification of systems (OASIS) framework where multiple SINDy models are constructed for the various regions of the state-space using single, short trajectories. However, due to the low data usage per model, the SINDy models were localized and could not extrapolate the entire state-space. Hence, a feedforward neural network was used to switch between the SINDy models based on partial state measurements and estimation of the remaining states via a Kalman filter. The authors demonstrated the effectiveness of the OASIS approach by building 100 different SINDy models for various sections of the operating region with relatively small data sets of single trajectories with 100 data points sampled every 0.01 hr. However, there was no update of the SINDy models themselves in real-time. [219] modified the original formulation of SINDy to handle output measurements and actuation in addition to state measurements and also proposed highly specific library terms that are relevant for power systems, demonstrating the superior performance of the proposed SINDy algorithm on synchronous generator models. In [220], the authors further combined SINDy with the manifold boundary approximation method to build models specifically favoring the identification of power systems and conducting their stability analyses. Specifically, the proposed algorithm had low data requirements and was ideal for updating models in real-time subject to changing dynamics, and this aspect was demonstrated via model reduction using limited data when applied to a number of synchronous generator models. However, much of the work was highly tailored to power systems, especially large power systems and their transient stability analysis. The structured online learning method was proposed in [221] where a quadratic value function was used to yield equations that were a more general form of the linear quadratic regulator with certain advantages and improvements when operating at unstable steady states in a pendulum example. Due to the quadratic formulation,

the parameters of the value function could be analytically computed with a low computational cost, while SINDy was used for the model identification part. However, the focus of [221] was on the real-time update of the value function rather than the SINDy models. Similarly to [219], SINDy was generalized in [222] to handle multi-input multi-output (MIMO) systems to model system outputs to sensor measurements, i.e., instead of obtaining ODEs in the states, the system output is obtained as functions of the state dynamics. Subsequently, a Kalman filter was used not to estimate the states but rather the model coefficients and update them in real-time using sensor measurements at every sampling time. Through two standard nonlinear examples, it was shown that this generalized SINDy with the Kalman filter approach could obtain better models than building a pure SINDy model using the sequential thresholded least-squares solver with similar or less amount of data.

The concept of real-time updates of SINDy models was first proposed in [117], where a method for re-identification was introduced, allowing for the updating of model coefficients or the addition/removal of terms, or any combination thereof, as needed. The re-identification process was triggered by a noticeable deviation between the local Lyapunov exponent and the prediction horizon estimate, although the definition of "prediction horizon" in their study differs from its usage in the current manuscript. Although the findings of [117] were mostly limited to the modeling context with no closed-loop data usage in the model updates, the methodology was recently adapted in [118] in the modeling of ducted fan aerial vehicles (DFAVs) for closed-loop control under MPC. Specifically, an offline model was first built using deep physical knowledge of DFAVs, and the model parameters were then updated based on the proposed paradigm of [117], using the same Lyapunov exponent-based trigger for the model update. The proposed methodology was shown to be able to control DFAVs, which are highly challenging to model due to their complex flow distribution, under the various cases and disturbances considered such as wind turbulence, which is of practical concern in such a setting. Although real-time update of SINDy models was also studied in [223], the goal was not to update the SINDy model to process changes but rather to build the process model itself for an unchanging process step-by-step, improving the model with newer data as it becomes available. Hence, the work of [223] can be considered an alternative to model building when there is no large data set from numerous simulations to build a high-fidelity SINDy model offline before deploying the controller. Finally, most recently, an MPC framework using SINDy models with updates to handle changing process conditions was investigated in [119]. The focus of the work was on model updates in the face of entirely unknown or first appearance process conditions and the operation of processes with multiple operating conditions. The discrete-time formulation of SINDy with actuation was used, and the greedy algorithm known as orthogonal matching pursuit was used to efficiently calculate only the matrix of changes to the model coefficients. During the model transition/update period, an elastic feedback correction method was used as a stopgap solution. The proposed error-triggered adaptive sparse identification for predictive control (ETASI4PC) method showed significant improvement over the state-of-the-art methods including SINDy without model updates as well as the aforementioned OASIS approach in the presence of large disturbances in the feed flow rate of a chemical reactor system. We note that the ETASI4PC method is a promising methodology for updating SINDy models in real-time in tracking MPC. At the moment, based on our survey of the literature, the operation of Lyapunov-based model predictive controllers, both Lyapunov-based tracking MPC (LMPC) and Lyapunov-based economic MPC (LEMPC) under SINDy models that are updated in real-time has not been investigated, which is the subject of the current manuscript. While a tracking MPC drives the state of a system to a desired set-point, economic MPC, a recent model-based control strategy, optimizes time-varying operation by considering future process states, economic objectives, and feedback. It incorporates economic factors and constraints to achieve improved process efficiency and desired closed-loop response characteristics. The potential benefits of economic MPC make it an attractive choice for industrial applications, as highlighted in studies such as [207, 224–226].

The rest of this manuscript is organized as follows: in Section 6.2, the notations, the class of nonlinear systems considered, and stability assumptions are provided. Section 6.3 provides a brief review of sparse identification and its implementation, followed by the design of Lyapunov-based tracking and economic MPCs using SINDy models as the predictive model. In Section 6.4, the error-triggering mechanism is introduced, the details of the SINDy model update procedure are given, the implementation strategy for adaptive SINDy models in LMPC and LEMPC is delineated, and rigorous closed-loop stability analyses are conducted for the two types of MPCs. Finally, a chemical reactor example is used in Section 6.5 to demonstrate the performance of the proposed adaptive SINDy-MPC methodology.

6.2 Preliminaries

6.2.1 Notation

If x is a vector, we denote its transpose as $x^\top$ and its weighted Euclidean norm as $|x|_Q$, where Q is a positive definite matrix. The standard Lie derivative $L_f V(x)$ is defined as $\frac{\partial V(x)}{\partial x} f(x)$. The operator "$\backslash$" represents set subtraction, such that $A \backslash B$ is the set of elements $x \in \mathbb{R}^{n_x}$ that belong to

A but not to B. A function $f(\cdot)$ is said to belong to the class $\mathcal{C}^1$ if it is continuously differentiable within its domain. A class $\mathcal{K}$ function is defined as a continuous function $\alpha : [0, a) \to [0, \infty)$ that is strictly increasing and takes the value of zero only when it is evaluated at zero.

6.2.2 Class of systems

We examine a broad category of continuous-time nonlinear systems, characterized by the equation,

$$\dot{x} = F(x, u, w) := f(x) + g(x)u + h(x)w, \qquad x(t_0) = x_0 \tag{6.1}$$

where $x \in \mathbb{R}^{n_x}$ denotes the state vector, $u \in \mathbb{R}^{n_u}$ represents the manipulated input vector, and $w \in W$ is the disturbance vector with $W := \{w \in \mathbb{R}^{n_w} \mid |w| \leq w_m,\ w_m \geq 0\}$. The functions $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are suitably smooth vector and matrix functions, respectively, with dimensions $n_x \times 1$, $n_x \times n_u$, and $n_x \times n_w$. Without loss of generality, we assume that the initial time t_0 and initial condition $f(0)$ are both equal to zero in this manuscript. Consequently, the steady-state of the nominal system of Eq. (6.1) is at the origin, specifically denoted as $(x_s^*, u_s^*) = (0, 0)$. Here, x_s^* and u_s^* represent the steady-state state and input vectors, respectively.

6.2.3 Stabilization via control Lyapunov function

Assuming noise-free state measurements and full state feedback for the nominal system described in Eq. (6.1), it is postulated that a stabilizing control law $u = \Phi(x) \in U$ exists, capable of exponentially stabilizing the origin of the closed-loop system mentioned in Eq. (6.1). According to converse Lyapunov theorems [200–202], this implies the existence of a $\mathcal{C}^1$ control Lyapunov function $V(x)$, along with four positive constants c_1, c_2, c_3, c_4, satisfying the conditions,

$$c_1|x|^2 \leq V(x) \leq c_2|x|^2, \tag{6.2a}$$

$$\dot{V}(x) = \frac{\partial V(x)}{\partial x} F(x, \Phi(x), 0) \leq -c_3|x|^2, \tag{6.2b}$$

$$\left|\frac{\partial V(x)}{\partial x}\right| \leq c_4|x| \tag{6.2c}$$

for all x in an open neighborhood D around the origin. In Eq. (6.2), $\dot{V}$ represents the time-derivative of the Lyapunov function, and $F(x, \Phi(x), 0)$ represents the nominal system from Eq. (6.1) under a candidate controller $\Phi(x)$, such as the universal Sontag control law [140]. Our first objective is to define a set of states ϕ_u, expressed as follows:

$$\phi_u = \{x \in \mathbb{R}^{n_x} \mid \dot{V}(x) = L_f V + L_g V u < -kV(x), u = \Phi(x) \in U, k > 0\} \cup \{0\}$$

under the controller $u = \Phi(x) \in U$, that satisfies the conditions described in Eq. (6.2). Subsequently, we define the closed-loop stability region Ω_ρ [203] for the nominal system presented in Eq. (6.1) as a sublevel set of V within ϕ_u, denoted as $\Omega_\rho := \{x \in \phi_u | V(x) \leq \rho\}$, where $\rho > 0$ and $\Omega_\rho \subset \phi_u$. Furthermore, from the Lipschitz continuity property of $F(x, u, w)$ and the given bounds on u, it can be deduced that there exist positive constants N_w, L_x, L'_x, L_w, and L'_w such that the following inequalities are satisfied for all $x, x' \in D$, $u \in U$, and $w \in W$:

$$|F(x,u,w)| \leq M \tag{6.3a}$$

$$|F(x,u,w) - F(x',u,0)| \leq L_x|x-x'| + L_w|w| \tag{6.3b}$$

$$\left|\frac{\partial V(x)}{\partial x}F(x,u,w) - \frac{\partial V(x')}{\partial x}F(x',u,0)\right| \leq L'_x|x-x'| + L'_w|w| \tag{6.3c}$$

6.3 Lyapunov-based MPC using sparse identification

In this section, the details of the sparse identification problem and its solution are provided, followed by the formulation of LMPC and LEMPC that utilize the SINDy model to predict the future states. Furthermore, closed-loop stability for the nonlinear system of Eq. (6.1) is discussed under the proposed LMPC and LEMPC.

6.3.1 Sparse identification

Sparse identification, a recent advancement in nonlinear system identification, has demonstrated its effectiveness in various engineering disciplines through numerous examples [71–76, 129]. The objective of the Sparse Identification of Nonlinear Dynamics (SINDy) approach is to utilize discrete measurement data from a physical system to identify a first-order ordinary differential equation (ODE) of the following form:

$$\dot{\hat{x}} = \hat{f}(\hat{x}) + \hat{g}(\hat{x})u \tag{6.4}$$

where $\hat{x} \in \mathbb{R}^{n_x}$ represents the state vector of the model obtained through sparse identification, while $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ are the vector fields that capture the underlying physical laws governing the system. We emphasize that the goal of using SINDy in this work is as a system identification tool, and the correct underlying physics need not necessarily be obtained exactly for the SINDy model to be accurate.

The fundamental assumption underlying SINDy is that the right-hand side of Eq. (6.4) typically comprises only a small number of nonlinear terms. As a result, when considering a large pool of potential nonlinear basis functions for $\hat{f}$ and $\hat{g}$, only a few terms will be active, with non-zero coefficients associated with them. This sparsity property of the candidate basis functions allows for efficient computation of the coefficients using convex optimization algorithms.

The application of SINDy begins with acquiring real-time data from the system of interest. This data can be collected through sensors in experimental or industrial setups, or generated from computer simulations based on theoretical models, such as first-principles or chemical process simulators. The collected data is then organized into two compact matrices: the data matrix X and the input matrix U,

$$X = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_{n_x}(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_{n_x}(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_{n_x}(t_m) \end{bmatrix} \tag{6.5a}$$

$$U = \begin{bmatrix} u_1(t_1) & u_2(t_1) & \cdots & u_{n_u}(t_1) \\ u_1(t_2) & u_2(t_2) & \cdots & u_{n_u}(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_m) & u_2(t_m) & \cdots & u_{n_u}(t_m) \end{bmatrix} \tag{6.5b}$$

where $x_i(t_\ell)$ represents the i^{th} state measurement at the ℓ^{th} sampling time, while $u_j(t_\ell)$ denotes the j^{th} input measurement at the ℓ^{th} sampling time. The indices i, j, and ℓ take values $i = 1, \ldots, n_x$, $j = 1, \ldots, n_u$, and $\ell = 1, \ldots, m$, respectively. The derivative of X, denoted as $\dot{X}$, is either directly measured or numerically estimated when direct measurement is not possible.

From the data matrices X and U, a function library matrix $\Theta(X, U)$ is constructed, containing p columns representing the nonlinear basis functions considered for the terms in $\hat{f}$ and $\hat{g}$. While polynomials and trigonometric functions are commonly used in engineering applications due to their universality, the basis set can be adapted based on performance and available knowledge of the system's structure. An example of a function library matrix that may be used as a starting point for system identification is as follows:

$$\Theta(X,U) = \begin{bmatrix} | & | & | & | & | & | & | \\ \mathbf{1} & X & \ln X & \sin(X) & \mathrm{e}^X & U & UX^2 \\ | & | & | & | & | & | & | \end{bmatrix} \tag{6.6}$$

Each candidate basis function associated with a variable or row in Eq. (6.4) is assigned a coefficient, and these coefficients are stored in the matrix $\Xi \in \mathbb{R}^{p\times n_x}$. The sparse identification algorithm solves the equation,

$$\dot{X} = \Theta(X,U)\Xi \tag{6.7}$$

to calculate the coefficients, Ξ. A popular method for solving this equation is the sequential thresholded least squares (STLSQ), where a threshold value known as the sparsification knob λ is specified, and coefficients in Ξ below λ are set to zero. Specifically, the least-squares problem associated with Eq. (6.7) can be formulated in the following general form:

$$\Xi = \arg\min_{\Xi'} \left\| \dot{X} - \Theta(X,U)\Xi' \right\|_2 + \lambda \|\Xi'\|_1 \tag{6.8}$$

where Ξ' is a notational substitute for Ξ, and the second term is an L_1 regularization term that enforces sparsity of Ξ. To implement the above step, we start by defining the matrix Ξ'' to be the matrix Ξ' with all coefficients with magnitudes below λ set to zero, which is the practical implementation of the L_1 regularization term in Eq. (6.8). Subsequently, the problem is reduced to the form,

$$\Xi = \arg\min_{\Xi''} \left\| \dot{X} - \Theta(X,U)\Xi'' \right\|_2 \tag{6.9}$$

which can be solved using, for example, MATLAB's built-in linear solver called with $A\backslash b$ where $A = \dot{X}$ and $b = \Theta(X, U)$. Equation (6.9) is solved repeatedly until the large/nonzero coefficients (greater than λ in each iteration) converge. Due to the efficiency of linear solvers as well as the sparse structure of Θ, the coefficients converge rapidly in this step. The STLSQ method is used in the current manuscript due to not only its efficiency but also simplicity, as described above, which facilitates the necessary modifications required for real-time model updates.

Finally, the calculated coefficient values in Ξ are then used to construct the continuous-time ordinary differential equation

$$\dot{x} = \Xi^\top (\Theta(x^\top, u^\top))^\top \tag{6.10}$$

where $\Theta(x^\top, u^\top)$ is not a matrix of data but a column vector of symbolic functions derived from the library of considered functions.

6.3.1.1 Solving for individual variables

The least-squares problem of Eq. (6.9) can also be solved not for the entire system but for a single variable of interest if only specific ODEs are required. From a programming perspective, it is identical whether the original full problem of Eq. (6.9) is solved to obtain all n_x ODEs in one computation or whether n_x problems are solved in a loop to identify the n_x ODEs. If Eq. (6.9) is solved individually for a variable we first define rows and columns of the relevant matrices as follows:

$$\dot{X} = [\dot{x}_1 \quad \dot{x}_2 \quad \cdots \quad \dot{x}_n] \tag{6.11a}$$

$$\Theta = [\theta_1 \quad \theta_2 \quad \cdots \quad \theta_p] \tag{6.11b}$$

$$\Xi = [\xi_1 \quad \xi_2 \quad \cdots \quad \xi_n] \tag{6.11c}$$

$$\Xi = \begin{bmatrix} \xi_1^\top \\ \xi_2^\top \\ \vdots \\ \xi_p^\top \end{bmatrix} \tag{6.11d}$$

where $\dot{x}_i \in \mathbb{R}^m$ and $\xi_i \in \mathbb{R}^p$ are the columns of $\dot{x}$ and Ξ, respectively, for $i = 1, \ldots, n_x$. The columns of Θ are denoted by $\theta_j \in \mathbb{R}^m$, while the rows of Ξ are represented by $\xi_j^\top \in \mathbb{R}^{n_x}$, for $j = 1, \ldots, p$. Using these notations, for the i^{th} variable, the sparse identification problem of Eq. (6.7) is of the form,

$$\dot{x}_i = \Theta(X, U)\xi_i \tag{6.12}$$

and the corresponding least-squares problem of Eq. (6.9) is

$$\xi_i = \arg\min_{\xi_i''} \left\| \dot{x}_i - \Theta(X, U)\xi_i'' \right\|_2 \tag{6.13}$$

where ξ_i represents the coefficients in front of each library function for the i^{th} variable, and ξ_i'' is the notational substitute for the vector ξ_i with all coefficients with magnitudes below λ set to zero. Solving the minimization of Eq. (6.13) for $i = 1, \ldots, n_x$ is identical to solving the full-state problem of Eq. (6.9) but more advantageous in terms of formulation when performing partial

SINDy modeling, i.e., model updates in real-time, as will be discussed in Section 6.4.2.

6.3.1.2 Scaling of library functions

In the case where the values of the functions in the library $\Theta(X, U)$ vary by orders of magnitudes for a data set, it may very likely be necessary to scale the library columns appropriately to yield a well-conditioned least-squares problem with a reasonable conditional number of $\Theta(X, U)$. When scaling the function library by a vector $\Lambda \in \mathbb{R}^p$, the sparse identification problem becomes

$$\dot{X} = \Theta(X, U)\Xi = \underbrace{\frac{\Theta(X, U)}{\Lambda}}_{\Theta_{\text{scaled}}} \underbrace{\Xi\Lambda}_{\Xi_{\text{scaled}}} \tag{6.14}$$

where Θ_{scaled} is the scaled library matrix with its i^{th} column divided by the i^{th} entry of Λ for $i = 1, \dots, p$. Similarly, Ξ_{scaled} is the scaled coefficient matrix where each of its rows has been multiplied by the corresponding scalar entry of Λ. The STLSQ iteration step of Eq. (6.9) must now use the scaled library and coefficient matrices, i.e.,

$$\Xi_{\text{scaled}} = \underset{\Xi''_{\text{scaled}}}{\arg\min} \left\| \dot{X} - \Theta_{\text{scaled}}(X, U)\Xi''_{\text{scaled}} \right\|_2 \tag{6.15}$$

Since Ξ_{scaled} is used in the STLSQ step, the threshold λ does not need to be scaled as all the entries of the scaled matrix Ξ_{scaled} are already of similar orders of magnitudes. At the end of the STLSQ algorithm, once Ξ_{scaled} has been calculated, the original Ξ can be recovered by dividing every row of Ξ_{scaled} by the corresponding value in the scaling vector Λ.

6.3.2 Lyapunov-based control using SINDy models

In this section, we outline the formulation of Lyapunov-based Model Predictive Control (LMPC) and Lyapunov-based Economic Model Predictive Control (LEMPC) utilizing SINDy models for future state prediction. Initially, a SINDy model is constructed to approximate the nonlinear dynamics of the system described by Eq. (6.1) within the operating region Ω_ρ using data obtained from extensive open-loop simulations. Subsequently, LMPC and LEMPC are developed by leveraging SINDy models to ensure closed-loop stability for the nonlinear system represented by Eq. (6.1).

In this study, we update the SINDy model given by Eq. (6.10) to capture the nonlinear dynamics of the system described by Eq. (6.1) in the presence of time-varying bounded disturbances (i.e., $|w(t)| \leq w_m$). Each SINDy model, denoted as $F^i_{si}(x,u)$ with $i = 1,2,...,N_T$, is updated using real-time data of closed-loop state trajectories and control actions. Here, N_T represents the total number of obtained SINDy models. We assume the existence of a set of stabilizing feedback controllers $u = \Phi^i_{si}(x) \in U$ that can render the origin of the SINDy models $F^i_{si}(x,u)$, with $i = 1,2,...,N_T$, of Eq. (6.10) exponentially stable within an open neighborhood $\hat{D}$ around the origin. Consequently, a $\mathcal{C}^1$ control Lyapunov function $\hat{V}(x)$ exists, satisfying the following inequalities for all x within $\hat{D}$:

$$\hat{c}^i_1|x|^2 \leq \hat{V}(x) \leq \hat{c}^i_2|x|^2, \tag{6.16a}$$

$$\frac{\partial \hat{V}(x)}{\partial x} F^i_{si}(x, \Phi^i_{si}(x)) \leq -\hat{c}^i_3|x|^2, \tag{6.16b}$$

$$\left|\frac{\partial \hat{V}(x)}{\partial x}\right| \leq \hat{c}^i_4|x| \tag{6.16c}$$

where $\hat{c}_1^i$, $\hat{c}_2^i$, $\hat{c}_3^i$, and $\hat{c}_4^i$ are positive constants, with $i = 1, 2, ..., N_T$. For simplicity, we will omit the superscript i in the symbols used to represent the SINDy models and controllers that satisfy Eq. (6.16) in the subsequent discussions. Similar to the approach used to characterize the closed-loop stability region Ω_ρ for the nonlinear system described by Eq. (6.1), we begin by characterizing a region denoted as $\hat{\phi}_u = \{x \in \mathbb{R}^n \mid \dot{\hat{V}}(x) < -\hat{c}_3|x|^2, u = \Phi_{si}(x) \in U\} \cup \{0\}$, from which the the origin of the SINDy model given by Eq. (6.10) can be rendered exponentially stable under the controller $u = \Phi_{si}(x) \in U$.

The closed-loop stability region for the SINDy model given by Eq. (6.10) is defined as a level set of the Lyapunov function within $\hat{\phi}_u$: $\Omega_{\hat{\rho}} := \{x \in \hat{\phi}_u \mid \hat{V}(x) \leq \hat{\rho}\}$, where $\hat{\rho} > 0$. It should be noted that $\Omega_{\hat{\rho}} \subseteq \Omega_\rho$ since the data set used to develop the SINDy model in Eq. (6.10) is generated from open-loop simulations with $x \in \Omega_\rho$ and $u \in U$. Additionally, there exist positive constants M_{si} and L_{si} such that the following inequalities hold for all $x, x' \in \Omega_{\hat{\rho}}$ and $u \in U$:

$$|F_{si}(x,u)| \leq M_{si} \tag{6.17a}$$

$$\left| \frac{\partial \hat{V}(x)}{\partial x} F_{si}(x,u) - \frac{\partial \hat{V}(x')}{\partial x} F_{si}(x',u) \right| \leq L_{si}|x - x'| \tag{6.17b}$$

Consider the existence of a bounded modeling error between the nominal system described by Eq. (6.1) and the SINDy model given by Eq. (6.10) (i.e., $|\nu| = |F(x,u,0) - F_{si}(x,u)| \leq \nu_m$, $\nu_m > 0$). The following proposition demonstrates that the feedback controller $u = \Phi_{si}(x) \in U$ can stabilize the nominal system of Eq. (6.1) if the modeling error is sufficiently small.

Proposition 6.1. *Under the assumption that the origin of the closed-loop SINDy model described by Eq.* (6.10) *is rendered exponentially stable under the controller* $u = \Phi_{si}(x) \in U$ *for all* $x \in$

$\Omega_{\hat{\rho}}$*, if there exists a positive real number* $\Gamma < \hat{c}_3/\hat{c}_4$ *that constrains the modeling error* $|\nu| = |F(x,u,0) - F_{si}(x,u)| \leq \Gamma|x| \leq \nu_m$ *for all* $x \in \Omega_{\hat{\rho}}$ *and* $u \in U$*, then the origin of the nominal closed-loop system described by Eq.* (6.1) *under* $u = \Phi_{si}(x) \in U$ *is also exponentially stable for all* $x \in \Omega_{\hat{\rho}}$.

Proof. To establish the exponential stability of the nominal system described by Eq. (6.1) under the controller based on the sparse-identified model from Eq. (6.10), we aim to demonstrate that the derivative of $\hat{V}$, which corresponds to the state of the nominal system, can be rendered negative for all x within the set $\Omega_{\hat{\rho}}$ under $u = \Phi_{si}(x) \in U$. By utilizing Eq. (6.16b) and Eq. (6.16c), we can compute the time-derivative of $\hat{V}$ as follows:

$$\begin{aligned}
\dot{\hat{V}} &= \frac{\partial \hat{V}(x)}{\partial x} F(x, \Phi_{si}(x), 0) \\
&= \frac{\partial \hat{V}(x)}{\partial x} \big(F_{si}(x, \Phi_{si}(x)) + F(x, \Phi_{si}(x), 0) - F_{si}(x, \Phi_{si}(x))\big) \\
&\leq -\hat{c}_3|x|^2 + \hat{c}_4|x|\big(F(x, \Phi_{si}(x), 0) - F_{si}(x, \Phi_{si}(x))\big) \\
&\leq -\hat{c}_3|x|^2 + \hat{c}_4|x|^2\nu_m
\end{aligned} \tag{6.18}$$

By appropriately selecting ν_m, such that $\nu_m < \hat{c}_3/\hat{c}_4$, we can ensure that $\dot{\hat{V}} \leq -\tilde{c}_3|x|^2 \leq 0$, where $\tilde{c}_3 = -\hat{c}_3 + \hat{c}_4\nu_m > 0$. This implies that the closed-loop state of the nominal system converges to the origin under $u = \Phi_{si}(x) \in U$ for all $x_0 \in \Omega_{\hat{\rho}}$. □

Upon integrating the SINDy model represented by Eq. (6.10) into the Lyapunov-based MPC designs, the control actions of the LMPC and LEMPC will be implemented using a sample-and-hold approach. Consequently, the subsequent propositions aim to establish the sample-and-hold characteristics of the Lyapunov-based controller $u = \Phi_{si}(x)$. Specifically, the next proposition derives an upper bound for the discrepancy between the states computed by the nominal system defined in Eq. (6.1) and the states predicted by the SINDy model given by Eq. (6.10).

Proposition 6.2. (c.f. proposition 3 in [3]) *For the nonlinear system described by* $\dot{x} = F(x, u, w)$ *in Eq.* (6.1) *and the SINDy model given by* $\dot{\hat{x}} = F_{si}(\hat{x}, u)$ *in Eq.* (6.10)*, assuming the same initial condition* $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$*, there exists a class* $\mathcal{K}$ *function* $f_w(\cdot)$ *and a positive constant* κ *such that the following inequalities hold for all* x *and* $\hat{x}$ *within* $\Omega_{\hat{\rho}}$*.*

$$|x(t) - \hat{x}(t)| \leq f_w(t) := \frac{L_w w_m + \nu_m}{L_x}\left(\mathrm{e}^{L_x t} - 1\right) \tag{6.19a}$$

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{6.19b}$$

Proof. Let us denote the error vector between the solutions of the system $\dot{x} = F(x, u, w)$ and the SINDy model $\dot{\hat{x}} = F_{si}(\hat{x}, u)$ as $e(t) = x(t) - \hat{x}(t)$. By taking the time derivative of $e(t)$, we obtain:

$$\begin{aligned} |\dot{e}(t)| &= |F(x, u, w) - F_{si}(\hat{x}, u)| \\ &\leq |F(x, u, w) - F(\hat{x}, u, 0)| + |F(\hat{x}, u, 0) - F_{si}(\hat{x}, u)| \end{aligned} \tag{6.20}$$

Using the Lipschitz condition from Eq. (6.3b), we have:

$$\begin{aligned} |F(x, u, w) - F(\hat{x}, u, 0)| &\leq L_x|x(t) - \hat{x}(t)| + L_w|w(t)| \\ &\leq L_x|x(t) - \hat{x}(t)| + L_w w_m \end{aligned} \tag{6.21}$$

The second term $|F(\hat{x}, u, 0) - F_{si}(\hat{x}, u)|$ in Eq. (6.20) represents the modeling error, which is bounded by $|\nu| \leq \nu_m$ for all $\hat{x} \in \Omega_{\hat{\rho}}$. Hence, combining Eq. (6.21) and the bound on the modeling error, we can bound $\dot{e}(t)$ as follows:

$$\begin{aligned} |\dot{e}(t)| &\leq L_x|x(t) - \hat{x}(t)| + L_w w_m + \nu_m \\ &\leq L|e(t)| + L_w w_m + \nu_m \end{aligned} \tag{6.22}$$

With the zero initial condition ($e(0) = 0$), we can bound the norm of the error vector for all $x(t), \hat{x}(t) \in \Omega_{\hat{\rho}}$ and $w(t) \leq w_m$:

$$|e(t)| = |x(t) - \hat{x}(t)| \leq \frac{L_w w_m + \nu_m}{L_x}(\mathrm{e}^{L_x t} - 1) \tag{6.23}$$

Next, to derive Eq. (6.19b) for all $x, \hat{x} \in \Omega_{\hat{\rho}}$, we expand $\hat{V}(x)$ using a Taylor series expansion around $\hat{x}$:

$$\hat{V}(x) \leq \hat{V}(\hat{x}) + \frac{\partial \hat{V}(\hat{x})}{\partial x}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{6.24}$$

where κ is a positive real number. Using Eq. (6.16a) and Eq. (6.16c), we can simplify Eq. (6.24) as follows:

$$\hat{V}(x) \leq \hat{V}(\hat{x}) \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{6.25}$$

This completes the proof of Proposition 6.2. □

6.3.2.1 LMPC using SINDy models

The formulation of a Lyapunov-based model predictive controller (LMPC) using a SINDy model can be expressed as follows [155]:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{x}(t), u(t)) \, \mathrm{d}t \tag{6.26a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{si}(\tilde{x}(t), u(t)) \tag{6.26b}$$

$$u(t) \in U, \; \forall \, t \in [t_k, t_{k+N}) \tag{6.26c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{6.26d}$$

$$\dot{\hat{V}}(x(t_k), u) \leq \dot{\hat{V}}\big(x(t_k), \Phi_{si}(x(t_k))\big), \text{ if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{si}} \tag{6.26e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{si}, \; \forall \, t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega_{\rho_{si}} \tag{6.26f}$$

where $\tilde{x}$ represents the predicted state trajectory. The set $S(\Delta)$ consists of piecewise constant functions with period Δ. N denotes the number of sampling periods in the prediction horizon. The term $\dot{\hat{V}}(x, u)$ denotes $\frac{\partial \hat{V}(x)}{\partial x} F_{si}(x, u)$. The optimal input trajectory computed by the LMPC, denoted as $u^*(t)$, is calculated over the entire prediction horizon $t \in [t_k, , t_{k+N})$. The control action computed for the first sampling period of the prediction horizon, $u^*(t_k)$, is applied during that period, and the LMPC is resolved at the next sampling time.

In the optimization problem defined by Eq. (6.26), the objective function in Eq. (6.26a) represents the integral of $L(\tilde{x}(t), u(t))$ over the prediction horizon. The constraint specified in Eq. (6.26b) describes the sparse-identified model of Eq. (6.10) used for state prediction in the closed-loop system. Equation (6.26c) defines the input constraints applied throughout the prediction horizon, and Eq. (6.26d) defines the initial condition $\tilde{x}(t_k)$ based on the state measurement at $t = t_k$. The constraint expressed in Eq. (6.26e) ensures that, if $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{si}}$, the closed-loop state will move towards the origin. However, once $x(t_k)$ enters $\Omega_{\rho_{si}}$, the states predicted by the SINDy model from Eq. (6.26b) will remain within $\Omega_{\rho_{si}}$ for the entire prediction horizon.

The proposition below demonstrates that the closed-loop state of the nominal system described in Eq. (6.1) remains bounded within the region $\Omega_{\hat{\rho}}$ for all times and can ultimately be bounded in a smaller subset $\Omega_{\rho_{\min}}$ that includes the origin. This result is obtained under the sample-and-hold implementation of the Lyapunov-based controller $u = \Phi_{si}(x) \in U$.

Proposition 6.3. *Consider the system described by Eq.* (6.1) *under the controller* $u = \Phi_{si}(\hat{x}) \in U$. *The controller is designed to stabilize the SINDy system represented by Eq.* (6.10) *and satisfies the conditions stated in Eq.* (6.16). *The controller operates in a sample-and-hold fashion, where* $u(t) = \Phi_{si}(\hat{x}(t_k))\ \forall\, t$ *within the interval* $[t_k, t_{k+1})$, *with* $t_{k+1} := t_k + \Delta$. *Let* ϵ_s, ϵ_w, Δ, *and* $\hat{\rho}$ *be positive values, and assume* $\rho_{\min}$, ρ_{si}, *and* ρ_s *satisfy the following conditions:*

$$-\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{si}M_{si}\Delta \leq -\epsilon_s \tag{6.27a}$$

$$-\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L'_x M\Delta + L'_w w_m \leq -\epsilon_w \tag{6.27b}$$

and

$$\rho_{si} := \max\{\hat{V}(\hat{x}(t+\Delta)) \mid \hat{x}(t) \in \Omega_{\rho_s}, u \in U\} \tag{6.28a}$$

$$\rho_{\min} \geq \rho_{si} + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}} f_w(\Delta) + \kappa(f_w(\Delta))^2 \tag{6.28b}$$

Then, the following inequality holds for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$*:*

$$\hat{V}(x(t)) \leq \hat{V}(x(t_k)),\ \forall t \in [t_k, t_{k+1}) \tag{6.29}$$

and the state $x(t)$ *of the nonlinear system of Eq.* (6.1) *is bounded in* $\Omega_{\hat{\rho}}$ $\forall$ t *for all times and ultimately bounded in* $\Omega_{\rho_{\min}}$.

Proof. Part 1: Let's assume that $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. We will now demonstrate that, under the controller $u(t) = \Phi_{si}(x(t_k)) \in U$, the value of $\hat{V}(\hat{x})$ decreases for $t \in [t_k, t_{k+1})$, where $x(t)$ and $\hat{x}(t)$ represent the solutions of the nonlinear system described by Eq. (6.1) in the presence of bounded disturbances and the SINDy system described by Eq. (6.10), respectively. We obtain the time-derivative of $\hat{V}(\hat{x})$ along the trajectory $\hat{x}(t)$ of the SINDy model within the interval $t \in [t_k, t_{k+1})$ as follows:

$$\begin{aligned}\dot{\hat{V}}(\hat{x}(t)) &= \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{si}(\hat{x}(t), \Phi_{si}(\hat{x}(t_k))) \\ &= \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{si}(\hat{x}(t_k), \Phi_{si}(\hat{x}(t_k))) + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{si}(\hat{x}(t), \Phi_{si}(\hat{x}(t_k))) \\ &- \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{si}(\hat{x}(t_k), \Phi_{si}(\hat{x}(t_k)))\end{aligned} \tag{6.30}$$

Using the inequalities of Eq. (6.16a) and Eq. (6.16b),

$$\begin{aligned}\dot{\hat{V}}(\hat{x}(t)) \leq -\frac{\hat{c}_3}{\hat{c}_2}\rho_s &+ \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{si}(\hat{x}(t), \Phi_{si}(\hat{x}(t_k))) \\ &- \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{si}(\hat{x}(t_k), \Phi_{si}(\hat{x}(t_k)))\end{aligned} \tag{6.31}$$

Utilizing the Lipschitz condition stated in Eq. (6.17) and considering the fact that $\hat{x} \in \Omega_{\hat{\rho}}$ and

$u \in U$, we can determine the upper bound of $\dot{\hat{V}}(\hat{x}(t))$ for all $t \in [t_k, t_{k+1})$ as follows:

$$\begin{aligned} \dot{\hat{V}}(\hat{x}(t)) &\leq -\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{si}|\hat{x}(t) - \hat{x}(t_k)| \\ &\leq -\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{si}M_{si}\Delta \end{aligned} \tag{6.32}$$

Hence, when Eq. (6.27a) is fulfilled, the subsequent inequality is valid for any $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(\hat{x}(t)) \leq -\epsilon_s \tag{6.33}$$

By integrating the aforementioned equation over $t \in [t_k, t_{k+1})$, we can conclude that $V(\hat{x}(t_{k+1})) \leq V(\hat{x}(t_k)) - \epsilon_s\Delta$. Hence, we have established that, for all $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, the state of the closed-loop SINDy system described by Eq. (6.10) remains bounded within the closed-loop stability region $\Omega_{\hat{\rho}}$ at all times and converges towards the origin when the controller $u = \Phi_{si}(\hat{x}) \in U$ is implemented in a sample-and-hold manner.

However, it should be noted that Eq. (6.33) may not hold when $x(t_k) = \hat{x}(t_k) \in \Omega_{\rho_s}$. This implies that the state may exit the region Ω_{ρ_s} within a single sampling period. To address this, we design $\Omega_{\rho_{si}}$ based on Eq. (6.28a) to ensure that the closed-loop state $\hat{x}(t)$ of the SINDy model remains within $\Omega_{\rho_{si}}$ for all $t \in [t_k, t_{k+1})$, $u \in U$, and $\hat{x}(t_k) \in \Omega_{\rho_s}$ during a sampling period. If the state $\hat{x}(t_{k+1})$ exits Ω_{ρ_s}, the controller $u = \Phi_{si}(x(t_{k+1}))$ will guide the state back towards Ω_{ρ_s} in the subsequent sampling period, as Eq. (6.33) is satisfied again at $t = t_{k+1}$. Consequently, we have demonstrated the convergence of the state to $\Omega_{\rho_{si}}$ for the closed-loop SINDy system described by Eq. (6.10) for all initial states $\hat{x}_0 \in \Omega_{\hat{\rho}}$. The next step is to establish that the closed-loop state of the actual nonlinear system governed by Eq. (6.1) can be bounded within $\Omega_{\hat{\rho}}$ for all times and ultimately bounded within a small neighborhood around the origin when the controller $u = \Phi_{si}(x) \in U$ is implemented using the sample-and-hold technique.

Part 2: Building upon the previous analysis of the SINDy system represented by Eq. (6.10), we now consider the nonlinear system described by Eq. (6.1) with $x(t_k) = \hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. We derive the time-derivative of $\hat{V}(x)$ for this nonlinear system, taking into account the presence of bounded disturbances w (with $|w| \leq w_m$), as shown below:

$$\begin{aligned}\dot{\hat{V}}(x(t)) &= \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{si}(x(t_k)), w) \\ &= \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k)), 0) + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{si}(x(t_k)), w) \\ &\quad - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k)), 0)\end{aligned} \tag{6.34}$$

By referring to Eq. (6.19), we have $\frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k)), 0) \leq -\tilde{c}_3 |x(t_k)|^2$ for all $x \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$. Utilizing Eq. (6.16a) and the Lipschitz condition stated in Eq. (6.17), we can derive the following inequality for $\dot{\hat{V}}(x(t))$ within the time interval $t \in [t_k, t_{k+1})$, given that $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$:

$$\begin{aligned}\dot{\hat{V}}(x(t)) &\leq -\frac{\tilde{c}_3}{\hat{c}_2} \rho_s + \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{si}(x(t_k)), w) - \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k)), 0) \\ &\leq -\frac{\tilde{c}_3}{\hat{c}_2} \rho_s + L'_x |x(t) - x(t_k)| + L'_w |w| \\ &\leq -\frac{\tilde{c}_3}{\hat{c}_2} \rho_s + L'_x M \Delta + L'_w w_m\end{aligned} \tag{6.35}$$

If the condition stated in Eq. (6.27b) is fulfilled, we can establish the following inequality for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(x(t)) \leq -\epsilon_w \tag{6.36}$$

From the inequality in Eq. (6.36), we can conclude that Eq. (6.29) holds, ensuring that the state of the closed-loop system described by Eq. (6.1) remains within the region $\Omega_{\hat{\rho}}$ for all times. Furthermore, this implies that the controller $u = \Phi_{si}(x)$ is capable of driving the state of the actual nonlinear system given by Eq. (6.1) towards the origin within each sampling period.

In addition to the above, if the initial state $x(t_k)$ belongs to the set Ω_{ρ_s}, we have already demonstrated in Part 1 that the state of the SINDy model described by Eq. (6.10) remains within the region $\Omega_{\rho_{si}}$ within one sampling period. Taking into account the bounded error between the SINDy model state and the actual nonlinear system state, as indicated by Eq. (6.19a), we can define a compact set $\Omega_{\rho_{\min}}$ such that $\Omega_{\rho_{si}} \subset \Omega_{\rho_{\min}}$, satisfying the condition stated in Eq. (6.28b). This guarantees that the state of the actual nonlinear system does not exit $\Omega_{\rho_{\min}}$ during a sampling period if the state of the SINDy model remains bounded within $\Omega_{\rho_{si}}$. If the state $x(t)$ enters the set $\Omega_{\rho_{\min}} \backslash \Omega_{\rho_s}$, we have shown that Eq. (6.36) holds, and thus, the state will be driven towards the origin again during the next sampling period under the controller $u = \Phi_{si}(x)$.

By establishing the above arguments, we have completed the proof of Proposition 6.3, demonstrating that for any initial state $x_0 = \hat{x}_0 \in \Omega_{\hat{\rho}}$, the closed-loop state trajectories of the nonlinear system described by Eq. (6.1) remain within the region $\Omega_{\hat{\rho}}$ and ultimately become bounded within $\Omega_{\rho_{\min}}$, provided that the assumptions of Proposition 6.3 are satisfied. $\square$

The aforementioned stability region, the various Lyapunov level sets and an example of a closed-loop state trajectory under the LMPC are depicted in Fig. 6.1.

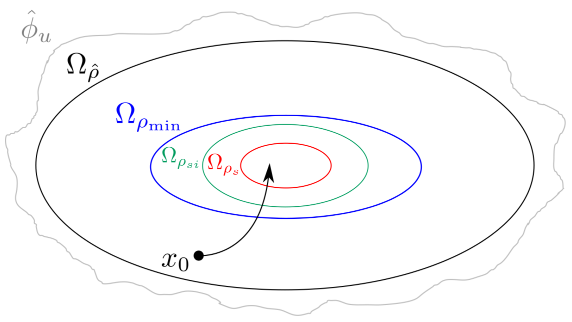


Figure 6.1: A diagram illustrating the sets $\hat{\phi}_u$, $\Omega_{\hat{\rho}}$, $\Omega_{\rho_{\min}}$, $\Omega_{\rho_{si}}$, and Ω_{ρ_s} in concentric ellipses, from outermost to innermost. The LMPC control strategy of Eq. (6.26) guides the closed-loop state towards the origin and ensures that it eventually remains bounded within $\Omega_{\rho_{\min}}$ for any initial state $x_0 \in \Omega_{\hat{\rho}}$.

6.3.2.2 LEMPC using SINDy models

The Lyapunov-based economic model predictive control (LEMPC) approach utilizing a SINDy model is designed to dynamically optimize the economic benefits of a process while ensuring that the closed-loop state remains within a specified stability region at all times [55]. The LEMPC can be formulated as the following optimization problem:

$$\mathcal{J} = \max_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} l_e(\tilde{x}(t), u(t))\, \mathrm{d}t \tag{6.37a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{si}(\tilde{x}(t), u(t)) \tag{6.37b}$$

$$u(t) \in U,\ \forall\, t \in [t_k, t_{k+N}) \tag{6.37c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{6.37d}$$

$$\hat{V}(\tilde{x}(t)) \le \hat{\rho}_e,\ \forall\, t \in [t_k, t_{k+N}),\ \text{if } x(t_k) \in \Omega_{\hat{\rho}_e} \tag{6.37e}$$

$$\dot{\hat{V}}(x(t_k), u) \le \dot{\hat{V}}\big(x(t_k), \Phi_{si}(x(t_k))\big),\ \text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\hat{\rho}_e} \tag{6.37f}$$

The notation used in Eq. (6.37) follows that of Eq. (6.26). The optimization problem presented in Eq. (6.37) aims to maximize the time integral of the stage cost function, denoted as $l_e(\tilde{x}(t), u(t))$, over the prediction horizon. The prediction model described in Eq. (6.37b) and the initial condition given in Eq. (6.37d) are the same as those used in the LMPC formulation of Eq. (6.26). The constraint stated in Eq. (6.37e) ensures that the predicted closed-loop states remain within the region $\Omega_{\hat{\rho}_e}$ over the prediction horizon when the initial state $x(t_k)$ is inside this region. However, if $x(t_k)$ enters the region $\Omega_{\hat{\rho}} \backslash \Omega_{\hat{\rho}_e}$, the contractive constraint expressed in Eq. (6.37f) drives the state towards the origin during the next sampling period, ultimately causing the state to enter $\Omega_{\hat{\rho}_e}$ within a finite number of sampling periods.

Proposition 6.4. *Consider the system described by Eq.* (6.1) *subject to the controller* $u = \Phi_{si}(\hat{x}) \in U$. *This controller satisfies the conditions specified in Eq.* (6.16) *and is implemented using a sample-and-hold approach, meaning that* $u(t) = \Phi_{si}(\hat{x}(t_k))\, \forall t \in [t_k, t_{k+1})$, *where* $t_{k+1} := t_k + \Delta$. *Further, let* ϵ_w *and* ϵ_s *be positive values, and let* Δ *be a positive time interval. Consider also that* $\hat{\rho} > \hat{\rho}_e > \rho_s > 0$, *satisfying the conditions given by the following equations:*

$$-\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{si}M_{si}\Delta \leq -\epsilon_s \tag{6.38a}$$

$$-\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L'_x M\Delta + L'_w w_m \leq -\epsilon_w \tag{6.38b}$$

$$\hat{\rho}_e > \max\{\hat{V}(\hat{x}(t_k + \Delta)) \mid \hat{x}(t_k) \in \Omega_{\rho_s}, u \in U, w \in W\} \tag{6.38c}$$

In that case, for any $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$*, the following inequalities hold:*

$$\hat{V}(\hat{x}(t)) \leq \hat{V}(\hat{x}(t_k)),\ \forall t \in [t_k, t_{k+1}) \tag{6.39a}$$

$$\hat{V}(x(t)) \leq \hat{V}(x(t_k)),\ \forall t \in [t_k, t_{k+1}) \tag{6.39b}$$

Proof. In order to demonstrate the decreasing value of $\hat{V}$ along the trajectory $\hat{x}(t)$ of the SINDy model described by Eq. (6.10) over the interval $t \in [t_k, t_{k+1})$, we evaluate the time derivative of $\hat{V}(\hat{x})$ with respect to $\hat{x}(t)$ as follows:

$$\begin{aligned} \dot{\hat{V}}(\hat{x}(t)) &= \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{si}(\hat{x}(t), \Phi_{si}(\hat{x}(t_k))) \\ &= \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{si}(\hat{x}(t_k), \Phi_{si}(\hat{x}(t_k))) \\ &+ \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{si}(\hat{x}(t), \Phi_{si}(\hat{x}(t_k))) \\ &- \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{si}(\hat{x}(t_k), \Phi_{si}(\hat{x}(t_k))) \end{aligned} \tag{6.40}$$

By exploiting the Lyapunov constraints of Eqs. (6.16a) and (6.16b) and the Lipschitz condition of Eq. (6.17), we obtain the following inequalities:

$$\begin{aligned} \dot{\hat{V}}(\hat{x}(t)) &\leq -\frac{\hat{c}_3}{\hat{c}_2}\rho_s + \frac{\partial \hat{V}(\hat{x}(t))}{\partial \hat{x}} F_{si}(\hat{x}(t), \Phi_{si}(\hat{x}(t_k))) \\ &- \frac{\partial \hat{V}(\hat{x}(t_k))}{\partial \hat{x}} F_{si}(\hat{x}(t_k), \Phi_{si}(\hat{x}(t_k))) \\ &\leq -\frac{\hat{c}_3}{\hat{c}_2}\rho_s + L_{si} M_{si} \Delta \end{aligned} \tag{6.41}$$

Hence, if the condition of Eq. (6.38a) is satisfied, we have $\dot{\hat{V}}(\hat{x}(t)) \leq -\epsilon_s$ for all $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$. Integrating the above inequality leads to $\hat{V}(\hat{x}(t)) \leq \hat{V}(\hat{x}(t_k)) - \Delta\epsilon_s$, for all $\hat{x}(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$ (referred to as Eq. (6.39a)).

To establish the inequality $\hat{V}(x(t)) \leq \hat{V}(x(t_k))$ for all $t \in [t_k, t_{k+1})$, we derive the time-derivative of $\hat{V}(x)$ for the nonlinear system described by Eq. (6.1) (where $\dot{x} = F(x, u, w)$) in the presence of bounded disturbances (i.e., $|w(t)| \leq w_m$) as follows:

$$\begin{aligned}\dot{\hat{V}}(x(t)) &= \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{si}(x(t_k)), w) \\ &= \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k)), 0) \\ &+ \frac{\partial \hat{V}(x(t))}{\partial x} F(x(t), \Phi_{si}(x(t_k)), w) \\ &- \frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k)), 0)\end{aligned} \tag{6.42}$$

By utilizing Eq. (6.19), which states that $\frac{\partial \hat{V}(x(t_k))}{\partial x} F(x(t_k), \Phi_{si}(x(t_k)), 0) \leq -\tilde{c}_3 |x(t_k)|^2$ holds for all $x \in \Omega_{\hat{\rho}}$, we can derive the following inequality for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$ using Eq. (6.16a) and the Lipschitz condition given by Eq. (6.17):

$$\begin{aligned}\dot{\hat{V}}(x(t)) &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L'_x |x(t) - x(t_k)| + L'_w |w| \\ &\leq -\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L'_x M \Delta + L'_w w_m\end{aligned} \tag{6.43}$$

Therefore, if the condition of Eq. (6.38b) is satisfied, we can derive the following inequality $\forall\, x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $\forall\, t \in [t_k, t_{k+1})$:

$$\dot{\hat{V}}(x(t)) \leq -\epsilon_w \tag{6.44}$$

Likewise, this implies that $\hat{V}(x(t)) \leq \hat{V}(x(t_k)) - \Delta\epsilon_w$ holds for all $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$ and $t \in [t_k, t_{k+1})$. Hence, if $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_s}$, the state of the nonlinear system described by Eq. (6.1) will enter Ω_{ρ_s} within a finite number of sampling periods. Moreover, if $x(t_k) \in \Omega_{\rho_s}$, where Eq. (6.43) and Eq. (6.44) do not hold, Eq. (6.38c) ensures that the state will not exit $\Omega_{\hat{\rho}_e}$ within one sampling period for any $u \in U$ and $w \in W$. If the state $x(t_{k+1})$ exits Ω_{ρ_s} but remains within $\Omega_{\hat{\rho}_e}$, then at the subsequent sampling period $t \in [t_{k+1}, t_{k+2})$, Eq. (6.44) is satisfied again, causing the state to be driven toward the origin. Consequently, the state of the nonlinear system given by Eq. (6.1) remains bounded within $\Omega_{\hat{\rho}}$ at all times. □

6.4 Error-triggered online update of SINDy models

In this section, we apply the LMPC the LEMPC methods described by Eq. (6.26) and Eq. (6.37), respectively, to the nonlinear system of Eq. (6.1) in the presence of bounded disturbances (i.e., $|w(t)| \leq w_m$) that grows due to changes in the dynamics of the nonlinear system of Eq. (6.1), potentially leading to instability in the closed-loop system. To mitigate the impact of disturbances, we update SINDy models through online learning to capture the nonlinear dynamics of the system described by Eq. (6.1) while accounting for the influence of disturbances $w(t)$. The subsequent subsections introduce the error-triggering mechanism employed for updating the SINDy models to ensure $|w(t)| \leq w_m$ for all time.

6.4.1 Error-triggering mechanism

In this subsection, we develop an event-triggering mechanism based on the errors between predicted states and measured states to update the SINDy model for all $x \in \Omega_{\hat{\rho}}$. This mechanism is referred to as the error-triggered on-line SINDy update throughout the manuscript. Specifically, following the error-triggering mechanism proposed in [227], we introduce a moving horizon error metric denoted as $e_d(t_k)$, which indicates the prediction accuracy of the SINDy model at time $t = t_k$.

$$e_d(t_k) = \sum_{r=0}^{N_w} \sum_{j=1}^{n_x} \frac{|x_{p,j}(t_{k-r}) - x_j(t_{k-r})|}{|x_j(t_{k-r})|} \tag{6.45}$$

where the quantity N_w represents the number of sampling periods prior to t_k that contribute to the estimation of the prediction error. At each sampling period between t_{k-N_w} and t_k, $x_j(t_{k-r})$ captures the historical measurements of the process states, where r ranges from 0 to N_w. Similarly, $x_{p,j}(t_{k-r})$ represents the predictions of the past states of the system obtained from the SINDy model. The moving horizon error detector triggers a model update if/when the error metric e_d surpasses a predefined threshold $e_{d,T}$, i.e., the following condition:

$$e_d(t_k) > e_{d,T} \tag{6.46}$$

Determining the parameters for this error-triggered approach involves defining the number of input and output data points N_d that should be retained for model update when it is triggered, the length N_w of the moving horizon used to calculate the error metric e_d, and the threshold $e_{d,T}$ that dictates when a model update should be initiated.

The following strategy is proposed for the selection of the parameter,

- N_d: The SINDy algorithm is highly dependent on the amount of dynamic information present in a data set rather than simply its size. While the initial SINDy model must be developed with a large data set consisting of many open-loop experiments/simulations, the model update procedure must use a much smaller number of data points to update the coefficients because, firstly, the updates must occur relatively soon after the disturbance takes effect and, secondly, the model is only being updated rather than being identified. Hence, the amount of data required for the initial offline model building procedure has no impact on the amount of data required to efficiently update the model in real-time. The actual amount of data will depend also on the sampling period since a smaller sampling period can yield a large volume of data within a short period of data acquisition. In summary, the value of N_d is highly process-specific, and some insights will be provided in the application section. We note, however, that even if N_d is relatively large for a certain process (or region of a process), one approach to mitigate process deterioration during the time between the error-triggering and the SINDy model update is to use a linear data-driven model as a stopgap solution [166].

- N_w: The selection of the appropriate value for N_w in the calculation of e_d requires finding a balance. On one hand, N_w should be sufficiently long to ensure that common disturbances during normal operation do not significantly impact e_d, which could lead to unnecessary triggering of errors. On the other hand, N_w should not be longer than necessary to avoid unnecessary data storage and processing. One approach to determine N_w is by evaluating the value of $e_d(t_k)$ at each sampling period for a set of input/output data collected during typical process operation. This evaluation is performed in closed-loop under the SINDy-based MPC, focusing on the region of operation where the initial SINDy model was developed and validated. By repeating this calculation for different values of N_w, it becomes possible to observe the range of minimum and maximum values of e_d. If N_w is small, the minimum and maximum values of e_d may differ significantly since any disturbance or measurement noise within the moving horizon has a considerable impact on e_d. However, as N_w increases, the influence of disturbances and measurement noise becomes less significant. At some point, the minimum and maximum values of e_d are expected to stabilize, indicating that further increases in N_w have minimal effect. In such a case, the smallest value of N_w for which the minimum and maximum values of e_d reach their approximate final value can be selected for use in Eq. (6.45). It is important to note that the value of N_w for a given process depends on the statistical properties of $w(t)$ and its influence on the system. Therefore, careful consideration of these factors is necessary when determining the appropriate value of N_w.

- $e_{d,T}$: The determination of the threshold value $e_{d,T}$ is performed offline, taking into account the chosen value of N_w. The goal is to set a threshold that avoids triggering model updates in the presence of measurement noise, small constant disturbances, and time-varying disturbances that still result in reasonably accurate predictions using the current model. One approach to achieve this is by analyzing the statistical properties of e_d using a set of closed-loop input/output data corresponding to normal process operation within the region where the initial SINDy model was developed and validated. For example, the maximum value of e_d can be calculated using the selected value of N_w. The threshold can then be set to be a reasonable percentage higher than this maximum value of e_d observed in the normal operating data. This approach ensures that the threshold includes disturbances and measurement noise that regularly affect the system. Alternatively, other statistical measures could be used, such as setting $e_{d,T}$ to be several standard deviations above the mean value of e_d calculated from the normal operating data. The choice of the appropriate measure depends on the specific system being analyzed. It is important to note that, even in the absence of disturbances or measurement noise, e_d may have a non-zero value if the SINDy model captured the dynamics of the system using a different set of basis functions than the actual nonlinearities of the process in consideration. If the exact basis functions and coefficients were correctly identified by the SINDy model, however, e_d can be expected to be very close to zero.

From a practical perspective, due to the continuous, online monitoring of the process performance, even if some parameters are not chosen optimally from the beginning, they can be adjusted based on the incoming data, and the lack of explicit formulae to determine the aforementioned parameters is not a limitation of the adaptive SINDy framework.

Remark 6.1. *While Eq.* (6.45) *assumes full-state feedback, if only output measurements are available, the SINDy modeling framework itself can be modified to obtain not the state derivative but the output measurements as functions of the states, i.e.,* $y = f(x, u)$*, where* y *are the output measurements. Such an approach has been proposed in [222] under the name of generalized SINDy. Hence, using output measurements as the target variables and nonlinear functions of the states and manipulated inputs as the library functions, once relationships are obtained for the output measurements, the predicted output measurements can be used to compute the error metric in Eq.* (6.45) *instead.*

6.4.2 SINDy model update procedure

Once the SINDy model update is triggered by the mechanisms of Section 6.4, the model update is carried out using the last N_d data points. The details of the update procedure are discussed in this subsection.

It is assumed that the structure of the original SINDy model obtained offline does not change due to the presence of disturbances or changes in the process. This is a practical assumption since many changes such as catalyst deactivation or fluctuations in feed flow rates would not add or remove terms from the process model but simply alter the terms, specifically the coefficients associated with the respective terms. Hence, the problem of updating SINDy models in real-time can be reduced to re-identifying a subset of the coefficients in the matrix Ξ. In doing so, however, the remaining coefficients must remain at their original values.

For real-time model updates of q out of p coefficients for each variable, the matrices $\Theta(X, U)$ and Ξ must be split into the part that will remain unchanged and the part that will be re-identified as follows:

$$\Theta = [\Theta_{\text{fixed}} \quad \Theta_{\text{update}}], \qquad \Xi = [\Xi_{\text{fixed}} \quad \Xi_{\text{update}}]^\top \tag{6.47}$$

where $\Theta_{\text{fixed}} \in \mathbb{R}^{m\times(p-q)}$ and $\Theta_{\text{update}} \in \mathbb{R}^{m\times q}$ contain the $m \times 1$ columns of Θ that are fixed and that are updated, respectively. Similarly, $\Xi_{\text{fixed}} \in \mathbb{R}^{(p-q)\times n_x}$ and $\Xi_{\text{update}} \in \mathbb{R}^{q\times n_x}$ contain the corresponding $1 \times n_x$ rows of Ξ. The sparse identification problem then becomes

$$\dot{X} = \Theta\Xi = \Theta_{\text{fixed}}\Xi_{\text{fixed}} + \Theta_{\text{update}}\Xi_{\text{update}} \tag{6.48}$$

which can be reformulated into the form,

$$\dot{X} - \Theta_{\text{fixed}}\Xi_{\text{fixed}} = \Theta_{\text{update}}\Xi_{\text{update}} \tag{6.49}$$

In Eq. (6.49), the left-hand side is a matrix that can be evaluated using the fixed part of the current

SINDy model, and Θ_{update} in the right-hand side of Eq. (6.49) is also known. Therefore, STLSQ can be used to solve for the only unknown in Eq. (6.49), which is Ξ_{update}, using the same procedure as described in Section 6.3.1. Specifically, the least-squares formulation now solves for Ξ_{update} as follows:

$$\Xi_{\text{update}} = \underset{\Xi''_{\text{update}}}{\arg\min} \left\| \dot{X} - \Theta_{\text{fixed}}\Xi_{\text{fixed}} - \Theta_{\text{update}}\Xi''_{\text{update}} \right\|_2 \tag{6.50}$$

In the above formulation, the notations use the full matrices for the parts of Ξ, which is done for simplicity. However, the same library terms do not need to be fixed between the different variables/ODEs. Since the STLSQ algorithm solves for the coefficients of each state variable sequentially in a loop, the different terms to be updated can be easily incorporated into the solution. Specifically, instead of altering the same q library function coefficients for every variables, let the i^{th} variable update q_i coefficients in its ODE right-hand side ($i = 1, \ldots, n_x$). In this case, the least-squares problem associated with the partial SINDy update for the i^{th} variable is expressed as follows:

$$\dot{x}_i - \Theta_{i,\text{fixed}}\Xi_{i,\text{fixed}} = \Theta_{i,\text{update}}\Xi_{i,\text{update}} \tag{6.51}$$

For example, consider a system where the first and third library functions are to be updated for the first variable ($q_1 = 2$) but only the second library function for the second variable ($q_2 = 1$). In this case, the least-squares problems to be solved for identifying the updated SINDy coefficients for the ODEs modeling the dynamics of the first two variables would be

$$\dot{x}_1 - [\theta_2 \quad \theta_4 \quad \theta_5 \quad \cdots \quad \theta_p] \begin{bmatrix} \xi_{2,1} \\ \xi_{4,1} \\ \xi_{5,1} \\ \vdots \\ \xi_{p,1} \end{bmatrix} = [\theta_1 \quad \theta_3] \begin{bmatrix} \xi_{1,1} \\ \xi_{3,1} \end{bmatrix} \tag{6.52}$$

and

$$\dot{x}_2 - [\theta_1 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \cdots \quad \theta_p] \begin{bmatrix} \xi_{1,2} \\ \xi_{3,2} \\ \xi_{4,2} \\ \xi_{5,2} \\ \vdots \\ \xi_{p,2} \end{bmatrix} = [\theta_2] \begin{bmatrix} \xi_{2,2} \end{bmatrix} \tag{6.53}$$

respectively, where $\xi_{i,j}$ denotes the i^{th} row of the j^{th} column of Ξ. Solving Eq. (6.52) will yield the updated values of $\xi_{1,1}$ and $\xi_{3,1}$, while solving Eq. (6.53) outputs the updated scalar, $\xi_{2,2}$. In this manner, any number and choice of library functions' coefficients can be updated using the partial re-identification algorithm described in this subsection.

6.4.3 Implementation strategy for error-triggered on-line model identification

After determining the values of N_d, N_w, and $e_{d,T}$ using the methodology described in the previous section, the implementation strategy for the proposed error-triggered on-line model identification is as follows:

Step 1: An initial SINDy model that captures the nonlinear process behavior in the operating region is developed using data from extensive open-loop simulations. This model is used to design the model predictive controller.

Step 2: The system is operated under the MPC that is designed based on the current SINDy model. During operation, N_d values of input/output data are collected and stored for potential future model identification. At t_{N_w}, the moving horizon error detector is activated to compute $e_d(t_k)$.

Step 3: When the current SINDy model becomes inadequate in capturing the dynamics of the process (due to factors such as plant variations or changes in the operating region), the value of $e_d(t_k)$ will rise. Once $e_d(t_k)$ surpasses the threshold $e_{d,T}$, the latest set of N_d input and output data values (collected up until time t_k) are employed for on-line model update. This newly updated SINDy model then replaces the current SINDy model and is utilized as the process model in the MPC.

Step 4: Steps 2 and 3 are repeated as process operation continues.

6.4.4 Stability analysis of error-triggered feedback systems

In this section, we use the propositions developed in Sections 6.3.2.1 and 6.3.2.2 to develop closed-loop stability results for the nonlinear system of Eq. (6.1) under the Lyapunov-based controllers.

6.4.4.1 Stability analysis for Lyapunov-based tracking MPC

Based on the LMPC formulation given by Eq. (6.26), the following theorem establishes that the LMPC optimization problem can be solved with recursive feasibility, ensuring closed-loop stability of the nonlinear system described by Eq. (6.1) when implementing the optimal control actions computed by LMPC using a sample-and-hold strategy (i.e., $u(t) = \Phi_{si}(\hat{x}(t_k))$, $\forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$ and Δ is the sampling period).

Theorem 6.1. *Consider the closed-loop system of Eq.* (6.1) *under the LMPC of Eq.* (6.26) *with on-line updates of the SINDy model. Let* $\Delta > 0$, $\epsilon_w > 0$ *and* $\hat{\rho} > \rho_{\min} > \rho_s$ *satisfy Eq.* (6.27) *and Eq.* (6.28). *Then, given any initial state* $x_0 \in \Omega_{\hat{\rho}}$, *if the conditions of Proposition 6.2 and Proposition 6.3 are satisfied, and if the SINDy model is updated following the implementation strategy in this section with the triggering event of Eq.* (6.46)*, then it is guaranteed that the LMPC of Eq.* (6.26) *has a feasible solution and that under the LMPC of Eq.* (6.26)*,* $x(t) \in \Omega_{\hat{\rho}}$, $\forall t \geq 0$, *and* $\lim_{t\to\infty} \hat{V}(x(t)) \leq \rho_{\min}$ *for the closed-loop system of Eq.* (6.1).

Proof. We will demonstrate that the optimization problem described by Eq. (6.26) is recursively feasible for all $x \in \Omega_{\hat{\rho}}$. Specifically, if at time $t = t_k$ the state $x(t_k)$ satisfies $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{si}}$, the control action $u(t) = \Phi_{si}(x(t_k)) \in U$, for $t \in [t_k, t_{k+1})$, obtained based on the state measurement $x(t_k)$, satisfies both the input constraint defined in Eq. (6.26c) and the Lyapunov-based constraint given by Eq. (6.26e). Moreover, if $x(t_k) \in \Omega_{\rho_{si}}$, the control actions computed using $\Phi_{si}(x(t_{k+i}))$ for $i = 0, 1, \ldots, N-1$ satisfy the input constraint in Eq. (6.26c) as well as the Lyapunov-based constraint specified in Eq. (6.26f). This result is derived from Proposition 6.3, which shows that the predicted states by the SINDy model defined in Eq. (6.26b) remain within $\Omega_{\rho_{si}}$ when subjected to the controller $\Phi_{si}(x)$. Consequently, for any initial state $x_0 \in \Omega_{\hat{\rho}}$, the LMPC optimization problem of Eq. (6.26) can be solved with recursive feasibility if the condition $x(t) \in \Omega_{\hat{\rho}}$ holds for all time instances.

We will now demonstrate that for any initial state $x_0 \in \Omega_{\hat{\rho}}$, the state of the closed-loop system described by Eq. (6.1) remains bounded within $\Omega_{\hat{\rho}}$ for all time and ultimately converges to a small neighborhood around the origin, denoted as $\Omega_{\rho_{\min}}$ and defined by Eq. (6.28b), under the LMPC controller specified by Eq. (6.26).

Let's consider the case where at time $t = t_k$, the state $x(t_k)$ satisfies $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{si}}$. In this situation, the constraint defined by Eq. (6.26e) is activated, ensuring that the control action u is selected to decrease the value of $\hat{V}(\hat{x})$ based on the predicted states provided by the SINDy model described by Eq. (6.26b) over the next sampling period. Additionally, according to Eq. (6.36), if the constraint of Eq. (6.26e) is satisfied, it follows that $\dot{\hat{V}}(x) \leq -\epsilon_w$ for $t \in [t_k, t_{k+1})$ after applying the control action $u^*(t_k)$ to the nonlinear system defined by Eq. (6.1). Therefore, the value of $\hat{V}(x)$ based on the state of the actual nonlinear system given by Eq. (6.1) decreases within the next sampling period, implying that the closed-loop state can be driven into $\Omega_{\rho_{si}}$ within a finite number of sampling steps.

Once the state enters $\Omega_{\rho_{si}}$, the constraint specified by Eq. (6.26f) is activated to ensure that the predicted states of the SINDy model in Eq. (6.26b) remain within $\Omega_{\rho_{si}}$ throughout the prediction horizon. Due to the existence of a mismatch between the SINDy model described by Eq. (6.26b) and the nonlinear system represented by Eq. (6.1), the state of the nonlinear system may exit $\Omega_{\rho_{si}}$ when subject to the constraint defined by Eq. (6.26f). However, by characterizing a region $\Omega_{\rho_{\min}}$ that satisfies Eq. (6.28b), Proposition 6.3 guarantees that if the predicted state by the SINDy model remains within $\Omega_{\rho_{si}}$, then the state $x(t)$ of the nonlinear system, for all $t \in [t_k, t_{k+1})$, will remain bounded within $\Omega_{\rho_{\min}}$. Consequently, at the next sampling step $t = t_{k+1}$, if the state $x(t_{k+1})$ is still bounded within $\Omega_{\rho_{si}}$, the constraint defined by Eq. (6.26f) ensures that the predicted state $\hat{x}$ of the SINDy model in Eq. (6.26b) remains within $\Omega_{\rho_{si}}$, guaranteeing that the actual state x of the nonlinear system described by Eq. (6.1) stays within $\Omega_{\rho_{\min}}$.

However, if $x(t_{k+1})$ belongs to $\Omega_{\rho_{\min}} \backslash \Omega_{\rho_{si}}$, similar to the proof provided for the case when $x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{si}}$, the constraint specified by Eq. (6.26e) is activated to drive the state towards the origin. This completes the proof that the states of the closed-loop system described by Eq. (6.1) remain bounded within $\Omega_{\hat{\rho}}$ and converge to $\Omega_{\rho_{\min}}$ for any initial state $x_0 \in \Omega_{\hat{\rho}}$.

We note that the theorem establishes that the closed-loop stability of the nonlinear system described by Eq. (6.1) is achieved through the implementation of the LMPC controller given by Eq. (6.26). This controller is designed based on the SINDy model represented by Eq. (6.10) and incorporates SINDy-based constraints.

It is important to note that the closed-loop state of the nonlinear system, as described by Eq. (6.1), can be driven to a small neighborhood around the origin. This is possible because the constraints defined by the LMPC controller in Eq. (6.26) ensure the decrease of $\hat{V}$ during each sampling period, accounting for various factors such as model mismatch (including the modeling error ν between the system in Eq. (6.1) and the SINDy model in Eq. (6.10)), the implementation of control actions using a sample-and-hold approach, and the presence of bounded disturbances $w(t)$ in Eq. (6.1). In other words, closed-loop stability can be maintained under the LMPC controller described by Eq. (6.26) if the modeling error ν, the sampling period Δ, and the disturbance bound w_m are sufficiently small. This requirement ensures the satisfaction of Proposition 6.2 and

Proposition 6.3.

However, the values of ν_m and w_m are generally not known as there is no formula to calculate them. Hence, if ν_m and w_m are not chosen to be sufficiently large and the process is subject to changes such that the modeling error $|\nu|$ exceeds ν_m and large enough disturbances such that the disturbance $|w|$ exceeds w_m, the condition of Eq. (6.28b) to quantify $\rho_{\min}$ may not hold, and the closed-loop state may exit $\rho_{\min}$. However, the goal of the error-triggering update of SINDy models is to mitigate the model mismatch such that, even if the process changes and $|\nu|$ increases, by choosing the value of $e_{d,T}$ conservatively, the error-triggering and model update procedure is rapidly carried out to reduce $|\nu|$ once again such that it never exceeds ν_m. In this way, the conditions of Proposition 6.3 are met and the proof provided in this theorem hold. Hence, under the sample-and-hold implementation of the LMPC with real-time SINDy model updates, the closed-loop state of the nonlinear system of Eq. (6.1) remains bounded in $\Omega_{\hat{\rho}}$ and ultimately converges to $\Omega_{\rho_{\min}}$. $\square$

6.4.4.2 Stability analysis for Lyapunov-based economic MPC

Theorem 6.2. *We examine the closed-loop system described by Eq.* (6.1) *when using the sample-and-hold implementation of the LEMPC of Eq.* (6.37) *with the stabilizing controller $\Phi_{si}(x)$ that fulfills Eq.* (6.16). *We consider parameters $\Delta > 0$, $\epsilon_w > 0$, and $\hat{\rho} > \hat{\rho}_e > 0$ that satisfy Eq.* (6.38) *along with the inequality,*

$$\hat{\rho}_e \leq \hat{\rho} - \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}} f_w(\Delta) - \kappa(f_w(\Delta))^2 \tag{6.54}$$

Assuming x_0 belongs to $\Omega_{\hat{\rho}}$ and the conditions stated in Proposition 6.2 and Proposition 6.4 are fulfilled, it is guaranteed that a feasible solution exists for the optimization problem presented *in Eq.* (6.37)*. Furthermore, the closed-loop state* $x(t)$ *remains bounded within the closed-loop stability region* $\Omega_{\hat{\rho}}$ *for all time* $t \geq 0$*.*

Proof. We will first establish the recursive feasibility of the optimization problem presented in Eq. (6.37) $\forall x \in \Omega_{\hat{\rho}}$. Specifically, if $x(t_k)$ is within $\Omega_{\hat{\rho}_e}$, the control actions $\Phi_{si}(x(t_{k+i}))$, where $i = 0, 1, \ldots, N-1$, satisfy both the input constraint of Eq. (6.37c) and the Lyapunov-based constraint of Eq. (6.37e). This is due to the fact that Eq. (6.39a) dictates that the states predicted by the SINDy model described in Eq. (6.37b) remain inside $\Omega_{\hat{\rho}_e}$ under the controller $\Phi_{si}(x)$.

Furthermore, if $x(t_k)$ falls within $\Omega_{\hat{\rho}} \backslash \Omega_{\hat{\rho}_e}$, the control action $u(t) = \Phi_{si}(x(t_k)) \in U$, for $t \in [t_k, t_{k+1})$, satisfies both the input constraint defined in Eq. (6.37c) and the Lyapunov-based constraint outlined in Eq. (6.37f). This ensures that the state can be driven towards the origin during the subsequent sampling period. As a result, the stabilizing controller $u = \Phi_{si}(x) \in U$ provides a feasible solution that adheres to all the constraints of the LEMPC optimization problem stated in Eq. (6.37) if $x(t)$ remains within $\Omega_{\hat{\rho}}$ for all time instances.

We will now establish that for $x_0 \in \Omega_{\hat{\rho}}$, the state of the closed-loop system described in Eq. (6.1) remains bounded within $\Omega_{\hat{\rho}}$ for all time instances. Specifically, if $x(t_k)$ falls within $\Omega_{\hat{\rho}_e}$, the predicted states $\hat{x}(t)$ obtained from the SINDy model presented in Eq. (6.37b) are guaranteed to stay within $\Omega_{\hat{\rho}_e}$ by adhering to the constraint specified in Eq. (6.37e). Based on Proposition 6.2, we can conclude that the actual state $x(t)$, where $t \in [t_k, t_{k+1})$, of the nonlinear system outlined in Eq. (6.1) is bounded by the following inequality:

$$\begin{aligned} \hat{V}(x) &\leq \hat{V}(\hat{x}) + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \\ &\leq \hat{V}(\hat{x}) + \frac{\hat{c}_4\sqrt{\hat{\rho}}}{\sqrt{\hat{c}_1}} f_w(\Delta) + \kappa(f_w(\Delta))^2 \end{aligned} \tag{6.55}$$

Hence, if we choose $\Omega_{\hat{\rho}_e}$ as a level set of $\hat{V}$ that satisfies the condition in Eq. (6.54), it guarantees that $V(x)$ based on the actual state $x(t)$ remains bounded within $\Omega_{\hat{\rho}}$ for all $t \in [t_k, t_{k+1})$. However, in the case where $x(t_k)$ lies within $\Omega_{\hat{\rho}} \backslash \Omega_{\hat{\rho}_e}$, the constraint stated in Eq. (6.37f) becomes active. Consequently, the control action u acts to decrease the value of $\hat{V}(\hat{x})$ based on the states predicted by the RNN model described in Eq. (6.37b) within the next sampling period.

From Eq. (6.39b) in Proposition 6.4, it follows that the value of $\hat{V}$ also decreases along the state trajectory of the actual nonlinear system described in Eq. (6.1) over $t \in [t_k, t_{k+1})$. Thus, we can conclude that for any initial condition within $\Omega_{\hat{\rho}}$, the closed-loop state of the system represented by Eq. (6.1) remains bounded within $\Omega_{\hat{\rho}} \forall t$ when subjected to the LEMPC of Eq. (6.37). $\square$

Remark 6.2. *We note that the condition of Eq.* (6.54)*, which dictates the size of the level set* $\hat{\rho}_e$*, includes the class* $\mathcal{K}$ *function* $f_w(\cdot)$*, which factors in the uncertainties from the modeling error and the disturbances,* ν *and* w*, respectively. Since the bounds on the modeling error and disturbance,* ν_m *and* w_m *are not estimable for a general system, if the process disturbance or plant-model mismatch is large and exceed the aforementioned bounds, the size of the level set* $\hat{\rho}_e$ *may be reduced, which can lead to lower economic benefits of the LEMPC. However,* $e_{d,T}$ *is chosen conservatively such that the SINDy model of Eq.* (6.37b) *is updated as soon as the error-triggering condition of Eq.* (6.46) *is violated, such that the modeling error* $|\nu|$ *is kept low throughout the operation under the LEMPC based on Theorem 6.2.*

6.5 Application of error-triggered on-line model update to plant variations: Application to a chemical process example

In this section, we present the application of the proposed error-triggered on-line model identification procedure to control a benchmark chemical reactor. The reactor experiences plant model changes, specifically catalyst deactivation. The system under consideration is a non-isothermal continuous stirred tank reactor (CSTR) involved in the catalytic conversion of reactant species A to product B (A $\rightarrow$ B). The reactor operates with an inlet concentration of A denoted by C_{A0}, an inlet temperature of T_0, and a feed volumetric flow rate of F. A heating jacket is employed in the CSTR to supply or remove heat at a rate of Q.

The dynamics of the CSTR are governed by material and energy balance equations given as follows:

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = \frac{F}{V}(C_{A0} - C_A) - k_0 \mathrm{e}^{\frac{-E}{RT}} C_A^2 \tag{6.56a}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 \mathrm{e}^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V} \tag{6.56b}$$

where C_A represents the concentration of reactant A in the reactor, V is the volume of the reacting liquid, T is the temperature of the reactor, and Q is the heat input rate. The concentration of reactant A in the feed is denoted as C_{A0}. The feed temperature and volumetric flow rate are T_0 and F, respectively. The reacting liquid has a constant density of ρ_L and a heat capacity of C_p. Parameters ΔH, k_0, E, and R correspond to the enthalpy of reaction, pre-exponential constant, activation energy, and ideal gas constant, respectively. The values of the process parameters are provided in Table 6.1. When the values of Table 6.1 are substituted into Eq. (6.56), the CSTR

Table 6.1: Parameter values for chemical process example

$F = 5.0\,\mathrm{m^3/h}$	$V = 1.0\,\mathrm{m^3}$
$k_0 = 8.46 \times 10^6\,\mathrm{m^3\,kmol^{-1}\,h^{-1}}$	$E = 5.0 \times 10^4\,\mathrm{kJ/kmol}$
$R = 8.314\,\mathrm{kJ\,kmol^{-1}\,K^{-1}}$	$\rho_L = 1000.0\,\mathrm{kg/m^3}$
$\Delta H_r = -1.15 \times 10^4\,\mathrm{kJ/kmol}$	$T_0 = 300.0\,\mathrm{K}$
$Q = 0\,\mathrm{kJ/h}$	$C_{A0} = 4\,\mathrm{kmol/m^3}$
$C_{As} = 1.95\,\mathrm{kmol/m^3}$	$T_s = 402\,\mathrm{K}$
$C_p = 0.231\,\mathrm{kJ\,kg^{-1}\,K^{-1}}$	

system can be written as

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = 5C_{A0} - 5C_A - 8.46 \cdot 10^6 \mathrm{e}^{\frac{-6013.95}{T}} C_A^2 \tag{6.57a}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = 1500 - 5T + 4.21 \cdot 10^8 \mathrm{e}^{\frac{-6013.95}{T}} C_A^2 + 0.00433Q \tag{6.57b}$$

We investigate the operation of the CSTR under LEMPC at an unstable steady-state characterized by $(C_{As},\ T_s) = (1.95\,\mathrm{kmol/m^3}, 402\,\mathrm{K})$ and $(C_{A0_s}\ Q_s) = (4\,\mathrm{kmol/m^3}, 0\,\mathrm{kJ/h})$. The manipulated inputs in this system are the inlet concentration of species A and the heat input rate, denoted as deviation variables $\Delta C_{A0} = C_{A0} - C_{A0_s}$ and $\Delta Q = Q - Q_s$, respectively. These manipulated inputs have the following bounds: $|\Delta C_{A0}| \leq 3.5\,\mathrm{kmol/m^3}$ and $|\Delta Q| \leq 5 \times 10^5\,\mathrm{kJ/h}$. Hence, the states and inputs of the closed-loop system are represented as $x^\top = [C_A - C_{As}\ T - T_s]$ and $u^\top = [\Delta C_{A0}\ \Delta Q]$, respectively. The equilibrium point of the system is located at the origin of the state-space, denoted as $(x_s^*, u_s^*) = (0, 0)$. In this study, we consider the model variations caused by catalyst deactivation during the operation of the CSTR described by Eq. (6.56). This deactivation leads to a reduction in the reaction pre-exponential factor k_0 within the constraint range of $0 < k_0 < 8.46 \times 10^6\,\mathrm{m^3\,kmol^{-1}\,h^{-1}}$.

The control Lyapunov function $V(x) = x^\top P x$ is designed, where the positive definite matrix P is given as:

$$P = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{6.58}$$

Using this Lyapunov function, the closed-loop stability region Ω_ρ for the CSTR can be defined as a level set of the Lyapunov function, where $\hat{\rho} = 368$ within the region ϕ_u. By employing the controller $u = \Phi(x) \in U$, the origin can be rendered exponentially stable inside this stability region.

To numerically simulate the dynamical model described by Eq. (6.56), we utilize the explicit Euler method with a time step of $h_c = 10^{-4}\,\mathrm{h}$. The nonlinear optimization problem of the LEMPC formulation in Eq. (6.37) is solved using the Python module of the IPOPT software package [145], specifically the PyIpopt module. The sampling period for the optimization problem is set to $\Delta = 10^{-2}\,\mathrm{h}$.

The main goal of the LEMPC is to maximize the profitability of the CSTR process described by Eq. (6.56) by manipulating the inlet concentration ΔC_{A0} and the heat input rate ΔQ. It aims to ensure that the closed-loop state trajectories remain within the stability region $\Omega_{\hat{\rho}}$ at all times under LEMPC. The objective function of the LEMPC is designed to optimize the production rate of product B, given by:

$$l_e(\tilde{x}, u) = k_0 \mathrm{e}^{-E/RT} C_A^2 \tag{6.59}$$

In addition, the LEMPC employs a material constraint, specified in Eq. (6.60), to maintain the average reactant material within an operating period t_p at its steady-state value C_{A0s}. This means that the averaged reactant material deviation, denoted as u_1, should equal zero:

$$\frac{1}{t_p}\int_0^{t_p} u_1(\tau)\,\mathrm{d}\tau = 0\ \text{kmol/m}^3 \tag{6.60}$$

By incorporating this material constraint, the LEMPC ensures that the average reactant material supplied during the operating period aligns with the steady-state value, facilitating stable and controlled production while maximizing the overall profitability of the CSTR process.

6.5.1 Data generation and SINDy model development

We follow the first type of data generation and model building process described in [228]. Specifically, we numerically integrate the system of Eq. (6.56) with an integration time step of $h_c = 10^{-4}\,\mathrm{h}$ and a sampling period of $\Delta = 10^{-2}\,\mathrm{h}$. 1000 different initial conditions are selected randomly with $C_A \in [0.2, 3.7]\ \mathrm{kmol/m^3}$ and $T \in [327, 477]\ \mathrm{K}$, while the inputs are taken to be step functions with amplitudes $C_{A0} \in [0.5, 7.5]\ \mathrm{kmol/m^3}$ and $Q \in [-500, 500]\ \mathrm{MJ/h}$. We note that, although the trajectories settled at the stable steady-states, since the dynamics of the reactor are independent of the specific steady-states, the model development did not suffer. Due to the large variation of the states when settling at other steady-states, however, finite-difference estimates of the time-derivative $\dot{X}$ can be poor when the temperature, for example, goes as low as 1 K or as high as 1000 K, despite the initial conditions and desired steady-state being extremely far from these states. Hence, only trajectories where the temperature never dropped below 300 K nor rose above 500 K were retained, which yielded 53 trajectories.

The function library for SINDy is chosen to be

$$\Theta(C_A, T, C_{A0}, Q) = [1 \quad C_A \quad C_A^2 \quad T \quad C_{A0} \quad Q \quad \mathrm{e}^{-\frac{6020}{T}} \quad C_A\mathrm{e}^{-\frac{6020}{T}} \quad C_A^2\mathrm{e}^{-\frac{6020}{T}}] \tag{6.61}$$

The choice for the basis functions is a central problem in SINDy modeling. Due to the presence of nonlinear reaction terms, especially Arrhenius dependence of the temperature and unknown reaction order with respect to reactant A, we consider monomial terms in C_A up to second order and a negative exponential term of the reciprocal of the temperature as well as all possible interactions of these two types of terms. For the input variables, since they often impact the dynamics of the system linearly, we use linear C_{A0} and Q terms in the basis functions to start with. The choice of -6020 as the numerator of the fraction in the exponential term (dependent on the activation energy), denoted by γ, is motivated by first conducting a coarse search of values between -7000 and 0 in steps of 1000. For each value of γ, the maximum absolute error (MAE) of the validation set is calculated for both C_A and T. The results are shown in Fig. 6.2. As can be seen, the error is very high for larger values of γ and decreases sharply at around -6000, rising again below -6000. For even lower values of $\gamma < -7000$, the models were found to be unstable and, hence, could not be integrated and did not yield an MAE against the validation set. Subsequently, a finer search is conducted in the vicinity of $\gamma = -6000$. Specifically, values of γ between -6500 and -5500 are chosen in steps of 20, yielding 100 values to assess. Similarly, the MAE for both states are recorded for each value of γ and plotted in Fig. 6.3, which clearly indicates $\gamma = -6020$ is the optimal value for this system. For comparison, the exact value of γ is -6013.95, as seen in Eq. (6.57). When using steps of 20, the closest value of γ in the trial set is -6020, which is currently identified via

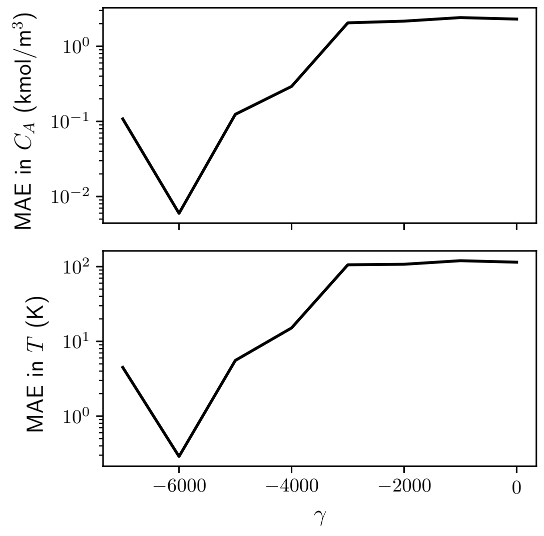


Figure 6.2: Validation error as a function of γ for a coarse search of $\gamma \in [-7000, 0]$.

the above procedure. An even finer search could be conducted to find $\gamma = -6014$ if using steps of 1 in the vicinity of -6020. However, due to the risk of overfitting and for practical considerations, especially for larger systems as detailed in [228], the value of -6020 is considered adequate for this work and is selected as the basis function.

For this system, it is imperative to scale the columns of $\Theta(X, U)$ to account for the multiple orders of magnitude of difference between the values of different columns. For instance, the Q column is on the order of magnitude of 10^5 while the final three columns of $\Theta(X, U)$, corresponding to the exponential terms, are on the order of magnitude of 10^{-6}–10^{-7}. We scale every column of Θ by its L_2 norm.

Using the above choice of candidate basis functions and library column normalization, we conduct a similar coarse-to-fine search to tune the sparsification knob λ. The first step is to conduct a coarse search, which is done for $\lambda \in [0, 500]$ in steps of 5. For each value of λ, the maximum MAE for both states are plotted in Fig. 6.4. As can be seen, for $\lambda > 450$, all terms of the C_A

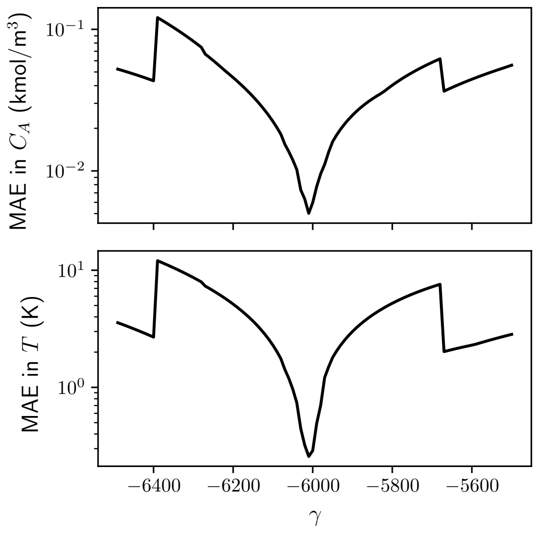


Figure 6.3: Validation error as a function of γ for a fine search of $\gamma \in [-6500, -5500]$.

ODE are zeroed, leading to an exponential increase in error, which remains constant thereafter. The lowest error is found to be at the lower values of λ, specifically below values of approximately 100. Conducting a finer search for $\lambda \in [0, 100]$ in steps of 1 and recording not only the MAE in both states but also the number of terms in the model, we obtain Fig. 6.5. It can be seen that the MAE for both states is consistently low throughout the entire range. However, the lowest number of terms in the model is at $\lambda = 72$. While a lower value of λ corresponding to a less parsimonious model with lower MAE can be used, in sparse identification, the balance between parsimony and accuracy is important to consider. Hence, since the error does not significantly decrease when λ is reduced further, the model corresponding to $\lambda = 72$ is taken as the optimal SINDy model for the CSTR system of Eq. (6.56) that balances the model sparsity with accuracy.

The final SINDy model obtained using the 53 open-loop trajectories using the above proce-

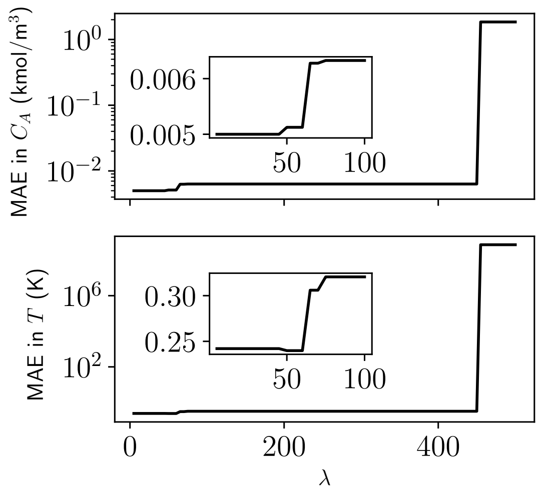


Figure 6.4: Validation error as a function of λ for a coarse search of $\lambda \in [0, 500]$ with zoomed-in subplot for $\lambda \in [0, 100]$.

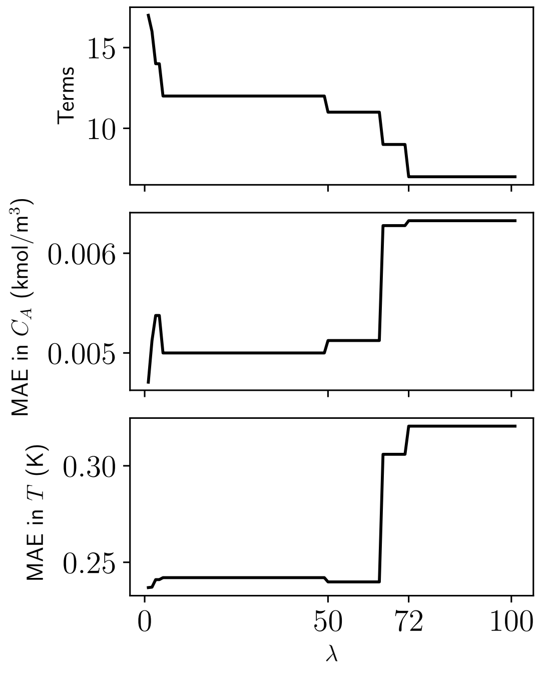


Figure 6.5: Validation error as a function of λ for a fine search of $\lambda \in [0, 100]$.

dure is as follows:

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = 5.045C_{A0} - 5.049C_A - 8.647 \cdot 10^6 \mathrm{e}^{\frac{-6020}{T}} C_A^2 \tag{6.62a}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = 1511.647 - 5.038T + 4.300 \cdot 10^8 \mathrm{e}^{\frac{-6020}{T}} C_A^2 + 0.00436Q \tag{6.62b}$$

where, compared to the first-principles model of Eq. (6.57), every term has been correctly identified, and all coefficients except the pre-exponential constants have been identified accurately with an error below 1%. As for the pre-exponential constants, since the exponential term in the basis function is $\mathrm{e}^{-\frac{6020}{T}}$ rather than $\mathrm{e}^{-\frac{6013.95}{T}}$, the pre-exponential constants are slightly larger for the SINDy model, possibly to "compensate for" the more negative exponential term reducing the values of the exponential terms themselves. Therefore, as a result, the final model is extremely accurate as can be seen in the maximum absolute errors in C_A and T, which are $0.006\,326\,12\,\mathrm{kmol/m^3}$ and 0.320559 K, respectively.

Remark 6.3. *The accuracy of the SINDy model obtained is most strongly dependent on the data generation and the choice of candidate library functions. Both of these were investigated in-depth in [228], Specifically, with respect to data generation, based on three different types of data generation methods studied in [228], open-loop step tests in the vicinity of the steady-state of interest were found to be the most appropriate type of data generation for SINDy modeling, which is why the 53 trajectories near the steady-state,* $(C_{As},\ T_s) = (1.95\,\mathrm{kmol/m^3}, 402\,\mathrm{K})$*, were used for model building in this work. Regarding the choice of library functions, two more general frameworks for more complex or larger-scale systems were proposed in [228]. One effective method was found to be non-dimensionalization, which is a standard practice in engineering modeling. When all vari-*

ables were scaled to similar orders of magnitudes, it was easier to identify the exact (dimensionless) model since the sensitivity of the library functions was reduced. In contrast, the most general framework proposed was to use polynomial terms in deviation variables since any nonlinearity can be expanded using Taylor expansion to the desired level of generalization using polynomials of a certain order. While the exact ODE model will not be recovered using this approach, from a modeling and control perspective, since only an accurate process model for the operating region is required, this approach is sufficient in terms of both accuracy and computation, possibly faster in terms of computation than the previous approach.

6.5.2 Closed-loop simulation results

6.5.2.1 Model assessment before plant disturbances

Using the chemical process example, we aim to illustrate the error-triggered on-line model update procedure in the presence of plant variations. However, before addressing the effect of catalyst deactivation, we first demonstrate that the initial SINDy model performs just as well as the first-principles model when there is no catalyst deactivation or disturbances. To compare their performance, we designed two LEMPC schemes, both following the structure of Eq. (6.37). One scheme utilized the exact first-principles model from Eq. (6.57) as the process model, while the other scheme employed the SINDy model from Eq. (6.62). Both LEMPC schemes employed the cost function defined in Eq. (6.59), the additional material constraint of Eq. (6.60), the upper and lower bounds on u_1 and u_2 described earlier, and the same Lyapunov-based controller and stability region.

For all simulations in this example, the LEMPC designs had a prediction horizon of $N = 5$,

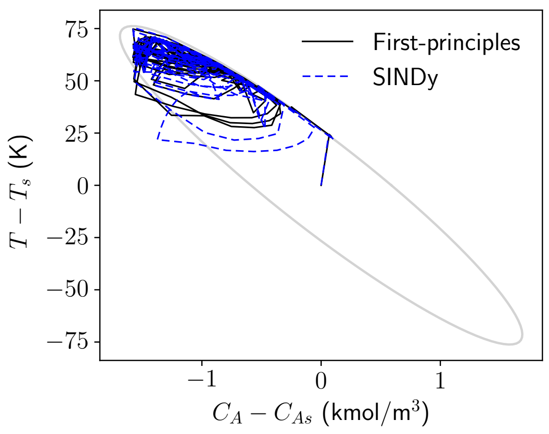


Figure 6.6: State-space trajectories of the CSTR without catalyst deactivation under an LEMPC based on the first-principles model and an LEMPC based on the initial SINDy model.

a sampling period of $\Delta = 0.01$ h, and an operating period of 20 sampling periods ($t_p = 0.2$ h). The first-principles LEMPC and the SINDy-based LEMPC were both applied to the CSTR model described by Eq. (6.56). The reactor was initialized at the unstable steady-state $(C_{As},\ T_s) = (1.95\,\mathrm{kmol/m^3}, 402\,\mathrm{K})$, and closed-loop simulations were conducted for 20 operating periods for each case. The resulting closed-loop trajectories for the CSTR under both LEMPC schemes are depicted in Figs. 6.6 and 6.7. The average yield of the first-principles LEMPC over twenty operating periods was 62.29, compared to 62.06 for the SINDy-based LEMPC. The agreement between the trajectories for most of the simulation duration and nearly identical yields obtained from the first-principles and SINDy-based LEMPCs further illustrates that the initial SINDy model adequately captures the process behavior in the absence of plant variations. It is worth noting that the periodic nature of the trajectories aligns with prior literature, which has reported that time-varying operation can be economically advantageous for certain processes [e.g., 229, 230].

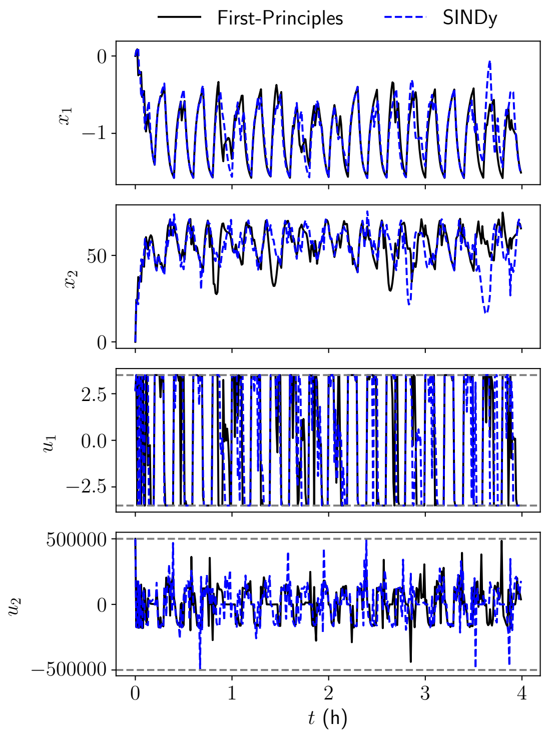


Figure 6.7: State and input trajectories of the CSTR without catalyst deactivation under an LEMPC based on the first-principles model and an LEMPC based on the initial SINDy model. The grey dashed lines represent the upper and lower bound for the inputs.

6.5.2.2 LEMPC performance in the presence of plant disturbances

Next, we investigate the effect of catalyst deactivation on the LEMPC performance using either process model. Specifically, the value of the pre-exponential constant k_0 in Eq. (6.56) is reduced by 20% of its original value from $8.46 \times 10^6\,\mathrm{m}^3\,\mathrm{kmol}^{-1}\,\mathrm{h}^{-1}$ to $6.77 \times 10^6\,\mathrm{m}^3\,\mathrm{kmol}^{-1}\,\mathrm{h}^{-1}$ after five operating periods (at $t = 1$ h). As a result, the CSTR system of Eq. (6.57) is altered to

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = 5C_{A0} - 5C_A - \mathbf{6.77} \cdot 10^6 \mathrm{e}^{\frac{-6013.95}{T}} C_A^2 \tag{6.63a}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = 1500 - 5T + \mathbf{3.37} \cdot 10^8 \mathrm{e}^{\frac{-6013.95}{T}} C_A^2 + 0.00433Q \tag{6.63b}$$

where the affected coefficients are boldfaced. In the remainder of the simulations, the material constraint of Eq. (6.60) is relaxed for the second half of the total simulation duration, from $t = 2$ h to $t = 4$ h, to allow a usage of u_1 that is $10\,\mathrm{kmol/m}^3$ greater than its steady-state value per operating period. This is to allow the LEMPC to find control actions that optimize the average yield further and provide a larger space of control actions for the LEMPC to highlight any possible differences between the LEMPCs with and without SINDy model updates. Hence, the material constraint now takes the form,

$$\frac{1}{t_p}\int_0^{t_p} u_1(\tau)\,\mathrm{d}\tau = \begin{cases} 0\,\mathrm{kmol/m}^3, & t < 2 \\ 10\,\mathrm{kmol/m}^3, & t \geq 2 \end{cases} \tag{6.64}$$

To effectively monitor the prediction error for the SINDy model in the presence of catalyst deactivation, a moving horizon error detector, as described in Section 6.4.1, was implemented early in the process operation. The detector was activated after collecting a sufficient amount of input/output data points, specifically N_w prior data points. At each sampling time, the detector calculated the value of e_d to assess whether it is necessary to trigger an update of the SINDy model.

The moving horizon error detector calculates the relative prediction error in the concentration of A and the reactor temperature. These errors are evaluated over the past 15 sampling periods (i.e., $N_w = 15$) and the current sampling time using the equation,

$$e_d(t_k) = \sum_{r=0}^{15} \frac{|x_{p,1}(t_{k-r}) - x_1(t_{k-r})|}{|x_1(t_{k-r})|} + \frac{|x_{p,2}(t_{k-r}) - x_2(t_{k-r})|}{|x_2(t_{k-r})|} \tag{6.65}$$

where the states from 15 sampling periods prior are used to initialize the integration of the current SINDy model over a duration of 15 sampling periods to calculate $x_{p,1}$ and $x_{p,2}$ over the entire window of $N_w = 15$ (single initialization). For $N_w = 15$, it was observed that significant discrepancies between the plant and the original SINDy model were marked by the value of e_d exceeding 2 (i.e., $e_{d,T} = 2$). Therefore, this threshold value was chosen to initiate model updates. The determination of $N_w = 15$ and $e_{d,T} = 2$ were as per the guidelines presented in Section 6.4.1. Specifically, if $N_w = 10$ (too low), the gap between the error values during normal operation and post-deactivation was not consistently large enough, and the error of the post-deactivated e_d trajectory without the model update periodically dipped very low, as shown in Fig. 6.8. Hence, if, for example, $e_{d,T}$ was still maintained at a value of 2 while $N_w = 10$, the post-deactivated e_d trajectory would often dip below 2. While this would still be possible to resolve by slightly reducing $e_{d,T}$ to, for example, 1.25, this would be too finely tuned and not easily generalized. In contrast, at $N_w = 15$, there is a large and consistent gap between the error trajectories with and without model updates throughout the simulation duration, with $e_d = 2 = e_{d,T}$ being a clear indicator of model performance. The gap can be further increased and the detection made even more robust by further increasing N_w. For example, $N_w = 20$ with $e_{d,T} = 3$ is also a valid and possibly even more robust choice of error detection. However, as N_w is increased, the amount of data required to be kept in storage as well as the length of time the SINDy models need to be integrated also increases. Due to the longer data collection, the time elapsed before the error is triggered and the new dynamics are detected, may also increase as the error will *accumulate* at a slower pace. Therefore, for computational considerations, we chose $N_w = 15$ with $e_{d,T} = 2$ as a reasonable trade-off that ensures error detection as well as reduces computational burden. Higher values of the window length N_w have been used with higher $e_{d,T}$, however, with success. For example, for the same CSTR system, in [67], a relatively high error threshold of $e_{d,T} = 15$ was used, which indicates the window length N_w must also have been quite large, such that normal process operation with an accurate process model could accumulate an error of up to 15 within N_w sampling periods, while, in our work, with $N_w = 15$, even under the altered process conditions with the old SINDy model used to carry out predictions, the error reached a maximum value of only 9 over the three hours of deactivated process run time.

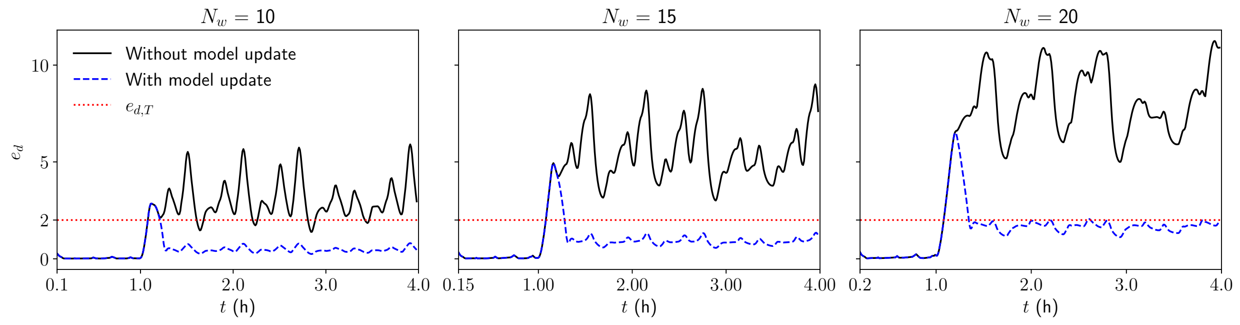


Figure 6.8: Value of error metric e_d using the detector of Eq. (6.65) over the simulation duration for various values of N_w with and without SINDy model updates. The dashed red line corresponds to $e_{d,T} = 2$.

When an on-line model update is triggered, a few pre-selected terms in the SINDy model are updated using the most recent input/output data. The coefficients to be updated are selected based on knowledge of the process operation or different subsets of the coefficients can be updated on a trial-and-error basis until the error is reduced to a value below $e_{d,T}$. In this case, since catalyst deactivation is a ubiquitous phenomenon in catalytic reactors, both ODE coefficients corresponding to the nonlinear reaction terms are chosen as candidates to be updated. There may be feed disturbances or other types of process changes as well. Hence, we also consider the coefficient corresponding to the reactor temperature T for both ODEs as coefficients subject to change. Hence, out of 18 coefficients (9 per ODE), 4 coefficients are updated using the model update procedure described in Section 6.4.2.

As for the input/output data used for the model update, although an immediate update is ideal if possible, this cannot be conducted since sufficient data from the new operating conditions must be present for the model update to succeed. On the other hand, letting the process run for an extended duration with the old model while the error remains high is undesirable as the LEMPC performance will likely deteriorate with time. Hence, the amount of data used to update the model should be the minimum amount required to accurately update the model using post-deactivation data, which will vary from one process to the other (e.g., 5000 data samples were used to update the same CSTR process in [223], while 200 data samples were required to re-identify new linear empirical models in [227]). For the reaction studied in this work, the input/output data to update the SINDy model is the state and input data of the the current operating period in which the moving horizon error detector was triggered, which would correspond to $N_d = 20$. However, using the entire operating period of input/output data yielded poor results. Upon further investigation, the

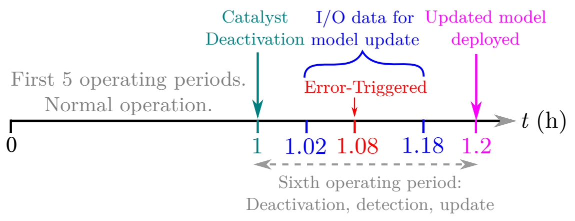

Catalyst Deactivation
I/O data for model update
Updated model deployed
First 5 operating periods. Normal operation.
Error-Triggered
t (h)
0
1
1.02
1.08
1.18
1.2
Sixth operating period: Deactivation, detection, update

Figure 6.9: Details of the error detection and model update procedure occurring in the sixth operating period from $t = 1.0$ h to $t = 1.2$ h.

cause was found to be the minimum/maximum values of u_1 used at the beginning and end of each operating period, leading to very large changes in the states, which produce accordingly large errors in the estimates of the time-derivative, $\dot{X}$. Hence, the first and last two data points (two due to the use of second-order finite-differences) of the operating period are omitted from the model update algorithm. For this process, it is found that the use of a single operating period without the endpoints is sufficient to update the four coefficients of the SINDy model chosen to be updated. Since the catalyst deactivation, error-triggering, and model update all occur during the sixth operating period, a detailed illustration of the model update procedure is shown in Fig. 6.9.

The LEMPC updates the SINDy process model at the end of the sixth operating period to reflect the catalyst deactivation and continues to operate for the remainder of the 4 h of simulation duration. The moving horizon error detector is used to monitor the modeling error throughout the process, and the value of e_d over the simulation duration is shown in Fig. 6.10. It can be observed that the error e_d exceeds the pre-determined threshold of $e_{d,T} = 2$ at $t = 1.08$ h, during the sixth operating period, and then rapidly decreases below the threshold at $t = 1.2$ h following the SINDy model update. On the other hand, the value of e_d calculated with the initial SINDy model (without update) continues to remain above $e_{d,t}$ for the remainder of the simulation, indicating that the initial SINDy model cannot very accurately predict the process states once the catalyst

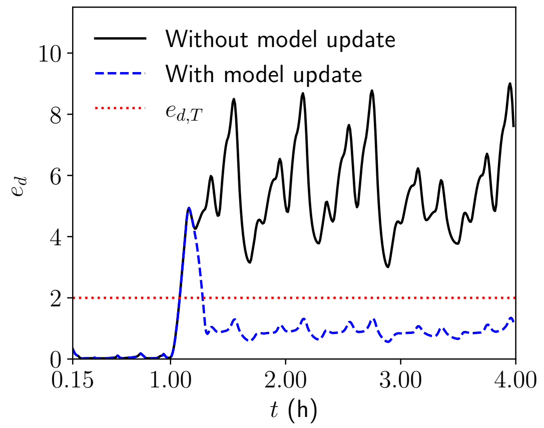


Figure 6.10: Value of error metric e_d using the detector of Eq. (6.65) and the integrated LEMPC design with error-triggered on-line model updates at each sampling time (the time axis starts from when there is sufficient data to begin calculating e_d, i.e., after $N_w = 15$ data points are collected).

is deactivated. Since Fig. 6.10 spans the entire simulation duration, and the exact details may be difficult to observe, a closer look at the sixth operating period, where all the changes occur, was provided in Fig. 6.9.

The closed-loop state and input trajectories for the LEMPCs with and without SINDy model updates are shown in Fig. 6.11 and Fig. 6.12, which depict the state-space and time-varying trajectories, respectively. From Fig. 6.11, it is observed that the closed-loop state quickly approaches the boundary of the stability region $\Omega_{\hat{\rho}}$ by using maximal input actions early on in the trajectory and continues to remain as close as possible to the boundary representing the highest possible temperature and lowest possible concentration of A, as this combination maximizes the production of B in the process. The closed-loop state is always maintained within the stability region $\Omega_{\hat{\rho}}$ for 99% of the simulation duration under both LEMPC. The manipulated input profiles indicate, as expected, the cyclic use of u_1, i.e., at the beginning of each operating period, the LEMPC uses the maximum value of ΔC_{A0} allowed to maximize production of B, while reducing the consumption

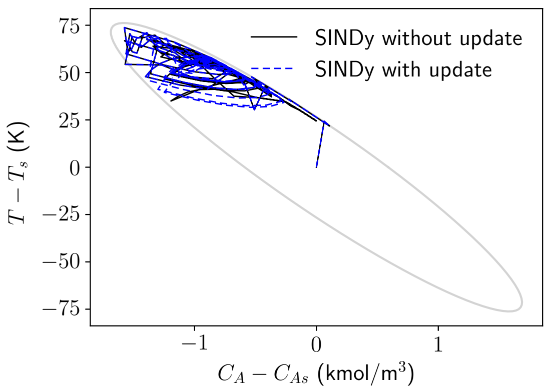


Figure 6.11: State trajectories for the closed-loop CSTR under the LEMPC of Eq. (6.37) with and without the online update of the SINDy model for the initial condition (0, 0). They grey ellipse represents the stability region $\Omega_{\hat{\rho}}$.

of the reactant at the end of each period to meet the material constraint. For the second half of the simulation, however, due to the relaxation of the material constraint as per Eq. (6.64), the sum of u_1 per operating period is equal to 10, which is why the initial period of maximum u_1 consumption is observed to be greater than the minimal consumption at the end of the last 10 operating periods.

Finally, the total economic benefits achieved over the 20 operating periods are calculated for three scenarios: the SINDy-based LEMPC without on-line updates, SINDy-based LEMPC with the on-line update, and steady-state operation where the system of Eq. (6.56) is operated at (C_{As}, T_s) for the entire duration. The economic benefits are evaluated using the equation,

$$L_E = \int_0^4 l_e(x, u)\, \mathrm{d}t \tag{6.66}$$

The closed-loop system under LEMPC with and without SINDy model updates achieves $L_E = 63.45$ and $L_E = 63.72$, respectively, within the four-hour period, while the steady-state operation

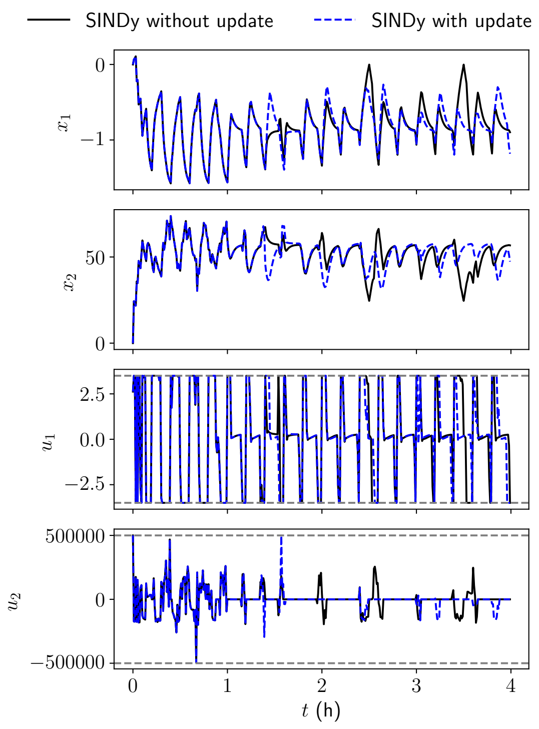


Figure 6.12: State and manipulated input profiles for the initial condition (0, 0) under the LEMPC of Eq. (6.37) with and without the online update of the SINDy, respectively. The grey dashed lines represent the upper and lower bound for the inputs.

yields $L_E = 34.79$. This comparison demonstrates that time-varying operation of the system of Eq. (6.56) under the LEMPC of Eq. (6.37) with or without on-line updating of SINDy models results in much higher economic benefits of 82% compared to steady-state operation. The economic benefits of the LEMPC with model updates are also 0.42% higher than the benefits of the LEMPC without model updates. Although the difference is small in this specific scenario, this is highly dependent on both the system as well as the model fidelity. For the same CSTR system modeled using RNN models, it was demonstrated in [67] that a large improvement is possible if the performance of the initial process model deteriorates severely after disturbances are introduced to the system. In our work, however, possibly since the model structure and coefficients were closely identified in the initial SINDy model, the performance deterioration of the LEMPC even without model updates was not sufficiently high to allow the LEMPC with model updates to improve upon. To investigate this, the CSTR was also simulated under an LEMPC using the first-principles model of Eq. (6.56) but with the relaxed material constraint of Eq. (6.64) and the process model updated immediately upon deactivation (i.e., at $t = 1$ h, not $t = 1.2$ h). Practically, the first-principles model, the exact moment of catalyst deactivation, and the exact percentage of catalyst deactivation are unknown. However, this is a hypothetical best-case scenario that should yield the highest possible economic benefits possible for this CSTR specification since the LEMPC process model "sees" the exact change immediately upon its occurrence. The economic benefits of this case was found to be $L_E = 63.53$. Since the value is actually lower than L_E for the LEMPC with SINDy, it can be inferred that all three LEMPC perform nearly identically and the minor improvements are more likely due to numerical issues or due to a very small number of sampling periods being significantly different from each other. Hence, the reason for the small, possible improvement of the LEMPC with SINDy model updates can be contributed to the LEMPC performance in general being upper-bounded by other factors due to the specific process, parameters, model structure, and initial SINDy model fidelity of this study.

Remark 6.4. *In this work, a Lyapunov-based tracking MPC using a SINDy model to drive the process to a steady-state was not presented because the development of the SINDy model and its real-time adaptation is similar to the case of economic MPC and would not add any new methodological and/or implementation insights.*

6.6 Conclusions

This study introduces a novel approach for on-line updates of nonlinear ODE models obtained using sparse identification to embed into a model predictive controller (MPC) for nonlinear process systems. The proposed methodology incorporates an error-triggering mechanism through a moving horizon error detector, which evaluates the relative prediction error within a specified horizon. When the prediction error surpasses a predefined threshold, the error-triggering mechanism is activated and the most recent yet sufficient input/output data is used to update specific coefficients of the SINDy model using an efficient algorithm. The results showcase the capability of the proposed approach to improve state predictions crucial for MPC in the presence of plant variations. A chemical process example under the framework of Lyapunov-based empirical model predictive control is employed to illustrate the effectiveness and implementation of real-time updates for the SINDy models. It was demonstrated that a small amount of real-time data could accurately update the SINDy model to adjust to the disturbances, greatly reducing thereby the model prediction error when monitoring the process via the moving horizon error detector. A slight improvement of the economic benefits was also observed when the SINDy model in the LEMPC was updated in real-time, compared to the SINDy-based LEMPC without model updates, although the improvement was limited by the catalyst deactivation and other compounding factors.

Chapter 7

Data-based modeling and control of nonlinear process systems using sparse identification: An overview of recent results

7.1 Introduction

A central objective of scientific and engineering research is the derivation of the laws governing physical systems in the form of equations. With the explosion in data and computational power over the last two decades, the construction of these equations empirically from data has become more tractable than deriving physics-based first-principles models, especially for highly complex systems, and is gaining momentum in the literature. For many physical systems, the laws governing their dynamics take the form of ordinary differential equations (ODE) or partial differential equations (PDE) with time and/or space as independent variables. Common examples include the Boltzmann equation in thermodynamics and the Navier–Stokes equations in fluid dynamics [7].

The development of such time-varying predictive models is often a prerequisite for other objectives in a plant/system engineering context, such as predictive maintenance in operations engineering and advanced control system design in any closed-loop system with strict product requirements. In chemical process systems, model predictive control (MPC) is an advanced control system that has been implemented and accepted widely in the industry [68]. As the name suggests, MPC uses a dynamical model such as an ODE to predict the process states (outputs) over a user-defined prediction horizon to be able to take the optimal control action based on anticipated possible future trajectories. A large body of literature on data-driven modeling in MPC can be found in [146]. Two of the most common, classical system identification algorithms include the singular value decomposition [156] and Numerical algorithms for Subspace State Space System Identification (N4SID) [30]. However, machine learning (ML) methods, a type of data-driven modeling with numerous parameters and tunable hyper-parameters, have demonstrated highly accurate results when applied to complex systems with multiple interacting nonlinearities due to their high degree of freedom. Some examples of ML methods include support vector regressors, extreme gradient boosting, and, particularly of interest in recent years, artificial neural networks. For example, in [3, 62], recurrent neural networks were used to model nonlinear processes, and subsequent closed-loop stability results under recurrent neural network-based model predictive control were derived. Autoencoders, which are feedforward neural networks (FNN) that replicate the input at its output, have been the subject of several studies. While linear autoencoders correspond exactly to PCA, [120] proposed the use of nonlinear autoencoders as a form of nonlinear PCA. Autoencoders, particularly undercomplete autoencoders, are a powerful tool for dimensionality reduction due to the enforced reduction of the dimension in the intermediate layers of the network. [231] used undercomplete autoencoders to carry out nonlinear dimensionality reduction and build reduced-order models for integrated scheduling and control of chemical process operations. When the system identification and optimal scheduling computations were conducted in the latent variables with reduced dimensionality, it was found that the computational efficiency as well as the level of dynamic information provided were improved. In [232], Koopman theory was used to derive a Wiener-type formulation for handling multiple-input multiple-output (MIMO) input-affine dynamical systems. Specifically, reduced-order surrogate models were developed by combining autoencoders with linear dynamic blocks. The models were hypothesized to be particularly useful in control applications due to the high accuracy and dimensionality reduction capabilities of the proposed Wiener-type Koopman models. The integration of a Gaussian process model with MPC was proposed in [233] and applied to a gas–liquid separation process. The simple model structure of the Gaussian process model and, more importantly, the statistical information such as the prediction uncertainty provided by such a model were found to be desirable qualities for control-centric applications.

A potential drawback of traditional ML models has been their black-box nature, which limits their applicability and adoption in process systems engineering. Therefore, the field of hybrid modeling, sometimes referred to as "gray-box" modeling, which aims to combine *a priori* first-principles knowledge or domain expertise with black-box approaches such as ML models to improve both the accuracy and interpretability of the overall model, has recently attracted significant attention. [234] outlines a number of approaches to incorporating physics into data-driven modeling including but not limited to: 1) feature engineering, which refers to domain experts selecting and/or creating physically meaningful features from the data set obtained from sensors rather than using the raw measurements directly, 2) residual modeling, which refers to building an ML model to model the residual between the known first-principles model and sensor measurements in order to build a model that captures the plant-model mismatch, and 3) linear meta-model of models, where the solutions from multiple sub-models, which correspond to various parts of the overall system and are obtained using feature engineering, are combined into a linear meta-model by taking a weighted linear combination of all the models to represent the overall system accurately once the weights are tuned. [235] investigated the use of FNNs to build state estimators in the absence of full-state feedback. Specifically, an FNN was used to model the nonlinear terms in the dynamics such as those corresponding to chemical reactions. In [192, 204], the links between layers of a recurrent neural network (RNN) were disconnected (i.e., corresponding weights zeroed) based on the process structure, leading to the elimination of erroneous model predictions and improved overall model accuracy. [236] provides a detailed overview of hybrid modeling and its evolution over the last three decades since it became a subject of interest in the scientific community. [236] reports that the *a priori* knowledge to be incorporated into hybrid modeling has typically been in the form of equations, and other forms of data/information such as plant floor experience and process flow sheets have not been investigated exhaustively. It was also found that data-driven modeling has typically been used to enhance previously known or derived mechanistic models, but the reverse, i.e., using mechanistic models to improve or constrain data-driven models, is largely unexplored. In the field of process monitoring and fault diagnostics, in particular, [236] highlighted the benefit of knowledge of causality that can be inferred via hybrid modeling.

The area of surrogate modeling in process systems engineering has, in parallel with the above directions, increased in research intensity. [237] provides a comprehensive overview of advances in surrogate modeling in chemical engineering over the past three decades. A primary reason for the surge of interest in surrogate models is the increasing complexity of modern, highly accurate models used to simulate or model the nonlinear processes, scheduling problems, and complex thermodynamics that are ubiquitous in chemical engineering. Despite their increasing accuracy, such models encounter a number of challenges in downstream optimization and control applications. Due to the model complexity, the computational expense in terms of both processing power and time required to evaluate such models is exorbitantly large in many cases. While single function evaluations may be feasible in a practical setting, if the models are to be embedded into an optimization problem, such as set-point optimization or closed-loop control under an MPC, the computational demand becomes prohibitive due to the large number of function evaluations (typically hundreds or thousands) required to find such solutions. This is further complicated by black-box models since no simplification, such as omission of a term or otherwise, may be performed to find a compromise between model complexity and accuracy. If the type of model used is noisy or has discontinuities, this further complicated the problem, especially since finite-differences cannot be used to estimmate derivatives, which are crucial in optimization. To overcome these challenges, mathematically simpler models known as surrogate models have been proposed to approximate the input/output relationship of the complex models using much fewer model parameters and with much lower computational costs.

Although surrogate models can be used to approximate more complex models, another approach is to start with simpler model structures to model the desired system of interest and only add complexity as required. While methods such as N4SID and MOESP have been widely used over the past decades with varying degrees of success depending on the application and severity of the nonlinearities present, sparse identification for nonlinear dynamical systems (SINDy) is a recent method that aims to identify nonlinear ODEs directly from data, which are explicit and in closed-form, allowing them to be directly incorporated into MPC or any other optimization problem. Due to the availability of efficient differential equation solvers, the computational cost of integrating such models is generally low, especially if the models are well-conditioned. In the field of chemical engineering, SINDy has been used to identify reaction networks [84] and to build reduced-order models for modeling and controlling a hydraulic fracturing process [82]. Despite the application of SINDy to several chemical process examples in the literature, a number of specific issues encountered in the modeling and control of chemical processes and plants remain to be addressed adequately, based on our review of the literature. Therefore, this paper provides a unified summary of recent advancements and novel extensions to SINDy to overcome numerous challenges that are encountered when applying SINDy to the domain of chemical engineering. Besides providing general guidance with respect to basis functions and numerical concerns, more specifically, the difficulties of modeling multiscale systems, noisy sensor data, and industrial processes are discussed.

In this manuscript, we apply SINDy to model and control three types of process systems: 1) processes with time-scale multiplicities, 2) simulated processes with high levels of sensor noise, and 3) large-scale processes corrupted with high levels of industrial noise. Each category of systems has associated challenges and are addressed using different improvements upon the original SINDy algorithm, the details of which will be discussed in the respective section. We note that, although SINDy was introduced with the intent of identifying the governing physical laws as closed-form differential equations consistent with known physics of the system of interest, the application of SINDy is not limited to such cases. As the product of SINDy is a closed-form

ODE model with explicit nonlinearities, the resulting model can be directly incorporated into an MPC for efficient computations. Therefore, in this work, we use SINDy as a system identification algorithm with the ultimate goal of building dynamical models for controllers. The rest of this manuscript is outlined as follows: in Section 7.2, the general class of nonlinear process systems under consideration is described. Section 7.3 details the SINDy algorithm along with general guidelines and tuning considerations for building SINDy models, and its formulation in a model predictive controller. In Section 7.4, the challenges of two-time-scale systems are discussed, while Sections 7.5 and 7.6 address the challenges of noisy data for simulated processes and large-scale industrial processes, respectively. Finally, Section 7.8 provides a number of research directions for furthering the application of SINDy for process modeling and control.

7.2 Class of nonlinear process systems

We consider the class of nonlinear process systems described by the following first-order ODE:

$$\dot{x}(t) = f(x) + g(x)u + w, \qquad x(t_0) = x_0 \tag{7.1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^r$ is the manipulated input vector, and $w \in \mathbb{R}^n$ is the noise vector. The unknown vector and matrix functions $f \in \mathbb{R}^n$ and $g \in \mathbb{R}^{n\times r}$, respectively, constitute the process model representing the inherent physical laws constraining the system and are assumed to be locally Lipschitz vector and matrix functions of their arguments with $f(0) = 0$. The manipulated input is constricted to be in r nonempty convex sets defined as $\mathcal{U}_i \subseteq \mathbb{R}, i = 1, \ldots, r$. The sensor noise w is assumed to be bounded within the set $W := w \in \mathbb{R}^n : \|w\|_2 \leq \theta, \theta > 0$. The class of systems of the form of Eq. (7.1) is further restricted to the family of stabilizable nonlinear systems, i.e., there exist a sufficiently smooth control Lyapunov function $V(x)$ and a control law $\Phi(x) = [\Phi_1(x) \cdots \Phi_r(x)]^\top$ that renders the nominal ($w \equiv 0$) closed-loop system of Eq. (7.1) asymptotically stable under $u = \Phi(x)$. The stability region Ω_ρ is defined as the largest level set of V where $\dot{V}$ is negative. Without loss of generality, the initial time t_0 is taken to be 0 throughout the article.

7.3 Methodology: Sparse identification of nonlinear dynamics

7.3.1 Overview of the sparse identification method

Based on sparse regression and compressive sensing, sparse identification of nonlinear dynamics (SINDy) is a novel method in the field of system identification [73, 77] and has been applied to a diverse array of engineering problems [218]. The aim of SINDy is to use only input/output data from a system to represent the dynamics in the form of the nominal system of Eq. (7.1),

$$\dot{\hat{x}}(t) = \hat{f}(\hat{x}) + \hat{g}(\hat{x})u \tag{7.2}$$

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

Since most physical systems contain only a few terms in the right-hand side of Eq. (7.2), if a large number of nonlinear basis functions are considered as possible terms in $\hat{f}$ and $\hat{g}$, the space of all candidate functions considered is rendered sparse. Hence, SINDy aims to identify the small number of active functions in $\hat{f}$ and $\hat{g}$ using algorithms that leverage sparsity. We first obtain a discrete set of full-state measurements from open-loop simulations or experiments and concatenate them into a data matrix X and an input matrix U,

$$X = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix} \tag{7.3a}$$

$$U = \begin{bmatrix} u_1(t_1) & u_2(t_1) & \cdots & u_r(t_1) \\ u_1(t_2) & u_2(t_2) & \cdots & u_r(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_m) & u_2(t_m) & \cdots & u_r(t_m) \end{bmatrix} \tag{7.3b}$$

where $x_i(t_\ell)$ and $u_j(t_\ell)$ represent the measurement of the i^{th} state and j^{th} input at the ℓ^{th} sampling time, respectively, where $i = 1, \ldots, n$, $j = 1, \ldots, r$, and $\ell = 1, \ldots, m$. $\dot{X}$, the time-derivative of X, is a required matrix in the sparse identification algorithm and is either measured if possible (e.g., velocity) or otherwise estimated from X. Subsequently, a function library $\Theta(X, U)$ is constructed with s nonlinear functions of X and U. These s functions are the candidate nonlinear functions that may be zero or nonzero in the right-hand side of Eq. (7.2). The sparse identification algorithm exploits sparsity to calculate the coefficients associated with the terms in the library, Θ. Given the universality of mononomials, polynomials, and trigonometric functions in engineering systems

[77], they are often selected as the initial library in Θ. An example of an augmented library is

$$\Theta(X,U) = \begin{bmatrix} | & | & | & | & | & | \\ \mathbf{1} & X & \sin(X) & \mathrm{e}^X & U & UX^2 \\ | & | & | & | & | & | \end{bmatrix} \tag{7.4}$$

The goal of sparse identification is to find each of the s coefficients associated with the s nonlinear functions considered in Θ for each row of Eq. (7.2). Each state x_i corresponds to a sparse vector of coefficients, $\xi_i \in \mathbb{R}^s$, that represent the nonzero terms in $\hat{f}_i$ and $\hat{g}_i$ in the respective ODE, $\dot{\hat{x}}_i = \hat{f}_i(\hat{x}_i) + \hat{g}_i(\hat{x}_i)u$. Consequently, there are n such coefficient vectors that must be calculated. In matrix notation, the unknown quantity is

$$\Xi = \begin{bmatrix} \xi_1 & \xi_2 & \cdots & \xi_n \end{bmatrix} \tag{7.5}$$

which is found by solving the following equation:

$$\dot{X} = \Theta(X,U)\Xi \tag{7.6}$$

Equation (7.6) may be solved using standard least-squares after reformulating the problem as such by setting all coefficients in Ξ below a certain threshold λ to zero. Specifically, the least-squares problem takes the form,

$$\Xi = \underset{\Xi'}{\arg\min} \left\| \dot{X} - \Theta(X,U)\Xi' \right\|_2 + \lambda \left\| \Xi' \right\|_1 \tag{7.7}$$

where the first term maximizes the fidelity of the model to the data, while the second term ensures sparsity of Ξ. In Eq. (7.7), Ξ' is a notational substitute for Ξ. To solve Eq. (7.7), the least-squares problem is written in the following form, which may be solved using a standard solver for a linear system of equations:

$$\Xi = \arg\min_{\Xi''} \left\| \dot{X} - \Theta(X, U)\Xi'' \right\|_2 \tag{7.8}$$

where the matrix Ξ'' is Ξ' with all coefficients having an absolute value below λ set to zero. Equation (7.8) is repeatedly solved until convergence of the non-zero coefficients. The iterations typically converge rapidly due to the sparse structure of Ξ. An alternate algorithm to solve Eq. (7.6) is known as Sparse Relaxed Regularized Regression, which is based on the well-known LASSO operator [184]. After finding Ξ using either method, the identified model can be formulated as the continuous-time differential equation,

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

where $\Theta(x^\top, u^\top)$ is a column vector containing symbolic functions of x and u from the chosen function library, and $x^\top$ represents the transpose of x.

7.3.2 Data generation and SINDy modeling considerations

When applying SINDy to an engineering problem, a number of factors affect the results and must be carefully considered before and during the construction of a SINDy model.

7.3.2.1 Data generation

Data for system identification methods is typically obtained from either open-loop simulations or open-loop experiments. The sampling period used to record the data, the variation of system inputs and outputs considered for data generation, and the distribution of the data set are some of the properties that affect the amount of dynamic information contained in the data set and, as a result, the model quality.

Firstly, as information is lost when continuous data is sampled into discrete data, a higher sampling rate (lower sampling period) generally leads to better system identification for any method including SINDy, especially since SINDy requires estimates of the time-derivative of the states using finite differences or some variant thereof. However, it is important to consider practical limitations in terms of sampling. While an extremely small sampling period of 10^{-5} units may produce a data set with high information density, from which derivative estimations can be made very accurately, leading to the identification of better models, such a high sampling rate is typically not possible to achieve in a chemical process application or even in many other engineering disciplines. Instead, the sampling period should be chosen to be as small as reasonably possible, which would also be desirable in practice. Manufacturer specifications of the relevant type of sensors for process variables may be used as a lower bound on the sampling period for simulations-based studies.

Secondly, the dynamic information captured in a data set is dependent on the initial conditions chosen, the input signal variation, and the total simulation duration. The chosen combinations of initial conditions and input variables must cover as much of the operating region of interest as possible, and the simulation should be run until it reaches the desired steady-state of operation, in order to maximize the dynamic information captured in the data set. In contrast, if the data collection is carried out using a narrow range of initial conditions and/or inputs, or if a large part of the trajectories are zero values at the steady-state due to excessive run time, the data set may be large but contain little dynamic information to build an adequate model from. Furthermore, based on our studies, SINDy modeling works best with longer trajectories, even if from fewer initial conditions, rather than an exponential number of extremely small trajectories from many random initial conditions. The open-loop runs, whether experimental or simulations-based, should also reflect the various types of actions that are relevant in a control setting. For example, a number of trajectories should use a nonzero input to drive the system to various regions of the state space, which will assist the model in identifying the input dynamics. However, a few runs should also initiate the system away from the steady-state and let the states approach the steady-state under no control action, provided that the steady-state of interest is a stable one. If the data set is generated following the above best practices, it should yield an independent and identically distributed (i.i.d.) data set with maximum dynamic information and the least redundancy/repetition.

Lastly, when dealing with a specific type of a system, any unique characteristics of the system that may hinder or facilitate data generation and quality should be considered. For example, since the goal is to capture as much dynamic information as possible and not collect redundant data over a large period of the simulation with constant values for all variables (i.e., after the system reaches a steady state), when dealing with multiscale systems, techniques such as “burst sampling” have been proposed in [98]. Burst sampling refers to the use of a short sampling time in regions with higher gradients and faster dynamics, such as the fast transient of the fast subsystem(s) of a multiscale system, while reducing the sampling rate once the fast states converge to the slow manifold. Such advanced sampling strategies greatly reduce data storage requirements, and allow the user to retain only the most informative bits of data to to be used for modeling and control. Such advanced data acquisition strategies should be used instead of mere iterative procedures. On the other hand, if the system operates at or near an unstable steady-state, integrating the system for extended periods of time may lead to the states diverging, which will cause errors during run time and hinder the data generation. Hence, for unstable operating points, it is desirable to use multiple shorter trajectories. Such facilitation and difficulties of data generation must be considered on a case-to-case basis for the system being studied.

7.3.2.2 Data preprocessing

In any machine learning (ML) application, it is essential to preprocess the data before training a model. The two preprocessing steps required to apply SINDy are the train/test split and the normalization of the data set.

With respect to the split, the data set must first be split into the training and test sets. Most of the training data set is used to regress the model coefficients, while a small fraction of the training set is reserved as the validation set, which is used to tune the hyper-parameters. Once the optimal set of hyper-parameters is found, the model is finalized on the entire training data with the selected hyper-parameters. The final model is then bench-marked against the unseen data, which is the test set. The train-validation-test split ratio is arbitrary to an extent, although general rules and best practices exist. The training set should generally be the largest because the model performance is mostly related to the training data set, which is used to find the model parameters, while the test set is only used to gauge the model accuracy post-training. In fact, as long as the data set is i.i.d., increasing the size of the training set will always lead to an improvement of the model accuracy. The train-validation-test proportions are also determined by the application. For example, for methods with a large number of hyper-parameters to tune, such as neural networks, where usually large volumes of data are usually available, it may be more valuable to have a larger validation set, e.g., a 50-25-25 train-validation-test split. On the other hand, for SINDy, since there is only one key hyper-parameter to tune, a larger training set may be warranted, such as a 60-20-20 or 70-15-15 split. When the data set poses additional challenges such as noise, it may warrant an 80-10-10 split, since training noisy data is often a difficult task and requires significant amounts of data, especially if any smoothing or other, extraneous preprocessing steps are required.

A number of methods exist to normalize, i.e., center and scale the data set. Three common methods for normalization are the z-score scaler, Min-Max scaler, and Max-Abs scaler, of which the first two methods both center and scale the data, while the Max-Abs scaler only scales it. Specifically, the z-score scaler first centers the data set to the origin by subtracting the mean, and then scales the data set to have unit variance by dividing by the standard deviation. The Min-Max scaler divides each number by the range of the data after subtracting the minimum value of the data set and then adds the minimum value back to the scaled number in order to transform all data points to values between a lower and upper limit, usually 0 and 1, respectively. The Max-Abs scaler only scales the data to be between ± 1 by dividing the data set by its maximum absolute value, without any subtraction or centering. While the methods are described for a single variable, for the multivariate case, the above operations are independently carried out on each variable or column of the data set. For chemical processes, as the process inputs and outputs are typically written in deviation form from their steady-state values, further centering may not be as crucial; all variables will attain a value of zero at the steady-state. However, due to the large differences in the orders of magnitudes between the variables, such as between concentrations and temperatures, scaling the data set is necessary in most cases. When using SINDy, where the sign of the coefficients associated with certain terms can contain information on the process dynamics (e.g., an increase in the input heating rate should lead to an increase in the temperature), methods that scale without centering such as the Max-Abs scaler can be a reasonable starting point when deciding on a normalization method, as was also observed in some of our results.

7.3.2.3 Hyper-parameter tuning

In the basic SINDy algorithm, the model structure and accuracy are simultaneously controlled and balanced by a single hyper-parameter, λ. Therefore, tuning it is essential and usually carried out via a fine search or coarse-to-fine search. The latter is computationally efficient and used in this work. A coarse-to-fine search can be justified by the fact that, for appropriately scaled data, for most systems, no nonzero terms will remain in the SINDy model for large values of λ, such as λ that is an order of magnitude greater than the scaled data set. In contrast, extremely small values of λ will yield dense models that are prone to instability as well as redundant in the basis functions. Hence, a coarse search can be used to bound the region where a finer search can be carried out to identify the optimal model that yields the lowest loss or error metric. Figure 7.1 demonstrates how this process can be used to select the optimal model (corresponding to the orange point) through a very fine search or even models very close to the optimal in terms of accuracy by a much coarser search (corresponding to the green region). As expected, it can be observed that values of $\lambda > 1.0$

zero all terms in the SINDy model, leading to a constant error for all such λ. At the lower extreme of values of λ, the model is no longer sparse, and some terms that may even lead to an unstable model can start to have nonzero coefficients, in which case the MSE rapidly increases even beyond the case of all the terms being zero. This is especially the case since Fig. 7.1 is based on the work in [217], where the case of noisy data is considered.

The basis functions chosen for the candidate library are another central element of the SINDy method, which may be treated similarly as a hyper-parameter, in the sense that it is not entirely arbitrary and may require addition/removal of basis functions as necessary. Expanding it without computational considerations is not recommended as the overall optimization problem will then suffer from the curse of dimensionality, while also rendering the model more prone to instabilities due to dense model structures. Therefore, if there is any physical insight on the type of nonlinearities that are potentially relevant to the system of interest, this physical insight should be incorporated into the optimization search (e.g., biasing the order with which the nonlinearities are considered in the optimization search in an approach similar to the ALAMO modeling technique [6]). For chemical processes with nonlinear reaction terms, a common consideration may be to include exponential terms involving the temperature as the Arrhenius rate law is widely used in deriving mass and energy balances for reactors.

For estimating the time-derivative $\dot{X}$ in the right-hand side of Eq. (7.6), which is typically unavailable from sensor measurements, the ideal method to be used depends on the nature of the data set. For clean data, any finite difference-based approach such as forward, backward, or centered finite difference is usually adequate and will eventually yield similar results for the model coefficients at the end of the SINDy algorithm. However, if the data is noisy, finite differences

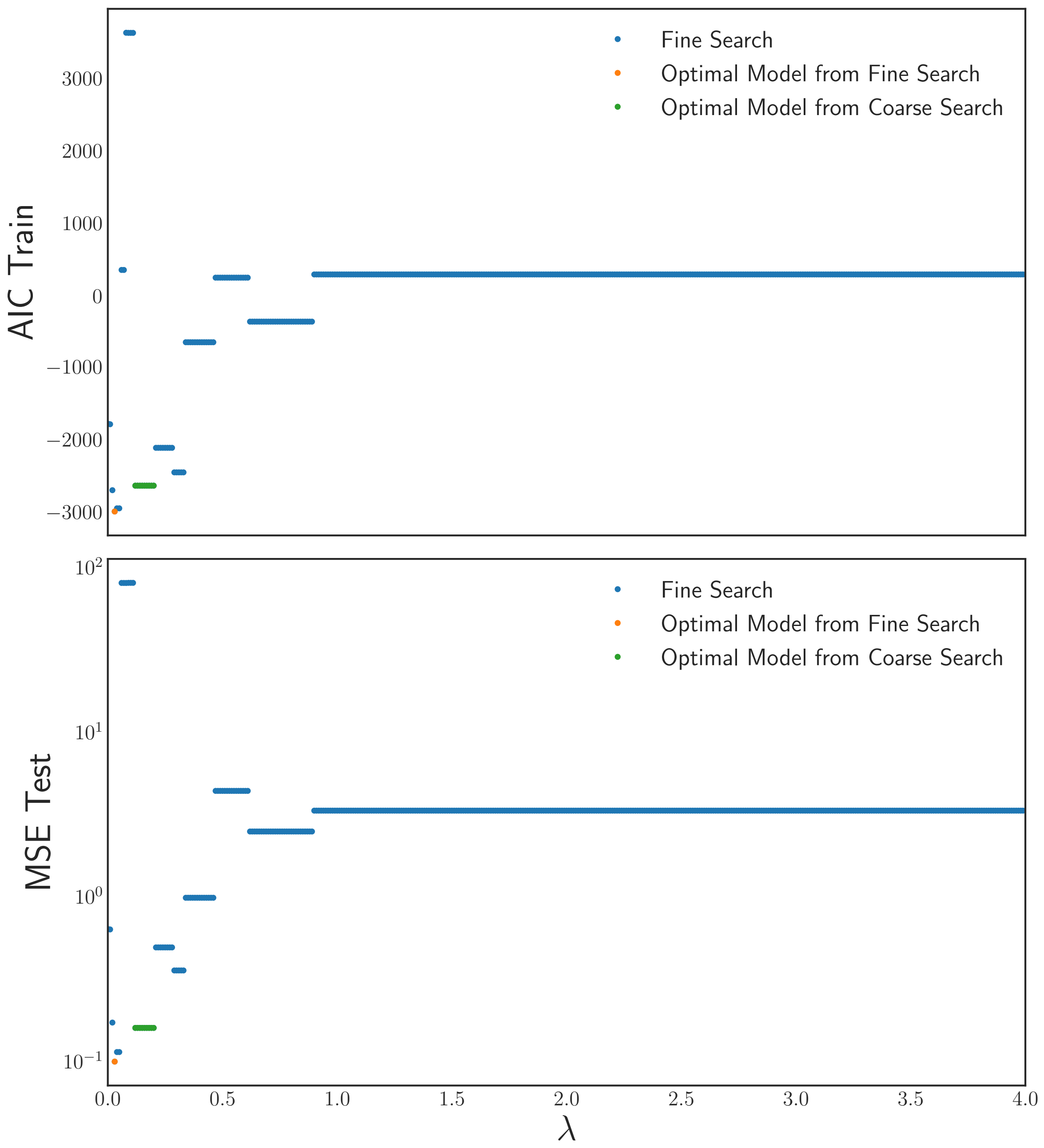


Figure 7.1: Values of two error metrics, the Akaike Information Criterion (AIC) and the mean-squared error, as functions of λ for model selection.

are unstable even at low noise levels. Hence, methods robust to noise such as the total variation regularized derivative (TVRD) and the smoothed finite-difference (SFD) have been proposed [77].

TVRD is based on the total-variation regularization, which has been widely used in image processing applications. In TVRD, the derivative is computed as the minimizer of a functional using gradient descent. In contrast, in SFD, the data set is first presmoothed using a filter, which may be a low-pass filter or the Savitzky-Golay filter, and finite-differences are then computed from the resulting, smoothed data set. As no gradient descent is involved, computationally, it is generally faster than TVRD. However, when both methods were used in [195], each method yielded some of the final, optimal models for the various cases studied, making them both reasonable choices to test.

7.3.3 Incorporation of SINDy within MPC

Model predictive control is an advanced control methodology that utilizes a model of the process to predict the states/output over a prediction horizon to compute the optimal control actions by solving an online optimization problem. The formulation of a Lyapunov-based model predictive controller (LMPC) that uses a sparse-identified ODE, $F_{si}(\cdot)$, as the process model is presented below:

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} \mathcal{C}(\tilde{x}(t), u(t)) \, \mathrm{d}t \tag{7.9a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{si}(\tilde{x}(t), u(t)) \tag{7.9b}$$

$$\tilde{x}(t_k) = x(t_k) \tag{7.9c}$$

$$u(t) \in \mathcal{U}, \; \forall \, t \in [t_k, t_{k+N}) \tag{7.9d}$$

$$\dot{\hat{V}}(x(t_k), u) \leq \dot{\hat{V}}\big(x(t_k), \Phi_{si}(x(t_k))\big), \quad \text{if } x(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{si}} \tag{7.9e}$$

$$\hat{V}(\tilde{x}(t)) \leq \rho_{si}, \; \forall \, t \in [t_k, t_{k+N}), \; \text{if } x(t_k) \in \Omega_{\rho_{si}} \tag{7.9f}$$

where $\tilde{x}$ is the predicted state trajectory, $S(\Delta)$ represents the set of piece-wise constant functions with a period of Δ, and N is the number of sampling periods within each prediction horizon. $\dot{\hat{V}}(x, u)$ is the time-derivative of the Lyapunov function and is equal to $\frac{\partial \hat{V}(x)}{\partial x} F_{si}(x, u)$. $u = u^*(t)$, $t \in [t_k, \, t_{k+N})$ denotes the optimal input sequence over the prediction horizon, which is provided by the optimizer. The LMPC applies only the first value in $u^*(t_k)$ over the next sampling period $t \in [t_k, \, t_{k+1})$, and solves the optimization again at the next sampling time t_{k+1}.

In the MPC formulation, Eq. (7.9a) is the objective function to be minimized and is chosen to be equal to the integral of $\mathcal{C}(\tilde{x}(t), u(t))$ over the prediction horizon. A typical cost function that achieves a value of zero at the steady-state in the absence of manipulated input action, while simultaneously weighing the deviation in both state and input from the origin is the quadratic stage cost, which is often used in LMPC and is formulated as follows:

$$\mathcal{C}(\tilde{x}(t), u(t)) = x^\top Q_1 x + u^\top Q_2 u \tag{7.10}$$

Equation (7.9b) describes the sparse-identified model that is used to predict the closed-loop states over the prediction horizon starting from the initial condition of Eq. (7.9c) while u is varied within the constraints defined by Eq. (7.9d). The last two constraints of Eq. (7.9e) based on the Lyapunov function, $V = x^\top P x$, guarantee that the closed-loop state either moves towards the origin at the next sampling time if the state is outside $\Omega_{\rho_{si}}$ or is contained within $\Omega_{\rho_{si}}$ for the entire prediction horizon once the state enters $\Omega_{\rho_{si}}$.

The generally nonlinear, non-convex optimization problem of Eq. (7.9) is solved at every sampling period, and the first entry of the optimal u^* calculated is sent to the actuator, following which the optimization is re-solved at the next sampling period using the new state measurement. The optimization is solved using the numerical solver Ipopt [145] with its Python front-end named PyIpopt. For the contractive constraint of Eq. (7.9e), the universal Sontag control law [140] or a well-tuned, stabilizing proportional-only controller may be used. It is important to note that the matrices P, Q_1, and Q_2 must be tuned for the LMPC to achieve the best results, and poorly tuned weight matrices may lead to the solver not converging to a solution within the sampling period or the maximum allowed number of iterations.

7.4 Reduced-order modeling for two-time-scale systems

Time-scale separation is a common phenomenon found in chemical processes such as distillation columns and catalytic continuous stirred-tank reactors (CSTRs) [91]. If the time-scale separation is not accounted for in a standard nonlinear feedback controller, the controller may be ill-conditioned or even unstable in closed-loop [99]. Due to the distinct slow and fast dynamics in such systems, the process will be represented by stiff ODEs in time when using SINDy without any modification. Such stiff ODEs, when integrated with an explicit integration method such as forward Euler, require a very small integration step size to prevent divergence and yield sufficiently accurate solutions. Hence, [135] used the mathematical framework of singular perturbations to propose the decomposition of the original two-time-scale system into two lower-order subsystems, each separately modeling the slow and fast dynamics of the original multiscale system. Specifically, following a short transient period, the fast states converge to a slow manifold and can be algebraically related to the slow states using nonlinear functional representations. In [135], we applied nonlinear principal component analysis (NLPCA) developed by [121] to capture the nonlinear relationship between the slow and fast states, while using sparse identification to derive well-conditioned, reduced-order ODE models for only the slow states that could then be integrated with much larger integration time steps due to their numerical stability. Once the slow states are predicted with the ODE model, it is possible to use NLPCA to algebraically predict the fast states without any integration.

Nonlinear principal component analysis is a nonlinear extension of principal component analysis (PCA). PCA is a commonly used dimensionality reduction technique that finds a linear mapping between a higher-dimensional space (of the data) and a lower-dimensional space with minimal loss of information by minimizing the squared sum of orthogonal distances between the data points and a straight line. NLPCA attempts to generalize this to the nonlinear case in two steps: first, a 1-D curve that passes through the "middle" of the data points known as the "principal curve" is found; second, the principal curve is parametrized in terms of distance of each point along the curve by using a feedforward neural network with nonlinear activation functions. Overall, to make a prediction of the state of the two-time-scale system, the measurement of the slow states at the current sampling time is passed to an explicit integrator (such as a Runge-Kutta scheme) that integrates the sparse-identified model to predict the slow states over the prediction horizon, which are then sent to the FNN to yield a prediction of the fast states.

Two-time-scale systems can be written in the form,

$$\dot{x}_s = f_s(x_s, x_f, u, \epsilon) \tag{7.11a}$$

$$\epsilon \dot{x}_f = f_f(x_s, x_f, u, \epsilon) \tag{7.11b}$$

where $x_s \in \mathbb{R}^{n_s}$ and $x_f \in \mathbb{R}^{n_f}$ denote the slow and fast states, respectively, with $n_s + n_f = n$. ϵ is a small positive parameter that represents the ratio of slow to fast dynamics of the original system. By making standard assumptions from the singular perturbation framework, the slow subsystem of Eq. (7.11a) can be rewritten in the form required for sparse identification,

$$\dot{\hat{x}}_s = F_{si}(\hat{x}_s, u) := \hat{f}(\hat{x}_s) + \hat{g}(\hat{x}_s)u, \qquad \hat{x}_s(t_0) = x_{s0} \tag{7.12}$$

where F_{si} is the sparse-identified slow subsystem.

In the first step of NLPCA, we capture the unidimensional principle curve in the n-dimensional state space to find the nonlinear algebraic relationship between the slow and fast states as shown in Fig. 7.2. The curve, denoted by $\mathcal{P}(\mu)$ is parametrized in terms of the ordered arc-length along the curve, μ. If $\bar{x} \in \mathbb{R}^n$ is the full-state vector, we can define the projection index $\mu_{\mathcal{P}} : \mathbb{R}^n \to \mathbb{R}$ as:

$$\mu_{\mathcal{P}}(\bar{x}) = \sup_{\mu \in \mathbb{R}} \{\mu : \big\|\bar{x} - \mathcal{P}(\mu)\big\| = \inf_{\mu' \in \mathbb{R}} \big\|\bar{x} - \mathcal{P}(\mu')\big\|\} \tag{7.13}$$

with μ' being a notational substitute for μ. Based on the above definition and denoting the expectation of a random variable by $\mathbb{E}$, the curve can be defined as:

$$\mathbb{E}\big(\bar{x}|\mu_{\mathcal{P}}(\bar{x}) = \mu\big) = \mathcal{P}(\mu) \tag{7.14}$$

where the expectation operator is approximated using a combination of scatter-plot smoothing and locally weighted regression when only discrete time-series data is available. Since the output of the first step of NLPCA, the principal curve, is a non-queryable model, an FNN is used to capture the identified principal curve.

With respect to the structure of the FNN, it is necessary to use at least one hidden layer with a sigmoid activation function, $\sigma(x) = 1/(1+\mathrm{e}^{-x})$, to exploit the universal approximation property of neural networks [60, 61]. To improve the network capability, a two-hidden-layer FNN was used in this work, as depicted in Fig. 7.3. The learning rate, which is the most influential hyper-parameter, requires careful tuning to obtain the optimal FNN model in the second step of NLPCA.

An LMPC that uses Eq. (7.12) as the process model of Eq. (7.9b) may be constructed. Such an LMPC will predict the slow states of the two-time-scale system and optimize the cost function

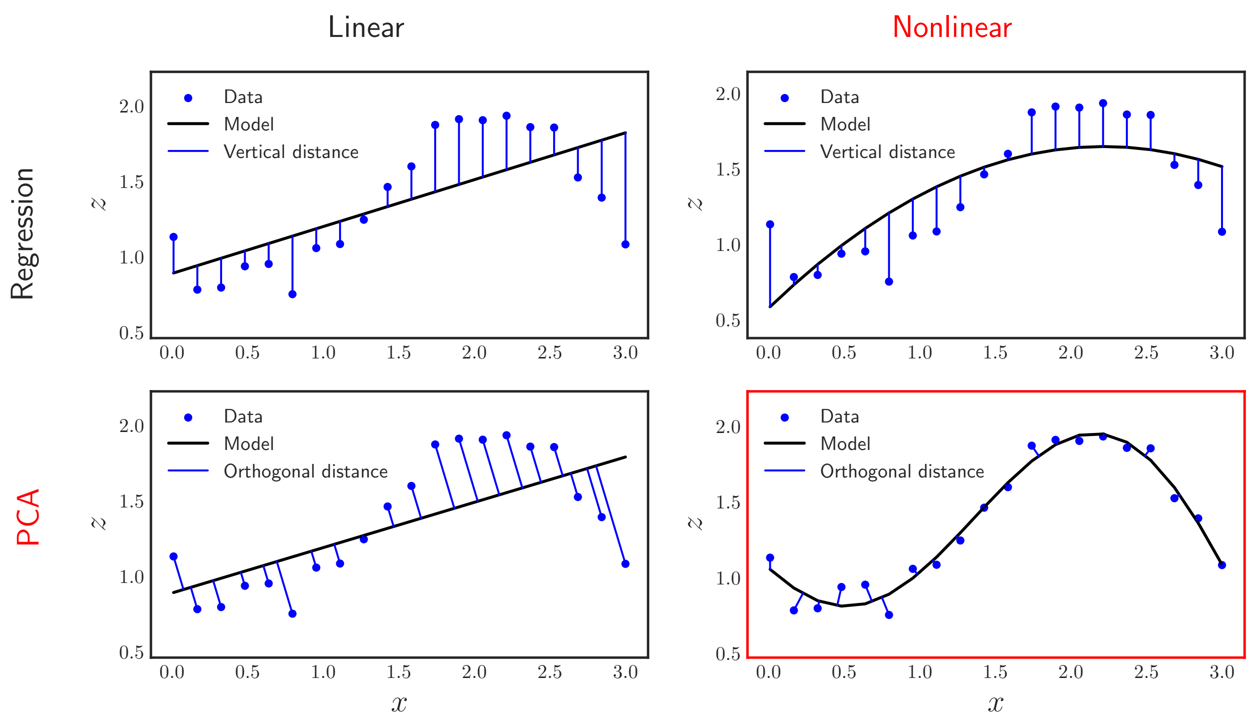


Figure 7.2: Demonstration of the evolution of NLPCA based on PCA and its relation to nonlinear regression.

based on the predicted slow states. Due to the coupled nature of the states, it is sufficient to stabilize the slow states to guarantee asymptotic stability for the entire system. However, if computational resources are available, the FNN may be used to predict the fast states, and the LMPC can then account for the full-state of the system. In [155], only the slow subsystem was used to ensure the LMPC optimization can be solved within every sampling period.

The primary advantage of the reduced-order model in LMPC is that the lower computational cost of the SINDy model inference, with nearly zero loss in model accuracy, directly impacts the difficulty of the optimization required to be solved by the LMPC. Hence, the LMPC based on the reduced-order SINDy model can use a higher prediction horizon, which has the potential to improve closed-loop performance in terms of faster convergence to the origin and a lower total

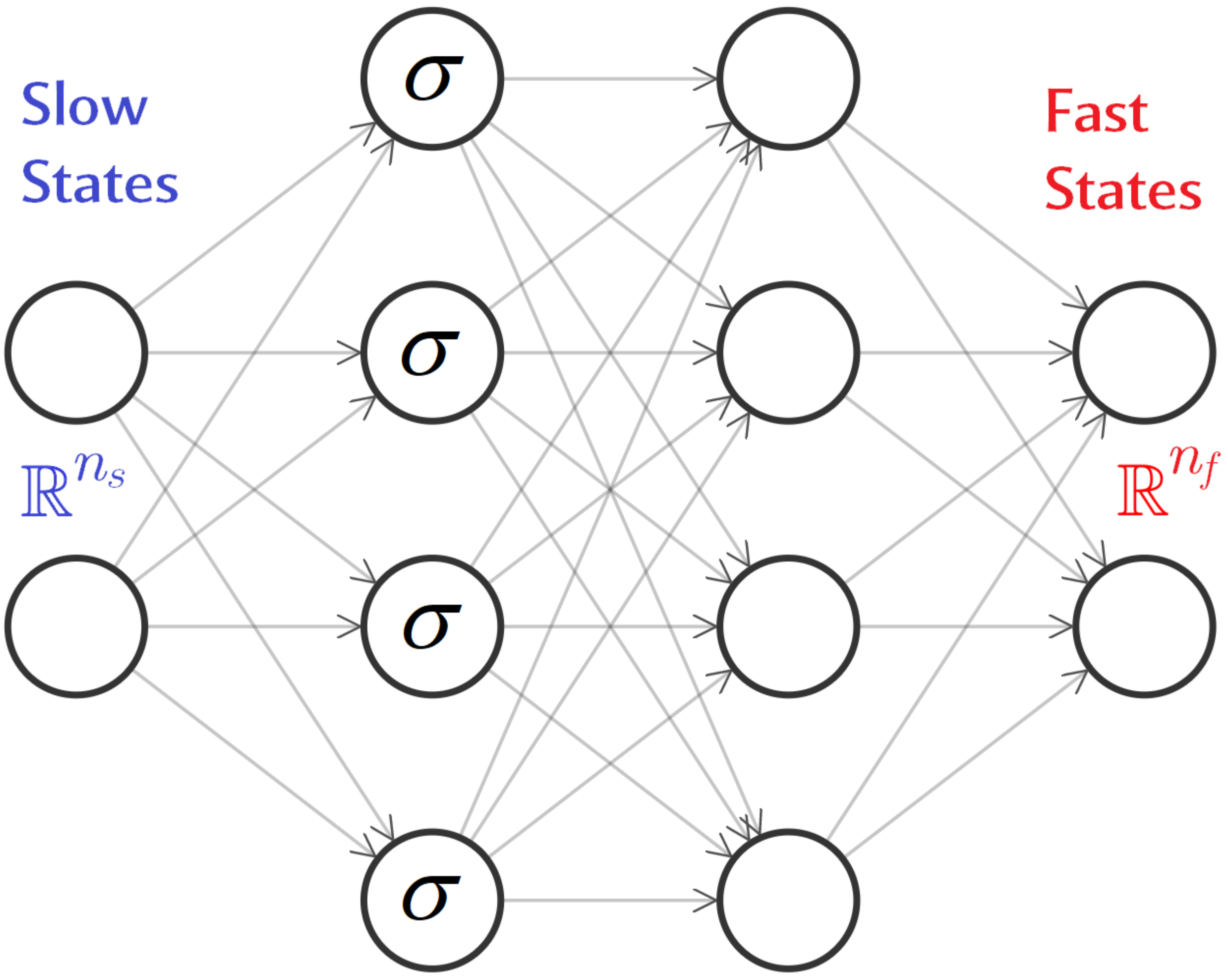


Figure 7.3: Structure of the neural network used for NLPCA-SI

cost function over the simulation duration, the former of which is demonstrated most clearly in the concentration profile in Fig. 7.4.

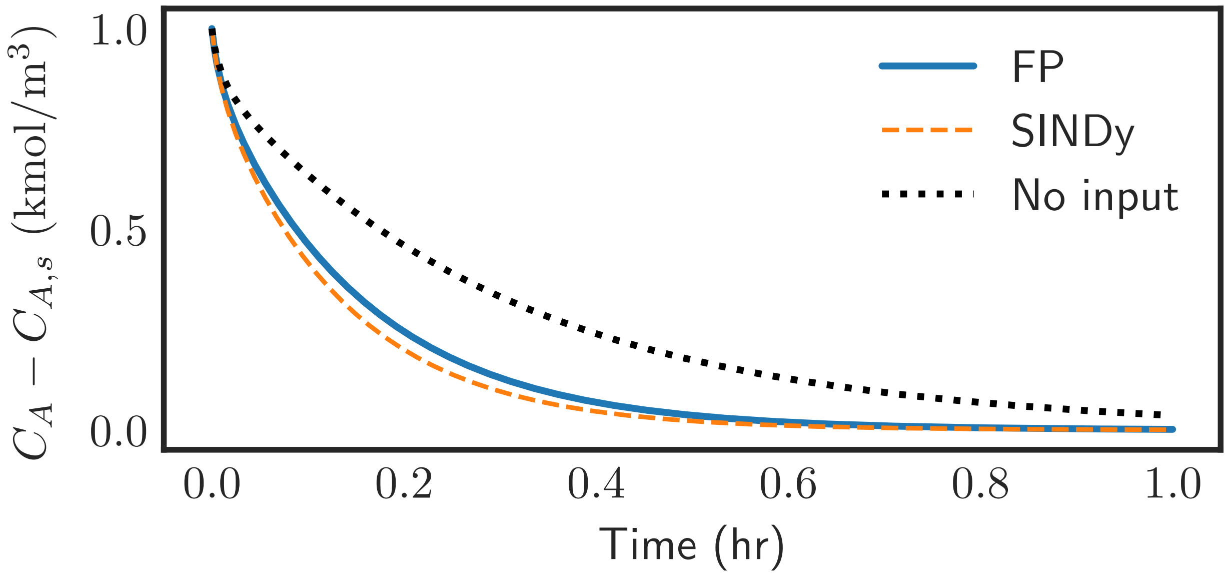


Figure 7.4: Concentration (state) profile for a CSTR in closed-loop under the LMPC utilizing the first-principles (FP) model with $N = 16$ (blue line) and the SINDy slow model with $N = 24$ (orange line).

7.5 Subsampling and co-teaching in the presence of high sensor noise

A key step in the sparse identification procedure is the estimation of the time-derivatives of the states when it cannot be measured directly, as is the case in most process systems. From a survey of the literature, since the conceptualization of SINDy in [77], several advancements in the algorithm have been proposed to handle noisy data. However, most articles that investigate the effect of noise on SINDy add noise to the pre-computed derivatives (from clean data) and/or use very low levels of noise that can be easily smoothened. One example is the SINDy-PI algorithm proposed by [113] and improved by [114]. Through case studies, [114] demonstrated that even the improved algorithm could only handle noise with a maximum variance of 10^{-4}, which is very small in the context of process systems. Although a number of works can be found that focus on alternate approaches to build dynamical models in the presence of noise, such as Runge-Kutta time-steppers with embedded neural networks to handle the nonlinear elements [185–188], these are alternatives to SINDy rather than improvements upon the original SINDy method. As a result, the methods to assist the modeling of noisy data as well as the subsequent results are largely different from SINDy and its extensions. For example, the unexpected results of [187] when using Runge-Kutta time-steppers were later explained using well-known characteristics of neural networks. More advanced time-steppers such as the work of [188] also emphasize their limitations when integrating the models from new initial conditions or attempting to capture dynamics away from a steady state, both of which are relevant in control-centric applications. While a detailed discussion of the comprehensive literature can be found in [195], in summary, one paper proposed an improvement upon the SINDy algorithm in the presence of moderate noise that demonstrated promise and could be developed further. This method, proposed by [116], termed subsampling-based threshold sparse Bayesian regression (SubTSBR), involved randomly subsampling a fraction of the entire data set multiple times and selecting the best model by using a model-selection criterion. The issue of noisy data has also been studied in the field of computer science, where fitting a neural network to noisy data often leads to the neural network overfitting the data and capturing the noisy pattern instead. A recent technique proposed to overcome this challenge is co-teaching, where a simplified first-principles process model is used to generate noise-free training data to assist the model training step by reducing overfitting. In this section, we propose a novel extension to SINDy by combining it with subsampling and co-teaching to handle highly noisy sensor data.

Subsampling is a classical statistical technique where a fraction of the total number of samples in a data set are randomly extracted and analyzed to estimate statistical parameters [181] or speed up algorithms [182]. However, subsampling can also be used to instead improve the modeling accuracy of SINDy when the data set is highly noisy. This is because common regression methods such as least squares utilize the complete data set by assuming that only a small fraction of the data samples are highly noisy or outliers. As a result, if the entire data set is used, the higher percentage of "good" data samples should smooth the large noise present in the data set. However, this assumption breaks down if the noise is either very high or uniformly present throughout the data set. In such a case, there are insufficient "good" data samples to smooth out the noise from the very highly corrupted data samples. In the context of SINDy, subsampling refers to selecting random fractions of the data set multiple times in order to sample only the less noisy data points for carrying out the sparse regression. The key requirement for subsampling is that the number of unknown weights to be estimated in the SINDy procedure have to be fewer than the number of total data samples available (i.e., the problem has to be overdetermined), which is the case for most practical data sets. Although as a standalone improvement, subsampling greatly improves the performance of SINDy under moderate noise levels, it is insufficient at higher noise levels, where co-teaching becomes incumbent.

Co-teaching is a method that has been used in the field of computer science, primarily in image recognition, where neural networks are trained to categorize images into pre-defined classes. However, often, a small proportion of the images in the training data set may be mislabeled, greatly deteriorating the performance of the neural network. As manually relabeling vast amounts of images is not feasible, the method of co-teaching was proposed wherein newly generated noise-free data is fed during model training to reduce the impact of the noisy data. The concept has recently been extended to regression problems, specifically the modeling of dynamical systems using long short-term memory (LSTM) networks [153, 154]. The central idea of co-teaching, which was highlighted and proven in [154], is that neural networks fit simpler patterns in the early iterations of model training, which implies that noise-free data will yield low values of the loss function, while noisy data will tend to produce high loss function values. Therefore, the training can be made more robust to noise and overfitting if the noisy data is augmented with a nonzero proportion of noise-free data from simulations of simplified, approximate first-principle models that can be derived for the complex, original nonlinear system.

Improving the sparse identification algorithm with both subsampling and co-teaching enables it to tackle consistently noisy data sets where subsampling alone is insufficient. This is because subsampling only subsamples, in the best case scenario, the least noisy data points, which are still too noisy to yield an adequate model. In the proposed method, first, a random subset of the entire data set X_{noisy} and its corresponding $\dot{X}_{\text{noisy}}$ are sampled, which are then mixed with noise-free data generated from approximate first-principles models of the process, X_{FP} and $\dot{X}_{\text{FP}}$. The resulting mixed data set is used to solve for the unknown weights of the s terms in the SINDy function library. Once a model is identified, a model-selection criterion is used to evaluate the model performance. Three parameters must be specified in the algorithm: $p \in (0, 1)$ or the subsampling fraction, $q \in (0, 1)$ or the noise-free subsampling fraction, and $L \geq 1$, which is the number of times to independently subsample and identify a SINDy model. The algorithm randomly subsamples and mixes $p \times m$ data points from the noisy data set with $q \times m$ data points from the noise-free data set to produce the data and derivative submatrices, X_i and $\dot{X}_i$, respctively, for subsample i with $i = 1, 2, \ldots, L$. U_i are the corresponding $(p + q) \times m$ points from the input matrix U. The sparse regression equation to be solved is then

$$\dot{X}_i = \Theta(X_i, U_i)\Xi_i \tag{7.15}$$

where Ξ_i are the coefficients associated with each library function that is identified using the data subset $\{X_i, \dot{X}_i, U_i\}$. Once Ξ_i is determined and, therefore, the i^{th} ODE model is found, the model selection criterion is used to extract the optimal model. An example of a model selection criterion that balances the error with the model sparsity, which is crucial for SINDy, is the Akaike Information Criterion given by the expression,

$$\text{MSE} = \frac{1}{m}\sum_{j=1}^{m}\big(x(t_j) - \hat{x}(t_j)\big)^2 \tag{7.16}$$

$$\text{AIC} = m \log \text{MSE} + 2L_0 \tag{7.17}$$

where MSE is the mean-squared error, and L_0 denotes the 0th norm, which is equal to the number of nonzero terms in the sparse-identified model.

The hyper-parameters unique to the subsampling with co-teaching algorithm, besides the ones described in Section 7.3.2.3, are the values of p, q, and L. It should be noted that the goal is to capture the original noisy data rather than the noise-free data from first-principles simulations. Hence, the fraction q should generally be small, while p can be any real number between 0 and 1 as long as both metrics satisfy $p + q \leq 1$. While increasing L will generally improve the model performance because a larger number of sub-models are identified for the optimal model to be chosen from, the computational costs of increasing L must be considered. Figure 7.5 shows the flow of the data throughout the algorithm.

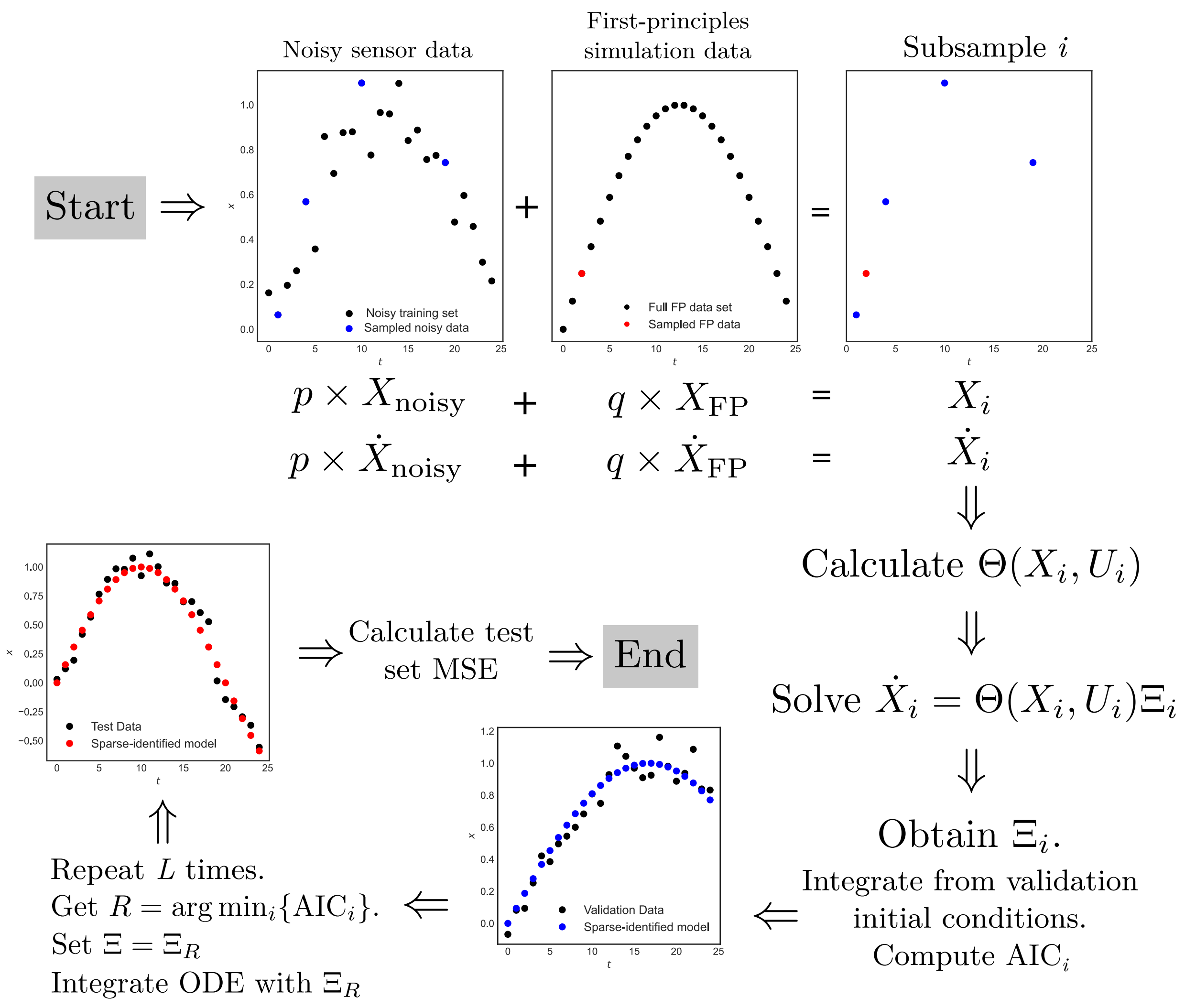

Start
Noisy sensor data
Noisy training set
Sampled noisy data
First-principles simulation data
Full FP data set
Sampled FP data
Subsample i
$p \times X_{\text{noisy}} + q \times X_{\text{FP}} = X_i$
$p \times \dot{X}_{\text{noisy}} + q \times \dot{X}_{\text{FP}} = \dot{X}_i$
Calculate $\Theta(X_i, U_i)$
Solve $\dot{X}_i = \Theta(X_i, U_i)\Xi_i$
Obtain Ξ_i.
Integrate from validation initial conditions.
Compute AIC$_i$
Validation Data
Sparse-identified model
Repeat L times.
Get $R = \arg\min_i\{\text{AIC}_i\}$.
Set $\Xi = \Xi_R$
Integrate ODE with Ξ_R
Test Data
Sparse-identified model
Calculate test set MSE
End

Figure 7.5: Data flow diagram of subsampling with co-teaching for noisy data.

Open-loop modeling results for a CSTR system are shown in Fig. 7.6, where the base SINDy model is observed to deteriorate in performance at the level of noise considered (Gaussian noise with a standard deviation of $\sigma_T = 4\,\mathrm{K}$ in the temperature). Subsampling, even by itself, greatly improves the SINDy model performance, while subsampling with co-teaching further improves the performance. The improvement using co-teaching is most significant at the highest levels of noise considered (Gaussian noise with a standard deviation of $\sigma_T = 6\,\mathrm{K}$ for the temperature) since models constructed using only subsampling even diverged in some cases [195]. Visually, the models can be assessed in terms of how close the model predictions are to the data as well as whether the states evolve in the correct direction. Although this is difficult to do for the entire simulation domain at the higher levels of noise, analyzing specific time domains in Fig. 7.6 can reveal differences between the models. In Fig. 7.6, in the ranges $t \in [2, 4] \cup [14, 16]$, the base SINDy model clearly deteriorates and deviates from the other models and the data, which is mostly concentrated much higher, near the steady-state, indicating the poor performance of the base SINDy model in these regions. The models using subsampling with and without co-teaching can be further differentiated in the regions $t \in [6, 10] \cup [12, 14]$, where the subsampling-only model predicts smaller deviations from the steady-state, but the data deviates further from the steady-state than predicted by either subsampling-based model. Therefore, the co-teaching-based subsampling model is closer to the data than the subsampling-only model in these ranges where the states deviate further from the steady-state. However, especially when dealing with noisy data, the modeling performance is best characterized quantitatively in terms of the MSE, which are shown in Table 7.1. The MSE for subsampling with co-teaching is consistently the lowest across all noise levels except the lowest noise level, where all methods show very similar MSE and the differences are insignificant because

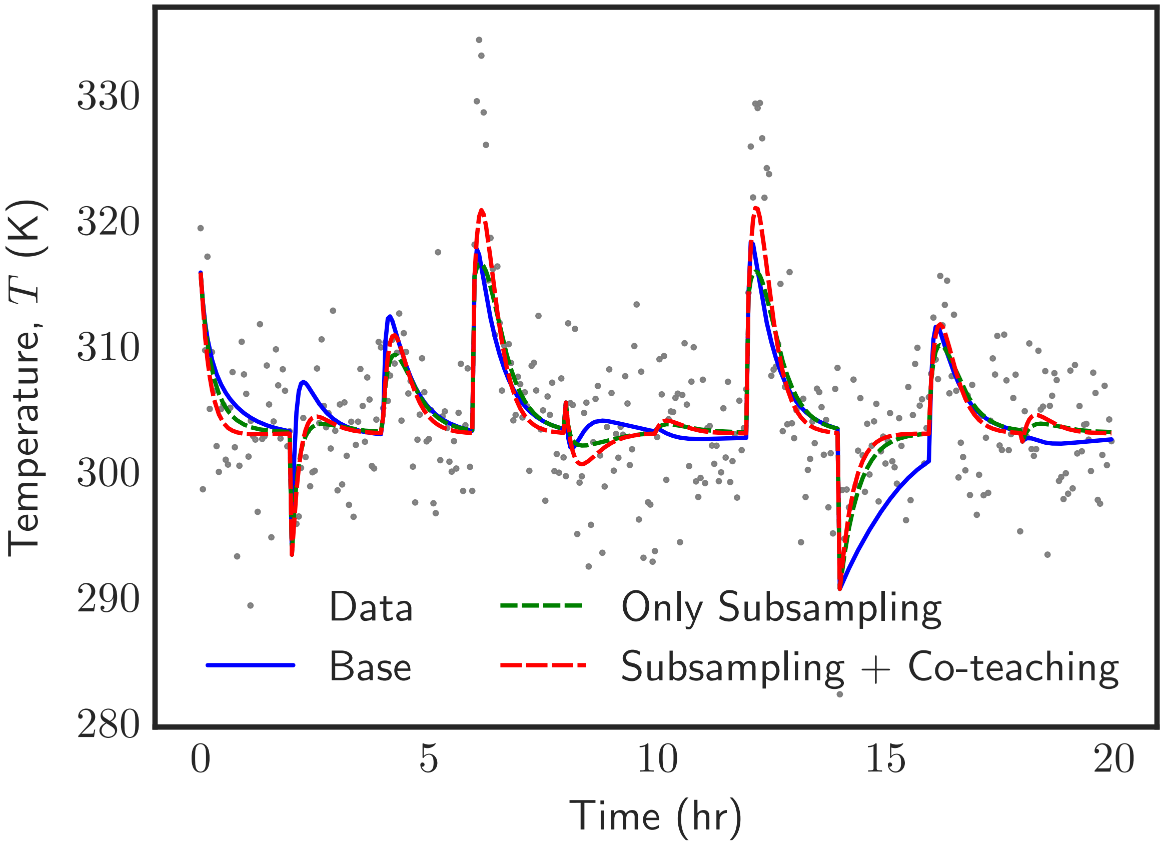


Figure 7.6: Comparison of original noisy data (grey dots) with results from sparse identification without any subsampling (blue line), subsampling without co-teaching with $p = 0.2$ (green line), and subsampling with co-teaching and $p = 0.16, q = 0.04$ (red line) for the temperature T of a CSTR system.

of the superior performance of the models from all three methods. At higher noise levels, the differences become more significant, with the subsampling-only based model even diverging when $\sigma_T = 6\,\mathrm{K}$. At low to moderate noise levels, however, the MSE of the models using subsampling, whether with co-teaching or not are very similar. Therefore, co-teaching should be used once the model performance from using only subsampling deteriorates.

Table 7.1: Test set MSE for the CSTR system for four noise levels

σ_T (K)	Base	Only Subsampling	Subsampling + Co-teaching
0.4	0.01113	0.01059	0.01102
2	0.10510	0.09922	0.10370
4	0.49837	0.40037	0.36283
6	0.98210	1.89607	0.77613

7.6 Ensembled-based dropout-SINDy to model highly noisy industrial data sets

While subsampling with co-teaching is a viable option to tackle the issue of high sensor noise in the data measurements, the primary drawback of co-teaching is its requirement for a first-principles process model that is at least similar to the original system with respect to the dynamics and the steady-state values. However, in the case of industrial data, the dynamics may be far too complex for any theoretically derived ODE to adequately capture the system. Therefore, for the case of dealing with high levels of industrial noise, a new direction and improvement on SINDy is proposed, which is a form of ensemble learning that we term “dropout-SINDy”.

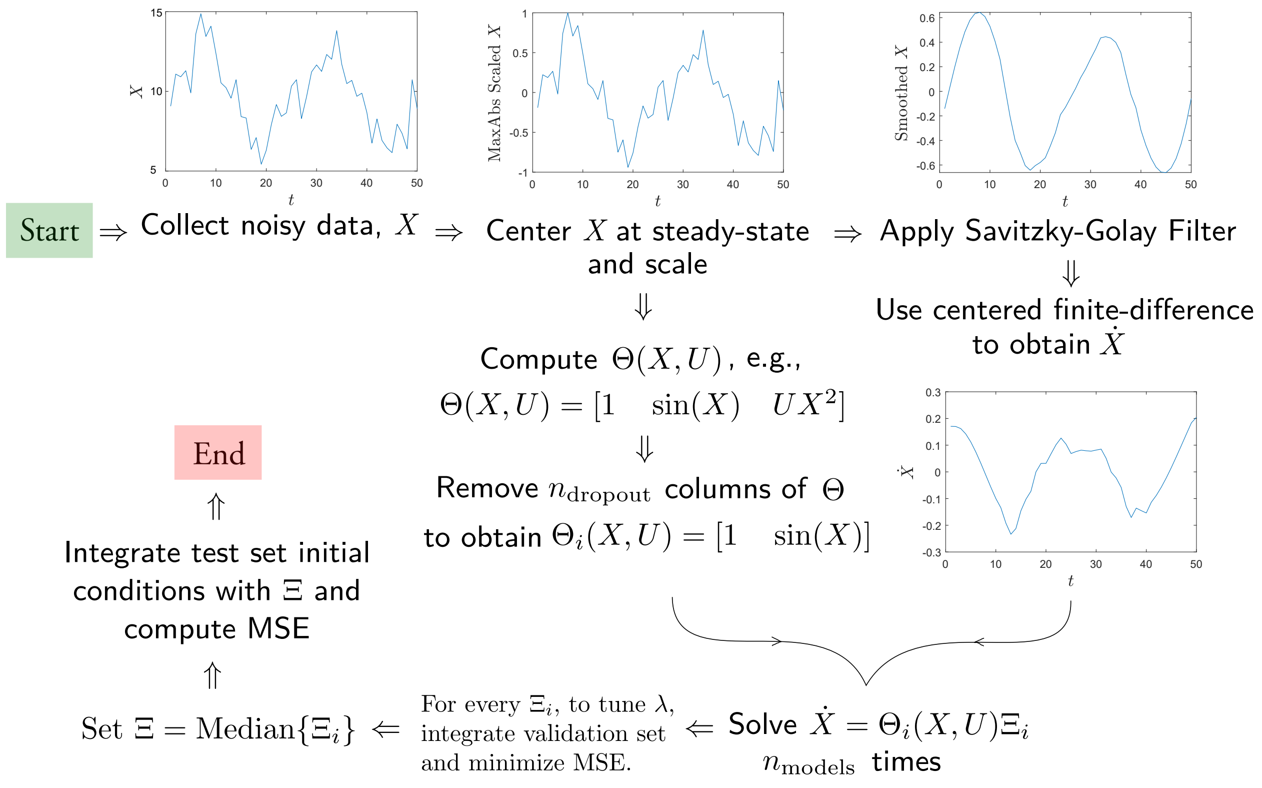
Ensemble learning refers to the use of multiple models in place of one model. Homogeneous ensemble learning involves the use of the multiple models of the same type, while heterogeneous ensemble learning strategies use a combination of different types of models to improve the predictive performance. In this work, only homogeneous ensemble learning is considered. However, in the context of SINDy, even the terminology, “homogeneous ensemble learning”, can refer to two distinct methods: either the data set can be subsampled to produce multiple models with the same underlying model structure, or multiple models with varying function libraries may be built using the same data set. The subsampling method described in Section 7.5 is an example of the former, but it was shown in [195] that subsampling, by itself, cannot improve the SINDy algorithm under high noise levels. In contrast, for the case of industrial noise, the proposed dropout-SINDy method uses only a fraction of the function library Θ to identify each submodel. Hence, multiple models can be identified, each with a random subset of the library. Similarly to co-teaching, this can reduce the impact of noisy data and, additionally, improve the stability properties of the SINDy models because a large number of nonzero terms (a dense coefficient matrix Ξ) can often lead to instabilities. The sparse regression equation to be solved for dropout-SINDy is similar to Eq. (7.15), but the state, input, and derivative data sets remain as X, U and $\dot{X}$, respectively, while only the library Θ_i and coefficients Ξ_i are varied between the n_{models} models in the ensemble, where $i = 1, 2, ..., n_{\text{models}}$:

$$\dot{X} = \Theta_i(X, U)\Xi_i \tag{7.18}$$

Equation (7.18) is solved using a different subset of the computed library Θ_i each time to find the corresponding set of model coefficients for the nonzeroed terms, Ξ_i. In each Θ_i, n_{dropout} library functions are randomly dropped out, with the corresponding entries in Ξ_i also being zeroed before solving Eq. (7.18). Once all the sub-models are found, the final model must be selected from the $\Xi_i, \ldots, \Xi_{n_{\text{models}}}$. In this case, although the mean, median, and mode are all possible methods to find the central tendency of all the Ξ_i, the mean is likely to yield dense models because even one nonzero value for a certain coefficient in any one of the sub-models will cause the coefficient to be nonzero. In contrast, the median and mode do not suffer from this. However, the mode may not be useful since even two sub-models with a zero coefficient for a library term will lead to the term being zeroed if none of the other nonzero values are repeated exactly equally, leading to excessive sparsity. Hence, the median is determined to be the most reasonable measure of central tendency for dropout-SINDy.

The number of functions of the candidate library to be omitted in each model, n_{dropout}, as well as the number of models, n_{models}, must be tuned when building a dropout-SINDy model. A very small value of n_{dropout} implies that the sub-models in the ensemble are very similar to the base SINDy model without any dropout, negating most if not all performance gains of the proposed method. But if n_{dropout} is too large, excessive sparsity will lead to models that lack the complexity required to capture the dynamics. Similarly, a small value of n_{models} may lead to the optimal model not being identified as the search is conducted over a smaller set, but increasing n_{models} also increases computational costs and might even promote instability if the median of the model coefficients is shifted by a larger proportion of poor models. Hence, this balance between computational cost, model improvement, model complexity, and stability must be considered when tuning n_{models} and n_{dropout} when using dropout-SINDy. The data flow throughout the algorithm is outlined in Fig. 7.7.

In this section, "industrial" data refers not to an experimental data set but data generated from a chemical process simulated in the high-fidelity chemical process simulator, Aspen Plus Dynamics, which is a widely used simulator in the chemical sector that has been used to build steady-state and dynamic simulations of chemical processes to aid chemical engineers in process design and optimization. Chemical process simulators have several advantages over first-principles models as they contain numerous built-in packages to handle most common unit operations, thermodynamic properties, molecular interactions, etc., which result in significantly more accurate models that



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 $\Theta(X,U)$, e.g., $\Theta(X,U) = [1 \quad \sin(X) \quad UX^2]$
Remove n_{dropout} columns of Θ to obtain $\Theta_i(X,U) = [1 \quad \sin(X)]$
Solve $\dot{X} = \Theta_i(X,U)\Xi_i$ 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
X
MaxAbs Scaled X
Smoothed X
$\dot{X}$
t

Figure 7.7: Data flow diagram of Dropout-SINDy for noisy, industrial data.

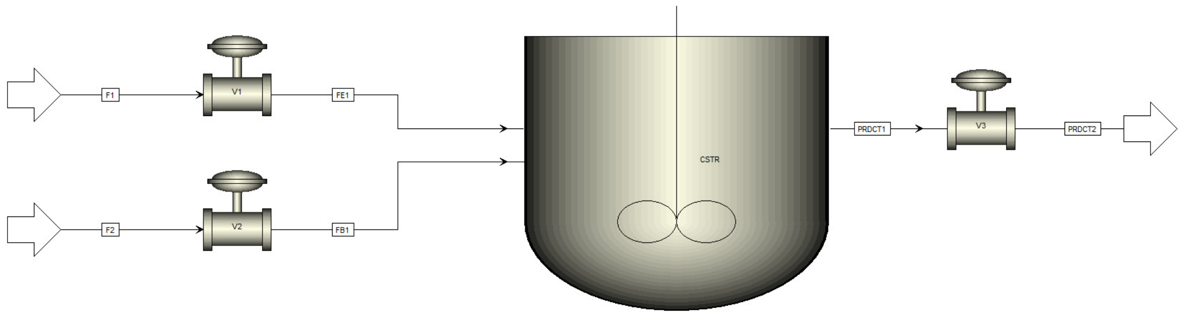


Figure 7.8: Aspen Plus model process flow diagram of an ethylbenzene production process.

more closely represent the plant process dynamics. In [217], Aspen Plus Dynamics was used to build the process flow diagram shown in Fig. 7.8, which was then used for both data generation as well as closed-loop simulations in order to imitate the industrial process.

When using the basic SINDy algorithm to model the highly noisy industrial data from Aspen Plus Dynamics, it is found that basic SINDy is unable to model the dynamics or even correctly predict the final steady-state of the open-loop system, the latter of which greatly affects the performance of a controller. However, when dropout-SINDy is used on the industrial data set, it is able to capture most of the dynamics and correctly predict the final steady-state values of the states. When an MPC is designed with the dropout-SINDy model, it can be demonstrated to achieve closed-loop stability and converge to the steady-state faster and with less energy and overshoot than a corresponding proportional-controller as shown in Fig. 7.9.

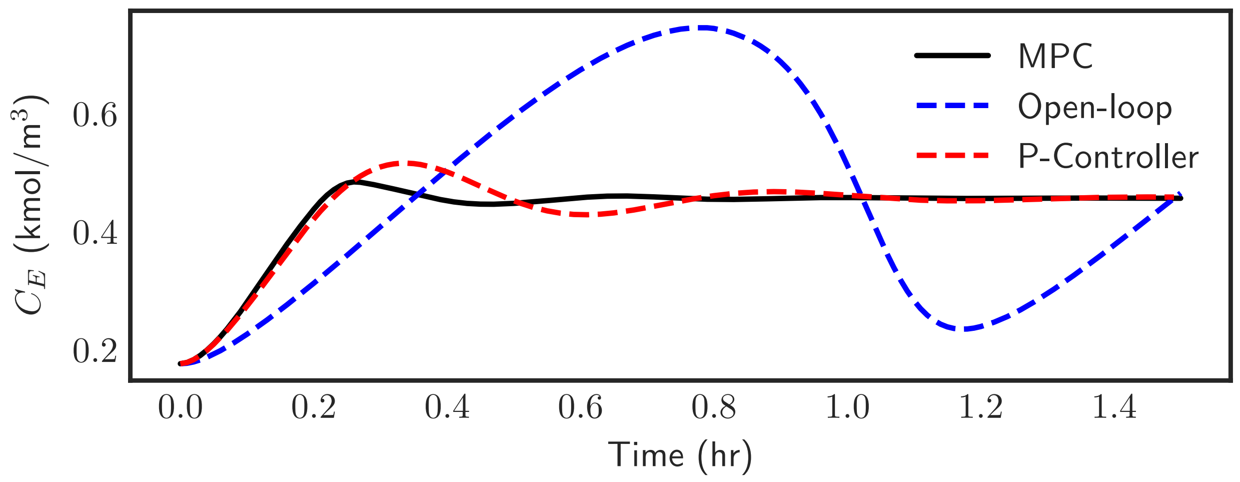


Figure 7.9: State and input profiles for a CSTR in closed-loop under no control (blue line), a P-controller (red line), and the LMPC utilizing the dropout-SINDy model (black line) throughout the simulation period $t_p = 1.5$ h.

7.7 Demonstration of the use of SINDy to model a nonlinear chemical process

In this section, the modeling of a highly nonlinear CSTR operating at an unstable steady-state using SINDy is considered. Specifically, a perfectly mixed, nonisothermal CSTR where an irreversible, exothermic reaction with second-order kinetics, A $\xrightarrow{k}$ B, takes place is studied. The rate constant of the reaction, k, is not assumed to be constant and, instead, an Arrhenius relation of the following form is used to determine the rate constant as a function of the Kelvin temperature, T:

$$k = k_0 \mathrm{e}^{-\frac{E}{RT}} \tag{7.19}$$

where k_0, E, and R represent the pre-exponential constant, activation energy of the reaction, and the ideal gas constant, respectively. Using material and energy balances, the differential equation model describing the CSTR dynamics is derived as follows:

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = \frac{F}{V}(C_{A0} - C_A) - k_0 \mathrm{e}^{-\frac{E}{RT}} C_A^2 \tag{7.20a}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{F}{V}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 \mathrm{e}^{-\frac{E}{RT}} C_A^2 + \frac{10^3 Q}{\rho_L C_p V} \tag{7.20b}$$

where C_A, V, and T denote the concentration of reactant A in the reactor, the volume of the reacting liquid inside the reactor, and the time-varying absolute temperature of the reactor. The concentration of species A in the inlet stream, the inlet temperature, and the volumetric flow rate fed to the reactor are represented by C_{A0}, T_0, and F, respectively. A heating jacket supplies/removes heat to/from the CSTR at a rate of Q. The density and heat capacity of the reacting liquid are assumed to have constant values of ρ_L and C_p, respectively, while ΔH denotes the enthalpy of the reaction. The values of the process parameters are provided in Table 7.2. With the values from Table 7.2

Table 7.2: Parameter values for nonisothermal CSTR example

$F = 5.0\,\mathrm{m^3/h}$	$V = 1.0\,\mathrm{m^3}$
$k_0 = 8.46 \times 10^6\,\mathrm{m^3\,kmol^{-1}\,h^{-1}}$	$E = 5.0 \times 10^4\,\mathrm{kJ/kmol}$
$R = 8.314\,\mathrm{kJ\,kmol^{-1}\,K^{-1}}$	$\rho_L = 1000.0\,\mathrm{kg/m^3}$
$\Delta H_r = -1.15 \times 10^4\,\mathrm{kJ/kmol}$	$T_0 = 300\,\mathrm{K}$
$C_{A0_s} = 4\,\mathrm{kmol/m^3}$	$Q_s = 0\,\mathrm{MJ/h}$
$C_{A_s} = 1.95\,\mathrm{kmol/m^3}$	$T_s = 402\,\mathrm{K}$
$C_p = 0.231\,\mathrm{kJ\,kg^{-1}\,K^{-1}}$	

substituted into Eq. (7.20), the exact ODE model to be identified using SINDy can be found to be

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = 5C_{A0} - 5C_A - 8.46 \times 10^6 \mathrm{e}^{-\frac{6013.95236949723}{T}} C_A^2 \tag{7.21a}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = 1500 - 5T + 4.211688 \times 10^8 \mathrm{e}^{-\frac{6013.95236949723}{T}} C_A^2 + 4.3Q \tag{7.21b}$$

The objective is to build a SINDy model for the CSTR system of Eq. (7.20), ideally for the entire state-space or at least a large region of the state-space around the desired operating point, which is the unstable steady-state, $(C_{A_s}, T_s) = (1.95\,\mathrm{kmol/m^3}, 402\,\mathrm{K})$. The factors that most significantly impact the quality of the SINDy model for this system were found to be the data generation and the candidate library of basis functions considered for $\Theta(X, U)$, both of which are discussed in detail over the next two subsections. To compare models quantitatively, for the sake of brevity, rather than reporting every model obtained from each data generation method or candidate library considered, in the rest of this section, the maximum absolute error in the Kelvin temperature will often be reported because the errors in the temperature are larger in terms of absolute value and intuitively understood.

Remark 7.1. *Due to the explicit nature of SINDy models, once the ODE models are obtained from SINDy, incorporating them into an MPC is generally straightforward. The challenge of SINDy-based MPC, however, lies in the modeling rather than MPC implementation, as opposed to entirely black-box approaches such as recurrent neural networks and other deep learning models, which can approximate practically any input/output data if provided with sufficient data and tuned thoroughly, but can encounter computational and technical challenges when implemented in closed-loop MPC. Hence, this section focuses solely on the modeling of the nonlinear CSTR, since past works [155, 195] have already demonstrated the application of SINDy models in MPC with open- and closed-loop simulation results for a diverse array of systems. The goal of this section is to familiarize the reader with the intricacies of building SINDy models in a chemical engineering paradigm.*

7.7.1 Data generation

For system identification, the data set used to identify the system is a crucial element. Hence, the data generation must be carried out in a practical method while also providing sufficient dynamic information for an algorithm to capture. Therefore, all simulations of Eq. (7.20) were carried out using an integration step size of $h_c = 10^{-4}$ h and sampled every $\Delta = 0.01$ h (36 seconds), which is a reasonable sampling period for such a chemical process. The simulations were carried out for a duration of $t_f = 1$ h since most trajectories reached a steady-state within 1 h of simulation duration.

Due to the various ways that one may generate or obtain data for this system, three types of data generation were carried out, and each data set was then used to attempt to build SINDy models. Representative trajectories for each data set are shown in Fig. 7.10. The types of data generation and their advantages/disadvantages are summarized as follows:

1. Method: Open-loop step tests are carried out using numerous, random initial conditions and input signals until a steady-state is reached.

    - 1000 such trajectories were generated in this data set.
    - Initial conditions were randomly selected with the following restrictions on the initial states: $C_A \in [0.2, 3.7]$ $\mathrm{kmol/m^3}$ and $T \in [327, 477]$ K
    - Input signals were randomly generated with the following restrictions on the inputs: $C_{A0} \in [0.5, 7.5]$ $\mathrm{kmol/m^3}$ and $Q \in [-500, 500]$ $\mathrm{MJ/h}$
    - This is a standard method of data generation within chemical engineering in simulations-based applications as well as experimental practices. As an established method, data

generation via this method is easily conducted, a wide area of the state space can be covered by exciting the input signals as desired, and a large amount of dynamic information is present in the data set.

- Due to the operating region being the unstable steady-state, the trajectories, being in open-loop, will settle at the stable steady-states. However, this was not found to deteriorate the performance, likely because the dynamics of the reactor itself are independent of the region.
- As the states may achieve extreme values when the input signals are varied too widely (such as temperatures as high as 1000 K or as low as 1 K for certain excessively large/small values of Q), the best practice is to limit the range of input signals when using this method of data generation. This is particularly important when using finite-differences to estimate $\dot{X}$, which is the only estimation method available in a practical setting. It was found that when data generated indiscriminately including trajectories that settle at 1000 K/1 K were included in the training data set, i.e., all 1000 trajectories were used in training, SINDy had difficulties identifying the correct model if the derivatives were estimated with finite-differences. If the exact derivatives were provided (which would not be available in most chemical engineering applications), then SINDy was able to identify the model correctly. Upon further analysis of the derivatives at the regions of the fastest dynamics, it was found that the states changed very abruptly within the sampling period of $\Delta = 0.01$ h, causing numerical instabilities in the derivative estimation. Hence, providing the exact derivatives resolved the issue. As

expected, the issue was also resolved if the data was sampled ten times as frequently, i.e., with $\Delta = 0.001$ h. However, using all the 1000 trajectories is not necessary to capture the dynamics of this system, as described next.

- When the data set was truncated to only retain trajectories that never exceeded a temperature or 500 K or dropped below 300 K, i.e. $T \in [300, 500]$ K $\forall$ t, in order to only select trajectories close to the desired steady-state, 53 out of the initial 1000 trajectories were retained. However, SINDy was able to identify the best model with these 53 trajectories, producing a model with a maximum absolute error in the temperature of only 0.5 K. A few representative open-loop simulations for the first-principles model and this sparse-identified model are shown in Fig. 7.11, demonstrating close agreement throughout the region of state-space. Hence, it can be concluded that 53 trajectories contain sufficient dynamic information to build a highly accurate SINDy model, and there is no necessity to use all 1000 trajectories, which introduce faster and more complex dynamics in certain regions, which eventually require a finer sampling to be practically useful.

2. Method: Open-loop step tests are carried out with the system initiated from the desired steady-state and excited using various input signals only until the desired level set, $\Omega_{\hat{\rho}}$, or operating region is exited.

   - 1000 such trajectories were generated in this data set.

   - The initial condition was fixed to be the unstable steady-state, $(C_A, T) = (C_{A_s}, T_s) = (1.95\,\mathrm{kmol/m^3}, 402\,\mathrm{K})$

- Input signals were randomly generated with the following restrictions on the inputs: $C_{A0} \in [0.5, 7.5]\ \mathrm{kmol/m^3}$ and $Q \in [-500, 500]\ \mathrm{MJ/h}$
- As data is generated only within the operating region of interest, an advantage is that almost the entire region of the state-space that is of interest can be captured via a large number of simulations.
- Due to the unstable nature of the steady-state, one disadvantage is that a very large number of the 1000 trajectories in the data set are incomplete and too short to provide sufficient dynamic information, especially for SINDy, which generally performs better with longer trajectories rather than short bursts of trajectories. Out of the 1000 trajectories, only 14 trajectories are able to be simulated until $t_f = 1$ h. Since second-order finite-differences are used for the gradient approximation in our work as well as due to the internal mechanisms of the integrator used, at least 4 data points are required to be able to use a trajectory for model identification. Only 381 of the 1000 trajectories had at least 4 data points and could be used to build a SINDy model. However, the data set of 381 trajectories did not contain enough dynamic information, producing a SINDy model with a maximum temperature prediction error of 10.4 K. However, if the size of the data set was increased to 2000 trajectories, 807 trajectories with at least 4 data points remained, which then produced a highly accurate SINDy model with a maximum temperature prediction error of 0.6 K.
- State-space profiles for some open-loop simulations are shown in Fig. 7.12 for the first-principles model and the identified SINDy model, showing close agreement through-

out.

3. Method: Closed-loop simulations are carried out under a proportional-only controller with the state initialized from various initial conditions.

   - Two data sets were used to attempt to build a SINDy model using this method of data generation, one data set with 121 trajectories, spanning an 11×11 grid for x_0 in the state-space, while the second data set consisted of 961 trajectories, covering a 31×31 grid for x_0 in the state-space.
   - Initial conditions were selected within the grid, $C_A \times T = [0.2, 3.7]\ \mathrm{kmol/m^3} \times [327, 477]\ \mathrm{K}$ with each range uniformly spaced into 10 or 30 intervals with 11 or 31 points, respectively.
   - Input signals were calculated using the equation for a proportional controller, $Q = -1000(T - T_s)$, where 1000 represents the controller gain, and C_{A0} was fixed at its steady-state value of $C_{A0_s} = 4\,\mathrm{kmol/m^3}$.
   - A purported advantage of this method of data generation is that, due to the presence of the controller, the state can be driven to the desired unstable steady-state from any initial condition, providing dynamic information for trajectories from any point in the state-space up to the unstable steady-state.
   - The models obtained using this data set could very accurately predict the derivatives within the test set, i.e., the right-hand side of the model evaluates to the correct value of the derivative of the test set trajectories. However, all the simulations diverged from the steady-state after a short period at the initial stages of the simulation duration.

This phenomenon was also observed in [77] with the glycolytic oscillator model and attributed to the identification of wrong basis terms for some of the variables. In the models obtained for Eq. (7.20) using SINDy with the data set generated using closed-loop simulations, the heat input rate, Q, erroneously appeared with a relatively large coefficient in the first ODE representing $\dot{C}_A$, which may be the cause of the divergence. While further analyses may allow such data to be used for SINDy model identification, based on our current results, this method of data generation was not found to produce accurate SINDy models.

Based on the above analysis, the first method of data generation was found to be the optimal method of data generation when using SINDy. Since the optimal results were obtained with limited trajectories that were able to be integrated to t_f and also stayed relatively close to the steady-state of interest, the best method of data generation for this system, based on the above analysis, seems to be conduct a modest number of step tests near the desired region. However, the second method can also be used if a much larger data set is used and caution is taken to only use trajectories with at least 4 data points when using a second-order finite-difference method for estimating the time-derivative of the states. The use of closed-loop data to identify SINDy models was not found to yield satisfactory results, and further analyses should be carried out in the future to assess the viability of such data for SINDy modeling of chemical processes.

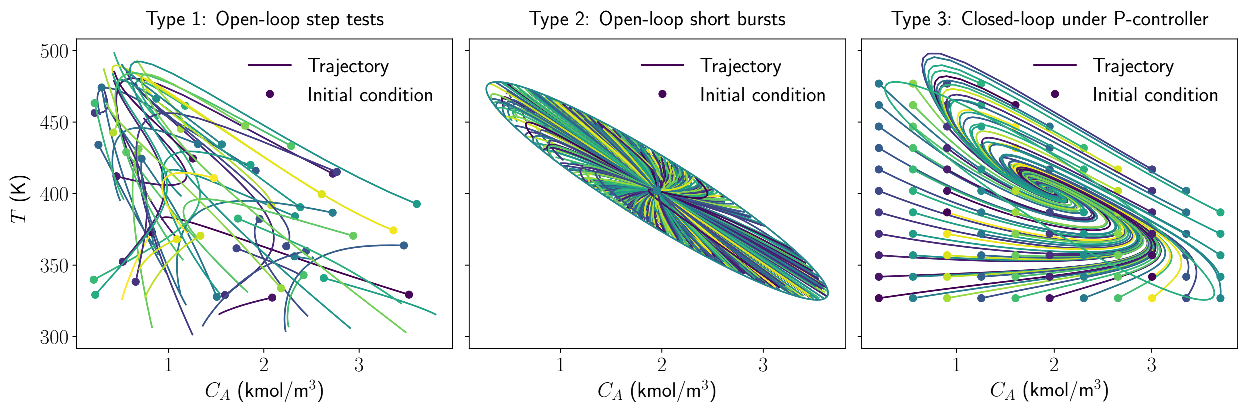


Figure 7.10: Three types of data generation for the nonisothermal CSTR operating at an unstable steady-state.

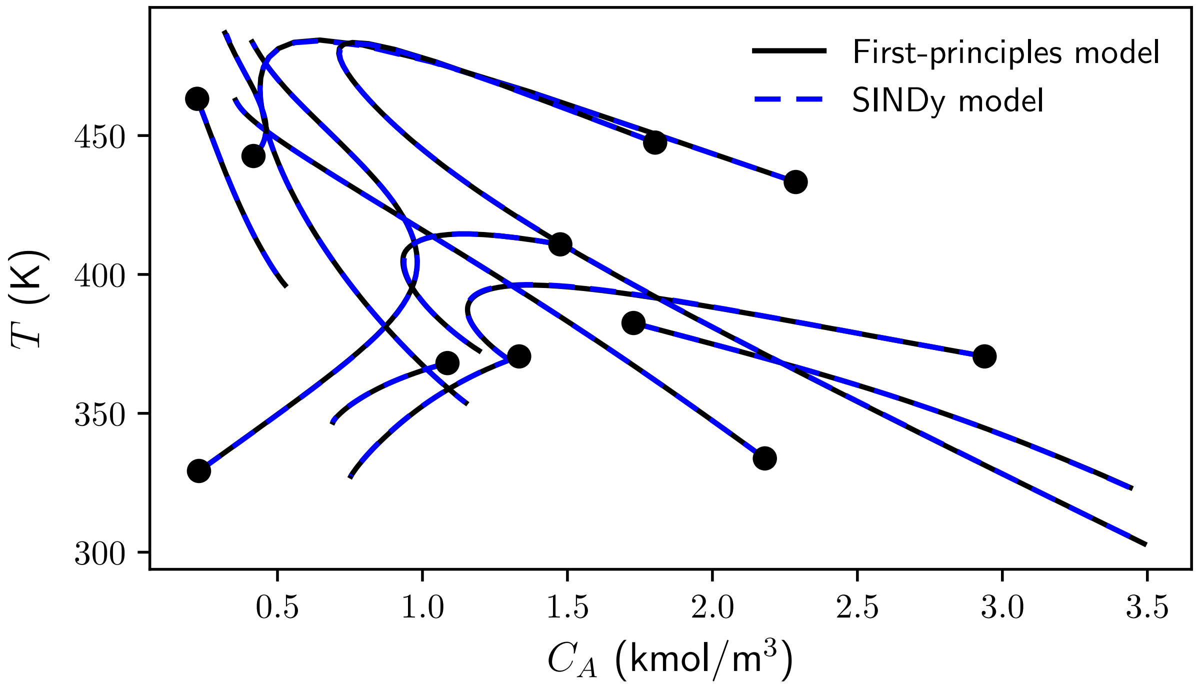


Figure 7.11: State-space profiles for open-loop simulation using the first-principles model of Eq. (7.20) and the SINDy model obtained using type 1 data generation, respectively, for various sets of inputs and initial conditions (marked as solid dots) x_0 in the vicinity of the desired operating point.

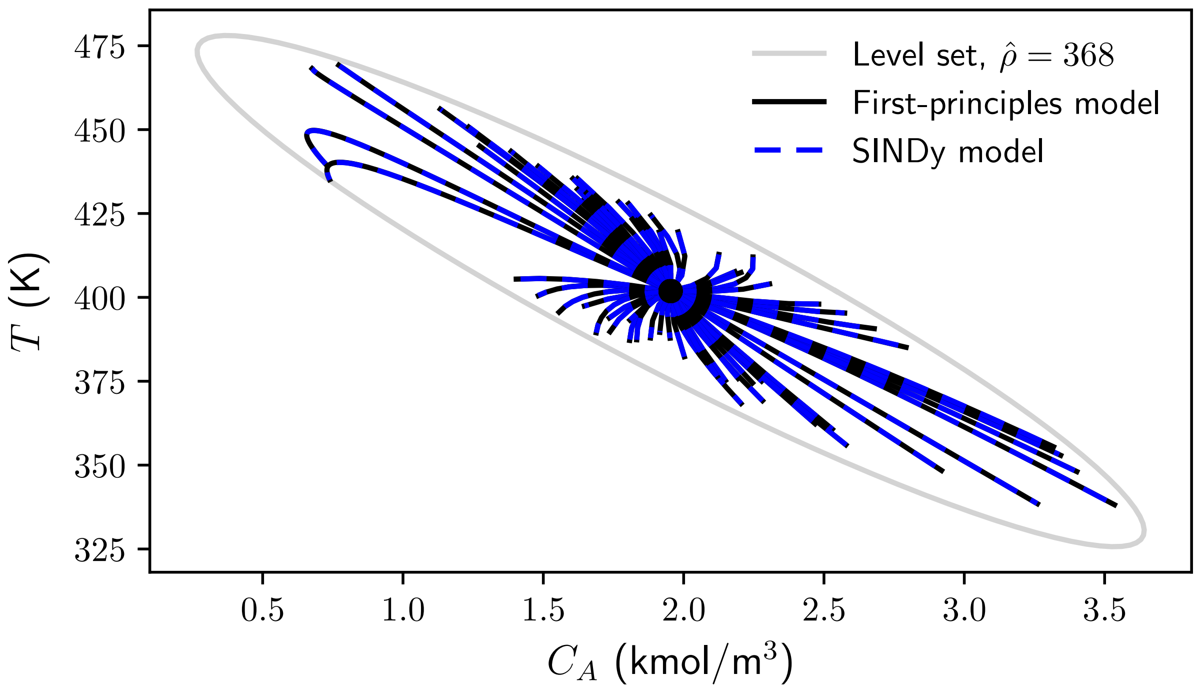


Figure 7.12: State-space profiles for open-loop simulation using the first-principles model of Eq. (7.20) and the SINDy model obtained using type 2 data generation, respectively, for various sets of inputs, starting from the steady-state.

7.7.2 Candidate library of basis functions

Since its inception, multiple studies have reported the central role of the candidate library, Θ, in the SINDy algorithm [77, 114]. In [77], for example, a standard benchmark problem for system identification, the glycolytic oscillator model, could only be partially identified, i.e., the dynamics of only four out of the seven states could be correctly identified. The reason was attributed to the presence of rational functions in the right-hand sides of the ODEs corresponding to the remaining three states, which were not considered in the polynomial basis set used. Hence, choosing the correct basis functions is critical to the success of SINDy. For the remainder of the section, the data set used to study the effect of the candidate library is the data set generated using the first type of data generation described in Section 7.7.1 (53 trajectories from open-loop step tests).

In the absence of any *a priori* knowledge, the nonisothermal CSTR of Eq. (7.20) is a particularly challenging system to obtain the correct basis for and, hence, model. This is primarily due to the fact that, by design, SINDy can only regress the pre-multiplying coefficients for each basis function, which appear linearly in the right-hand side of the ODE. The basis functions themselves must be selected and the Θ calculated before carrying out the regression step for identifying the pre-multiplying coefficients by solving Eq. (7.8). Since the activation energy is generally unknown, the exponential term must be carefully selected. For the set of parameters chosen, from Eq. (7.21), it can be observed that the numerator of the argument of the exponential term is -6013.95236949723. Due to the extreme dissimilarity between $\mathrm{e}^{-\frac{1}{T}}$ and $\mathrm{e}^{-\frac{6013.95236949723}{T}}$, using the former exponential term as a basis function will not yield an accurate SINDy model. The dynamics of the $\mathrm{e}^{-\frac{6013.95236949723}{T}}$ term cannot be captured by any linear multiple of $\mathrm{e}^{-\frac{1}{T}}$. However, choosing large numbers over a wide range is also inadvisable since only a narrow range of the exponent can yield an accurate model with a maximum absolute error in the temperature below 1 K as shown in Fig. 7.13. While it may be possible to tune the exponent in this particular case by conducting a fine search for the exponent over a wide range with shortly spaced intervals of approximately 10 units, this is generally intractable when the exponent is even larger in magnitude (increasing the required search region) or if there are multiple reactions, in which case tuning each exponent term using a multidimensional grid search at such a high resolution becomes prohibitively expensive. Therefore, two approaches are proposed to overcome this challenge, both of which are shown to yield accurate SINDy models.

Remark 7.2. *This challenge has been overcome in some past studies by assuming that the activation energy is known* a priori*, and the exact term,* $\mathrm{e}^{-\frac{6013.95236949723}{T}}$*, is included in the candidate library, largely simplifying the modeling problem [e.g., 218, 223]. In other studies using SINDy to model reaction networks, the objective was to identify the reactions rather than investigate any temperature dependence [84]. Hence, the specific challenge of obtaining an appropriate basis for SINDy to model nonisothermal reactors is considered here.*

7.7.2.1 Non-dimensionalization of the temperature

The first approach we consider is non-dimensionalizing the temperature by scaling it by a reference temperature, T_{ref}. We consider, for simplicity and without loss of generality, $T_{\text{ref}} = T_s$, and define the new dimensionless temperature as $\bar{T} = T/T_s$. Hence, the ODE system of Eq. (7.20),

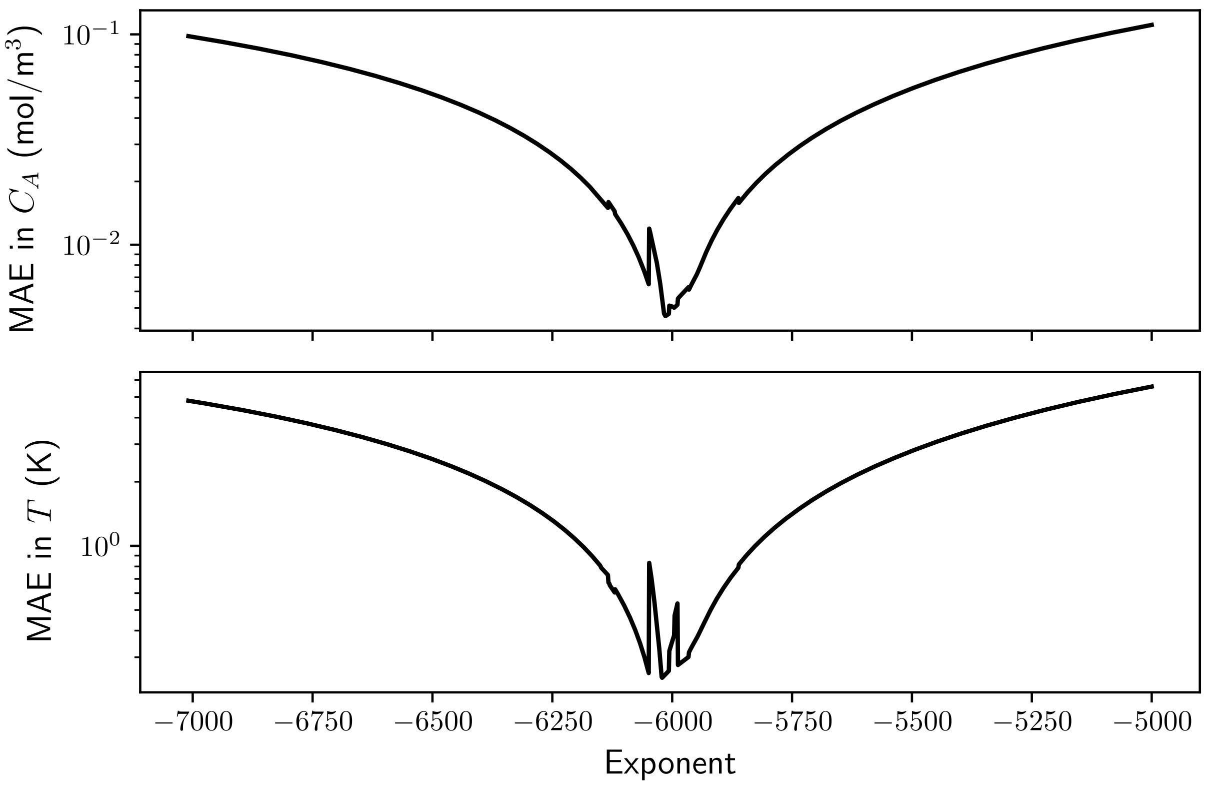


Figure 7.13: Validation error as a function of the numerator of the argument of the exponential function in the candidate library for the original data set, (C_A, T).

after the appropriate substitutions, takes the form,

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = \frac{F}{V}(C_{A0} - C_A) - k_0 \mathrm{e}^{\frac{\gamma}{T}} C_A^2 \tag{7.22a}$$

$$\frac{\mathrm{d}\bar{T}}{\mathrm{d}t} = \frac{F}{V}\left(\frac{T_0}{T_s} - \bar{T}\right) + \frac{-\Delta H}{\rho_L C_p T_s} k_0 \mathrm{e}^{\frac{\gamma}{T}} C_A^2 + \frac{10^3 Q}{\rho_L C_p V T_s} \tag{7.22b}$$

where $\gamma = -E/RT_s$. For the set of process parameters and reference temperature chosen, $\gamma = -14.96$. Due to the much smaller value of γ and the lower sensitivity of γ, it is possible to conduct a fine search for a value of γ that produces an accurate SINDy model. The maximum absolute errors in the variables for the validation set for $\gamma \in [-20, 0]$ are shown in Fig. 7.14. A value of $\gamma = -15$ yields the highly accurate model,

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = 5.051 C_{A0} - 5.058 C_A - 8.8 \times 10^6 \mathrm{e}^{-\frac{15}{T}} C_A^2 \tag{7.23a}$$

$$\frac{\mathrm{d}\bar{T}}{\mathrm{d}t} = 3.768 - 5.046\bar{T} + 1.09 \times 10^6 \mathrm{e}^{-\frac{15}{T}} C_A^2 + 0.011 Q \tag{7.23b}$$

where every coefficient is within 5% of the true values. There are two further advantages of non-dimensionalization in this case. Firstly, when multiple reactions are present, in many practical cases, since the reference temperature is similar to the specific process temperatures, all the γ will often be approximately of the same order of magnitude or within an order of magnitude difference [e.g., 227]. Therefore, a "mean" or representative value of the γ values of all the reactions will produce an accurate SINDy model, owing to the greatly reduced sensitivity of the basis functions to γ. Secondly, even if the (nearly) exact value of $\gamma = -15$ is not found using the search methodology described, simply using every integer value of $\gamma \in [-20, -10]$ to create 11 basis functions also yields an accurate (but dense) SINDy model with a maximum absolute error in the temperature of

0.4 K. In contrast, in the original variables, if multiple basis functions, for example, the set $\gamma \in \{-7000, -6000, -5000, -4000, -3000, -2000, -1000\}$ is chosen to be in the candidate library, the results are poor due to the large dissimilarities between successive basis functions as noted previously. We reiterate that the goal of using SINDy in this work is not to capture the underlying ODE but to use SINDy as a system identification method. Although the original system was nearly reproduced when $\gamma = -15$ was correctly identified, this need not be the case to apply SINDy, especially when the models will subsequently be used for model-based feedback control. Hence, the "bruteforce" approach of using all 11 basis functions with $\gamma \in [-20, -10]$ is considered a satisfactory model as well. This latter approach may also handle multiple reactions more easily since it is likely that the correct value of γ for each reaction is captured in the candidate library.

Remark 7.3. *To apply non-dimensionalization to the system when applying SINDy, the only change that must be made is that the temperature data must be scaled by T_s before providing the data set to the SINDy algorithm. Since finite-differences are used to estimate the time-derivative, $\dot{X}$, the derivative estimates will scale accordingly once the data set itself is scaled.*

7.7.2.2 Higher-order Taylor series approximation

A possibly more general approach that can handle any value of the activation energy or any number of reactions is to express the exponential term using its Taylor series expansion such that the activation energy appears as a pre-multiplier, which can then be regressed using SINDy. As SINDy is a nonlinear method, any order of the Taylor series can be retained. If multiple reactions are present, the pre-multiplier should account for all the reactions since the temperature variable is independent of the activation energy, and all the approximated terms can be summed to yield one

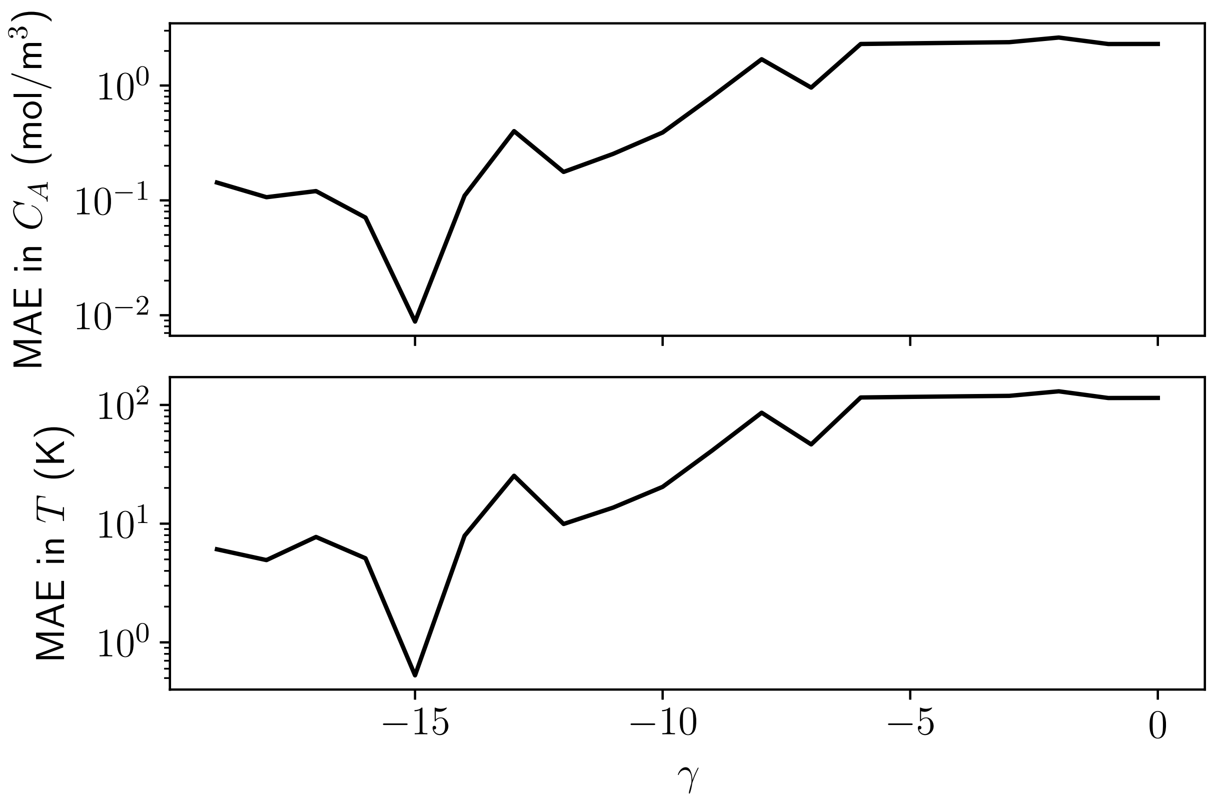


Figure 7.14: Validation error as a function of γ for the data set with dimensionless temperature, $(C_A, \bar{T})$.

final pre-multiplying coefficient value for each term of the Taylor expansion.

Due to the length of the models involving Taylor expansion, only error metrics and discussions are provided for this method. When the candidate library includes up to 5th-order terms of the Taylor expansion, i.e., $(T - 402)^5$, an accurate SINDy model with a maximum absolute error in the temperature of 0.4 K is obtained, with open-loop test results nearly identical to Fig. 7.11. When the sparse-identified model is compared to the original ODE of Eq. (7.21) with parameters substituted in and the exponential term replaced by 5th-order Taylor series, it is found that the SINDy model neglects terms above third-order, which are of the order of 10^{-8} and 10^{-6} for C_A and T, respectively. As for terms up to third-order, the SINDy model correctly identifies all terms for C_A and identifies the terms in T correctly as well, but also identifies a few erroneous terms such as linear C_A and C_{A0} terms. However, the contribution of the extra terms are extremely minor and do not affect the accuracy, as seen in the extremely low maximum absolute error.

Remark 7.4. *While this method may be reminiscent of linearization of a nonlinear ODE, there are two key differences. Firstly, a nonlinear higher-order Taylor expansion is used to approximate the exponential function rather than a linear approximation. This greatly affects the region of accurate model predictions compared to a linearized model. When the open-loop tests shown in Fig. 7.12 were repeated with the linear state-space model obtained for this system in [62] using N4SID, all trajectories were found to diverge, while Fig. 7.12 demonstrates the high accuracy of the nonlinear SINDy model. Secondly, only the exponential term in the Arrhenius relationship is approximated using the Taylor expansion, but the remaining terms in the ODE model and candidate library remain in their original, nonlinear forms. Hence, all other nonlinear terms can still be* *identified exactly without any approximation, while model linearization includes linearizing even such polynomial and trigonometric terms.*

7.7.3 Summary of data generation and candidate library guidelines and final steps to build the SINDy model

Based on extensive results from using the various types of data generation and basis functions considered, the following points can be summarized:

1. Open-loop step tests were found to be the optimal method of data generation for obtaining a SINDy model for the system studied, although short bursts within the desired stability region can yield a good model if a sufficient number of trajectories with at least 4 data points are obtained.

2. Data from closed-loop simulations did not yield an accurate model for the system studied.

3. A sampling period of $\Delta = 0.01$ h or 36 s is sufficient for obtaining an accurate SINDy model as long as enough dynamic information is captured via open-loop tests with a large number of input signals.

4. Due to the sensitivity of the argument of the exponential term, γ, the exponential basis term should be selected carefully.

    - Specifically, if *a priori* knowledge of the reaction (such as an estimate of the activation energy) is available, the system may be modeled directly without any modifications as long as the correct values of the activation energy are used to build the candidate library.

- In the event that the no *a priori* knowledge is available, the system should be either non-dimensionalized with respect to temperature or a higher-order Taylor series used to approximate the exponential terms.

5. Non-dimensionalization of the temperature has potential to reproduce the exact system.

6. Using Taylor series approximations of the exponential term can yield highly accurate SINDy models, but their performance is expected to deteriorate when sufficiently far from the point of expansion. However, since a nonlinear, higher-order approximation is used, the region where the model performs accurately will be significantly larger than any model obtained from a linearization of the original system, and likely large enough for any practical application.

Once the data set and method of handling the exponential term are finalized based on the aforementioned guidelines, the SINDy model is obtained by using the PySINDy package in Python [209, 210]. Specifically, the data set is loaded into Python and split into an 80%/10%/10% training/validation/test set. The time-derivative of the states, $\dot{X}$, is estimated using second-order central finite differences (except the first and last points, which use second-order forward and backward finite differences, respectively). The optimizer is chosen to be the sequential thresholded least squares described in [77], with λ tuned via a coarse search to a value of 5.0, although similar results were obtained for the SR3 optimizer as well. The candidate library, for both the non-dimensionalization and Taylor series approaches, was chosen to include up to second-order polynomial terms for the concentration C_A, the bias term, and linear input terms. The remaining terms for the non-dimensionalization method included a linear temperature term, the exponential term with $\gamma = -15$, and interaction terms between the polynomial C_A terms and the exponential term. Specifically, the candidate library for the non-dimensionalization approach takes the following form:

$$\Theta(C_A, \bar{T}, C_{A0}, Q) = [1 \quad C_A \quad C_A^2 \quad \bar{T} \quad C_{A0} \quad Q \quad \mathrm{e}^{-\frac{15}{T}} \quad C_A\mathrm{e}^{-\frac{15}{T}} \quad C_A^2\mathrm{e}^{-\frac{15}{T}}] \tag{7.24}$$

For the Taylor series approach, the only change is that the exponential term is replaced with $(T - 402)$, $(T - 402)^2, \ldots, (T - 402)^5$. Hence, the last three functions and the $\bar{T}$ function in Eq. (7.24) are replaced by 15 terms (five exponential approximation terms and ten interaction terms with C_A), producing a library of 20 functions. Once all of the above selections are made, the SINDy model can be obtained by calling the model fitting method in PySINDy. The SINDy model of Eq. (7.23), for example, is obtained by using the first type of data generation (53 open-loop step tests) and the candidate library of Eq. (7.24).

7.8 Future directions

7.8.1 Neural network basis functions

For highly complex systems, it may be possible that the initially chosen nonlinear basis functions do not produce adequate results, but no prior knowledge is available to intelligently expand the function library. Moreover, adding random, additional nonlinear candidate functions may fail to improve the SINDy model performance if the functions added are completely dissimilar to the relevant functions that are required to model the system. An example is the challenge of the exponential basis term encountered and discussed in Section 7.7.2 of the nonisothermal CSTR example. In such cases, one option is to add more powerful and general function approximators such as feedforward neural networks, which are well-known for their universal approximation property, which dictates that they can approximate any static nonlinear function if they are designed with enough neurons and at least one sigmoidal hidden layer [60, 61]. Such hybrid models consisting of partly first-principles/ODE models and partly data-based black-box models are increasingly being used [238–243]. Specifically, hybrid models involving ODE models and FNNs have been successfully applied to state estimation problems in the recent work of [235]. Therefore, a similar approach may be proposed for SINDy, where the right-hand side of the SINDy model of Eq. (7.2) may be modified to

$$\dot{\hat{x}}(t) = \hat{f}(\hat{x}) + \hat{g}(\hat{x})u + \text{FNN}(x, u) \tag{7.25}$$

where FNN denotes a feedforward neural network model that can capture any nonlinearities not modeled by the function library. One advantage of such a model, as opposed to a purely FNN model for the right-hand side of Eq. (7.25), may include reduced computational time due to the requirement of simpler models with fewer parameters, since only a fraction of the model must be captured by an FNN. Moreover, neural network training generally requires large volumes of data with wide variation and coverage of the operating region, which may be difficult to obtain in an experimental or plant setting. In contrast, when only a fraction of the overall model requires an FNN to be modeled, the data acquisition may be eased as well.

Once a model of the form of Eq. (7.25) is identified, if possible, converting the FNN part of the SINDy model back to symbolic functions will greatly improve the model inference time as explicit nonlinearities are computationally desirable. Such advances have already been initiated in recent papers on modeling biological systems [244].

7.8.2 Real-time model updates

In the presence of disturbances or changes in process behavior due to, for example, catalyst deactivation or feed stream disruptions, the process model in a model-based controller such as MPC must be updated in real-time to reflect the changes. Much of the research in model re-identification is concentrated on the mathematical details of the algorithms used for the model update, such as recursive least-squares or recursive singular value decomposition [245–247] rather than developing a rigorous framework for the triggering of the model re-identification procedure. Research on the triggering procedure include error-triggered as well as event-triggered model re-identification [67, 227, 248], but mostly use first-principles process models. In the context of SINDy, [117] proposed a model re-identification procedure, where the SINDy model coefficients could be updated or terms could be added or deleted as required. The trigger for re-identification was a significant divergence between the local Lyapunov exponent and the prediction horizon estimate (although the definition of "prediction horizon" in [117] differs from its usage in this manuscript). However, the results of [117] were only in the context of modeling. Hence, a future direction for research in sparse identification would be to consider real-time updates to a data-based SINDy model based on the error- or event-triggering mechanism of [67].

7.9 Conclusions

In this paper, we have provided an overview of several recent advancements in the sparse identification for nonlinear dynamics (SINDy) method to overcome the challenges of modeling and controlling two-time-scale systems and noisy data. The methods considered included combining SINDy with nonlinear principal component analysis, feedforward neural networks, subsampling, co-teaching, and ensemble learning. The novel methods were described in detail, and best practices, tuning guidelines, as well as common pitfalls to avoid, for their successful application in process systems engineering were provided for control practitioners. To demonstrate their effectiveness, results from applying the proposed algorithms to chemical processes were provided. Finally, a number of future research directions were outlined.

Chapter 8

Conclusion

This dissertation focused on the application of sparse identification for nonlinear dynamics (SINDy) for modeling nonlinear chemical processes and subsequently embedding the dynamic models developed into model-based control techniques such as model predictive control (MPC). Challenges specifically encountered in chemical engineering such as time-scale multiplicities, noisy sensor data, lack of first-principle models, and process disturbances when using SINDy were addressed. First, a reduced-order modeling framework to build well-conditioned ordinary differential equation (ODE) models for two-time-scale systems using SINDy was proposed and incorporated into Lyapunov-based MPC (LMPC), followed by a closed-loop stability analysis of the system under the proposed SINDy-based LMPC. Next, to handle sensor noise that corrupts most measurement data in process systems and can cause numerical instabilities when differentiated naively without accounting for the noise, several methods from the recent literature are discussed, investigated, and proposed along with a novel ensemble-based SINDy algorithm. Thirdly. to account for real-time process shifts due to disturbances such as catalyst deactivation or equipment fouling, which may be reflected via changes in the model terms or structure, an error detector was designed and used in conjunction with economic MPC to update the initially identified SINDy models online, based on limited new data following the introduction of the process disturbance. Finally, the various advancements to the SINDy algorithm developed over the course of the dissertation are summarized along with a chemical process example providing detailed insights and guidance for process modeling practitioners, based on the challenges, successes, and limitations encountered throughout the dissertation.

In **Chapter 2**, reduced-order models were developed for nonlinear two-time-scale systems using measurement data. Nonlinear principal component analysis, a technique to extract nonlinear relationships between variables, was combined with sparse identification, an algorithm to reconstruct the underlying ODEs governing the system, to build a nonlinear model that can be used to predict the full state of the original system. The effectiveness of this method was demonstrated using two reactor-based examples and comparing the results with the original system and a purely sparse identified system. In both examples, the results were in close agreement with the original system and also the purely sparse identified system. It was observed, however, that the accuracy of the sparse identification method for the full state was strongly linked to the sampling period at which the original data was sampled and also, in some cases, required a very specific degree of sparsity in the algorithm. Furthermore, the sparse identified stiff ODEs required a very short integration time step of 10^{-6}, while the NN model in NLPCA-SI could predict the fast state (after a short transient) to at least a similar degree of accuracy without any integration. In the future, we will focus on the designs of controllers for nonlinear two-time-scale systems using data-based reduced-order models. The predicted states using NLPCA-SI can be used as the process model in model-based control methods such as MPC.

Chapter 3 focused on the design of a Lyapunov-based MPC for a class of nonlinear singularly perturbed systems using only measurement data from processes. In singularly perturbed systems, due to the presence of time-scale multiplicities, a direct application of MPC without accounting for the evolution of the states in different time scales can lead to closed-loop performance deterioration or even closed-loop instability due to controller ill-conditioning. Hence, we proposed a method to first separate the slow and fast variables in the system and then design the MPC based on the reduced-order slow subsystem. Furthermore, due to the lack of a first-principles model in most practical applications, our method used only sampled experimental/industrial simulation data to reconstruct the reduced slow subsystem via a machine-learning method known as sparse identification. Subsequently, the theory was developed by deriving sufficient conditions for closed-loop stability under sample-and-hold implementation. Finally, the proposed LMPC design was applied to a non-isothermal reactor that exhibited time-scale separation. It was observed that the controllers yielded nearly identical performance for the same controller parameters. However, the LMPC based on the sparse-identified slow subsystem could implement superior controller parameters, such as a longer prediction horizon, due to its reduced complexity and, hence, lower computational time. As a result, the SI based LMPC outperformed the LMPC utilizing the first-principles model when the superior parameters were used for the former controller, demonstrating the practicality and benefits of designing MPC by reconstructing the reduced slow subsystem from measurement data.

In **Chapter 4**, a novel algorithm was devised to build dynamical models that capture nonlinear process dynamics given only highly noisy sensor data. The noise was assumed to follow a white

Gaussian distribution with different variances. A predator-prey model and a chemical process were used to demonstrate the performance and applicability of the new algorithm. It was shown that the basic sparse identification algorithm was inadequate in identifying the model in the presence of high noise in the data, particularly above a variance of 0.01 for normalized data. However, when the subsampling technique was introduced, without co-teaching, by randomly subsampling to leave out the more noisy data in some iterations, it could identify the dynamics satisfactorily up to a noise variance of 0.04. Finally, the proposed algorithm combining subsampling with co-teaching, where the original data is subsampled but also mixed with some noise-free data from first-principles model simulations was used. Using the third algorithm, the performance improved slightly in the presence of noise with variance up to 0.04. However, at the highest noise level studied, which was characterized by a variance of 0.09, both the base case and the subsampling without co-teaching failed and could not identify the models using the extremely noisy data. The subsampling with co-teaching could accurately identify the models in this case, even when only 20% of the subsamples consisted of noise-free data generated from first-principles model simulations. The performance was evaluated based on plots of the outputs as well as the mean squared error (MSE) on the testing data sets. The results were qualitatively similar in both systems investigated, with more accurate models predicting the testing data set more accurately and yielding lower MSE values.

In **Chapter 5**, 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 as compared to the open-loop performance and a P-controller in terms of faster convergence.

Chapter 6 introduces a novel approach for on-line updates of nonlinear ODE models obtained using sparse identification to embed into a model predictive controller (MPC) for nonlinear process systems. The proposed methodology incorporates an error-triggering mechanism through a moving horizon error detector, which evaluates the relative prediction error within a specified horizon. When the prediction error surpasses a predefined threshold, the error-triggering mechanism is activated and the most recent yet sufficient input/output data is used to update specific coefficients of the SINDy model using an efficient algorithm. The results showcase the capability of the proposed approach to improve state predictions crucial for MPC in the presence of plant variations. A chemical process example under the framework of Lyapunov-based empirical model predictive control is employed to illustrate the effectiveness and implementation of real-time updates for the SINDy models. It was demonstrated that a small amount of real-time data could accurately update the SINDy model to adjust to the disturbances, greatly reducing thereby the model prediction error when monitoring the process via the moving horizon error detector. A slight improvement of the economic benefits was also observed when the SINDy model in the LEMPC was updated in real-time, compared to the SINDy-based LEMPC without model updates, although the improvement was limited by the catalyst deactivation and other compounding factors.

In **Chapter 7**, we have provided an overview of several recent advancements in the sparse identification for nonlinear dynamics (SINDy) method to overcome the challenges of modeling and controlling two-time-scale systems and noisy data. The methods considered included combining SINDy with nonlinear principal component analysis, feedforward neural networks, subsampling, co-teaching, and ensemble learning. The novel methods were described in detail, and best practices, tuning guidelines, as well as common pitfalls to avoid, for their successful application in process systems engineering were provided for control practitioners. To demonstrate their effectiveness, results from applying the proposed algorithms to chemical processes were provided. Finally, a number of future research directions were outlined.

Bibliography

[1] S. S. Ge and C. Wang. Adaptive neural control of uncertain MIMO nonlinear systems. *IEEE Transactions on Neural Networks*, 15:674–692, 2004.

[2] H. W. Ge, Y. C. Liang, and M. Marchese. A modified particle swarm optimization-based dynamic recurrent neural network for identifying and controlling nonlinear systems. *Computers & Structures*, 85:1611–1622, 2007.

[3] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine learning-based predictive control of nonlinear processes. Part I: Theory. *AIChE Journal*, 65:e16729, 2019.

[4] J. Luo, B. Çıtmacı, J. B. Jang, F. Abdullah, C. G. Morales-Guio, and P. D. Christofides. Machine learning-based predictive control using on-line model linearization: Application to an experimental electrochemical reactor. *Chemical Engineering Research and Design*, 197:721–737, 2023.

[5] A. Cozad, N. V. Sahinidis, and D. C. Miller. A combined first-principles and data-driven approach to model building. *Computers & Chemical Engineering*, 73:116–127, 2015.

[6] Z. T. Wilson and N. V. Sahinidis. The ALAMO approach to machine learning. *Computers & Chemical Engineering*, 106:785–795, 2017.

[7] S. Zhang and G. Lin. Robust data-driven discovery of governing physical laws with error bars. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474:20180305, 2018.

[8] P. Whittle. *Hypothesis testing in time series analysis*. PhD thesis, Uppsala University, 1951.

[9] V. Volterra. *Theory of functionals and of integral and integro-differential equations*. Blackie and Sons, Ltd., 1930.

[10] A. Patrikar and J. Provence. Nonlinear system identification and adaptive control using polynomial networks. *Mathematical and computer modelling*, 23:159–173, 1996.

[11] F. J. Doyle, R. K. Pearson, and B. A. Ogunnaike. *Identification and control using Volterra models*. Springer, 2002.

[12] S. A. Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[13] S. J. Norquay, A. Palazoglu, and J. Romagnoli. Model predictive control based on Wiener models. *Chemical Engineering Science*, 53:75–84, 1998.

[14] K. Fruzzetti, A. Palazoğlu, and K. McDonald. Nolinear model predictive control using Hammerstein models. *Journal of Process Control*, 7:31–41, 1997.

[15] R. S. Patwardhan, S. Lakshminarayanan, and S. L. Shah. Constrained nonlinear MPC using Hammerstein and Wiener models: PLS framework. *AIChE Journal*, 44:1611–1622, 1998.

[16] H. Bloemen, T. Van Den Boom, and H. Verbruggen. Model-based predictive control for Hammerstein? Wiener systems. *International Journal of Control*, 74:482–495, 2001.

[17] G. R. Sriniwas and Y. Arkun. A global solution to the nonlinear model predictive control algorithms using polynomial ARX models. *Computers & Chemical Engineering*, 21:431–439, 1997.

[18] S. Aumi and P. Mhaskar. Integrating data-based modeling and nonlinear control tools for batch process control. *AIChE journal*, 58:2105–2119, 2012.

[19] M. ławryńczuk. Computationally efficient nonlinear predictive control based on neural Wiener models. *Neurocomputing*, 74:401–417, 2010.

[20] M. Ławryńczuk. Practical nonlinear predictive control algorithms for neural Wiener models. *Journal of Process Control*, 23:696–714, 2013.

[21] S. Narasimhan, N. H. El-Farra, and M. J. Ellis. A reachable set-based scheme for the detection of false data injection cyberattacks on dynamic processes. *Digital Chemical Engineering*, 7:100100, 2023.

[22] C. T. Chou and M. Verhaegen. Subspace algorithms for the identification of multivariable dynamic errors-in-variables models. *Automatica*, 33:1857–1869, 1997.

[23] S. J. Qin. An overview of subspace identification. *Computers & Chemical Engineering*, 30:1502–1513, 2006.

[24] B. Huang and R. Kadali. *Dynamic modeling, predictive control and performance monitoring: a data-driven subspace approach*. Springer, 2008.

[25] B. L. Pence, H. K. Fathy, and J. L. Stein. Recursive maximum likelihood parameter estimation for state space systems using polynomial chaos theory. *Automatica*, 47:2420–2424, 2011.

[26] T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. *Automatica*, 47:39–49, 2011.

[27] P. Van Overschee and B. De Moor. *Subspace identification for linear systems: Theory—Implementation—Applications*. Springer Science & Business Media, 2012.

[28] L. Vandenberghe. Convex optimization techniques in system identification. *IFAC Proceedings Volumes*, 45:71–76, 2012.

[29] A. Hansson, Z. Liu, and L. Vandenberghe. Subspace system identification via weighted nuclear norm optimization. In *Proceedings of the 51st IEEE Conference on Decision and Control (CDC)*, pages 3439–3444. IEEE, 2012.

[30] P. Van Overschee and B. De Moor. N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30:75–93, 1994.

[31] W. Favoreel, B. De Moor, and P. Van Overschee. Subspace state space system identification for industrial processes. *Journal of Process Control*, 10:149–155, 2000.

[32] M. Verhaegen and E. Deprettere. A fast, recursive MIMO state space model identification algorithm. In *Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1349–1354. IEEE, 1991.

[33] M. Verahegen and P. Dewilde. Subspace model identification Part 1. The output-error state-space model identification class of algorithms. *International Journal of Control*, 56:1187–1210, 1992.

[34] M. Viberg. Subspace-based methods for the identification of linear time-invariant systems. *Automatica*, 31:1835–1851, 1995.

[35] W. E. Larimore. Canonical variate analysis in identification, filtering, and adaptive control. In *Proceedings of the 29th IEEE Conference on Decision and Control*, pages 596–604. IEEE, 1990.

[36] N. A. Mardi and L. Wang. Subspace-based model predictive control of time-varying systems. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 4005–4010. IEEE, 2009.

[37] J. D. Farmer and J. J. Sidorowich. Predicting chaotic time series. *Physical Review Letters*, 59:845, 1987.

[38] T. D. Sauer. Reconstruction of shared nonlinear dynamics in a network. *Physical Review Letters*, 93:198701, 2004.

[39] S. V. Lapin. Identification of time-varying nonlinear systems using Chebyshev polynomials. *Journal of Computational and Applied Mathematics*, 49:121–126, 1993.

[40] G. P. Liu, V. Kadirkamanathan, and S. A. Billings. On-line identification of nonlinear systems using Volterra polynomial basis function neural networks. *Neural Networks*, 11:1645–1657, 1998.

[41] S. Mahmoodi, J. Poshtan, M. R. Jahed-Motlagh, and A. Montazeri. Nonlinear model predictive control of a pH neutralization process based on Wiener–Laguerre model. *Chemical Engineering Journal*, 146:328–337, 2009.

[42] A. Van Mulders, J. Schoukens, M. Volckaert, and M. Diehl. Two Nonlinear Optimization Methods for Black Box Identification Compared. *IFAC Proceedings Volumes*, 42:1086–1091, 2009.

[43] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, and R. Pintelon. Identification of nonlinear systems using Polynomial Nonlinear State Space models. *Automatica*, 46:647–656, 2010.

[44] W. Widanage, J. Stoev, A. Van Mulders, J. Schoukens, and G. Pinte. Nonlinear system-identification of the filling phase of a wet-clutch system. *Control Engineering Practice*, 19:1506–1516, 2011.

[45] A. Marconato, J. Sjöberg, and J. Schoukens. Initialization of nonlinear state-space models applied to the Wiener–Hammerstein benchmark. *Control Engineering Practice*, 20:1126–1132, 2012.

[46] J. Paduart, L. Lauwers, R. Pintelon, and J. Schoukens. Identification of a Wiener–Hammerstein system using the polynomial nonlinear state space approach. *Control Engineering Practice*, 20:1133–1139, 2012.

[47] A. Dutta, Y. Zhong, B. Depraetere, K. Van Vaerenbergh, C. Ionescu, B. Wyns, G. Pinte, A. Nowe, J. Swevers, and R. De Keyser. Model-based and model-free learning strategies for wet clutch control. *Mechatronics*, 24:1008–1020, 2014.

[48] A. Alanqar, H. Durand, and P. D. Christofides. On identification of well-conditioned nonlinear systems: Application to economic model predictive control of nonlinear processes. *AIChE Journal*, 61:3353–3373, 2015.

[49] S. Yin and O. Kaynak. Big data for modern industry: challenges and trends [point of view]. *Proceedings of the IEEE*, 103:143–146, 2015.

[50] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[51] F. Chollet et al. Keras. https://keras.io, 2015.

[52] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[53] J. M. Ali, M. A. Hussain, M. O. Tade, and J. Zhang. Artificial Intelligence techniques

applied as estimator in chemical process systems–A literature survey. *Expert Systems with Applications*, 42:5915–5931, 2015.

[54] Z. Wu, A. Tran, Y. M. Ren, C. S. Barnes, S. Chen, and P. D. Christofides. Model Predictive Control of Phthalic Anhydride Synthesis in a Fixed-Bed Catalytic Reactor via Machine Learning Modeling. *Chemical Engineering Research and Design*, 145:173–183, 2019.

[55] Z. Wu and P. D. Christofides. Economic Machine-Learning-Based Predictive Control of Nonlinear Systems. *Mathematics*, 7:494, 2019.

[56] J. B. Rawlings and C. T. Maravelias. Bringing new technologies and approaches to the operation and control of chemical process systems. *AIChE Journal*, 65, 2019.

[57] V. Venkatasubramanian. The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE Journal*, 65:466–478, 2019.

[58] A. Alnajdi, A. Suryavanshi, M. S. Alhajeri, F. Abdullah, and P. D. Christofides. Machine learning-based predictive control of nonlinear time-delay systems: Closed-loop stability and input delay compensation. *Digital Chemical Engineering*, 7:100084, 2023.

[59] A. Alnajdi, F. Abdullah, A. Suryavanshi, and P. D. Christofides. Machine Learning-Based Model Predictive Control of Two-Time-Scale Systems. *Mathematics*, 11:3827, 2023.

[60] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3:551–560, 1990.

[61] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

[62] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine-learning-based predictive control of nonlinear processes. Part II: Computational implementation. *AIChE Journal*, 65:e16734, 2019.

[63] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1:4–27, 1990.

[64] T. W. S. Chow and Y. Fang. A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics. *IEEE Transactions on Industrial Electronics*, 45:151–161, 1998.

[65] K. Gurney. *An introduction to neural networks*. CRC press, 2014.

[66] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[67] Z. Wu, D. Rincon, and P. D. Christofides. Real-Time Adaptive Machine-Learning-Based Predictive Control of Nonlinear Processes. *Industrial & Engineering Chemistry Research*, 59:2275–2290, 2020.

[68] K. S. Holkar and L. M. Waghmare. An overview of model predictive control. *International Journal of Control and Automation*, 3:47–63, 2010.

[69] W. Tang and P. Daoutidis. Data-driven control: Overview and perspectives. In *Proceedings of the American Control Conference (ACC)*, pages 1048–1064. IEEE, 2022.

[70] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002.

[71] W.-X. Wang, R. Yang, Y.-C. Lai, V. Kovanis, and C. Grebogi. Predicting Catastrophes in Nonlinear Dynamical Systems by Compressive Sensing. *Physical Review Letters*, 106:154101, 2011.

[72] S. L. Brunton, J. H. Tu, I. Bright, and J. N. Kutz. Compressive Sensing and Low-Rank Libraries for Classification of Bifurcation Regimes in Nonlinear Dynamical Systems. *SIAM Journal on Applied Dynamical Systems*, 13:1716–1732, 2014.

[73] Z. Bai, T. Wimalajeewa, Z. Berger, G. Wang, M. Glauser, and P. K. Varshney. Low-Dimensional Approach for Reconstruction of Airfoil Data via Compressive Sensing. *AIAA Journal*, 53:920–933, 2015.

[74] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher. Sparse dynamics for partial differential equations. *Proceedings of the National Academy of Sciences*, 110:6634–6639, 2013.

[75] V. Ozolinš, R. Lai, R. Caflisch, and S. Osher. Compressed modes for variational problems in mathematics and physics. *Proceedings of the National Academy of Sciences*, 110:18368–18373, 2013.

[76] A. Mackey, H. Schaeffer, and S. Osher. On the Compressive Spectral Method. *Multiscale Modeling & Simulation*, 12:1800–1827, 2014.

[77] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113:3932–3937, 2016.

[78] B. M. de Silva, D. M. Higdon, S. L. Brunton, and J. N. Kutz. Discovery of Physics From Data: Universal Laws and Discrepancies. *Frontiers in Artificial Intelligence*, 3:25, 2020.

[79] Y. Cai, X. Wang, G. Joós, and I. Kamwa. An online data-driven method to locate forced oscillation sources from power plants based on sparse identification of nonlinear dynamics (SINDy). *IEEE Transactions on Power Systems*, 38:2085–2099, 2022.

[80] D. Bhattacharya, R. Hashem, L. K. Cheng, and W. Xu. Nonlinear model predictive control of a robotic soft esophagus. *IEEE Transactions on Industrial Electronics*, 69:10363–10373, 2021.

[81] L. Boninsegna, F. Nüske, and C. Clementi. Sparse learning of stochastic dynamical equations. *The Journal of Chemical Physics*, 148:241723, 2018.

[82] A. Narasingam and J. S.-I. Kwon. Data-driven identification of interpretable reduced-order models using sparse regression. *Computers & Chemical Engineering*, 119:101–111, 2018.

[83] T. A. Tamba and Y. Y. Nazaruddin. Data-Driven Construction of Chemical Reaction Network Graph Using Constrained LASSO. In *Proceedings of the 6th International Conference on Instrumentation, Control, and Automation (ICA)*, pages 226–230. IEEE, 2019.

[84] M. Hoffmann, C. Fröhner, and F. Noé. Reactive SINDy: Discovering governing reactions from concentration data. *The Journal of Chemical Physics*, 150:025101, 2019.

[85] F. Harirchi, D. Kim, O. Khalil, S. Liu, P. Elvati, M. Baranwal, A. Hero, and A. Violi. On sparse identification of complex dynamical systems: A study on discovering influential reactions in chemical reaction networks. *Fuel*, 279:118204, 2020.

[86] L. Scheffold, T. Finkler, and U. Piechottka. Gray-Box system modeling using symbolic regression and nonlinear model predictive control of a semibatch polymerization. *Computers & Chemical Engineering*, 146:107204, 2021.

[87] R. Subramanian, R. R. Moar, and S. Singh. White-box Machine learning approaches to identify governing equations for overall dynamics of manufacturing systems: A case study on distillation column. *Machine Learning with Applications*, 3:100014, 2021.

[88] J. Rubio-Herrero, C. O. Marrero, and W.-T. L. Fan. Modeling atmospheric data and identifying dynamics Temporal data-driven modeling of air pollutants. *Journal of Cleaner Production*, 333:129863, 2022.

[89] W. Farlessyost and S. Singh. Reduced order dynamical models for complex dynamics in manufacturing and natural systems using machine learning. *Nonlinear Dynamics*, 110:1613–1631, 2022.

[90] B. Bhadriraju, J. S.-I. Kwon, and F. Khan. An adaptive data-driven approach for two-timescale dynamics prediction and remaining useful life estimation of Li-ion batteries. *Computers & Chemical Engineering*, 175:108275, 2023.

[91] H.-C. Chang and M. Aluko. Multi-scale analysis of exotic dynamics in surface catalyzed

reactions—I: Justification and preliminary model discriminations. *Chemical Engineering Science*, 39:37–50, 1984.

[92] F. G. Heineken, H. M. Tsuchiya, and R. Aris. On the mathematical status of the pseudo-steady state hypothesis of biochemical kinetics. *Mathematical Biosciences*, 1:95–113, 1967.

[93] S. J. Merrill. A model of the stimulation of B-cells by replicating antigen—II. *Mathematical Biosciences*, 41:143–156, 1978.

[94] P. D. Christofides and P. Daoutidis. Feedback control of two-time-scale nonlinear systems. *International Journal of Control*, 63:965–994, 1996.

[95] J. Lévine and P. Rouchon. Quality control of binary distillation columns via nonlinear aggregated models. *Automatica*, 27:463–480, 1991.

[96] C. Georgakis. A quasi-modal approach to model reduction. In *Proceedings of the Joint American Control Conference*, volume 14, pages 639–644, San Fransisco, CA, USA, 1977.

[97] J. J. Monge and C. Georgakis. The Effect of Operating Variables on the Dynamics of Catalytic Cracking Processes. *Chemical Engineering Communications*, 60:1–26, 1987.

[98] K. P. Champion, S. L. Brunton, and J. N. Kutz. Discovery of Nonlinear Multiscale Systems: Sampling Strategies and Embeddings. *SIAM Journal on Applied Dynamical Systems*, 18:312–333, 2019.

[99] P. Kokotović, H. K. Khalil, and J. O'Reilly. *Singular Perturbation Methods in Control: Analysis and Design*. Society for Industrial and Applied Mathematics, 1999.

[100] A. M. Bruckstein, D. L. Donoho, and M. Elad. From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images. *SIAM Review*, 51:34–81, 2009.

[101] A. Doostan and H. Owhadi. A non-adapted sparse approximation of PDEs with stochastic inputs. *Journal of Computational Physics*, 230:3015–3034, 2011.

[102] J. Hampton and A. Doostan. Compressive sampling of polynomial chaos expansions: Convergence analysis and sampling strategies. *Journal of Computational Physics*, 280:363–386, 2015.

[103] E. J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346:589–592, 2008.

[104] H. Rauhut and R. Ward. Sparse Legendre expansions via $\ell 1$-minimization. *Journal of Approximation Theory*, 164:517–533, 2012.

[105] J. Peng, J. Hampton, and A. Doostan. On polynomial chaos expansion via gradient-enhanced $\ell 1$-minimization. *Journal of Computational Physics*, 310:440–458, 2016.

[106] D. Nguyen, S. Ouala, L. Drumetz, and R. Fablet. Assimilation-based Learning of Chaotic Dynamical Systems from Noisy and Partial Data. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 3862–3866, Barcelona, Spain, 2020.

[107] M. Didonna, M. Stender, A. Papangelo, F. Fontanela, M. Ciavarella, and N. Hoffmann. Reconstruction of Governing Equations from Vibration Measurements for Geometrically Nonlinear Systems. *Lubricants*, 7:64, 2019.

[108] G. Leylaz, S. Wang, and J.-Q. Sun. Identification of nonlinear dynamical systems with time delay. *International Journal of Dynamics and Control*, pages 1–12, 2021.

[109] M. Lin, C. Cheng, Z. Peng, X. Dong, Y. Qu, and G. Meng. Nonlinear dynamical system identification using the sparse regression and separable least squares methods. *Journal of Sound and Vibration*, 505:116141, 2021.

[110] A. T. Sarić, A. A. Sarić, M. K. Transtrum, and A. M. Stanković. Symbolic Regression for Data-Driven Dynamic Model Refinement in Power Systems. *IEEE Transactions on Power Systems*, 36:2390–2402, 2021.

[111] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for Time-Dependent Problems*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, 2007.

[112] H. Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473:20160446, 2017.

[113] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Inferring Biological Networks by Sparse Identification of Nonlinear Dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2:52–63, 2016.

[114] K. Kaheman, J. N. Kutz, and S. L. Brunton. SINDy-PI: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476:20200279, 2020.

[115] A. Cortiella, K.-C. Park, and A. Doostan. Sparse identification of nonlinear dynamical systems via reweighted $\ell 1$-regularized least squares. *Computer Methods in Applied Mechanics and Engineering*, 376:113620, 2021.

[116] S. Zhang and G. Lin. SubTSBR to tackle high noise and outliers for data-driven discovery of differential equations. *Journal of Computational Physics*, 428:109962, 2021.

[117] M. Quade, M. Abel, J. Nathan Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for rapid model recovery. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28:063116, 2018.

[118] T. Manzoor, H. Pei, Z. Sun, and Z. Cheng. Model Predictive Control Technique for Ducted Fan Aerial Vehicles Using Physics-Informed Machine Learning. *Drones*, 7:4, 2022.

[119] K. Huang, Z. Tao, Y. Liu, D. Wu, C. Yang, and W. Gui. Error-Triggered Adaptive Sparse Identification for Predictive Control and Its Application to Multiple Operating Conditions Processes. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2023.

[120] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37:233–243, 1991.

[121] D. Dong and T. McAvoy. Nonlinear principal component analysis—Based on principal curves and neural networks. *Computers & Chemical Engineering*, 20:65–78, 1996.

[122] G. Kimaev and L. A. Ricardez-Sandoval. Artificial Neural Network Discrimination for Pa-

rameter Estimation and Optimal Product Design of Thin Films Manufactured by Chemical Vapor Deposition. *The Journal of Physical Chemistry C*, 124:18615–18627, 2020.

[123] G. Kimaev and L. A. Ricardez-Sandoval. Artificial Neural Networks for dynamic optimization of stochastic multiscale systems subject to uncertainty. *Chemical Engineering Research and Design*, 161:11–25, 2020.

[124] G. Kimaev and L. A. Ricardez-Sandoval. Nonlinear model predictive control of a multiscale thin film deposition process using artificial neural networks. *Chemical Engineering Science*, 207:1230–1245, 2019.

[125] M. Paluš and I. Dvořák. Singular-value decomposition in attractor reconstruction: Pitfalls and precautions. *Physica D: Nonlinear Phenomena*, 55:221–234, 1992.

[126] L. Xu, E. Oja, and C. Y. Suen. Modified Hebbian learning for curve and surface fitting. *Neural Networks*, 5:441–457, 1992.

[127] T. Hastie and W. Stuetzle. Principal Curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

[128] W. S. Cleveland. Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association*, 74:829–836, 1979.

[129] J. L. Proctor, S. L. Brunton, B. W. Brunton, and J. N. Kutz. Exploiting sparsity and equation-free architectures in complex systems. *The European Physical Journal Special Topics*, 223:2665–2684, 2014.

[130] I. Arnaldo, U.-M. O'Reilly, and K. Veeramachaneni. Building Predictive Models via Feature Synthesis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, page 983–990, New York, NY, USA, 2015.

[131] E. D. Sontag. Feedback stabilization using two-hidden-layer nets. *IEEE Transactions on Neural Networks*, 3:981–990, 1992.

[132] A. Kumar, P. D. Christofides, and P. Daoutidis. Singular perturbation modeling of nonlinear processes with nonexplicit time-scale multiplicity. *Chemical Engineering Science*, 53:1491–1504, 1998.

[133] X. Chen, M. Heidarinejad, J. Liu, D. M. de la Peña, and P. D. Christofides. Model predictive control of nonlinear singularly perturbed systems: Application to a large-scale process network. *Journal of Process Control*, 21:1296–1305, 2011.

[134] M. Ellis, M. Heidarinejad, and P. D. Christofides. Economic model predictive control of nonlinear singularly perturbed systems. *Journal of Process Control*, 23:743–754, 2013.

[135] F. Abdullah, Z. Wu, and P. D. Christofides. Data-based reduced-order modeling of nonlinear two-time-scale processes. *Chemical Engineering Research and Design*, 166:1–9, 2021.

[136] L. A. Ricardez-Sandoval, H. M. Budman, and P. L. Douglas. Simultaneous design and control of processes under uncertainty: A robust modelling approach. *Journal of Process Control*, 18:735–752, 2008.

[137] L. A. Ricardez-Sandoval, H. M. Budman, and P. L. Douglas. Application of Robust Control

Tools to the Simultaneous Design and Control of Dynamic Systems. *Industrial & Engineering Chemistry Research*, 48:801–813, 2009.

[138] K. B. Sanchez-Sanchez and L. A. Ricardez-Sandoval. Simultaneous Design and Control under Uncertainty Using Model Predictive Control. *Industrial & Engineering Chemistry Research*, 52:4815–4833, 2013.

[139] P. Kokotović, H. K. Khalil, and J. O'Reilly. *Singular Perturbation Methods in Control: Analysis and Design*. Academic Press, London, 1986.

[140] Y. Lin and E. D. Sontag. A universal formula for stabilization with bounded controls. *Systems & Control Letters*, 16:393–397, 1991.

[141] R. Chartrand. Numerical Differentiation of Noisy, Nonsmooth Data. *ISRN Applied Mathematics*, 2011:1–11, 2011.

[142] A. Savitzky and M. J. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36:1627–1639, 1964.

[143] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Sparse identification of nonlinear dynamics with control (SINDYc). *IFAC-PapersOnLine*, 49:710–715, 2016.

[144] P. D. Christofides and A. R. Teel. Singular perturbations and input-to-state stability. *IEEE Transactions on Automatic Control*, 41:1645–1650, 1996.

[145] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search

algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.

[146] E. Aggelogiannaki and H. Sarimveis. Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models. *Computers & Chemical Engineering*, 32:1225–1237, 2008.

[147] R. Al Seyab and Y. Cao. Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation. *Journal of Process Control*, 18:568–581, 2008.

[148] J. Zeng, C. Gao, and H. Su. Data-driven predictive control for blast furnace ironmaking process. *Computers & Chemical Engineering*, 34:1854–1862, 2010.

[149] S. Aumi, B. Corbett, T. Clarke-Pringle, and P. Mhaskar. Data-driven model predictive quality control of batch processes. *AIChE Journal*, 59:2852–2861, 2013.

[150] W. Xie, I. Bonis, and C. Theodoropoulos. Data-driven model reduction-based nonlinear MPC for large-scale distributed parameter systems. *Journal of Process Control*, 35:50–58, 2015.

[151] D. Chaffart and L. A. Ricardez-Sandoval. Optimization and control of a thin film growth process: A hybrid first principles/artificial neural network based multiscale modelling approach. *Computers & Chemical Engineering*, 119:465–479, 2018.

[152] A. Garg and P. Mhaskar. Utilizing big data for batch process modeling and control. *Computers & Chemical Engineering*, 119:228–236, 2018.

[153] Z. Wu, J. Luo, D. Rincon, and P. D. Christofides. Machine learning-based predictive control using noisy data: evaluating performance and robustness via a large-scale process simulator. *Chemical Engineering Research and Design*, 168:275–287, 2021.

[154] Z. Wu, D. Rincon, J. Luo, and P. D. Christofides. Machine learning modeling and predictive control of nonlinear processes using noisy data. *AIChE Journal*, 67:e17164, 2021.

[155] F. Abdullah, Z. Wu, and P. D. Christofides. Sparse-identification-based model predictive control of nonlinear two-time-scale processes. *Computers & Chemical Engineering*, 153:107411, 2021.

[156] C. Moore. Application of Singular Value Decomposition to the Design, Analysis, and Control of Industrial Processes. In *Proceedings of the American Control Conference*, pages 643–650, Seattle, WA, USA, 1986.

[157] J. Huusom, N. Poulsen, S. Jørgensen, and J. Jørgensen. Tuning SISO offset-free Model Predictive Control based on ARX models. *Journal of Process Control*, 22:1997–2007, 12 2012.

[158] J. M. P. Menezes and G. A. Barreto. Long-term time series prediction with the NARX network: An empirical evaluation. *Neurocomputing*, 71:3335–3343, 2008.

[159] H. Siegelmann, B. Horne, and C. Giles. Computational capabilities of recurrent NARX neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27:208–215, 1997.

[160] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou. High-order neural network structures for identification of dynamical systems. *IEEE Transactions on Neural Networks*, 6:422–431, 1995.

[161] A. P. Trischler and G. M. D'Eleuterio. Synthesis of recurrent neural networks for dynamical system simulation. *Neural Networks*, 80:67–78, 2016.

[162] W. Wong, E. Chee, J. Li, and X. Wang. Recurrent Neural Network-Based Model Predictive Control for Continuous Pharmaceutical Manufacturing. *Mathematics*, 6:242, 2018.

[163] M. Dam, M. Brøns, J. Juul Rasmussen, V. Naulin, and J. S. Hesthaven. Sparse identification of a predator-prey system from simulation data of a convection model. *Physics of Plasmas*, 24:022310, 2017.

[164] H. Schaeffer, G. Tran, and R. Ward. Extracting Sparse High-Dimensional Dynamics from Limited Data. *SIAM Journal on Applied Mathematics*, 78:3279–3295, 2018.

[165] G. Tran and R. Ward. Exact Recovery of Chaotic Systems from Highly Corrupted Data. *Multiscale Modeling & Simulation*, 15:1108–1129, 2017.

[166] E. Kaiser, J. N. Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474:20180335, 2018.

[167] N. M. Mangan, T. Askham, S. L. Brunton, J. N. Kutz, and J. L. Proctor. Model selection for hybrid dynamical systems via sparse regression. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475:20180534, 2019.

[168] H. Schaeffer, G. Tran, R. Ward, and L. Zhang. Extracting Structured Dynamical Systems Using Sparse Optimization With Very Few Samples. *Multiscale Modeling & Simulation*, 18:1435–1461, 2020.

[169] J.-C. Loiseau and S. L. Brunton. Constrained sparse Galerkin regression. *Journal of Fluid Mechanics*, 838:42–67, 2018.

[170] L. Zhang and H. Schaeffer. On the Convergence of the SINDy Algorithm. *Multiscale Modeling & Simulation*, 17:948–972, 2019.

[171] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. W. Tsang, and M. Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8536–8546, Montréal, QC, Canada, 2018.

[172] R. Diversi, R. Guidorzi, and U. Soverini. Identification of ARX and ARARX Models in the Presence of Input and Output Noises. *European Journal of Control*, 16:242–255, 2010.

[173] P. Wu, H. Pan, J. Ren, and C. Yang. A New Subspace Identification Approach Based on Principal Component Analysis and Noise Estimation. *Industrial & Engineering Chemistry Research*, 54:5106–5114, 2015.

[174] B. C. Juricek, W. E. Larimore, and D. E. Seborg. Reduced-Rank ARX and Subspace System Identification for Process Control. *IFAC Proceedings Volumes*, 31:247–252, 1998.

[175] S. C. Patwardhan, S. Narasimhan, P. Jagadeesan, B. Gopaluni, and S. L. Shah. Nonlinear

Bayesian state estimation: A review of recent developments. *Control Engineering Practice*, 20:933–953, 2012.

[176] K. Yeo and I. Melnyk. Deep learning algorithm for data-driven simulation of noisy dynamical system. *Journal of Computational Physics*, 376:1212–1231, 2019.

[177] M. Hadigol and A. Doostan. Least squares polynomial chaos expansion: A review of sampling strategies. *Computer Methods in Applied Mechanics and Engineering*, 332:382–407, 2018.

[178] A. Cohen, M. A. Davenport, and D. Leviatan. On the Stability and Accuracy of Least Squares Approximations. *Foundations of Computational Mathematics*, 13:819–834, 2013.

[179] J. Hampton and A. Doostan. Coherence motivated sampling and convergence analysis of least squares polynomial Chaos regression. *Computer Methods in Applied Mechanics and Engineering*, 290:73–97, 2015.

[180] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60:259–268, 1992.

[181] B. Efron and C. Stein. The Jackknife Estimate of Variance. *The Annals of Statistics*, 9:586 – 596, 1981.

[182] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3:e1602614, 2017.

[183] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor. Model selection for dynamical

systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473:20170009, 2017.

[184] P. Zheng, T. Askham, S. L. Brunton, J. N. Kutz, and A. Y. Aravkin. A Unified Framework for Sparse Relaxed Regularized Regression: SR3. *IEEE Access*, 7:1404–1423, 2019.

[185] R. González-García, R. Rico-Martínez, and I. Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & Chemical Engineering*, 22:S965–s968, 1998.

[186] R. Fablet, S. Ouala, and C. Herzet. Bilinear Residual Neural Network for the Identification and Forecasting of Geophysical Dynamics. In *Proceedings of the 26th European Signal Processing Conference*, pages 1477–1481, Rome, Italy, 2018.

[187] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems. *arXiv:1801.01236*, 2018.

[188] S. H. Rudy, J. Nathan Kutz, and S. L. Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics*, 396:483–506, 2019.

[189] H. Schaeffer and S. G. McCalla. Sparse model selection via integral terms. *Phys. Rev. E*, 96:023302, 2017.

[190] A. A. R. AlMomani, J. Sun, and E. Bollt. How entropic regression beats the outliers problem in nonlinear system identification. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30:013107, 2020.

[191] Z. Zhang, Z. Wu, D. Rincon, and P. D. Christofides. Real-Time Optimization and Control of Nonlinear Processes Using Machine Learning. *Mathematics*, 7:890, 2019.

[192] M. S. Alhajeri, F. Abdullah, Z. Wu, and P. D. Christofides. Physics-informed machine learning modeling for predictive control using noisy data. *Chemical Engineering Research and Design*, 186:34–49, 2022.

[193] Y. M. Ren, M. S. Alhajeri, J. Luo, S. Chen, F. Abdullah, Z. Wu, and P. D. Christofides. A tutorial review of neural network modeling approaches for model predictive control. *Computers & Chemical Engineering*, 165:107956, 2022.

[194] D. Shah, J. Wang, and Q. P. He. Feature engineering in big data analytics for IoT-enabled smart manufacturing – Comparison between deep learning and statistical learning. *Computers & Chemical Engineering*, 141:106970, 2020.

[195] F. Abdullah, Z. Wu, and P. D. Christofides. Handling noisy data in sparse model identification using subsampling and co-teaching. *Computers & Chemical Engineering*, 157:107628, 2022.

[196] J. O. Ramsay, G. Hooker, D. Campbell, and J. Cao. Parameter estimation for differential equations: A generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69:741–796, 2007.

[197] Z. Feng, W. Shen, G. Rangaiah, and L. Dong. Closed-loop identification and model predictive control of extractive dividing-wall column. *Chemical Engineering and Processing - Process Intensification*, 142:107552, 2019.

[198] D. F. Mendoza, L. M. Palacio, J. E. Graciano, C. A. Riascos, A. S. Vianna, and G. A. Le Roux. Real-time optimization of an industrial-scale vapor recompression distillation process. model validation and analysis. *Industrial & Engineering Chemistry Research*, 52:5735–5746, 2013.

[199] M. Sharifzadeh. Integration of process design and control: A review. *Chemical Engineering Research and Design*, 91:2515–2549, 2013.

[200] Y. Lin, E. Sontag, and Y. Wang. A smooth converse Lyapunov theorem for robust stability. *SIAM Journal on Control and Optimization*, 34:124–160, 1996.

[201] J. L. Massera. Contributions to stability theory. *Annals of Mathematics*, 64:182–206, 1956.

[202] P. D. Christofides and N. H. El-Farra. *Control of Nonlinear and Hybrid Process Systems: Designs for Uncertainty, Constraints and Time-Delays*. Springer-Verlag, Berlin, Germany, 2005.

[203] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, New Jersey, third edition, 2002.

[204] M. S. Alhajeri, J. Luo, Z. Wu, F. Albalawi, and P. D. Christofides. Process structure-based recurrent neural network modeling for predictive control: A comparative study. *Chemical Engineering Research and Design*, 179:77–89, 2022.

[205] R. Polikar. Ensemble learning. In *Ensemble Machine Learning*, pages 1–34. Springer, 2012.

[206] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa. Ensemble approaches for regression: A survey. *ACM Computing Surveys*, 45:1–40, 2012.

[207] M. Heidarinejad, J. Liu, and P. D. Christofides. Economic model predictive control of nonlinear process systems using Lyapunov techniques. *AIChE Journal*, 58:855–870, 2012.

[208] Z. T. Wilson and N. V. Sahinidis. Data Driven Modeling in Alamo: Feature Selection and Non-Parametric Modeling Applications. In *Proceedings of the AIChE Annual Meeting*, Pittsburgh, PA, USA, 2018.

[209] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton. PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5:2104, 2020.

[210] A. A. Kaptanoglu, B. M. de Silva, U. Fasel, K. Kaheman, A. J. Goldschmidt, J. Callaham, C. B. Delahunt, Z. G. Nicolaou, K. Champion, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton. PySINDy: A comprehensive Python package for robust sparse system identification. *Journal of Open Source Software*, 7:3994, 2022.

[211] G. Leibovitz and R. Giryes. Efficient Least Residual Greedy Algorithms for Sparse Recovery. *IEEE Transactions on Signal Processing*, 68:3707–3722, 2020.

[212] D. Coughanowr. *Process Systems Analysis and Control*. McGraw-Hill International Editions. McGraw-Hill, 1991.

[213] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice–A survey. *Automatica*, 25:335–348, 1989.

[214] D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50:2967–2986, 2014.

[215] S. L. Brunton and B. R. Noack. Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews*, 67:050801, 2015.

[216] Z. Wu and P. D. Christofides. *Process Operational Safety and Cybersecurity*. Springer, 2021.

[217] F. Abdullah, M. S. Alhajeri, and P. D. Christofides. Modeling and Control of Nonlinear Processes Using Sparse Identification: Using Dropout to Handle Noisy Data. *Industrial & Engineering Chemistry Research*, 61:17976–17992, 2022.

[218] B. Bhadriraju, M. S. F. Bangi, A. Narasingam, and J. S.-I. Kwon. Operable adaptive sparse identification of systems: Application to chemical processes. *AIChE Journal*, 66:e16980, 2020.

[219] A. M. Stanković, A. A. Sarić, A. T. Sarić, and M. K. Transtrum. Data-driven symbolic regression for identification of nonlinear dynamics in power systems. In *Proceedings of the IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5, Montréal, QC, Canada, 2020.

[220] A. T. Sarić, A. A. Sarić, M. K. Transtrum, and A. M. Stanković. Symbolic regression for data-driven dynamic model refinement in power systems. *IEEE Transactions on Power Systems*, 36:2390–2402, 2020.

[221] M. Farsi and J. Liu. Structured online learning-based control of continuous-time nonlinear systems. *IFAC-PapersOnLine*, 53:8142–8149, 2020.

[222] J. Wang, J. Moreira, Y. Cao, and B. Gopaluni. Time-Variant Digital Twin Modeling through the Kalman-Generalized Sparse Identification of Nonlinear Dynamics. In *Proceedings of the American Control Conference (ACC)*, pages 5217–5222, Atlanta, GA, USA, 2022. IEEE.

[223] B. Bhadriraju, A. Narasingam, and J. S.-I. Kwon. Machine learning-based adaptive model identification of systems: Application to a chemical process. *Chemical Engineering Research and Design*, 152:372–383, 2019.

[224] R. Amrit, J. B. Rawlings, and D. Angeli. Economic optimization using model predictive control with a terminal cost. *Annual Reviews in Control*, 35:178–186, 2011.

[225] R. Huang, E. Harinath, and L. T. Biegler. Lyapunov stability of economically oriented NMPC for cyclic processes. *Journal of Process Control*, 21:501–509, 2011.

[226] M. Ellis, H. Durand, and P. D. Christofides. A tutorial review of economic model predictive control methods. *Journal of Process Control*, 24:1156–1178, 2014.

[227] A. Alanqar, H. Durand, and P. D. Christofides. Error-triggered on-line model identification for model-based feedback control. *AIChE Journal*, 63:949–966, 2017.

[228] F. Abdullah and P. D. Christofides. Data-based modeling and control of nonlinear process systems using sparse identification: An overview of recent results. *Computers & Chemical Engineering*, 174:108247, 2023.

[229] J. E. Bailey. Periodic operation of chemical reactors: A review. *Chemical Engineering Communications*, 1:111–124, 1973.

[230] P. L. Silveston. Periodic operation of chemical reactors – A review of the experimental literature. *Sādhanā*, 10:217–246, 1987.

[231] C. Tsay and M. Baldea. Integrating production scheduling and process control using latent variable dynamic models. *Control Engineering Practice*, 94:104201, 2020.

[232] J. C. Schulze, D. T. Doncevic, and A. Mitsos. Identification of MIMO Wiener-type Koopman models for data-driven model reduction using deep learning. *Computers & Chemical Engineering*, 161:107781, 2022.

[233] B. Likar and J. Kocijan. Predictive control of a gas–liquid separation plant based on a Gaussian process model. *Computers & Chemical Engineering*, 31:142–152, 2007.

[234] T. Bikmukhametov and J. Jäschke. Combining machine learning and process engineering physics towards enhanced accuracy and explainability of data-driven models. *Computers & Chemical Engineering*, 138:106834, 2020.

[235] M. S. Alhajeri, Z. Wu, D. Rincon, F. Albalawi, and P. D. Christofides. Machine-learning-based state estimation and predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 167:268–280, 2021.

[236] J. Sansana, M. N. Joswiak, I. Castillo, Z. Wang, R. Rendall, L. H. Chiang, and M. S. Reis. Recent trends on hybrid modeling for Industry 4.0. *Computers & Chemical Engineering*, 151:107365, 2021.

[237] K. McBride and K. Sundmacher. Overview of Surrogate Modeling in Chemical Process Engineering. *Chemie Ingenieur Technik*, 91:228–239, 2019.

[238] G. Porru, C. Aragonese, R. Baratti, and Alberto Servida. Monitoring of a CO oxidation reactor through a grey model-based EKF observer. *Chemical Engineering Science*, 55:331–338, 2000.

[239] R. Oliveira. Combining first principles modelling and artificial neural networks: A general framework. In A. Kraslawski and I. Turunen, editors, *European Symposium on Computer Aided Process Engineering-13*, volume 14 of *Computer Aided Chemical Engineering*, pages 821–826. Elsevier, 2003.

[240] M. von Stosch, R. Oliveira, J. Peres, and S. Feyo de Azevedo. Hybrid semi-parametric modeling in process systems engineering: Past, present and future. *Computers & Chemical Engineering*, 60:86–101, 2014.

[241] S. Zendehboudi, N. Rezaei, and A. Lohi. Applications of hybrid models in chemical, petroleum, and energy systems: A systematic review. *Applied Energy*, 228:2539–2566, 2018.

[242] M. S. F. Bangi and J. S.-I. Kwon. Deep hybrid modeling of chemical process: Application to hydraulic fracturing. *Computers & Chemical Engineering*, 134:106696, 2020.

[243] D. Lee, A. Jayaraman, and J. S. Kwon. Development of a hybrid model for a partially known intracellular signaling pathway through correction term estimation and neural network modeling. *PLOS Computational Biology*, 16:1–31, 2020.

[244] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[245] M. Moonen, B. De Moor, L. Vandenberghe, and J. Vandewalle. On-and off-line identification of linear state-space models. *International Journal of Control*, 49:219–232, 1989.

[246] M. Lovera, T. Gustafsson, and M. Verhaegen. Recursive subspace identification of linear and non-linear Wiener state-space models. *Automatica*, 36:1639–1650, 2000.

[247] G. Mercere, S. Lecoeuche, and M. Lovera. Recursive subspace identification based on instrumental variable unconstrained quadratic optimization. *International Journal of Adaptive Control and Signal Processing*, 18:771–797, 2004.

[248] A. Alanqar, H. Durand, and P. D. Christofides. Fault-Tolerant Economic Model Predictive Control Using Error-Triggered Online Model Identification. *Industrial & Engineering Chemistry Research*, 56:5652–5667, 2017.