University of California
Santa Barbara

Feedback controller design and process modeling methods using machine learning

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Chemical Engineering

by

Pratyush Kumar

Committee in charge:

Professor James B. Rawlings, Chair
Professor Michael F. Doherty
Professor Todd M. Squires
Professor João P. Hespanha

March 2023

The Dissertation of Pratyush Kumar is approved.

Professor Michael F. Doherty

Professor Todd M. Squires

Professor João P. Hespanha

Professor James B. Rawlings, Committee Chair

January 2023

Feedback controller design and process modeling methods using machine learning

To my family.

Acknowledgments

I express my sincere gratitude to my advisor Prof. James Rawlings for his guidance and support throughout my PhD studies. Without his supervision, knowledge, and experience, I would not have been able to accomplish the work in this thesis. I am thankful to him for always challenging me to work on impactful research problems, and also for his patience and encouragement during my graduate school. His eagerness to teach and learn from students has always inspired me. I have grown both personally and professionally during my time in his research group.

I thank Profs. Mike Doherty, Todd Squires, and Joao Hespanha for taking the time to serve on my PhD committee. I am also grateful to Prof. Stephen Wright from UW Madison for his advice and collaboration on some machine learning aspects of my research.

During my PhD studies, I had an opportunity to closely work with a few industrial practitioners. I am thankful to Drs. Robert Turney, Michael Wenzel, Mohammad Elsbat, and Michael Risbeck from Johnson Controls International (JCI). My time as an intern at JCI was highly useful to gain an industrial perspective on control systems technologies and generate impactful research ideas. It was also a pleasure to work with Dr. Pierre Carrette from Shell on a project about MPC-PI cascade control systems in industrial applications.

I also thank my undergraduate Profs. Ravindra Gudi, Mani Bhushan, Sachin Patwardhan, and Vinay Prasad who initially introduced me to the field of process control and optimization.

I have been fortunate to work with a few great colleagues in the Rawlings group. In my first year at UW Madison, Michael Risbeck and Nishith Patel provided assistance on group software and were helpful in MPC related questions. I thank Michael additionally for engaging in technical discussions during my internship at JCI, and for creating highly useful group software tools that saved valuable time in research projects. Travis Arnold and Douglas Allan were great colleagues at both UW Madison and UCSB. They were always willing to engage in technical discussions on controls topics and research ideas. Koty McAllister has been an amazing colleague and friend throughout my graduate school. The move from UW Madison to UCSB was smooth due to his support and good humor. I also thank him for the many outside of work fun activities he engaged with me in Madison and Santa Barbara. It has been my pleasure to work with the younger Rawlings group members Steven Kuntz, Chris Kuo-Leblanc, Davide Mannini, and Titus Quah. I wish them all the best in the rest of their PhD studies and future endeavors.

My PhD work would have been very different without access to the powerful scientific computing software I used for my research. I also thank the developers of CasADi, TensorFlow, MPCTools, and Matplotlib for creating valuable software.

My time at Santa Barbara will be memorable for many years because of all fun activities I did with my friends. I am grateful to Patrick Leggieri, Michael Schmithorst, Candice Macabuhay, Thomas Lankiewicz, Kellie Heom, and Sally Jiao. Thanks for including me in all the beach days, cliff dye games, pool nights, downtown events, thanksgiving dinners, and Superbowl days, etc. All the activities kept me sane during a graduate school journey away from home. I will cherish all the memories I have made for a long time. I am also thankful to Akash Deep, Alec Linot, Jonathan Sheavly, and Mike Jindra for their friendship in my first year at UW Madison.

Finally, I am grateful to my mom, dad, and younger brother for their endless love and support. Throughout my years growing up, my parents made strong efforts so that I can get good education, and taught me the value of hard work. I thank them for letting me pursue graduate school in a different country across the globe, and I cannot imagine completing my PhD without their support.

Pratyush Kumar
Santa Barbara, CA
January 2023

Curriculum Vitæ
Pratyush Kumar

Education

Sept, 2017 - Jan, 2023	**University of California, Santa Barbara** Ph.D. in Chemical Engineering (Expected) First year 2017-2018 at the University of Wisconsin, Madison
July, 2013 - Aug, 2017	**Indian Institute of Technology Bombay** Bachelors of Technology in Chemical Engineering

Publications

2021	Industrial, large-scale model predictive control with structured neural networks P. Kumar, J. B. Rawlings, S. J. Wright *Published* in Computers and Chemical Engineering.
2022	Modeling proportional-integral controllers in tracking and economic model predictive control P. Kumar, J. B. Rawlings, P. Carrette *Published* in Journal of Process Control.
Under review	Grey-box model and neural network disturbance predictor identification for economic MPC in building energy systems. P. Kumar, J. B. Rawlings, M. J. Wenzel, M. J. Risbeck *Submitted* to Energy and Buildings.
Under review	Unconstrained feedback controller design using Q-learning from noisy process data P. Kumar, J. B. Rawlings *Submitted* to Computers and Chemical Engineering.
Under review	Hybrid process modeling strategies using neural networks and application to economic optimization P. Kumar, J. B. Rawlings *Submitted* to Computers and Chemical Engineering.

Internships

June - Sept, 2022	**Data Center Engineering Intern, Google LLC** Advanced technology innovation to improve data center efficiency.
June - Sept, 2019	**Controls Research Intern, Johnson Controls International** First principles and machine learning based dynamic modeling of building systems.

May - Aug, 2016	**Research Intern, University of Alberta, Edmonton, Canada** Structural controllability and observability analysis of graphical systems with application to oil reservoirs.

Accolades

2022	Awarded a quarter long fellowship from the Center for control, dynamical systems, and computation (CCDC) at UCSB.
2017	Received the Timothy C. Scott fellowship at UW Madison.
2017	Granted the undergraduate research award at IIT Bombay.
2016	Awarded the MITACS Globalink scholarship to pursue a research internship in Canada.

Abstract

Feedback controller design and process modeling methods using machine learning

by

Pratyush Kumar

The recent advances in the field of machine learning, the availability of powerful computing resources, and growing data collection capabilities across industries present new opportunities to upgrade the existing feedback control and automation methods implemented in applications. This thesis presents novel approaches and methods to use machine learning algorithms to develop feedback controllers and process models for dynamical systems. In the first half of this thesis, we develop approaches to use reinforcement learning and neural networks to design feedback controllers for multivariable systems. We develop a novel model-free Q-learning approach suitable to estimate linear, unconstrained feedback controllers from noisy process data. We present a neural network (NN) design approach to approximate the model predictive control (MPC) feedback law for large-scale applications that may be out of reach with available QP solvers. The proposed NN design approach is applied to a large industrial crude distillation unit model, and we demonstrate that NNs can be used to execute MPC orders of magnitude faster compared to an available QP solver.

The next half of this thesis focuses on developing hybrid model identification approaches that utilize both the advantages of neural networks and some first principles process knowledge usually available in applications. We consider building systems affected by large occupancy induced heat disturbances. For these systems, we develop a novel two step grey-box dynamic and NN disturbance model identification framework. We use a NN to model the heat disturbance so that it can be used to pro-

vide feedforward predictions of the disturbance in an MPC controller for improved energy cost optimization. We also present a hybrid modeling approach for nonlinear chemical engineering processes. For this class of systems, we use NNs to approximate some functions in the overall dynamic model, e.g, reaction kinetics, which may be challenging to parameterize using the available domain knowledge. The estimated hybrid models are used for steady-state economic optimization at the real time optimization layer. Throughout this thesis, we present examples with heating, ventilation, and air-conditioning and chemical engineering systems to demonstrate the effectiveness of the proposed controller design and process modeling methods. We compare the proposed methods with existing approaches and illustrate their potential to design high-performance control systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The automation of large, constrained industrial processes in modern markets with dynamically changing commodity prices is performed using an online optimization approach. This online optimization of a process is carried out using a hierarchy (Figure 1.1) of several layers that are designed based on a time scale separation of the process. The real time optimization (RTO) and the model predictive control (MPC) layers are crucial elements of the hierarchy, both of which aim to optimize process performance online. These layers use mathematical models describing the process dynamics in an optimization problem. The quality of the model used at the layers and the design of the optimization problem solved are both influential to the performance of the overall process operations system.

The RTO layer solves a steady-state optimization problem based on raw material and product prices, product goals, etc, to determine an economically profitable steady state of the plant. This layer uses a nonlinear model between the actuators and measurements of the process, and the problem at the layer is solved approximately on a time scale of hours. The computed steady state is passed as a collection of setpoints to the MPC layer, which then dynamically drives and maintains the plant at the setpoints requested by the RTO layer. The MPC controller uses an approximate linear model of the plant and solves a quadratic program (QP) at a time scale of minutes. The

solutions of the MPC optimization problem are further passed as setpoints to regulatory layer proportional-integral (PI) controllers, which then regulate some single input single output (SISO) processes and manipulate the final plant actuators.

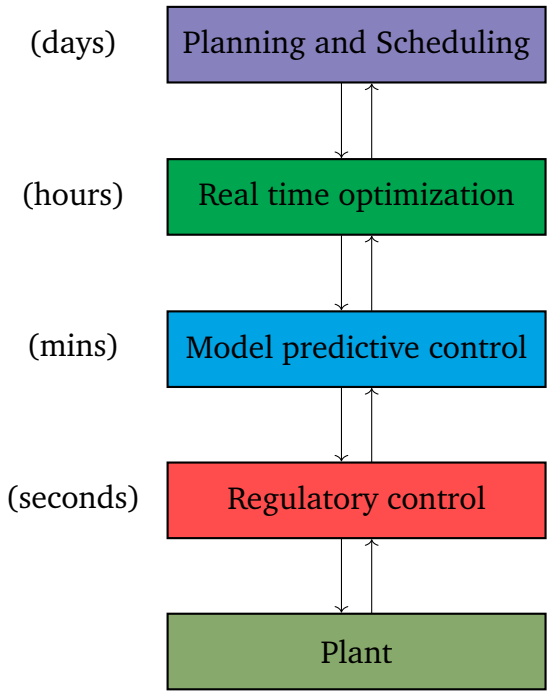

Figure 1.1: Diagram of a process operations hierarchy (Seborg et al., 2017, Chapter 19) typically implemented for the automation of large industrial systems.

The research in the field of process operations and control is often directed towards developing technologies to improve the efficiency and performance of the process operations hierarchy. In particular, process modeling or system identification is an important area because accurate dynamic models are valuable to improve economic performances of both the RTO and MPC layers. The development of advanced MPC controller formulations is also an important topic that has an industrial impact for designing high performance automation systems. In this thesis, we use modern machine learning algorithms to develop (i) controller design methods for implementation at the MPC layer, and (ii) novel approaches to construct dynamic process models for use in both the RTO and MPC layers.

The motivations for using ML algorithms for process modeling and control are as follows. First, several algorithmic advances have been made in the field over the past few decades that nowadays conveniently allow the development of sophisticated ML models. Most notable advances are neural network (NN) training methods (LeCun et al., 2015; Hinton et al., 2006; Srivastava et al., 2014), symbolic differentiation based software (Abadi et al., 2015), and deep reinforcement learning approaches (Mnih et al., 2015). The researchers in the field have leveraged the advances to demonstrate impressive capabilities of ML algorithms for computer vision (Krizhevsky et al., 2017) and autonomy tasks (Silver et al., 2016; Kober et al., 2013). Second, the computing power available to implement ML algorithms using big data has also grown significantly in the recent years. Powerful resources are available in applications for both online and offline computations. And cloud computing services offered by large corporations provide unparalleled computational resources than available a few decades ago. Lastly, the volume of data collection is also growing in this modern age across a range of industries.

This thesis aims to leverage the above opportunities to develop advanced feedback controller design and process modeling methods. In this introductory chapter, we first give a brief overview of the existing methods for process modeling, model predictive control, and machine learning. In Section 1.2, we discuss how the machine learning algorithms can be used for feedback controller design and dynamic process modeling. Then we outline the rest of the chapters and contributions of this thesis in that section. The mathematical notation used for the rest of the thesis are discussed in Section 1.3.

1.1 Background

1.1.1 Process modeling

Dynamic process modeling is the task of developing a continuous or discrete time representation of the dynamics of some process. For example, consider a process described by the ordinary differential equations (ODEs)

$$\frac{dx}{dt} = f(x, u) + w \tag{1.1}$$

$$y = h(x) + v \tag{1.2}$$

in which, x is the state, y is the measurement, u is the control input, and w and v are the process and measurement noises. In process modeling, we aim to develop a model between the inputs (u) and measurements (y) that closely approximates the input to measurement relationship of the above true dynamics (1.1) – (1.2). This task is also called system identification (Ljung, 1999) and is studied widely in the control systems literature due to the importance of dynamic models in control systems design.

A dynamic model can be developed using a first principles, grey-box, or a black-box modeling approach. In the first principles approach, the model is developed using rigorous domain knowledge about the process dynamics. The approach does not use any data collected from the true process, and it is sometimes also referred as the white-box approach. The grey-box and black-box modeling methods use some data collected from the process and solve a data fitting problem to develop the dynamic model. To perform grey-box modeling, a dynamic model is parameterized using the available first principles knowledge. The unknown parameters in the model are subsequently estimated by solving the data fitting problem. The black-box approach does not use any first principles knowledge. A model is first parameterized using completely black-

box linear or nonlinear function approximators, and the parameters are estimated by solving the data fitting problem.

As mentioned previously, the RTO layer uses a nonlinear process model and the MPC controller uses a linear dynamic model. In standard industrial practice, the nonlinear model for the RTO layer is typically developed using one of the first principles or grey-box modeling approaches. These modeling approaches, however, have an issue that the available first principles knowledge in applications can be often incorrect or incomplete. And a model developed using these approaches can have a plant-model mismatch, leading to a suboptimal economic performance. For the MPC controller, a linear dynamic model can be estimated using black-box linear system identification (Qin, 2006) methods available in the literature. The linear model almost always has a mismatch with the actual plant, so integrating disturbance models are used in conjunction with the main dynamic model. The disturbance model helps to achieve a zero setpoint tracking error in the primary measurements during the online closed-loop implementation.

1.1.2 Model predictive control

The MPC controller is a crucial element of the process operations hierarchy. The controller uses a linear model and solves a dynamic optimization problem to manipulate the actuators for process operation. The ability of the MPC approach to systematically handle large multivariable systems and constraints led to its widespread adoption in the process industries (Qin and Badgwell, 2003).

An industrial MPC controller has three components: a state estimator, a target selector, and a regulator. The state estimator uses past observations of the control inputs and measurements to obtain the best possible estimates of the state and disturbance.

These two estimates correspond to the states in the main dynamic model and the disturbance model. The target selector then uses the disturbance estimate and the desired setpoints requested by the RTO layer to determine a target steady state. This target is computed such that the actual plant measurements achieve a zero setpoint tracking error at that steady state. The MPC regulator next solves a dynamic optimization problem based on the state estimate, the linear model forecasts, and the target steady state to determine the control input to apply to the plant.

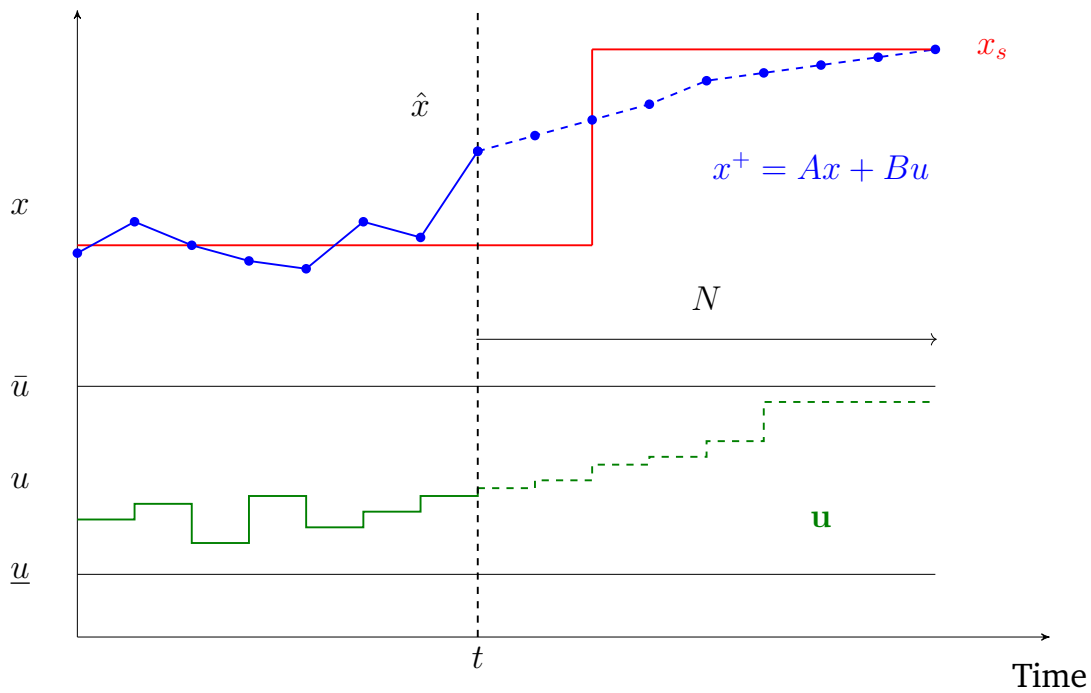

Figure 1.2: Pictorial representation of a closed-loop MPC implementation. After every sampling instant, the MPC optimizes over a future control input sequence based on the state estimate, the linear model forecasts, and the target steady state. The first input is applied to the plant and the optimization process is repeated online.

In Figure 1.2, we show a pictorial representation of an online MPC controller execution. After every sampling instant, the MPC regulator uses the state estimate ($\hat{x}$) and the linear model ($x^+ = Ax + Bu$) in a dynamic optimization problem to decide the best input sequence to drive the system to the target steady state (x_s, u_s). The first input is

applied to the system, and the dynamic optimization problem is resolved after the next sampling instant.

The MPC regulator solves the following optimization problem

$$\min_{\mathbf{u}} \sum_{k=0}^{N-1} \Big((x(k) - x_s)' Q (x(k) - x_s) + (u(k) - u_s)' R (u(k) - u_s) \Big)$$

$$+ (x(N) - x_s)' P (x(N) - x_s) \tag{1.3}$$

$$\text{s.t} \quad x^+ = Ax + Bu, \tag{1.4}$$

$$\underline{u} \leq u \leq \overline{u} \tag{1.5}$$

$$x(0) = \hat{x} \tag{1.6}$$

in which, N is the forecasting horizon length, $[\underline{u},\ \overline{u}]$ denote the actuator constraints, Q, R, and P are the penalty matrices used in the stage costs. The future control input sequence $\mathbf{u} = [u(0)', u(1)', \ldots, u(N)']'$ is the decision variable. This optimization problem is a quadratic program (QP). The first element of the optimal control input sequence $u^0(0)$ is applied to the plant, and the QP is resolved online after the next sampling instant. We note that the solution of the MPC optimization problem can be explicitly characterized as a piecewise affine function over the state estimate and the target steady-state pair. We refer to the first element of that function as the MPC feedback law, denoted as $u^0(0) = \kappa_N(\hat{x}, x_s, u_s)$.

The MPC controller formulation above is a type of setpoint tracking formulation. The MPC aims to drive the measurements to the setpoints requested by the RTO layer, which are based on plant economics. An alternative strategy to the multi-layered RTO and MPC approach can be to combine the two layers and use an “economic MPC" formulation (Rawlings et al., 2012). In this formulation, the objective function in the MPC optimization problem can be chosen directly based on the plant economics.

The MPC approach has strong stability and robustness properties, which make the technology reliable for implementation in industrial applications. Mayne et al. (2000) reviews that an MPC formulated using an appropriate terminal cost and constraint has a nominal stability property. The optimal cost function of the QP can be used as a Lyapunov function to establish the nominal stability property. Allan et al. (2017) discusses the inherent robustness property of a nonlinear MPC controller formulation to small disturbances. The term "inherent robustness" refers to the robustness provided by the MPC controller even when the optimization problem is not explicitly designed to be robust to disturbances.

1.1.3 Machine learning

The field of machine learning focuses on developing methods and algorithms that use data to improve performance on a specified task (Mitchell et al., 1997). This task may be prediction, pattern recognition, control, etc. The types of ML algorithms can be broadly categorized into supervised, unsupervised, and reinforcement learning.

Supervised learning algorithms aim to solve the problem of identifying relationships between some labelled input and output data. That is, given a set of labelled pairs (x_i, y_i), estimate a function that best approximates the true relationship between the pairs. The estimated function may then be used subsequently to predict the output when a label is not available. Examples of the types of supervised learning algorithms are linear regression, artificial neural networks, support vector machines, etc. The unsupervised learning algorithms are presented with only the inputs and without any labelled outputs. The goal of these algorithms is to discover patterns in the inputs.

Reinforcement learning (RL) algorithms solve a decision-making problem that maximizes a reward when operating in an environment. The algorithms are allowed to in-

teract with the environment using an action and obtain a measure of performance via a reward function. The algorithm can choose a series of actions, and the final goal is to learn the optimal way to choose the action that maximizes the reward. This objective of RL algorithms is very similar to that of classical feedback control methods. In which we assume that given the ability to interact with a process using a manipulated control input, we would like to decide how to choose the control input in a way that minimizes a cost function. Modern reinforcement learning algorithms have achieved remarkable results for decision-making purposes when playing Go (Silver et al., 2016) and Atari games (Mnih et al., 2015).

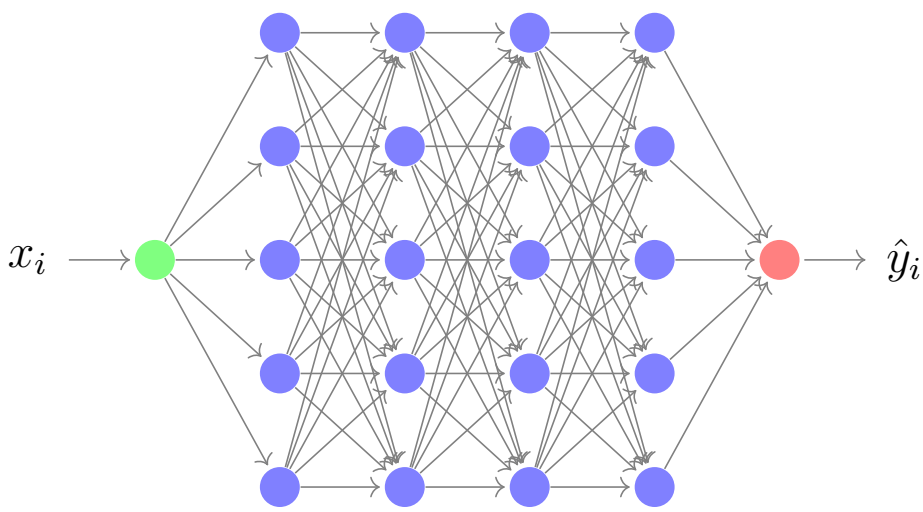

Figure 1.3: Diagram of a feedforward neural network. An input vector x_i is propagated through the layers in the network to predict the output $\hat{y}_i$.

We use neural networks (NNs) as function approximators for the majority of the approaches developed in this thesis. So we next describe the architecture and training process of a standard feedforward NN. Figure 1.3 shows a diagram of a standard feedforward network. An input vector x_i is provided to the network, which then undergoes a series of operations to predict the output $\hat{y}_i$. The following operations are performed

in the network

$$z_0 = x_i \tag{1.7}$$

$$z_{l+1} = g_l(W_l z_l + b_l), \quad l \in [1,\ h] \tag{1.8}$$

$$\hat{y}_i = z_{h+1} \tag{1.9}$$

in which, W_l and b_l are the weights and biases in the network at the layer l, and h is the number of layers in the network. The function $g_l(\cdot)$ is a nonlinear activation function, which we consider as either the hyperbolic tangent or the ReLU activation function defined as $g(a) = \text{max}(0, a)$. We do not use an activation function in the last layer, i.e., $g_h(a) = a$.

The recursive application of the series of operations at each layer enables a feed-forward NN to represent complex nonlinear functions. And they are well-known for their function approximation capabilities. The weights and biases in the network can be estimated using a set of labelled data (x_i, y_i) by minimizing the prediction error of the network as follows

$$\min_{W_l,\ b_l} \sum_{i=1}^{N_s} |y_i - \hat{y}_i(x_i)|^2 \tag{1.10}$$

in which, y_i is the true label corresponding to the input x_i, $\hat{y}_i(x_i)$ is the NN prediction corresponding to that input, the symbol $|\cdot|$ denotes the norm of a vector, and N_s is the total number of training data samples. The decision variables in this problem are the weights and biases of all the layers (W_l, b_l) in the network. The optimization problem can be solved using a stochastic gradient algorithm such as Adam (Kingma and Ba, 2014).

The optimization algorithm uses gradients of the objective function computed over

some small "batches" of the entire training data in the descent step. This approach enables the algorithm to make faster progress to an optimum compared to if the entire training data were used to compute the gradient in the descent step. The size of the data over which the gradient is computed is referred as the "batch size". The algorithm uses gradients from multiple batches to complete one traversal over the entire training data. Each complete traversal over the entire training data is referred as one "epoch". The batch size and the number of epochs are hyperparameters that can be tuned to achieve a good model fit.

During the training, one must take precaution that the model does not overfit to the training data. And that it also generalizes to similar data not available in the training set. For this purpose, we always separate a holdout data set from the entire training set. On the holdout data set, the optimization problem loss metric is computed at the end of every epoch. At the end of training, we use the NN weights and biases that have the best loss metric on the holdout data set for further validation. This approach ensures that the developed NN is not overfitted to the training data and also generalizes to samples not available in the training set.

1.2 Overview of thesis

Machine learning methods can be used for process modeling and control in the following three approaches. First, reinforcement learning methods may be used for model-free feedback controller design from process data without estimating a dynamic model. The motivation for this approach is that a model-free method avoids the model identification step, which can sometimes be considered a challenging step for feedback controller design in process control applications. Second, neural networks (NNs) may be used to approximate the MPC feedback law ($\kappa_N(\cdot)$) offline using the solutions of

the MPC optimization problem as labelled data. The offline trained network can then be used online to implement an "approximate" MPC controller in place of using QP solvers. The advantage of this approach is that NNs can execute MPC faster than online QP solvers, and allow practitioners to implement MPC controllers on large-scale applications. Third, neural networks may be used to estimate dynamic process models from data, which can then be used at the RTO and MPC layers. The motivation for this approach is that NNs can allow the development of accurate models, and thus provide an improved economic performance of the process operations hierarchy.

We follow the above three approaches to develop the process modeling and controller design methods proposed in this thesis. For reinforcement learning, we focus on estimating the linear quadratic regulator (LQR) using Q-learning from *noisy data*. For the MPC feedback law approximation, we consider *large-scale* applications that may be out of reach for available online QP solvers. For process modeling, we develop *hybrid* modeling approaches using NNs that also utilize some first principles knowledge to develop the process model.

The rest of this thesis is organized into five more chapters. The first two chapters focus on using reinforcement learning and neural networks for feedback controller design for multivariable processes. And the next two chapters develop hybrid modeling methods for building applications and nonlinear chemical engineering processes. We next highlight the contributions and contents of each chapter.

Chapter 2: Model-free controller design using Q-learning from noisy data.

In this chapter, we start with a discussion on the related research in the area of model-free RL for feedback controller design. We discuss an existing Q-learning algorithm to estimate the nominal LQR feedback law from data. We highlight via a simulation study that this RL algorithm is not suitable to estimate the LQR feedback

law from noisy data.

Next, we discuss an available algorithm to estimate the stochastic LQR feedback law, which was developed for linear systems with Gaussian process noise of *known* covariance. We build upon this algorithm and propose to extend it to treat the case of an *unknown* noise covariance. Then, we use the extended algorithm to estimate a feedback controller for linear systems with both process and measurement noise and only output measurements. For comparative studies, we also discuss a maximum likelihood estimation (MLE) algorithm to estimate a model-based controller. Simulation studies are presented using a linear heating, ventilation, and air-conditioning (HVAC) example system. We demonstrate the suitability of the developed model-free Q-learning approach to estimate a reasonable feedback controller from a viable amount of training data. We show that a controller estimated with the proposed model-free Q-learning approach provides a very similar closed-loop performance as a controller estimated using the model-based MLE method.

Chapter 3: Fast, large-scale model predictive control using deep neural networks.

This chapter presents the design of neural networks (NNs) to approximate the MPC feedback law so that NNs can be used to implement MPC for large-scale industrial applications.

We start with examining the MPC feedback law approximation approach with a small double integrator example and on the control problem of regulation to the origin. We analyze the quality and the nature of the function approximations by NNs in this example. Then, we present the design of NNs to approximate the feedback law for the industrially relevant offset-free MPC formulation. We propose a novel structured NN architecture that can be used to achieve offset free closed-loop performance in applications. We discuss the data generation approach to sample the relevant state space for

NN training based on the anticipated plant setpoints and disturbances. In this chapter, we also establish an inherent robustness property of approximate NN controllers using the input-to-state stability results available within the MPC literature. We show that if NNs are sufficiently trained and have low enough MPC feedback law approximation errors, then they are robust to small disturbances.

The application of the proposed NN controller design approach is demonstrated via simulation studies on two large-scale application examples. Specifically, we apply the NN design approach on an industrial crude distillation unit model with 252 states, 32 control inputs, and a control sample horizon length of 140. We demonstrate that NNs can be used to execute MPC around four orders of magnitude faster than an available QP solver.

Chapter 4: Grey-box modeling and disturbance forecasting in building energy systems.

Building systems comprise a large portion of the United States energy usage. For modern markets with dynamically changing energy prices, researchers have proposed the use of economic MPC to optimize energy cost in real time. Buildings systems are affected by disturbances such as the ambient temperature and the heat load generated by occupants. Both these disturbances have a significant contribution to the building dynamics. And the performance of an economic MPC controller depends on the quality of both the building dynamic model and the disturbance forecasts.

In this chapter, we propose a novel two-step method to estimate a grey-box building dynamic model and a neural network to predict the large occupancy generated heat disturbance. In the first step, a grey-box building model is estimated using some input excitation data. We treat the unmeasured heat disturbance in this step by estimating a piecewise constant signal for the disturbance in the same grey-box building

model identification problem. We also present an approach to compute approximate confidence intervals on the physical parameters in the grey-box building model. In the second identification step, we use historical operational data to identify patterns in the occupancy generated heat disturbance using a NN.

We present case studies using a two time scale building system to demonstrate the efficacy of the proposed model identification approach. The subsequent use of the composite building and NN disturbance model for economic MPC is also demonstrated via closed-loop simulation studies. We illustrate that modeling and forecasting the occupancy generated heat disturbance in the MPC problem is valuable to attain improved energy cost savings.

Chapter 5: Hybrid process modeling with application to economic optimization.

The chapter develops a hybrid modeling approach to estimate nonlinear process models for use in steady-state economic optimization at the RTO layer.

As mentioned earlier, developing a fully first principles based or a grey-box dynamic model can be challenging for many industrial processes due to incomplete or incorrect process knowledge. We present a modeling approach that utilizes both the available process knowledge and the advantages of neural networks. The NNs are used to approximate some complex unknown functions in the dynamic model that may be challenging to parameterize using the available process knowledge. We consider two nonlinear chemical process examples and demonstrate the suitability of the modeling approach to estimate accurate dynamic models from process data.

We also examine the performance of the hybrid models when used in a steady-state economic optimization problem typically solved at the RTO layer in the process industries. We elucidate the type of data that should be collected from the process if the final goal is to use an estimated hybrid model in steady-state optimization. In the case

studies, we also develop multiple hybrid models with different NN function parameterization choices along with a fully black-box NN model. We show that hybrid models that have the most possible information about the functions being approximated by the NNs provide good steady-state economic performance. We emphasize that structural insights are crucial to obtain high performance using the hybrid models.

Chapter 6: Concluding remarks.

This chapter first summarizes the results and contributions of this thesis. Then, we provide future research directions for each approach discussed in this thesis on using machine learning methods for process modeling and control.

1.3 Notation

We use the symbols $\mathbb{I}$ and $\mathbb{R}$ denote integers and reals respectively. Subscripts on these symbols denote restrictions, e.g., $\mathbb{I}_{a:b}$ denotes integers in the closed interval $[a, b]$. The symbol $\mathbb{R}^n$ denotes a vector of real numbers in n dimensions.

We use a bold symbol $\mathbf{d}$ to denote a time sequence, and a vector $d(k)$ denotes an element of $\mathbf{d}$ at time $k \geq 0$. We use $\mathbf{d}_i$ to denote a subsequence or the collection of elements of $\mathbf{d}$ for $k \in \mathbb{I}_{0:i-1}$. We also define the norm of a sequence (or subsequence) as follows $||\mathbf{d}_i|| = \max_{k \in \mathbb{I}_{0:i-1}} |d(k)|$.

The symbol $x_{i:j}$ denotes a sub-vector of $x \in \mathbb{R}^n$ containing the elements in the index range i to j, with elements at both the end indices included. The symbol $\otimes$ denotes the Kronecker product. We use vec(M) to represent the vector obtained by a vertical concatenation of all the columns in the matrix M. We use I_n to denote an identity matrix of size $n \times n$. The symbol $|M|$ denotes the Frobenius norm of a matrix M. The symbol $\mathrm{diag}(v)$ denotes a diagonal matrix containing the elements of the vector v on

the diagonal. We use the notation $\lceil x \rceil$ to represent the ceiling function that maps the argument x to the smallest integer that is greater than or equal to x.

For a given vector $a \in \mathbb{R}^n$, and lower and upper bounds $[\underline{a}, \overline{a}]$, we define the saturation function as sat$(a, \underline{a}, \overline{a}) = \{a$ if $\underline{a} \leq a \leq \overline{a};\ \underline{a}$ if $a < \underline{a};\ \overline{a}$ if $\overline{a} < a\}$.

Chapter 2

Model-free controller design using Q-learning from noisy data

As discussed in Chapter 1, the standard approach to develop an MPC controller for multivariable processes is to first estimate a dynamic model of the process from data, then solve an optimal control problem in real time for the process operation. Reinforcement learning algorithms (RL) that avoid the model identification step have gained significant attention recently for model-free controller design as an alternative to the model-based approach. The drivers behind this interest are, (i) the impressive results obtained by model-free RL algorithms for playing Go (Silver et al., 2016) and Atari games (Mnih et al., 2015), and performing robotic tasks (Kober et al., 2013; Levine and Koltun, 2014; Levine et al., 2016), and (ii) dynamic model development can sometimes be considered a challenging step in industrial applications, and model-free RL methods may have the potential to avoid that step altogether. Several researchers in the field of process systems engineering are currently investigating the applications of model-free RL algorithms for process control (Shin et al., 2019; Nian et al., 2020; Jiang et al., 2021; Raman et al., 2020; Buşoniu et al., 2018), real time optimization (Powell et al., 2020), and scheduling (Hubbs et al., 2020).

For an industrial deployment of a controller design algorithm, it must be suitable to

estimate a controller from noisy data containing both process and measurement noise. To the best of our knowledge, no previous work in the literature has demonstrated the efficacy of a model-free RL algorithm to give a reasonable controller from data sets that contain both those types of noise sources. Standard model-based methods conveniently handle, and are robust to both the noise sources in the model identification step. Therefore, the model-free RL methods must demonstrate the same capability for them to be competitive for consideration in an industrial deployment.

In this chapter, we develop a novel model-free approach to estimate linear, unconstrained feedback controllers from noisy process data. The developed approach is based on an extension of an available Q-learning algorithm in the literature developed to estimate the linear quadratic regulator (LQR) for linear systems with Gaussian process noise of known covariance. In applications, however, the noise covariance is almost always unknown. So we first extend the algorithm to handle the case of an unknown noise covariance. Then, we use the extended algorithm to estimate a feedback controller for linear systems with both process and measurement noise and only output measurements. We use a linear heating, ventilation, and air-conditioning (HVAC) example to demonstrate the suitability of the proposed approach to estimate a controller from noisy data.

For the above-mentioned developments, we build upon the Q-learning algorithm from Tu and Recht (2018) that estimates the stochastic LQR feedback law from data. The LQR formulation considers linear systems with Gaussian process noise and uses a discounted, infinite horizon, expected value objective. In the literature, there is also a model-free Q-learning algorithm to estimate the nominal LQR feedback law from data (Bradtke et al., 1994). Nominal MPC type controller implementations that do not explicitly account for disturbances in the MPC optimization problem are more prevalent in applications. A practitioner may be encouraged to apply the model-free RL algorithm

for the nominal LQR to estimate a feedback controller in applications. The model-free algorithm available in the literature to estimate the nominal LQR, however, cannot handle noisy training data because it does not model any noise sources in the data generating system. In addition to the main contribution of this chapter on estimating a feedback controller based on a stochastic LQR formulation, we also present a simulation study to examine the performance of the model-free RL algorithm to estimate the nominal LQR. We show that the algorithm to estimate the nominal LQR cannot handle noisy data and is not suitable for industrial implementations.

In the next section, we begin with a discussion on the different types of model-free RL algorithms and previous work on their applications in feedback controller design. Then, in Section 2.2, we discuss the Q-learning algorithm for the nominal LQR and demonstrate that it cannot be used to estimate a feedback controller from noisy data. We discuss the Q-learning algorithm to estimate the stochastic LQR feedback law and our proposed modifications to treat both process and measurement noise and only output measurements in Section 2.3. We also present a maximum likelihood estimation (MLE) method to estimate a linear dynamic model and noise covariances for comparison with the model-free Q-learning approach in Section 2.4. Simulation studies to demonstrate the effectiveness of the proposed model-free controller estimation approach are presented in Section 2.5. We discuss the conclusions of this thesis chapter in Section 2.6.

Portions of the developments and results presented in this chapter appear in Rawlings and Maravelias (2019) and are to appear in Kumar and Rawlings (2023b). The mathematical notation used in this chapter are given in Section 1.3.

2.1 Literature review

Researchers have studied model-free RL algorithms in the machine learning literature for decades using Markov decision processes as the underlying system. Sutton and Barto (2018) give an introduction to the theory and applications of modern RL algorithms. Broadly, the model-free RL algorithms for decision-making and control purposes can be categorized into value function and policy gradient based methods. For the value function methods, some data collected from the system and an appropriate Bellman equation is used to estimate a state or state-control value function. The estimated value function is used to determine the optimal feedback policy or the control input in real time. To determine the optimal feedback policy, an iterative approach typically known as policy iteration is used. The state-action value function is known as the Q-function, and the algorithms that use this function are also called Q-learning (Watkins and Dayan, 1992). For the policy gradient based model-free RL methods (Williams, 1992; Silver et al., 2014; Lillicrap et al., 2015), a feedback policy is first parameterized using a function approximator. Then, an optimization problem is solved to determine the parameters in the policy. This problem is solved using an approximate gradient descent approach.

Several approaches have been proposed in the literature to apply both the value function and policy gradient methods for feedback control problems. Bradtke et al. (1994) proposed a Q-learning based algorithm to estimate the nominal LQR feedback law from process data. This algorithm uses a policy iteration approach based on successive Q-function approximations. The convergence of the iterative approach to the optimal LQR feedback law is also established in that work. Lewis and Vamvoudakis (2011) and Rizvi and Lin (2017) treat the case of output measurements in the Q-learning algorithm to estimate the nominal LQR feedback law. A surrogate state containing a recent

history of past measurements and control inputs is used as the state to treat the output measurement case. All these papers that develop approaches to estimate the nominal LQR feedback law assume that the plant generating the training data is a deterministic linear system. The algorithms in the papers cannot handle noise in the training data (Rawlings and Maravelias, 2019) used to estimate the feedback controller. This limitation means that the algorithms are not suitable for an industrial deployment.

Tu and Recht (2018) applies the least squares policy iteration (LSPI) approach of Lagoudakis and Parr (2003) to estimate the stochastic LQR (Bertsekas, 1995) feedback law from data. The work treats linear systems driven by Gaussian process noise of known covariance. This approach is also not suitable for an industrial implementation, because the both process and measurement noise are present in applications. Additionally, the covariances in the noise statistics are almost always unknown. The recent work Yaghmaie et al. (2022) presents two value function based algorithms to estimate the LQR feedback law. The paper considers linear systems with state measurements and both process and measurement noise. The proposed algorithms are implemented in an on-policy approach (which we discuss subsequently), and can require infeasible amounts of training data for an industrial deployment.

The applications of the policy gradient methods to estimate the LQR feedback law has also been studied in the literature (Fazel et al., 2018; Hambly et al., 2021). These algorithms use an iterative approach to estimate the optimal feedback law. At each iteration of the algorithm, an approximate gradient of the value function is computed based on the data collected by implementing the current feedback law in the iteration to the system. This approach of data collection at each iteration of the algorithm is called an *on-policy* approach, which can be impractical for industrial deployment because a new training data set should be generated for implementing each iteration of the algorithm. In addition, the entire controller estimation and training data set

collection process should be repeated even for changes in the LQR problem tuning parameters. This repetition may be required because the tuning parameters affect the stage costs, and thus also the value functions used to compute the successive feedback laws in the iterative algorithm. We note that the Q-learning algorithms, on the other hand, can also be implemented in an *off-policy* approach (Krauth et al., 2019). Here, the training data for the controller estimation can be generated using a totally independent control input sequence. Hence, an off-policy based model-free RL algorithm can have an improved data efficiency and is more suitable for industrial implementation. Due to this reason, we focus on using Q-learning for model-free controller estimation in this thesis as opposed to policy gradient.

Other researchers have applied *deep* RL algorithms for a nonlinear MPC controller formulation, feedback controller tuning, and merging the RL and MPC approaches. The term "deep" is used when a deep NN is employed to approximate the value function or the feedback policy in the model-free RL algorithm. Spielberg et al. (2019) and Wang et al. (2018) propose to use deep Q-learning for feedback control of nonlinear chemical engineering processes. Yoo et al. (2021) develops a policy gradient algorithm using Monte-Carlo simulations to control batch processes. All these above works, however, have only examined the performance of the algorithm in *noise-free* simulation studies. Morinelly and Ydstie (2016) and Zanon and Gros (2020) develop methods to utilize the advantages of both model-free RL and model-based MPC approaches. The work by Dogru et al. (2022) develops an approach to use model-free RL for the tuning of PI controllers in industrial applications.

2.2 Estimation of nominal linear quadratic regulator

In this section, we discuss the Q-learning algorithm from Bradtke et al. (1994) to estimate the nominal LQR feedback law. Then, we illustrate via a simulation study that the algorithm cannot handle noise in the training data and is therefore not suitable for industrial implementation.

2.2.1 Linear quadratic regulator formulation

The linear quadratic regulator has been studied in the control systems literature for decades. The control objective is to regulate the system to a desired steady state, which is often redefined as the origin after transforming all the variables in deviation from that steady state. The following infinite horizon optimization problem is solved to regulate the system state to the origin

$$\min_{\mathbf{u}} \quad \sum_{k=0}^{\infty} (x(k)'Qx(k) + u(k)'Ru(k)) \tag{2.1}$$

$$x^+ = Ax + Bu \tag{2.2}$$

$$x(0) = x \tag{2.3}$$

in which, A and B are the dynamic model matrices, x is the initial state, Q and R are the penalty matrices used for the stage costs, and the infinite horizon sequence $\mathbf{u}$ is the decision variable. We assume that the pair (A, B) is stabilizable, $R > 0$, $Q \geq 0$, and (A, Q) is detectable for the LQR feedback law to result in a stabilizing controller (Rawlings et al., 2020, Exercise 1.20b). The optimization problem can be solved using a dynamic programming approach (Rawlings et al., 2020, Pages 18-20), which yields

the solution

$$u(k) = K^\star x(k) \tag{2.4}$$

Here, the matrix $K^\star$ is the optimal LQR feedback law that can be obtained by solving the equations

$$\Pi^\star = Q + A'\Pi^\star A - A'\Pi^\star B(B'\Pi^\star B + R)^{-1}B'\Pi^\star A \tag{2.5}$$

$$K^\star = -(B'\Pi^\star B + R)^{-1}B'\Pi^\star A \tag{2.6}$$

The equation (2.5) is a type of the discrete algebraic Riccati equation (DARE). We also define the value function for any suboptimal, stabilizing feedback law K as follows

$$V_K(x) = \sum_{k=0}^{\infty}\Big(x(k)'Qx(k) + (Kx(k))'R(Kx(k))\Big) = x'\Pi x \tag{2.7}$$

in which, Π is the cost-to-go matrix corresponding to the feedback law K. This matrix is obtained by solving the Lyapunov equation

$$\Pi = Q + K'RK + (A + BK)'\Pi(A + BK) \tag{2.8}$$

The standard model-based approach to determine the optimal LQR feedback law $K^\star$ is to first estimate the dynamic model matrices A and B using black-box linear system identification methods. Then, solve the equations (2.6) – (2.5) to determine the feedback law.

2.2.2 Model-free Q-learning for nominal LQR

We now discuss the model-free Q-learning algorithm proposed in Bradtke et al. (1994) to estimate the nominal LQR feedback law $K^\star$ from data. The algorithm is based on a policy iteration approach. Each iteration of the algorithm consists of a policy evaluation step and an improvement step. In the policy evaluation step, a Q-function for the current feedback law in the iteration is estimated. The estimated function is then used in the policy improvement step to determine the feedback law for the next iteration. The algorithm is initialized with a stabilizing feedback law and the iterations are repeated until the convergence of the feedback law. In this subsection, we first derive the Q-function structure for the nominal LQR problem, and subsequently discuss both the policy evaluation and improvement steps.

The Q-function for any state $x \in \mathbb{R}^n$, control input $u \in \mathbb{R}^m$, and a feedback law K_i is defined as: the infinite horizon cost when the linear system ($x^+ = Ax + Bu$) is initialized at the state x, the control input u is implemented at the first time step, and the feedback law K_i is followed thereafter. Based on this definition, the Q-function can be mathematically written as

$$Q_{K_i}(x,u) = x'Qx + u'Ru + \sum_{k=1}^{\infty} \Big(x(k)'Qx(k) + (K_i x(k))'R(K_i x(k)) \Big) \tag{2.9}$$

$$Q_{K_i}(x,u) = x'Qx + u'Ru + V_{K_i}(x^+) \tag{2.10}$$

$$Q_{K_i}(x,u) = x'Qx + u'Ru + (x^+)'\Pi_i x^+ \tag{2.11}$$

We use the subscript i to denote the iteration number in the algorithm. The equation (2.11) is obtained by substituting for the value function corresponding to the feedback law K_i using (2.7) and (2.8). The matrix Π_i denotes the cost-to-go matrix corresponding to the feedback law K_i. The state x^+ can be substituted using the dynamic model

($x^+ = Ax + Bu$) to derive the following structure of the Q-function

$$Q_{K_i}(x,u) = \begin{bmatrix} x \\ u \end{bmatrix}' \begin{bmatrix} Q + A'\Pi_i A & A'\Pi_i B \\ B'\Pi_i A & R + B'\Pi_i B \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \tag{2.12}$$

This Q-function for the nominal LQR is quadratic in the state (x) and control input (u). We subsequently refer to the matrix characterizing the quadratic Q-function as

$$S_i = \begin{bmatrix} S_{ixx} & S_{ixu} \\ S_{iux} & S_{iuu} \end{bmatrix} = \begin{bmatrix} Q + A'\Pi_i A & A'\Pi_i B \\ B'\Pi_i A & R + B'\Pi_i B \end{bmatrix} \tag{2.13}$$

Policy evaluation. Based on the structural knowledge of the Q-function, the goal of the policy evaluation step is to estimate this function from process data. This estimation step utilizes the following Bellman equation for the Q-function

$$Q_{K_i}(x,u) = x'Qx + u'Ru + Q_{K_i}(x^+, K_i x^+) \tag{2.14}$$

We note that this equation can be obtained by observing that $V_{K_i}(x) = Q_{K_i}(x, K_i x)$ and substituting this relation in (2.10). The Q-function function is next parameterized using a linear function approximator and the Bellman equation is used to estimate the unknown parameters in the function from process data. The following Q-function approximation architecture is used

$$Q_{K_i}(x,u) = \left(\begin{bmatrix} x \\ u \end{bmatrix}' \otimes \begin{bmatrix} x \\ u \end{bmatrix}' \right) \text{vec}(S_i) \tag{2.15}$$

$$Q_{K_i}(x,u) = \phi(x,u)'\text{svec}(S_i) \tag{2.16}$$

in which, $\phi(\cdot)$ is a basis function that contains all the unique quadratic terms in the Kronecker product in (2.15). The notation svec(S_i) denotes a vector transformation of the symmetric matrix S_i containing all the diagonal elements and doubled off diagonal elements. This vector characterizes all the unknown parameters in the Q-function. To estimate these parameters, we first collect a training data trajectory by simulating the linear system $x^+ = Ax+Bu$ using some control input sequence. Based on the generated training data and the Bellman equation (2.14), we construct the following equation for each time step k in the available training data

$$\left[\phi(x(k), u(k))' - \phi(x(k+1), K_i x(k+1))'\right] \text{svec}(S_i) = x(k)'Qx(k) + u(k)'Ru(k) \tag{2.17}$$

Further, we define the following two matrices

$$\tilde{A} = \begin{bmatrix} \phi(x(0), u(0))' - \phi(x(1), K_i x(1))' \\ \phi(x(1), u(1))' - \phi(x(2), K_i x(2))' \\ \vdots \\ \phi(x(N_s), u(N_s))' - \phi(x(N_s+1), K_i x(N_s+1))' \end{bmatrix} \tag{2.18}$$

$$\tilde{b} = \begin{bmatrix} x(0)'Qx(0) + u(0)'Ru(0) \\ x(1)'Qx(1) + u(1)'Ru(1) \\ \vdots \\ x(N_s)'Qx(N_s) + u(N_s)'Ru(N_s) \end{bmatrix} \tag{2.19}$$

in which, N_s is the total number of time steps in the data set used to estimate the Q-function. The unknown parameters in the Q-function can be estimated by solving the

least squares problem

$$\tilde{A}\ \text{svec}(S_i) = \tilde{b} \tag{2.20}$$

The estimated parameters are transformed back to the symmetric matrix form to characterize the complete estimate of the quadratic Q-function at the current iteration i.

Policy improvement. Based on the Q-function estimate, the goal of the policy improvement step is to obtain an improved feedback law that has a smaller infinite horizon cost than the current feedback law in the iteration. This improved feedback law is defined as follows

$$K_{i+1}x = \min_u \hat{Q}_{K_i}(x, u),\ \forall x \in \mathbb{R}^n \tag{2.21}$$

This step yields the following feedback law for the next iteration in the algorithm

$$K_{i+1} = -\hat{S}_{iuu}^{-1}\hat{S}_{iux} \tag{2.22}$$

in which, $\hat{S}_i$ is the estimate of the Q-function matrix obtained by solving the least squares problem (2.20).

The above described policy evaluation and improvement steps can be repeated for some specified number of iterations or until the estimated feedback law stops changing significantly. The algorithm proposed in Bradtke et al. (1994) is implemented in an on-policy approach. At each iteration i of the algorithm, a new training data set containing N_s samples is generated by applying control inputs using the relation $u = K_i x + d$ to the plant. Here, d can be sampled from a Gaussian distribution with large enough covariance such that the plant dynamics are sufficiently excited in the training data. The least squares problem (2.20) is solved based on the N_s training data samples generated

for the current iteration of the algorithm. The feedback law is improved based on the solution of the least squares problem, and the data collection and policy improvement processes are repeated online.

We also note that the paper Bradtke et al. (1994) uses a recursive least squares approach to solve the problem (2.20) iteratively as new training data samples become available. In this chapter, we solve the problem with a batch least squares approach after all the training samples for the current iteration have been collected.

2.2.3 Simulation study

We now present a simulation study to compare the model-based and model-free approaches to estimate the nominal LQR feedback law. The main purpose of this study is to elucidate that the model-free algorithm to estimate the nominal LQR feedback law fails with noisy training data.

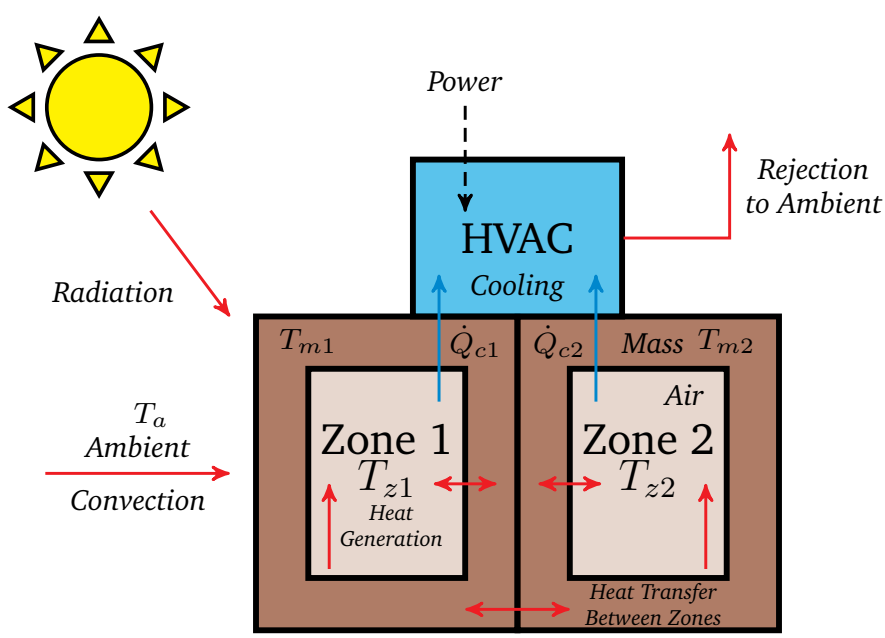

Figure 2.1: A diagram of the HVAC building system considered for simulation studies in this chapter.

$$C_i \frac{dT_i}{dt} = -H_i(T_i - T_a) - \sum_{j \neq i} \beta_{i-j}(T_i - T_j) - \dot{Q}_{ci} + \dot{Q}_{ai} \tag{2.23}$$

$$T_i \in \{T_{z1}, T_{m1}, T_{z2}, T_{m2}\}$$

We consider a linear heating, ventilation, and air-conditioning (HVAC) system shown in Figure 2.1, which is simulated using the ODEs (2.23). The plant has four states $x = \begin{bmatrix} T_{z1}, & T_{m1}, & T_{z2}, & T_{m2} \end{bmatrix}'$, and two control inputs $u = \begin{bmatrix} \dot{Q}_{c1}, & \dot{Q}_{c2} \end{bmatrix}'$. We assume that the disturbances ambient temperature (T_a) and heat loads ($\dot{Q}_{ai}$) remain constant. The plant model is converted into the following linear, discrete time state space form

$$x^+ = Ax + Bu \tag{2.24}$$

The sample time used to obtain this discrete time model is 0.5 hours. The state and control inputs in this model are considered in deviation from a fixed steady state.

To develop a model-based controller, we first estimate the linear model matrices (A, B) using training data collected from the HVAC system. The model estimates are then used to obtain the nominal LQR feedback law by solving the DARE (2.5). We use MATLAB's linear system identification toolbox for this model identification step. To determine the LQR feedback law without estimating a dynamic model, we apply the Q-learning algorithm discussed in the previous subsection.

We first examine the performances of the model-based and model-free algorithms with noise-free training data. Figure 2.2 shows a comparison of the two algorithms for this noise-free case. We show the control inputs (top left), state measurements (top right), and the errors in the estimated feedback laws (center bottom) compared to the optimal LQR feedback law for both the algorithms. We also show the training data

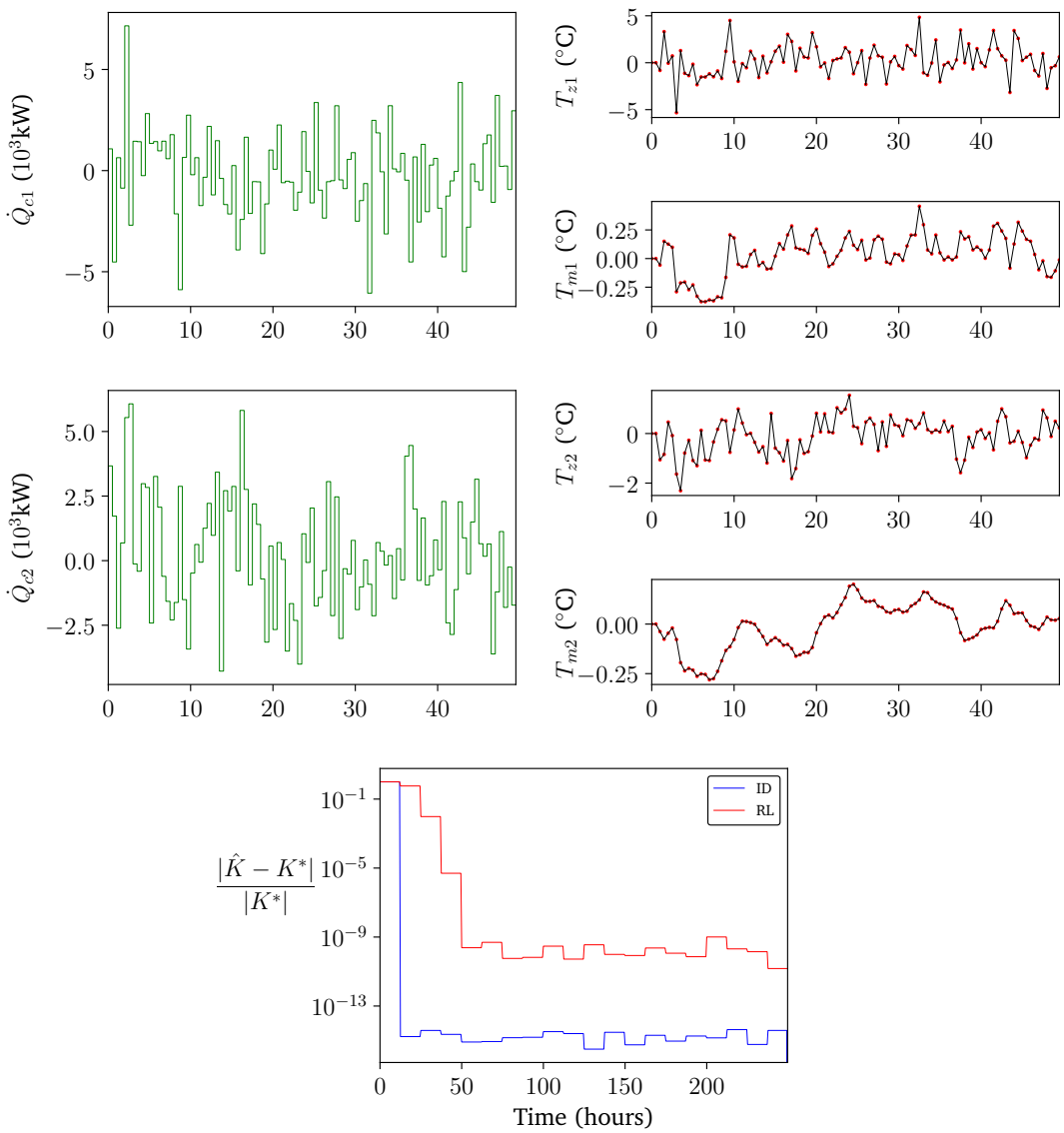

Figure 2.2: Comparison of model-based system identification (ID) and model-free (RL) controller design methods using noise-free training data. Input data (top left), state measurements (top right), model fit using system identification (solid lines in top right), log plot of the error in estimated optimal feedback laws from both the methods (center bottom).

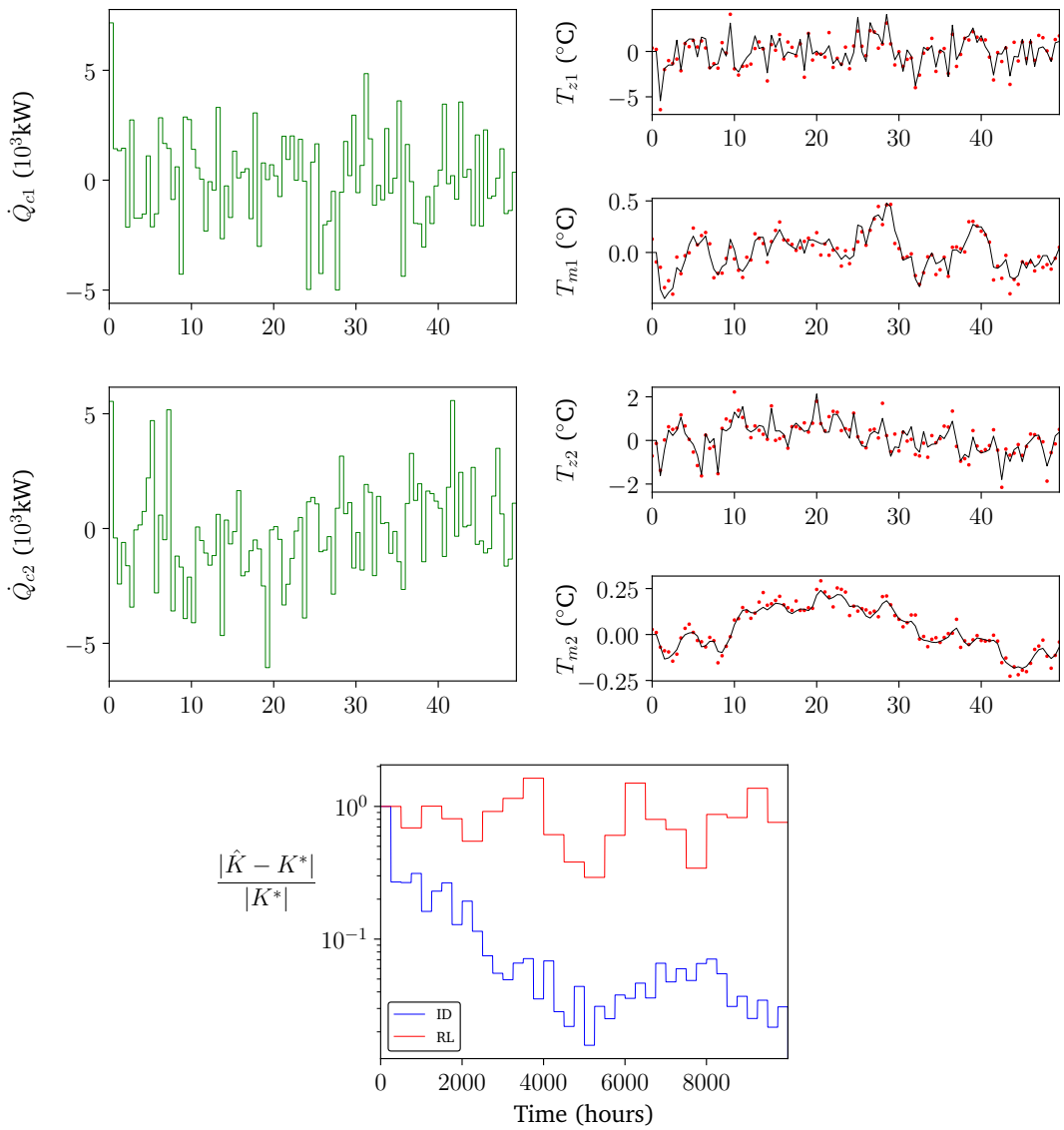

Figure 2.3: Comparison of model-based system identification (ID) and model-free (RL) controller design methods using noisy training data. Input data (top left), state measurements (top right), model fit using system identification (solid lines in top right), log plot of the error in estimated optimals feedback laws from both the methods (center bottom).

fit (solid black lines in top right) obtained by the estimated model in the model-based algorithm in the state measurement plots. The log plots of the feedback law estimation errors (center bottom) show that both the model-based and model-free methods are successful to obtain the optimal LQR feedback law. Both the methods show convergence to the optimal feedback law in a reasonable number of data samples.

Next, we add measurement noise to all the states and reimplement the model-based and model-free controller estimation algorithms. Figure 2.3 shows a comparison of the two algorithms with noisy data. We observe that the model-free RL algorithm does not find the LQR feedback law and has almost 100% estimation errors even with large number of data samples. The model-based algorithm shows convergence to the optimal feedback law upto a reasonable accuracy from a viable number of data samples.

The feedback law estimation error plots in Figure 2.3 suggests that the model-free Q-learning algorithm to estimate the nominal LQR cannot handle noise in the training data. This failure is particularly because the algorithm assumes that a deterministic linear system without any noise term is generating the training data. The goal of the rest of this thesis chapter is to propose a model-free Q-learning approach that does appropriately model the noise to overcome the failure in this section with noisy data.

2.3 Estimation of stochastic linear quadratic regulator

To estimate a model-free controller from noisy data, we use an algorithm that also models the noise in the data generating system. Tu and Recht (2018) developed a Q-learning algorithm to estimate the stochastic LQR feedback law for linear systems with full state measurements driven by Gaussian process noise. The algorithm, however, cannot be applied directly in applications because it assumes that the process noise covariance is known a-priori. We first propose an extension of the algorithm in Tu

and Recht (2018) using a modified Q-function approximation architecture to treat the case of an unknown covariance. Then, we use the extended algorithm to design a feedback controller for the more general class of linear systems with both process and measurement noise and only output measurements.

2.3.1 Linear quadratic regulator formulation

We now discuss the stochastic LQR formulation considered in the rest of this chapter for controller design. The objective function of this LQR problem is the expected value of an infinite horizon sum of discounted, quadratic stage costs. We solve the following optimization problem to determine the LQR feedback law

$$\min_{\mathbf{u}} \quad \mathbb{E}_{\mathbf{w}}\Big[\sum_{k=0}^{\infty}\gamma^k(x(k)'Qx(k)+u(k)'Ru(k)+2x(k)'Mu(k))\Big] \tag{2.25}$$

$$\text{subject to} \quad x^+ = Ax+Bu+w, \quad w\sim N(0,Q_w) \tag{2.26}$$

$$x(0)=x \tag{2.27}$$

in which, x is the state, u is the control input, and w is the process noise. The matrices A, B characterize the linear dynamic model, and Q, R, M are used to penalize the state, control input, and the cross term in the stage costs. The process noise is of zero mean and its covariance is $Q_w \geq 0$. We use the discount factor $\gamma < 1$ in the stage costs to ensure that the objective function remains finite for any stabilizing feedback law. The expectation in the objective function is performed over the process noise sequence $\mathbf{w}$. The decision variable in the optimization problem is the control input sequence $\mathbf{u}$. The problem can be solved using dynamic programming (Bertsekas, 1995, Pages 150-152)

to obtain the solution

$$u^0(k) = K^\star x^0(k) \tag{2.28}$$

in which, $x^0(k)$ and $u^0(k)$ denote the optimal state and control input at the time step k. The matrix $K^\star$ is the optimal stochastic LQR feedback law, which can be obtained by solving the equations

$$\Pi^\star = Q + \gamma A'\Pi^\star A - (M + \gamma A'\Pi^\star B)(\gamma B'\Pi^\star B + R)^{-1}(M' + \gamma B'\Pi^\star A) \tag{2.29}$$

$$K^\star = -(R + \gamma B'\Pi^\star B)^{-1}(\gamma B'\Pi^\star A + M') \tag{2.30}$$

The first equation is also a type of discrete algebraic Riccati equation (DARE), modified for the stochastic LQR problem considered in this section. Further, we also define the value function corresponding to any stabilizing feedback law K as follows

$$V_K(x) = \mathbb{E}_{\mathbf{w}}\Big[\sum_{k=0}^{\infty} \gamma^k \ell(x(k), Kx(k))\Big] \tag{2.31}$$

$$V_K(x) = x'\Pi x + \eta\,\mathrm{tr}(\Pi Q_w) \tag{2.32}$$

Here, $\ell(\cdot)$ is the stage cost (without the discount factor) in the LQR problem (2.25), and $\eta = \gamma/(1-\gamma)$. The value function denotes the expected value of the infinite horizon objective when control inputs according to the feedback law K, starting from the state x are applied to the linear system $x^+ = Ax + Bu + w$. We note that the value function of this stochastic LQR formulation has an additional trace term, which corresponds to the contribution of the process noise. The matrix Π is called the cost-to-go matrix that

can be obtained by solving the Lyapunov equation

$$\Pi = Q + K'RK + MK + K'M' + \gamma(A + BK)'\Pi(A + BK) \tag{2.33}$$

The cross term in the LQR problem is used particularly to implement a rate-of-change penalty on the control input. For example, we may be interested in implementing a rate-of-change penalty on the control input using a matrix S_R as follows: $(u(k) - u(k-1))'S_R(u(k) - u(k-1))$. To include this penalty, we augment the system state $x(k)$ with the previous control input $u(k-1)$. The dynamic model matrices and the control problem penalty matrices are derived for the augmented state such that the final LQR problem is of the type (2.25) (Rawlings et al., 2020, Exercise 1.25). We solve the DARE equation for the problem with the augmented state to determine the optimal LQR feedback law.

2.3.2 Least squares policy iteration

Next, we outline the least squares policy iteration (LSPI) algorithm proposed in Tu and Recht (2018) to estimate the stochastic LQR feedback law $K^\star$ in (2.30). This algorithm is also implemented in a policy iteration framework, similar to the algorithm discussed previously to estimate the nominal LQR feedback law. We derive the Q-function equations below and describe the LSPI algorithm for the stochastic LQR described in the previous subsection. We note that the Q-function equations derived here differ slightly from Tu and Recht (2018) because we consider also the cross term in the LQR problem and any general positive semidefinite process noise covariance.

The Q-function for the stochastic LQR formulation for any state x, control input u,

and feedback law K_i is defined as follows

$$Q_{K_i}(x,u) = \ \ell(x,u) + \mathbb{E}_{\mathbf{w}}\Big[\sum_{k=1}^{\infty}\gamma^k \ell(x(k), K_i x(k))\Big] \tag{2.34}$$

which is equivalent to the expected infinite horizon cost obtained for the linear system $x^+ = Ax + Bu + w$ when starting from the state x, applying the control input u at the first time step, and following the feedback law K_i thereafter. The subscript i is used to denote the current iteration number in the LSPI algorithm. The second term in (2.34) can also be written as the discount factor times the expectation over the value function for the state x^+ and feedback law K_i as follows

$$Q_{K_i}(x,u) = \ \ell(x,u) + \gamma \mathbb{E}_w[V_{K_i}(x^+)] \tag{2.35}$$

The expectation in this equation is now performed over the process noise at only the first time step. We substitute for the value function corresponding to the feedback law K_i using (2.32). And after evaluating the expectation over the process noise w, we obtain the following structure of the Q-function

$$Q_{K_i}(x,u) = \begin{bmatrix} x \\ u \end{bmatrix}' \begin{bmatrix} Q + \gamma A'\Pi_i A & \gamma A'\Pi_i B + M \\ \gamma B'\Pi_i A + M' & R + \gamma B'\Pi_i B \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \eta \operatorname{tr}(\Pi_i Q_w) \tag{2.36}$$

in which, Π_i is the cost-to-go matrix corresponding to the feedback law K_i. This Q-function also quadratic in the state (x) and control input (u). We note that an additional trace term appears in this equation compared to the Q-function of the nominal LQR in (2.12). We denote the matrix characterizing the quadratic term in the

Q-function as follows

$$S_i = \begin{bmatrix} S_{ixx} & S_{ixu} \\ S_{iux} & S_{iuu} \end{bmatrix} = \begin{bmatrix} Q + \gamma A'\Pi_i A & \gamma A'\Pi_i B + M \\ \gamma B'\Pi_i A + M' & R + \gamma B'\Pi_i B \end{bmatrix} \tag{2.37}$$

In addition, we also note the following relation between the cost-to-go matrix Π_i and the Q-function matrix S_i

$$\Pi_i = \begin{bmatrix} I \\ K_i \end{bmatrix}' S_i \begin{bmatrix} I \\ K_i \end{bmatrix} \tag{2.38}$$

This relation can be verified by substituting for the matrix S_i from (2.37) and using the Lyapunov equation (2.33) for the feedback law and cost-to-go matrix pair (K_i, Π_i).

Policy evaluation. The goal of the policy evaluation step is to utilize the structural knowledge of the Q-function and a Bellman equation to estimate that function from data. We use the following Bellman equation

$$Q_{K_i}(x, u) = \ \ell(x, u) + \gamma \mathbb{E}_w[Q_{K_i}(x^+, K_i x^+)] \tag{2.39}$$

This equation can be obtained by noting that $V_{K_i}(x) = Q_{K_i}(x, K_i x)$, and substituting this relation in (2.35). The Q-function is first parameterized using a linear function approximation architecture. Then, the unknown parameters in the architecture are estimated by solving a least squares problem formulated based on the Bellman equation. We use the following linear function approximation architecture to estimate the

Q-function

$$Q_{K_i}(x,u) = \left[\begin{bmatrix} x \\ u \end{bmatrix}' \otimes \begin{bmatrix} x \\ u \end{bmatrix}' + \eta \text{vec}(Q_w)' \Big(\begin{bmatrix} I \\ K_i \end{bmatrix}' \otimes \begin{bmatrix} I \\ K_i \end{bmatrix}' \Big) \right] \text{vec}(S_i) \tag{2.40}$$

$$Q_{K_i}(x,u) = \phi(x,u)' \text{svec}(S_i) \tag{2.41}$$

The first equation is obtained by expanding the quadratic and trace terms in (2.36). Equation (2.38) is used to expand the trace term. The basis function $\phi(x,u)$ contains all the unique quadratic terms of the row vector in (2.40) that has the Kronecker products and the process noise covariance. The unknown parameters in the function approximation architecture are denoted by svec(S_i), which contains all the diagonal elements and doubled off diagonal elements of the symmetric Q-function matrix S_i.

To estimate the Q-function from data, we collect a training data trajectory by simulating the linear system $x^+ = Ax + Bu + w$ using a specified control input and Gaussian process noise sequences $\mathbf{u}$ and $\mathbf{w}$. The data generation step gives a state sequence $\mathbf{x}$. For the LSPI algorithm to estimate the stochastic LQR, we generate the training data using an arbitrary control input sequence that is independent of the feedback laws generated for the iterations of the algorithm. Additionally, the entire training data set is used to estimate the Q-functions in all the iterations of the algorithm.

We use the linear function approximation architecture in (2.41) and the generated training data to construct the following two matrices

$$\tilde{A} = \frac{1}{N_t} \sum_{k=0}^{N_t} \phi(x(k),u(k)) \Big(\phi(x(k),u(k))' - \gamma \phi(x(k+1), K_i x(k+1))' \Big) \tag{2.42}$$

$$\tilde{b} = \frac{1}{N_t} \sum_{k=0}^{N_t} \phi(x(k),u(k)) \ell(x(k),u(k)) \tag{2.43}$$

in which, N_t is the total number of time steps in the training data trajectory, and $x(k)$ and $u(k)$ are the state and control input at the time step k in the training data. These two matrices are used to solve the following least squares problem to estimate the unknown parameters in Q-function

$$\tilde{A}\ \text{svec}(S_i) = \tilde{b} \tag{2.44}$$

The parameter estimate obtained by solving this problem is transformed back to the symmetric matrix form to characterize the full Q-function estimate at the current iteration i. We refer the reader to Lagoudakis and Parr (2003) for a detailed derivation of the regressor ($\tilde{A}$) and target ($\tilde{b}$) matrices for this Q-learning least squares problem. The matrices are derived by enforcing the Q-function approximation to be a fixed point of the Bellman equation (2.39) in the space of representable functions by the linear function approximation architecture (2.41). The Q-function parameters estimated using the above least squares problem converge to the true parameters for large number of training data samples. This convergence property and the treatment of the noise using the stochastic LQR formulation is an initial step towards developing an industrially implementable algorithm. We show in Subsections 2.3.3 – 2.3.4 how to further modify this algorithm so that it is suitable for use in applications.

Policy improvement. Similar to the policy evaluation step in the previous algorithm for the nominal LQR, we use the estimated Q-function to obtain an improved feedback law as follows

$$K_{i+1}x = \min_u \hat{Q}_{K_i}(x,u),\ \forall x \in \mathbb{R}^n \tag{2.45}$$

$$K_{i+1} = -\hat{S}_{iuu}^{-1}\hat{S}_{iux} \tag{2.46}$$

in which, $\hat{S}_i$ is the estimate of the Q-function matrix obtained by solving the least squares problem (2.44).

The two policy evaluation and improvement steps can be implemented for some specified large number of iterations or until the feedback law stops changing noticeably. Tu and Recht (2018) generate the training data for the LSPI algorithm by simulating the linear system in an episodic approach. The initial states in the episodes are sampled from some specified Gaussian distribution. We note that “resetting" the initial plant state in this approach is typically not possible for industrial processes. We also emphasize that the algorithm does not require the training data to be generated using an episodic approach. The data can also be collected from the plant in the form of a single of multiple trajectories without any control on the initial states.

2.3.3 Q-function approximation under unknown process noise covariance

The LSPI algorithm for the stochastic LQR discussed in the previous subsection requires a-priori knowledge of the process noise covariance Q_w, which is used in the linear function approximation architecture (2.41) to formulate the Q-learning least squares problem. In applications, however, the noise covariance is almost always unknown. And the methods available in the literature to estimate noise covariances, such as the autocovariance least squares (Odelson et al., 2003), assume that a linear dynamic model is available. Therefore, the algorithm in Tu and Recht (2018) cannot be directly applied in process control applications. We now propose an approach to handle an unknown noise covariance in the LSPI algorithm for linear systems of the type $x^+ = Ax + Bu + w$, driven by Gaussian process noise and full state measurements.

To estimate the Q-function for the stochastic LQR under an unknown process noise

covariance, we propose to use the following modified linear approximation architecture

$$Q_{K_i}(x,u) = \left[\begin{bmatrix} x \\ u \end{bmatrix}' \otimes \begin{bmatrix} x \\ u \end{bmatrix}', 1 \right] \begin{bmatrix} \text{vec}(S_i) \\ \eta \operatorname{tr}(\Pi_i Q_w) \end{bmatrix} \tag{2.47}$$

$$Q_{K_i}(x,u) = \phi(x,u)' \begin{bmatrix} \text{svec}(S_i) \\ \eta \operatorname{tr}(\Pi_i Q_w) \end{bmatrix} \tag{2.48}$$

in which, $\phi(x,u)$ is a basis function that contains all the unique quadratic terms of the Kronecker product in (2.47), and concatenated with the scalar unity. An LSPI algorithm using this modified Q-function approximation architecture can be implemented in a similar approach as the algorithm discussed in the previous subsection. The data generation and policy improvement steps remain the same, and we only change the basis function to the one in (2.48) to construct the regressor ($\tilde{A}$) and target ($\tilde{b}$) matrices for the least squares problem.

In the modified least squares problem for this unknown process noise covariance case, we estimate another parameter $\eta \operatorname{tr}(\Pi_i Q_w)$ in addition to the parameters in the Q-function matrix S_i. The additional parameter corresponds to the contribution of the process noise to the Q-function. The parameter is not used in the policy improvement step to obtain the feedback law for the next iteration. However, an estimation of this parameter is required to systematically estimate the Q-function matrix S_i. We refer to the noise parameter as β in the rest of this chapter.

2.3.4 Extension to output measurements and process and measurement noises

Both the algorithms discussed in the previous subsections assume that the linear system is driven by only process noise, and full state measurements are available. In process control applications, the sensor noise also contributes significantly to the measurements. In addition, not all but only some subset or linear combination of the states are measured. For these reasons, we aim to develop a feedback controller for the following class of linear systems

$$x^+ = Ax + Bu + w, \tag{2.49}$$

$$y = Cx + v, \tag{2.50}$$

$$w \sim N(0, Q_w), \quad v \sim N(0, R_v) \tag{2.51}$$

in which, $y \in \mathbb{R}^p$ denote the measurements, $C \in \mathbb{R}^{p\times n}$ is the matrix that characterizes the measurements, and v is the measurement noise of zero mean and covariance R_v. To develop a feedback controller for this output measurement case, we use a vector containing a recent history of past measurements and control inputs as a surrogate state. Then, we apply the LSPI algorithm discussed in Subsection 2.3.3 that does not assume any value of the noise covariance. The surrogate state is defined for every time step k as follows

$$z(k) = \Big[y(k-N_p)', ..., y(k-1)', u(k-N_p)', ..., u(k-1)'\Big]' \tag{2.52}$$

in which, N_p is a parameter that characterizes the number of past measurements and control inputs used to construct the state. This method of using a recent history of mea-

surements and control inputs to replace the state has been used in both the system identification (Ho and Kalman, 1966; Qin, 2006) and Q-learning (Lewis and Vamvoudakis, 2011; Rizvi and Lin, 2017) research literatures for many years for model and controller identification purposes.

The parameter N_p used to construct the surrogate state characterizes the assumed model order of the plant. That parameter should be chosen large enough such that the state contains enough information about the plant dynamics. But small enough so that the Q-function parameter estimates do not become sensitive to noise in the training data. We choose the parameter by examining the condition number of the data matrix that is constructed using samples of the surrogate state. We describe this approach in detail in Appendix 2.6.

For feedback controller design using output measurements, we consider an output tracking penalty $y(k)'Q_y y(k)$ in the stage cost of the LQR problem. The training data is generated from a linear system of the type in equations (2.49) – (2.50). We use observations of the surrogate state $z(k)$ and stage cost values with the output tracking penalty to construct the matrices $\tilde{A}$ and $\tilde{b}$ for the Q-learning least squares problem. Both the policy evaluation and improvement steps are implemented similarly as the algorithms discussed in the previous two subsections. The algorithm yields a feedback controller that uses the surrogate z to compute the control input u during the online closed-loop implementation.

We note that a linear system with the surrogate state does not evolve similarly as the assumed model in the LSPI algorithm. The linear model assumed in the algorithm is of the type $x^+ = Ax + Bu + w$. So the use of the surrogate state results in both a dynamic and noise model *mismatch* between the plant and model assumed in the LSPI algorithm. However, we show in the simulation studies in Section 2.5 that this model mismatch is not crucial. The proposed approach of using a surrogate state in the

LSPI algorithm for the output measurement case is still suitable to yield a reasonable feedback controller.

All the LSPI algorithms discussed in this chapter estimate only a regulator and not an estimator that can be used for noise filtering during the online closed-loop implementation. A notable advantage of the model-based controller design approach is that since it also estimates a dynamic model (possibly also noise covariances), a Kalman filter can be constructed using the estimated model and noise covariances. The filter can then be used to perform optimal state estimation during the online controller estimation. The LSPI algorithms lose this ability of using a Kalman filter for state estimation. So for the online closed-loop implementation in the simulation studies with the model-free controllers, we implement a heuristic noise filtering approach. We use an exponential moving average of the measurements during the online implementation. The filtered estimates of the measurements are obtained using

$$\hat{y}^+ = \alpha\hat{y} + (1-\alpha)y^+ \tag{2.53}$$

in which, $\hat{y}$ and $\hat{y}^+$ denote the filtered measurements at the current and next time step respectively, y^+ denotes the measurements at the next time step, and α is a parameter in the heuristic filter. This parameter controls the sensitivity of the filtered estimates to new measurements. During the online implementation, the filtered measurements obtained with this approach are used to construct the surrogate state z, which is then finally used to compute the control input. In the simulation studies, we also consider the case of full state measurements when the plant as a linear system of the type $x^+ = Ax + Bu + w$. For this case, the heuristic noise filtering approach is applied over the state measurements. The filtered estimates are then used to compute the control inputs during the online implementation.

2.4 Maximum likelihood estimation of linear dynamic model and noise covariances

For comparative simulation studies in this chapter, we wish to compare the performance of the model-free LSPI algorithms developed above with a model-based method that also systematically treats the noise in the training data. We now discuss a maximum likelihood estimation (MLE) method to estimate a linear dynamic model and noise covariances from plant data. We choose this method because it also estimates the noise covariances in addition to the dynamic model. The noise covariances are used to systematically develop a Kalman filter to perform state estimation during the closed-loop implementation. The MLE algorithm is discussed next. The algorithm is similar to the Larimore's subspace method (Larimore, 1990; Qin, 2006), and was proposed recently in Kuntz and Rawlings (2022).

We assume that a linear dynamic model of the type (2.49) – (2.50) is used to generate the training data. We construct the following equation for each time step k in the training data

$$s(k) = \Theta t(k) + e(k), \quad e(k) \sim N(0, S_{wv}) \tag{2.54}$$

in which, $s = \begin{bmatrix} x^+ \\ y \end{bmatrix}$, $t = \begin{bmatrix} x \\ u \end{bmatrix}$, $e = \begin{bmatrix} w \\ v \end{bmatrix}$, and $\Theta = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$. The dynamic model Θ and noise covariances S_{wv} are assumed to be unknown. The MLE estimates can be determined by maximizing the probability of the observed measurements (Graham and Rawlings, 2022, Chapter 4. 7) over the unknown parameters. The estimates can then

be computed using the equations

$$\hat{\Theta} = \Big(\sum_{k=0}^{N_s-1} s(k)t(k)' \Big) \Big(\sum_{k=0}^{N_s-1} t(k)t(k)' \Big)^{-1} \tag{2.55}$$

$$\hat{S}_{wv} = \frac{1}{N_s - (n+m)} \sum_{k=0}^{N_s-1} (s(k) - \hat{\Theta}t(k))(s(k) - \hat{\Theta}t(k))' \tag{2.56}$$

in which, N_s is the number of training data samples. The state x of the linear system (2.49) – (2.50) is not measured, so we use the surrogate state defined in (2.52) to construct samples of the vectors s and t for the above equations. This approach is similar to the method discussed previously for the output measurement case in the LSPI algorithm. The parameter N_p to construct the surrogate state can be chosen similarly as the approach used for the LSPI algorithm, which is discussed in Appendix 2.6. For the simulation case in which we assume full state measurements and the plant is a linear system of the type $x^+ = Ax + Bu + w$, we use $s = x^+$ and $t = x$. The MLE algorithm for this case is used to estimate only the dynamic model matrices A and B, and the process noise covariance Q_w.

2.5 Simulation studies

In this section, we demonstrate the effectiveness of the proposed LSPI algorithms to successfully estimate reasonable feedback controllers from noisy process data. We use the same linear heating, ventilation, and air-conditioning (HVAC) example considered for the previous simulation study to estimate the nominal LQR feedback law.

We present simulation studies for two cases of the plant used for the data generation and closed-loop simulations. In the first case, we assume that the plant is a linear system of the type $x^+ = Ax + Bu + w$ driven by only Gaussian process noise and all

Parameter	Value	Parameter	Value
H_1/C_{z1}	6×10^{-4} sec^{-1}	H_2/C_{z2}	1×10^{-3} sec^{-1}
β_{z1-m2}/C_{z1}	2×10^{-4} sec^{-1}	β_{z2-m2}/C_{z2}	5×10^{-4} sec^{-1}
β_{z1-m2}/C_{m1}	3×10^{-4} sec^{-1}	β_{m1-m2}/C_{m2}	2×10^{-4} sec^{-1}
β_{m1-m2}/C_{m1}	6×10^{-4} sec^{-1}	β_{z2-m2}/C_{m2}	4×10^{-4} sec^{-1}
$1/C_{z1}$	2.5×10^{-4} °C/kJ	$1/C_{z2}$	2×10^{-4} °C/kJ
$u_s = [50\text{kW},\ 40\text{kW}]'$			
$p_s = [60\text{kW},\ 40\text{kW},\ 22°\text{C}]'$			
$x_s = [25.53°\text{C},\ 23.65°\text{C},\ 22.23°\text{C},\ 22.70°\text{C}]'$			

Table 2.1: Parameters used to simulate the ODEs (2.23) of the HVAC building system, and the steady state used to obtain deviation variables. The interaction coefficients (β_{ij}) for the combination of temperature states not shown in this Table are zero.

the states are measured. This case is considered so that the performances of the LSPI algorithms can be examined in a scenario when there is no mismatch between the plant and the model assumed in the algorithms. For the second simulation case, we assume that the HVAC plant is a linear system of the type (2.49) – (2.50) with both process and measurement noise and only output measurements. For both the simulation cases, we examine (i) the data requirements of the algorithms to obtain a reliable controller, and (ii) the closed-loop performance compared to a perfect model-based controller that uses the true plant model and noise covariances.

The HVAC model parameters used for the simulation study in this section are given in Table 2.1. Similar to the previous example, we assume that the disturbances ambient temperature (T_a) and heat loads ($\dot{Q}_{a1}, \dot{Q}_{a2}$) to the HVAC system remain constant during the training data generation and closed-loop simulations. We convert the plant model to the linear discrete time state space form for the simulation study. The states and control inputs in the discrete time linear model are considered in deviation from the steady state shown in Table 2.1. The sample time used to convert the ODEs to the discrete time model and to obtain measurements from the HVAC system is 1 minutes. For

the full state measurement case, we measure all the zone and mass temperature states $x = [T_{z1}, T_{m1}, T_{z2}, T_{m2}]'$. While for the output measurement case, the zone temperatures are the only measurements $y = [T_{z1}, T_{z2}]'$.

2.5.1 State measurements with process noise

First, we examine the performances of the LSPI and MLE algorithms to obtain a reliable controller for the case in which the HVAC plant is driven by only Gaussian process noise and all the states are measured. We generate a total of 60 hours (3600 samples) of training data by simulating the plant using Gaussian random control input and process noise sequences. The HVAC plant is initialized at the origin for this training data generation. Both the random sequences are of zero mean, and the covariances are given in Table 2.2. The first 2 hours from the entire training data set is shown in Figure 2.4.

After the training data generation, we implement the following three algorithms to estimate a feedback controller

1. SYSID: The MLE algorithm discussed in the previous section is implemented to estimate the linear dynamic model matrices (A, B) and the process noise covariance Q_w. The estimates are then used to develop an LQR feedback law for regulation and a Kalman filter for state estimation.

2. LSPI-KQW: The LSPI algorithm discussed in Subsection 2.3.2 that assumes a known value of the process noise covariance is implemented to estimate the LQR feedback law. We assume that the noise covariance used in the plant for the data generation is known exactly for this algorithm.

3. LSPI-UQW: We implement the LSPI algorithm in Subsection 2.3.3 that does not

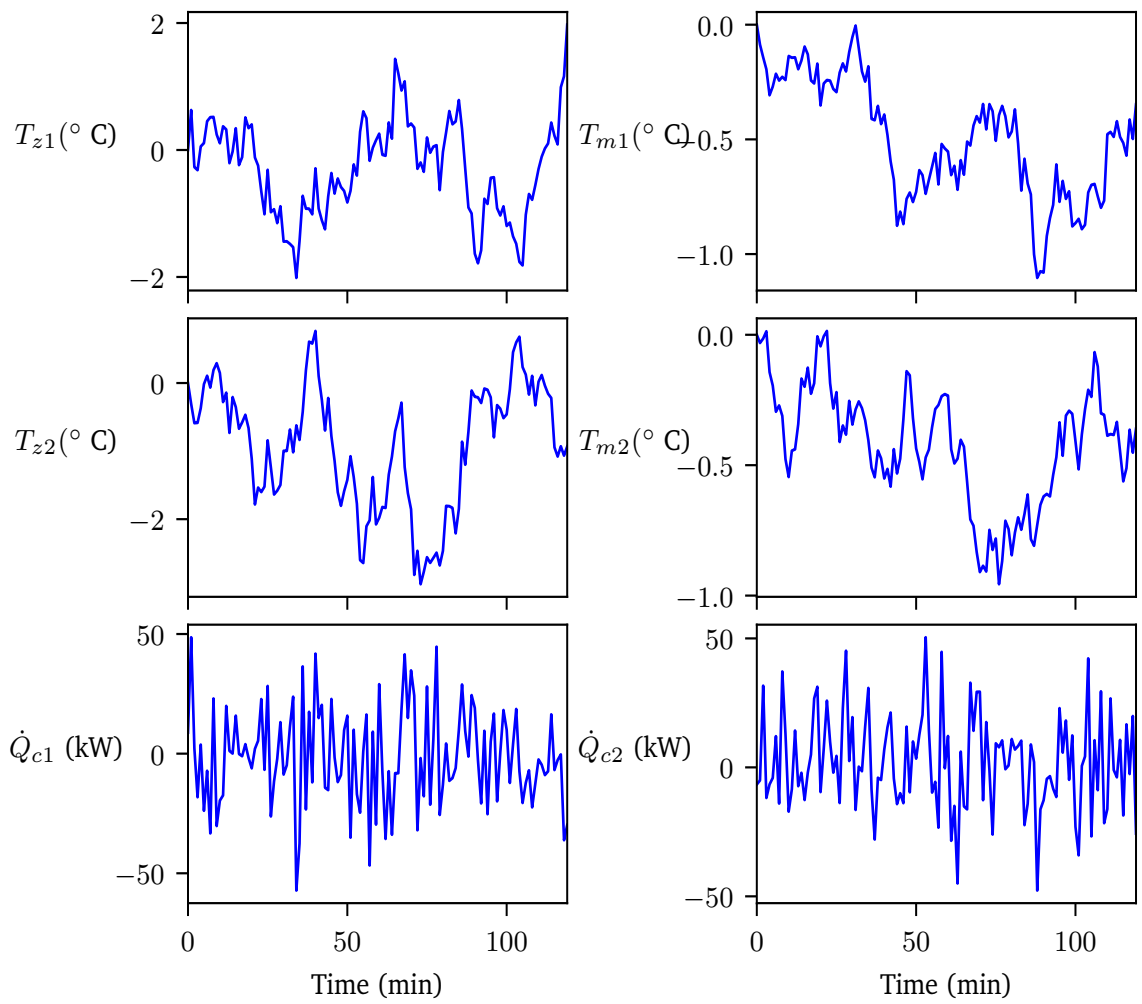

Figure 2.4: Sample training data set used for the case of full state measurements with process noise.

assume a known value of the process noise covariance, and also estimates the contribution of the noise (β) to the Q-function.

A comparison between the LSPI-KQW and LSPI-UQW algorithms elucidates the achievable performance with an LSPI algorithm that does not utilize any knowledge of process noise statistics.

The three algorithms discussed above are implemented with varying amounts of

Parameter	Value
State measurement with process noise	
Σ_u	$400I_2$
Q_w	diag([0.08, 0.01, 0.09, 0.01]')
Q	$10^3 Q_s$
R	$10^2 R_s$
S_R	$10^2 R_s$
γ	0.98
Output measurement with process and measurement noises	
Σ_u	$625I_2$
Q_w	diag([0.08, 0.01, 0.09, 0.01]')
R_v	diag([0.3, 0.2]')
Q_y	$10^3 Q_{ys}$
R	$10^2 R_s$
S_R	$10^2 R_s$
γ	0.98

Table 2.2: Covariances used to generate the training data, and the tuning parameters for the LQR used for the simulation studies in Section 2.5. The covariance used to generate the control input sequence is Σ_u. The penalty matrices Q_s, Q_{ys}, and R_s are diagonal and contain the inverse of the squares of the state, measurement, and control input at the steady state shown in Table 2.1.

training data starting from 3 to 60 hours, in increments of 3 hours. The tuning parameters used for the LQR problem are provided in Table 2.2. A rate-of-change penalty is also considered in the LQR problem, so we use samples of the augmented state $\tilde{x} = \begin{bmatrix} x', & u'_{-1} \end{bmatrix}'$ in the two LSPI algorithms. The optimal LQR feedback law uses this state to compute the control input. The feedback law is of size 2×6, and the Q-function matrix S required to be estimated in the two LSPI algorithms is of dimensions 8×8. We use a matrix of zeros as the initial feedback law for the LSPI algorithms. And the two policy evaluation and improvement steps in the algorithms are executed for a total of 15 iterations.

The quality of the optimal Q-function and the LQR feedback law obtained with all

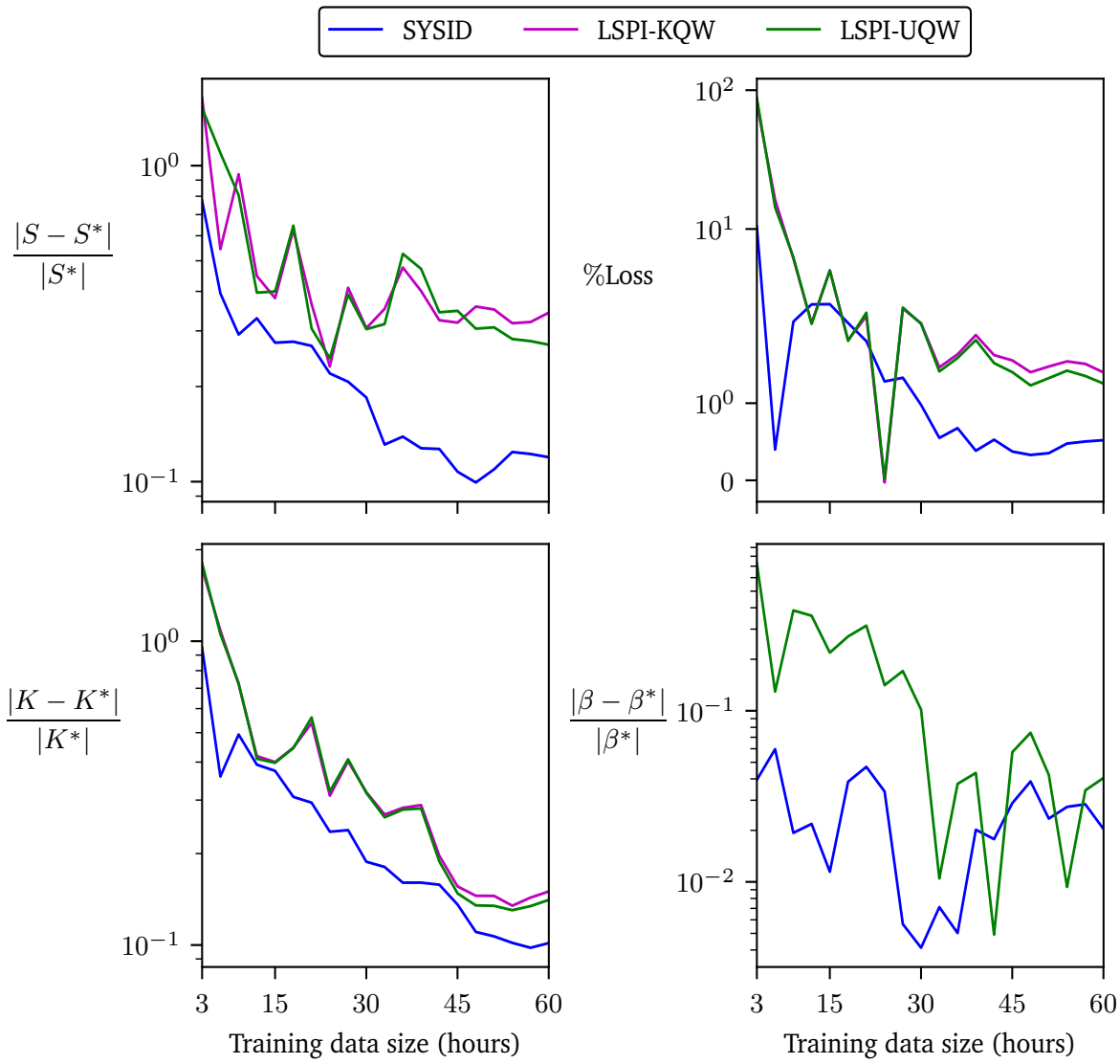

Figure 2.5: The errors in the Q-function and LQR feedback law estimates obtained using the SYSID, LSPI-KQW, and LSPI-UQW algorithms with varying amounts of training data for the case of full state measurements and process noise. We also show the variation in the % closed-loop performance loss compared to the perfect model-based controller that uses the true plant model and noise covariance.

the algorithms for varying amounts of training data is examined first. To gauge the quality of the Q-function estimates, we compare the estimated S matrix and the noise term β with the corresponding true quantities obtained using the actual plant model. The SYSID algorithm does not directly estimate these two quantities, so we use the estimated model and noise covariance to construct the S matrix and noise term β. In Figure 2.5, we show the variations in the estimation errors of the S matrix, noise term β, and the LQR feedback law $K^\star$. We observe that for a small amount of 3 hours of training data, the estimation errors in the three quantities obtained with the two LSPI algorithms are unacceptably large, and around $60 - 110\%$. The quality of the estimates obtained with the LSPI algorithms improve with an increase in the training data. When using the full 60 hours of training data, the LQR feedback law estimation errors with the two LSPI algorithms is around $10 - 20\%$. The SYSID algorithm consistently provides similar or better estimates of all the three quantities for varying amounts of training data. The performances of the LSPI-KQW and LSPI-UQW are similar, which illustrates that the approach proposed in this chapter of using a modified Q-function approximation architecture to treat an unknown noise covariance is effective.

Next, we examine the closed-loop performances of the controllers estimated with the LSPI and MLE algorithms compared to a perfect model-based controller that uses the true dynamic model and noise covariances. For these comparisons, we perform multiple closed-loop simulations with each estimated controller starting from some different initial states of the HVAC plant. All these initial states are sampled randomly from a uniform distribution in the range -10 to 10°C. We use (2.53) for noise filtering during the closed-loop implementation using the model-free controllers, with $\alpha = 0.2$. Based on all the closed-loop simulations, we compute the following performance metric

to gauge the overall performance of an estimated controller

$$\Lambda_T^C = \frac{1}{N_{tr} N_t} \sum_{j=1}^{N_{tr}} \sum_{k=0}^{N_t} \ell(x_j(k), u_j(k), \Delta u_j(k)) \tag{2.57}$$

in which, we use the subscript j to denote the trajectory number of a closed-loop simulation, N_{tr} is the total number of simulations, and N_t is the number of time steps in each closed-loop simulation. We perform $N_{tr} = 500$ simulations, and each for $N_t = 240$ (4 hours) number of time steps. We choose this length of each simulation so that enough closed-loop transient and steady-state data is captured in the performance metrics. For each estimated controller, we also compute the following metric that quantifies the performance loss compared to the perfect model-based controller

$$\% \text{ Performance Loss} = 100(\lambda_T^C - \Lambda_T^P)/\Lambda_T^P \tag{2.58}$$

in which, Λ_T^P denotes the overall performance of the perfect model-based controller obtained using (2.57). In Figure 2.5 (top-right), we show the variation in the above performance loss metric for the estimated controllers using the three algorithms for varying amounts of training data. We notice that after approximately 15 hours of data, the performances of the controllers estimated using the two LSPI algorithms are close to the perfect model-based controller, with the loss metrics less than 5%. The performances further improve with an increase in the training data.

In Table 2.3, we summarize the loss metrics obtained by the estimated controllers using the three algorithms when using the full 60 hours of training data. The controller obtained with the SYSID algorithm provides the best performance with the loss metric of only 0.52%. The controllers obtained with the two LSPI algorithms also provide good performance with the loss metrics of 1.4% and 1.25%. The higher performance of

Controller	% Performance Loss	Controller estimation time (seconds)
State measurement with process noise		
SYSID	0.52	0.05
LSPI-KQW	1.4	12.02
LSPI-UQW	1.25	6.97
Output measurement with process and measurement noises		
SYSID	2.68	0.02
LSPI	3.55	3.2

Table 2.3: Summary of the simulation study performed to compare the performances of the SYSID and LSPI algorithms.

the model-based controller obtained using the SYSID algorithm is because it estimates a more accurate LQR feedback law as observed in Figure 2.5. In addition, we also perform state estimation systematically using a Kalman filter during the closed-loop implementation of the model-based controller obtained using the SYSID algorithm.

We also summarize the time required by the three algorithms for the controller estimation in Table 2.3. The estimation times are shown for the case when the entire 60 hours of training data is used by the algorithms. We notice that the SYSID algorithm is the fastest and only requires 0.05 seconds. The two LSPI algorithms require around 7 and 12 seconds for the controller estimation. The LSPI algorithms solve multiple least squares problems in an iterative approach, which leads to the larger time required for estimating a feedback controller. We notice that the LSPI-KQW uses slightly more time than the LSPI-UQW, which is due to the fact that the Q-function approximation architecture used for the LSPI-KQW has more Kronecker products that need to be evaluated when constructing the least squares problem. Nonetheless, all the algorithms are suitable for controller estimation in applications.

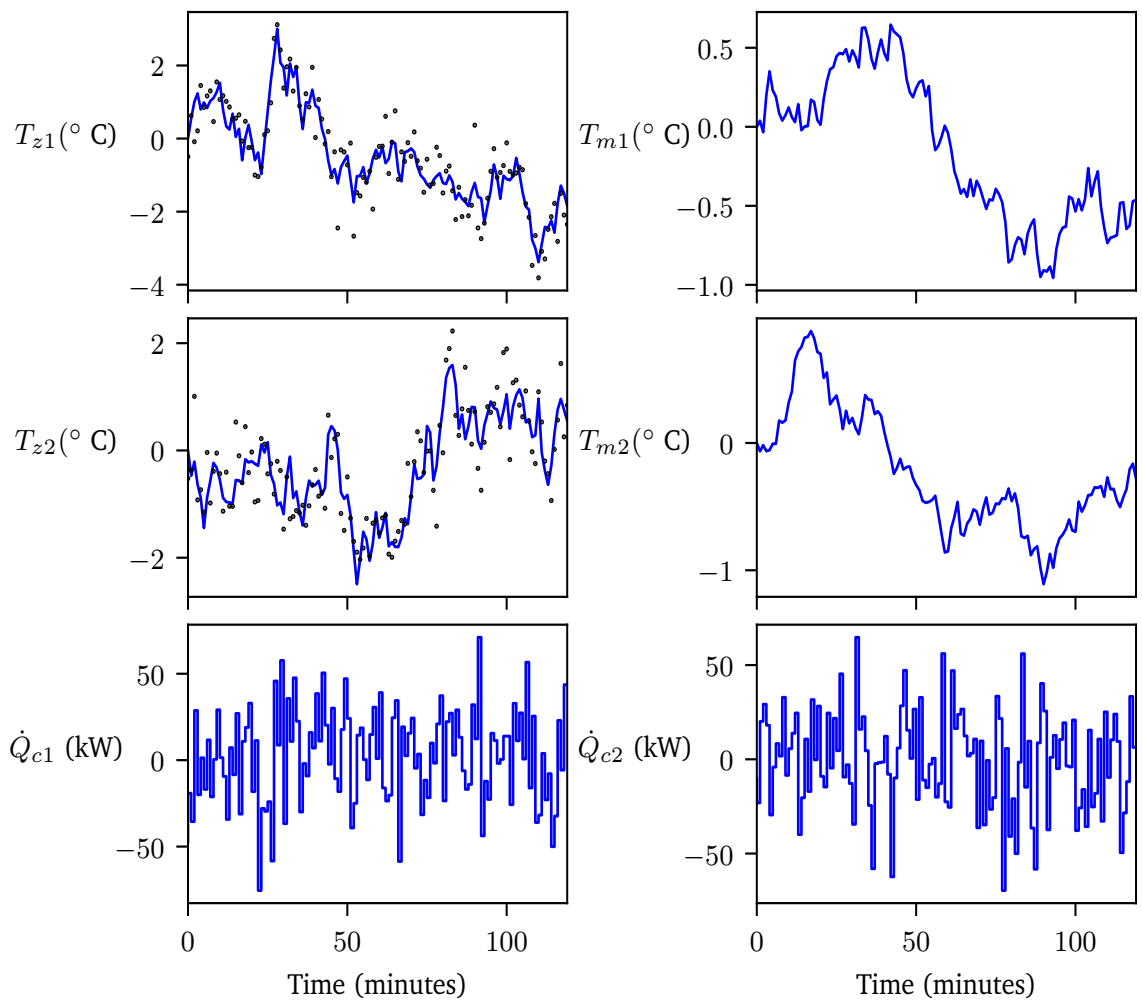

Figure 2.6: Sample training data set for the case of output measurements and both process and measurement noise. The black dots in the zone temperature plots show the measurements, and the blue lines depict the actual state.

2.5.2 Output measurements with both process and measurement noises

The effectiveness of the proposed LSPI algorithms to estimate a reasonable feedback controller is demonstrated next on the case of only output measurements and both process and measurement noise. We generate a total of 24 hours (1440 samples) of

training data set by simulating the HVAC plant using Gaussian random control input, process noise, and measurement noise sequences. The mean of these sequences are chosen to be zero and covariances are given in Table 2.2. The HVAC plant is initialized at the origin for the training data generation. We show the first 2 hours of the generated training data in Figure 2.6.

We implement the following two controller estimation algorithms based on the generated training data

1. SYSID: The MLE algorithm discussed in Section 2.4 is used to estimate the linear dynamic model matrices (Θ) and the noise covariances (S_{wv}), which are then used to construct a regulator and state estimator.

2. LSPI: We implement the model-free algorithm for the output measurement case discussed in Subsection 2.3.4 to estimate a feedback controller.

For both the algorithms, we choose $N_p = 2$ using the method discussed in Appendix 2.6. The LSPI algorithm is started with a matrix of zeros as the initial feedback law. And the two policy evaluation and improvement steps in the algorithm are implemented for a total of 15 iterations.

An output tracking penalty is considered in the LQR problem for this output measurement case. The tuning parameters used for the LQR problem are given in Table 2.2. The dimension of the surrogate state z is 8, which requires the LSPI algorithm to estimate a Q-function matrix S of dimensions 10×10, and a feedback law matrix of size 2×8. As noted previously, the use of the surrogate state z in the LSPI and SYSID algorithm results in a dynamic and noise model mismatch between the plant and model assumed in the algorithms. Thus, the estimated Q-function and feedback law cannot be compared to some true quantities. So to evaluate the quality of the estimated feedback controllers, we directly compare the closed-loop performances of the estimated

controllers with a perfect model-based controller that uses the true HVAC plant model and noise covariances. We refer to this perfect controller as the PLQG controller in the subsequent discussion in this chapter.

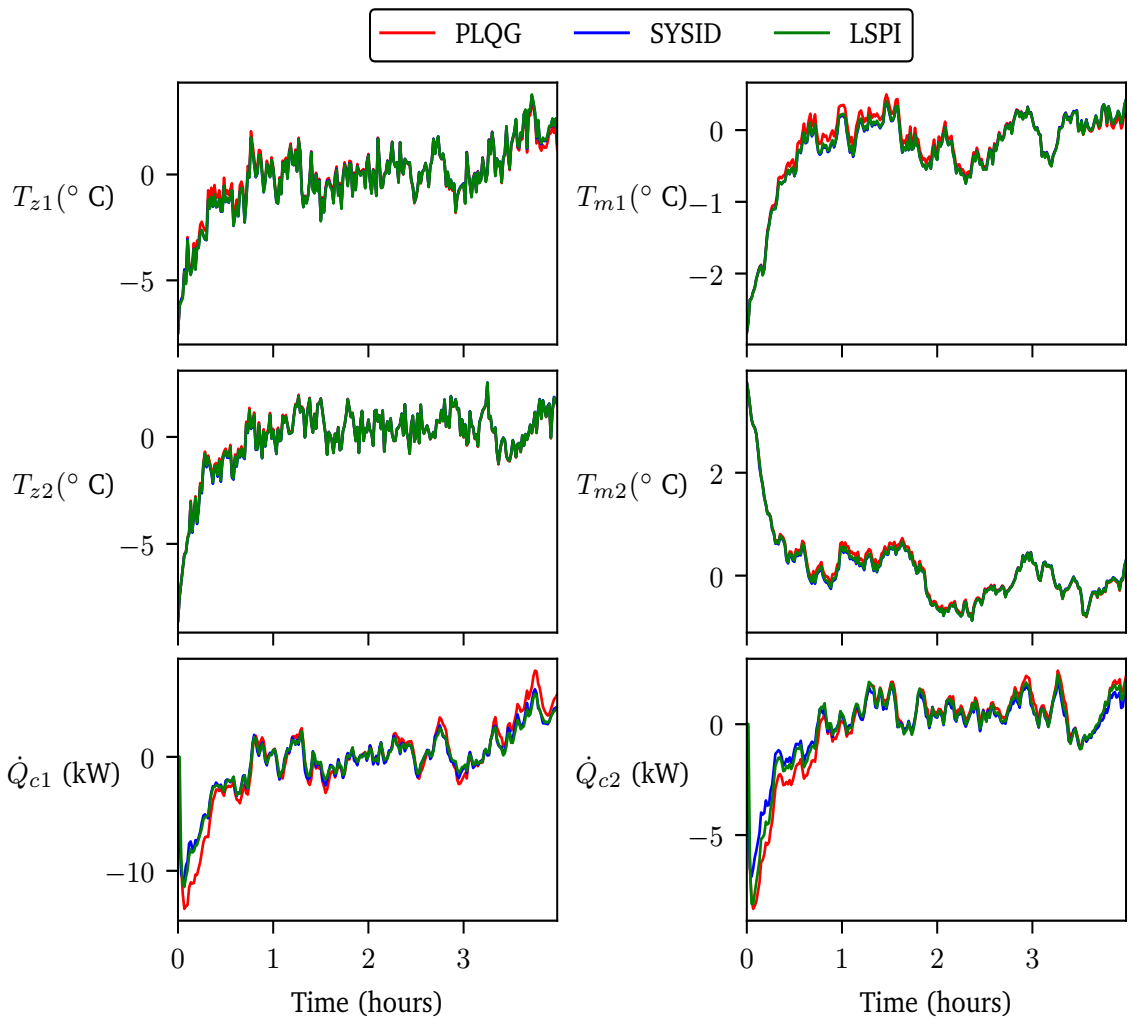

Figure 2.7: A sample closed-loop simulation performed using the PLQG controller, and the controllers estimated with the SYSID and LSPI algorithms. The performance of the controller estimated using the LSPI algorithm is almost the same as the other two model-based controllers.

We first compare the closed-loop performances obtained with the controllers estimated using the two algorithms with the PLQG controller. For this study, the two algorithms are implemented using the full 24 hours of the generated training data. Next,

we perform multiple closed-loop simulations using the estimated controllers starting from some random initial states. All the initial states for these closed-loop simulations are sampled from a uniform distribution in the bounds -10 to $10°\mathrm{C}$. We conduct 500 closed-loop simulations, each for a length of 4 hours. We choose $\alpha = 0.2$ for noise filtering using (2.53) during the closed-loop implementation of the model-free feedback controller estimated using the LSPI algorithm. In Figure 2.7, we show one sample closed-loop simulation obtained with the three controllers. We notice that model-free controller estimated using the LSPI algorithm provides almost the same performance as the other two model-based controllers.

In addition to examining the closed-loop trajectories, we also compute the following metric for each closed-loop simulation conducted using the three controllers

$$\Lambda_{T_j}^{C} = \frac{1}{N_t} \sum_{k=0}^{N_t} \ell(y_j(k), u_j(k), \Delta u_j(k)) \tag{2.59}$$

in which, we use the subscript j to denote the trajectory number of a simulation. Figure 2.8 shows the histograms of the above performance metric for the three controllers. The histograms illustrate that the model-free controller estimated using the LSPI algorithm has a similar performance as the other two model-based controllers across all the conducted closed-loop simulations.

The data efficiency of the SYSID and MLE algorithms is studied next. For this analysis, we implement the two algorithms for varying amounts of training data starting from 2 to 24 hours, in increments of 2 hours. For each estimated controller, we conduct multiple closed-loop simulations similar to the previous analysis. Then, we evaluate the loss metric shown in (2.58) that compares the performance of the estimated controllers compared to the PLQG controller. We plot the loss metric for the estimated controllers using the two algorithms for varying amounts of training data in Figure 2.9.

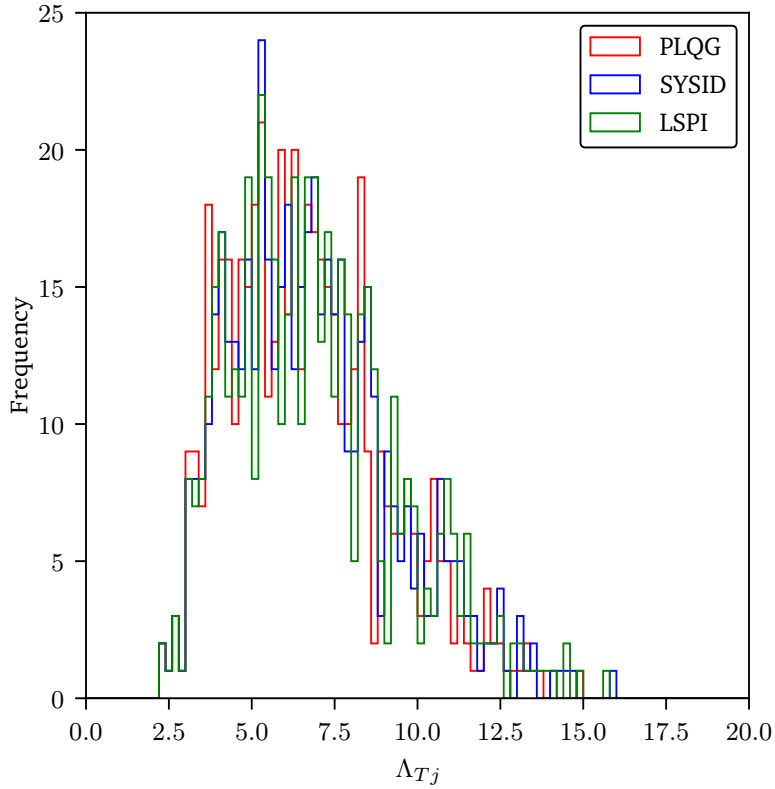

Figure 2.8: Histogram of the closed-loop performance metrics obtained with the controllers estimated using the SYSID and LSPI algorithms, and the PLQG controller.

We observe that for a small amount of 2 hours of training data, the performance of the controller estimated using the LSPI algorithm is noticeably worse with about 14% loss. The performance of the controllers estimated using the LSPI algorithm improves with an increase in the training data. The SYSID algorithm is more data efficient than the LSPI approach. With only 2 hours of training data, the controller estimated with SYSID algorithm provides a performance loss of less than 4%

We summarize the loss metric and the controller estimation times of the two algorithms when using the full 24 hours of training data in Table 2.3. The controller obtained using the SYSID approach provides a performance loss of 2.68$\%$, whereas, the LSPI algorithm also provides a good performance with a loss metric of only 3.55$\%$. The time required for the controller estimation by the SYSID and LSPI algorithms are

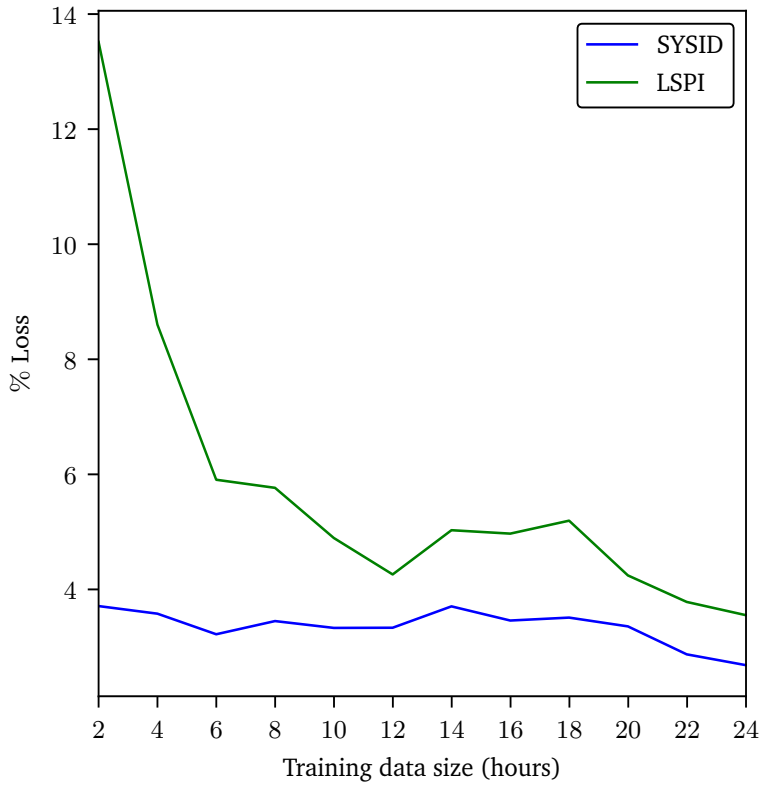

Figure 2.9: Plot of the overall closed-loop performance loss for the controllers estimated using the SYSID and LSPI algorithms with varying amounts of training data.

0.02 and 3.2 seconds, which show that both the algorithms are suitable for implementation in process control applications.

2.6 Conclusions

In this chapter, a new model-free controller estimation algorithm using Q-learning and least squares policy iteration (LSPI) has been presented.

We started with discussing an available algorithm in the literature to estimate the nominal LQR feedback law. A simulation study was presented to elucidate that the algorithm to estimate the nominal LQR is not suitable to estimate a controller from noisy data, and thus cannot be deployed in industrial applications.

Then, we proposed an approach that is suitable to estimate a feedback controller from noisy data. The approach is based on an extension of an available algorithm to estimate the stochastic LQR feedback law for linear systems driven by Gaussian process noise and full state measurements. We discussed how that algorithm can be extended to design a controller from data containing both process and measurement noise and only output measurements. We developed a modified Q-function approximation architecture to first treat the case of an unknown process noise covariance in the full state measurement case. Then, we used a surrogate state containing a recent history of past measurements and control inputs and applied the extended algorithm to design a controller for the output measurement case with both process and measurement noise. Such a model-free controller estimation algorithm that can handle both those noise sources has not been previously proposed in the literature.

We presented simulation studies using a linear HVAC example system to demonstrate the effectiveness of the proposed model-free feedback controller design approach. In the case of full state measurements with process noise, we illustrated that the controllers estimated using the LSPI algorithms provide good performance with the loss metrics of 1.2 and 1.4% compared to the perfect PLQG controller. For the output measurement case with both process and measurement noise, we demonstrated that the controllers estimated using the SYSID and LSPI algorithm provide performance losses of 2.68% and 3.55%. A total of 24 hours of training data was used to achieve this performance using the LSPI algorithm. The performance and data requirement demonstrate that the model-free controller design approach proposed in this chapter can be a viable option for implementation in process control applications.

The reasons for the success of the algorithm proposed in Section 2.3 compared to the failure of the algorithm of Bradtke et al. (1994) as pointed out (Rawlings and Maravelias, 2019) in Section 2.2 are two-fold. First, we build upon an algorithm that

considers a stochastic LQR formulation to estimate a feedback controller. Linear systems with Gaussian process noise are considered in the algorithm. This treatment of the noise enables some robustness of the algorithm to noisy data, however, it cannot be applied directly in applications because it assumes full state measurements and that the covariance of the process noise is known. Second, we extended that algorithm to treat an unknown process noise covariance. The modified approach was then used to estimate a feedback controller for systems with both process and measurement noise and only output measurements.

The model-free approach discussed in this chapter is a step towards an algorithm that may be suitable for deployment in process control applications. We discuss some directions for future work by pointing out the remaining advantages of the model-based approaches in Chapter 6 of this thesis.

The next chapter develops a method to use neural networks to approximate the constrained, nonlinear MPC feedback law to enable MPC implementation in large-scale industrial applications.

Appendix

Model order selection

The parameter N_p in (2.52) to construct the surrogate state (z) characterizes the assumed model order of the plant. To choose this parameter for both the LSPI and MLE algorithms, we first construct the data matrix

$$H = \Big[z(N_p),\ z(N_p+1),\ ...,\ z(N_t) \Big] \tag{2.60}$$

in which, N_t is the total number of time steps in the training data. Each column in this data matrix contains N_p number of past measurements and control inputs. For a large N_p, the rows of the data matrix become nearly linearly independent, and the condition number becomes large. In this case, the estimates of the linear dynamic model in the MLE algorithm or the Q-function in the LSPI approach become sensitive to noise. For a small N_p, the surrogate state does not contain enough information about the plant dynamics, and accurate estimates of the linear dynamic model or the Q-function cannot be obtained. So the tradeoff in choosing the value of N_p is accuracy vs sensitivity to noise. To evaluate this tradeoff, we plot the condition number of the data matrix for various values of the parameter N_p starting from unity to a large integer. We choose the value of N_p around the "knee" of the curve when the condition number increases rapidly and saturates for larger values of N_p.

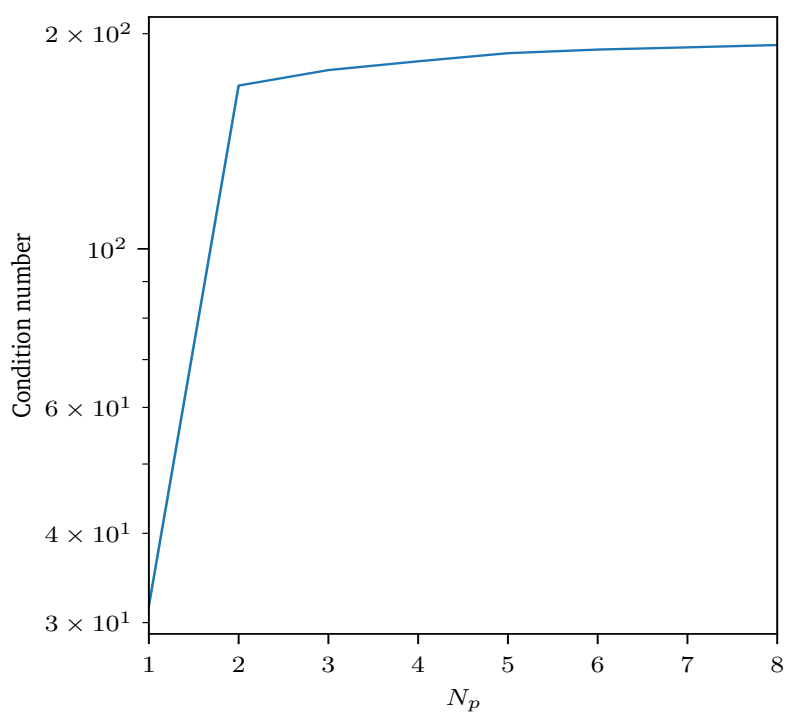

Figure 2.10: Plot of the condition number of the data matrix (2.60) for various values of the parameter N_p in the HVAC example.

In Figure 2.10, we show the condition number plot used to select the order in the output measurement case studies presented in Subsection 2.5.2. We choose $N_p = 2$, which is approximately around the value at which the condition number of the data matrix is increasing rapidly before saturating to large values.

Chapter 3

Fast, large-scale model predictive control using deep neural networks

3.1 Introduction

A model predictive controller (MPC) is implemented by solving an optimization problem in real time. A dynamic model of the plant is used to predict the future measurements in response to the actuators. The optimization problem determines the best possible future actuator sequence to drive the plant to some steady state. The first move is applied to the plant, and the optimization problem is re-solved after every measurement sampling instant in real time. A majority of industrial MPC controllers use a linear plant model, and require solutions to quadratic programs (QP) in real time. Several advances have been made over the years to efficiently solve QPs (Kouzoupis et al., 2018; Wright, 2019), which have enabled control practitioners to implement the MPC technology in industrial applications.

The solution of the QP solved by a linear MPC controller can be characterized as a piecewise affine function defined over some set of feasible initial states and steady-state targets computed for offset-free control (Bemporad et al., 2002; Seron et al., 2003). We refer to this optimal solution of the MPC optimization problem as the MPC feedback

law in this chapter. When implementing an MPC controller in applications, an alternative approach to online optimization can be to store all the regions in the piecewise affine MPC feedback law. And in real time, traverse all the regions to determine in which region the current state lies. The control input can then be computed using the feedback law for the region in which the current state lies. A drawback of this approach is that the number of regions in the MPC feedback law grows exponentially with both the model dimensions and the horizon length in the MPC problem. Thus, the approach cannot be implemented for feedback control of large-scale systems.

To overcome the above issue with storing and traversing all the regions in the entire MPC feedback law, Pannocchia et al. (2007) proposed a partial enumeration approach in which only some relevant regions of the state space are stored. During the online implementation, the shorter number of relevant regions are traversed to determine the control input. These relevant regions are determined based on the states that are recently encountered during the closed-loop implementation. If a state is not found in the list of relevant regions, then the MPC optimization problem is solved to find the region and the feedback law corresponding to that state. The new region is inserted to the list of relevant regions. And to prevent the list from growing further, another region in which a state has not been encountered recently is replaced from the list.

The MPC feedback law may be approximated using parametric function approximators such as polynomials (Kvasnica et al., 2011), other types of piecewise affine functions (Bemporad et al., 2011; Wen et al., 2009), and neural networks (Cavagnari et al., 1999). The motivation of these approaches is to use the approximate MPC feedback law represented by the function approximator during the online closed-loop implementation. In this approximate MPC approach, the use of NNs to approximate the MPC feedback law has gained significant attention recently (Chen et al., 2018; Karg and Lucia, 2020; Paulson and Mesbah, 2020; Lovelett et al., 2020). Feedforward NNs that use

the rectified linear unit (ReLU) as the activation function represent a piecewise affine function. The complexity of the representable functions also grows exponentially with the addition of more layers and nodes in the NN (Montufar et al., 2014). This property makes NNs an attractive candidate to approximate the MPC feedback law. The NN design approach proposed in many of the recent literature is to generate the training data for the NN by solving numerous QPs offline for a set of feasible initial states of the QP. A standard feedforward NN is trained offline using the labelled state to optimal control input samples, and the NN is finally used online for the MPC controller implementation in place of a QP solver.

To enlarge the class of applications achievable using MPC, large-scale problems in which the state-of-the-art QP solvers fail to deliver the control input in the available sample time should be addressed. The literature in the area of MPC feedback law approximation using NNs has not demonstrated the scalability of the approach to such large problems. The following two issues arise in these problems

- In a large dimensional state space, the entire feasible state space of the QP cannot be sampled *densely* due to the well-known curse of dimensionality.
- The state space can be sampled partially to avoid this problem, however, the time required to solve a QP for each sampled state may still be too large to render the offline NN design approach challenging.

We demonstrate in this chapter that the first issue can be resolved by sampling only the relevant state space based on the anticipated plant operational scenarios. The second issue can be mitigated with the use of parallel computing when solving the large QPs for the sampled states during the data generation. For an industrial deployment of the NN design approach proposed in this chapter, the following feature must be present in the large-scale application of interest: In a particular operating mode of the process,

the number of time varying disturbances and setpoints should be reasonably small such that only a small fraction of the entire state space is visited during that operation mode. The MPC feedback law can then be approximated for the different operating modes of the plant, each driven by their respective set of setpoints and disturbances.

Based on the above viewpoint, this thesis chapter presents the design of NNs to approximate the MPC feedback law for large-scale applications. The industrially relevant offset-free MPC formulation is considered. For the NN controller design, we propose to use a modified structured NN architecture to achieve offset-free closed-loop performance. We emphasize that the structured NN architecture is advantageous, and high quality closed-loop performance cannot be obtained using a standard feedforward architecture. Simulation studies are presented with large-scale application examples to demonstrate the scalability of the proposed NN controller design approach. In particular, we present a large-scale industrial crude distillation example with 252 states, 32 control inputs, and a horizon length of 140 in the MPC problem. We utilize parallel computing for the offline data generation and graphical processing units (GPU) for the NN training. We show that after the offline design step, the trained NNs execute MPC around 4 orders of magnitude faster than an available QP solver. The performance loss of the NNs compared to the optimal MPC controller is also less than 1%.

We begin by considering a small scale double integrator example to gain insights into the optimal MPC feedback law, and the corresponding possible approximations that can be attained using NNs. In this chapter, we also establish conditions under which a closed-loop system under feedback with a NN controller is robust to disturbances and MPC feedback law approximation errors. Input-to-state stability results available in the MPC literature are used to analyze the robustness of NN controllers.

A related work Chen et al. (2022) considers the MPC feedback law approximation using NNs on the control problem of regulation to the origin. The paper examines the

approach on an example system with a state dimension of 36. By contrast, we consider the offset-free MPC formulation and demonstrate the scalability of the NN controller design approach on an industrial example with a state dimension of 252. Drgoňa et al. (2018) also treat large MPC applications using NNs. This work proposes to reduce the dimensionality of the state in the QP using principal component analysis, and develops the approximate NN controller in the low dimensional state space. Chan et al. (2021) present a similar structured NN architecture as the one proposed in this chapter. However, the work does not consider plant disturbances and does not demonstrate the scalability of the approach to large applications.

The rest of this thesis chapter is organized as follows. In Section 3.2, we analyze the nature of the MPC feedback law and possible approximations by NNs on a small scale double integrator example. In Section 3.3, we discuss the offset-free MPC formulation considered for NN controller design for large-scale application examples. We discuss the structured NN architecture, the offline data generation, and the network training in Section 3.4. The robustness property of approximate NN controllers is analyzed in Section 3.5. Simulation studies to demonstrate the scalability and effectiveness of the proposed NN controller design approach on large-scale examples are presented in Section 3.6. Conclusions of this chapter are given in Section 3.7.

Results from this chapter appear in the papers Rawlings and Maravelias (2019) and Kumar et al. (2021). The mathematical notation used in this chapter are given in Section 1.3.

3.2 Illustrative double integrator example

We begin by examining the MPC feedback law approximation approach on a two dimensional example, in which the optimal and approximate NN feedback laws can

be examined visually. We consider a double integrator system (unstable) with the following linear dynamics (Mayne and Raković, 2003)

$$x^+ = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0.0787 \end{bmatrix} u \tag{3.1}$$

The goal of the MPC controller is to stabilize the system using the control input (u) for some set of initial states. The following MPC optimization problem is solved

$$\min_{\mathbf{u}} \sum_{k=0}^{N-1} \ell(x(k), u(k)) + V_f(x(N)) \tag{3.2}$$

$$\text{s.t} \quad x^+ = Ax + Bu, \tag{3.3}$$

$$-1 \leq u \leq 1 \tag{3.4}$$

$$A_N x(N) \leq b_N \tag{3.5}$$

$$x(0) = x \tag{3.6}$$

in which, $\ell(\cdot)$ is the stage cost, $V_f(\cdot)$ is the terminal cost, A, B are the linear dynamic model matrices shown in (3.1), $N = 10$ is the control horizon length, and the matrices A_N, b_N are used to implement a terminal state constraint. These matrices are chosen so that the terminal constraint represents the maximal output admissible set (Gilbert and Tan, 1991) for the optimal unconstrained LQR. We use the following stage cost in the optimization problem

$$\ell(x, u) = x_1^2 + 0.1u^2 \tag{3.7}$$

The terminal cost is also quadratic, i.e, $V_f(x) = x'Px$, and the penalty matrix P is chosen as the solution of the DARE equation.

Figure 3.1 (left) shows the polytopic regions in the piecewise affine MPC feedback law, which is defined over a set of feasible initial states of the QP ($\mathcal{X}_N$). We note that the polytopic regions in the MPC feedback law are typically defined in the literature such that the optimal solution for the full control input sequence ($\mathbf{u}^0(\cdot)$) in the horizon length is different across the regions. But the optimal solution of the first control input ($u^0(\cdot)$) can still be the same across the regions. For Figure 3.1, we have merged the regions for which the optimal solution of the first control input were the same.

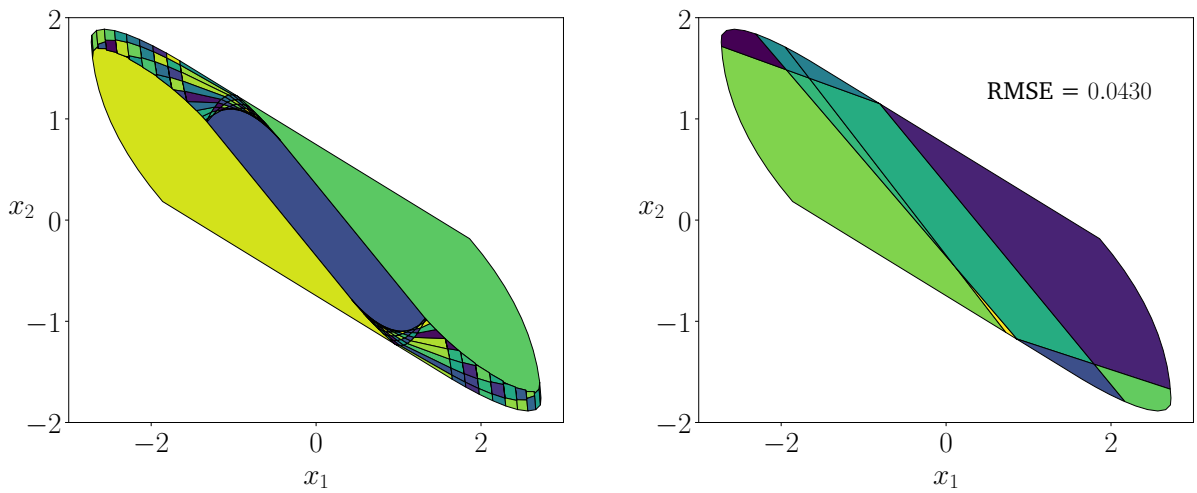

Figure 3.1: Partition of the state space for the optimal feedback law (left); partition of the state space in the NN approximation of the feedback law (right).

To obtain an NN approximation of the MPC feedback law, we sample a set of initial states in the feasible region ($\mathcal{X}_N$), and solve the MPC problem to compute the corresponding optimal control inputs. We sample 100 initial states uniformly in the feasible region. For each sampled state, we perform closed-loop simulations using the optimal MPC controller for $N_t = 10$ time steps and obtain a total of 1000 states shown in Figure 3.2 (left) for the NN training. A standard feedforward network is trained using the 1000 state (x) and control input ($u^0(x)$) pairs as labelled data. We choose a NN of architecture [2, 6, 1], i.e, an input layer of 2 nodes, one hidden layer containing 6

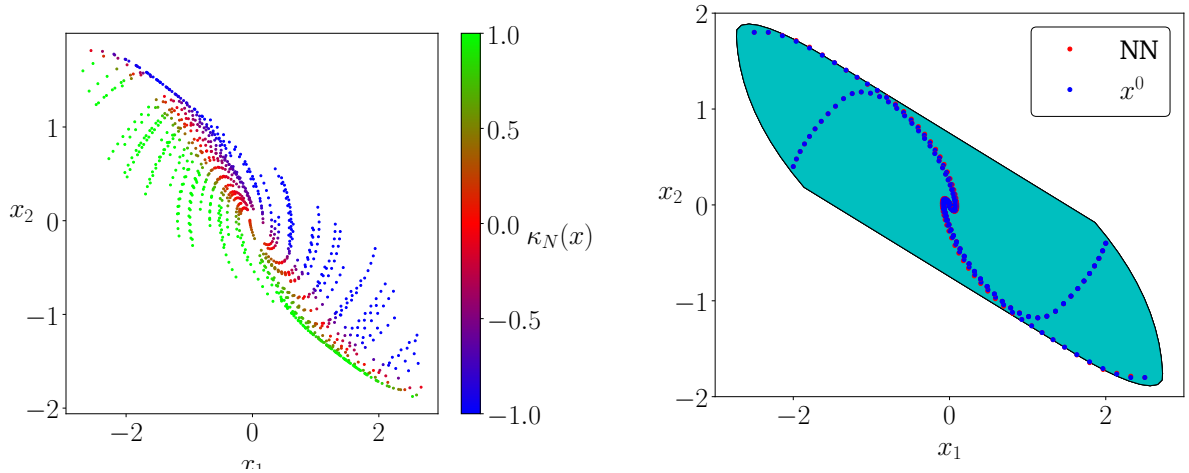

Figure 3.2: Data set used for training, in which the color indicates the value of feedback law $\kappa_N(x)$ (left); closed-loop trajectories obtained using the optimal and approximate NN controllers (right).

nodes, and an output layer of 1 node. The NN is trained using the stochastic gradient algorithm Adam (Kingma and Ba, 2014).

After the NN training, we examine the polytopic regions in the NN approximation of the MPC feedback law. To determine the polytopic regions of the piecewise affine feedback law represented by the NN, we enumerate all the possible combinations of active/inactive nodes in the hidden layer. Each such combination maps to a polytopic region of the two dimensional state space and a corresponding feedback law. We show the polytopic regions in the approximate NN feedback law in the feasible region $\mathcal{X}_N$ in Figure 3.1 (right). We observe that the NN does not recover all the regions in the optimal MPC feedback law, but retains a crude structure of the regions. We also show the root mean squared error metric (RMSE) obtained by the NN on the training data in the Figure 3.1. Next, we use the NN approximation of the MPC feedback law in some closed-loop simulations and examine its performance. Figure 3.2 (right) shows closed-loop trajectories obtained using the optimal and NN controllers starting from four arbitrary initial states. We observe that the difference in the closed-loop

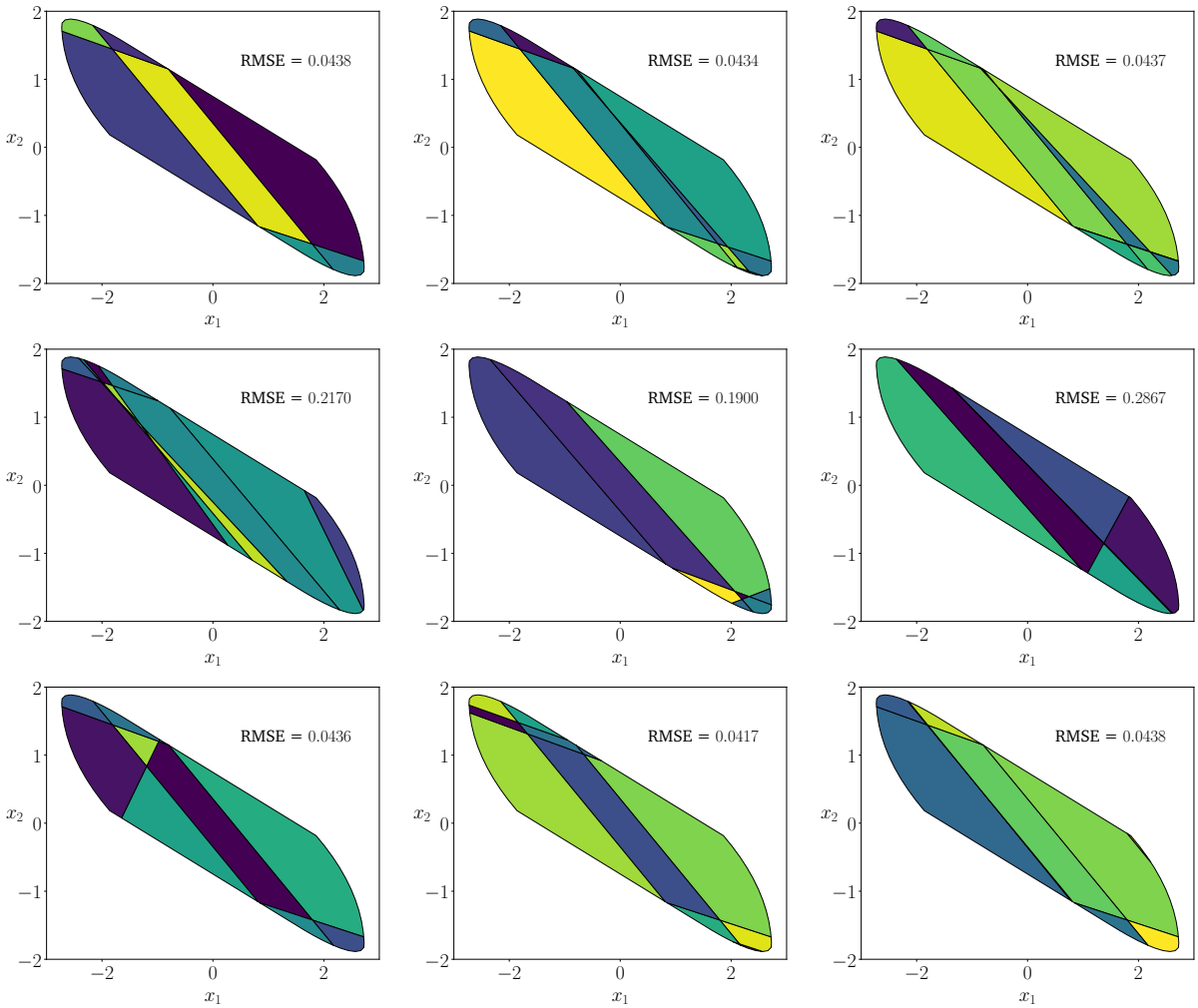

Figure 3.3: Other partitions of the state space obtained using the approximate NN feedback laws for identical training data but with different initialization of the weights and biases during training.

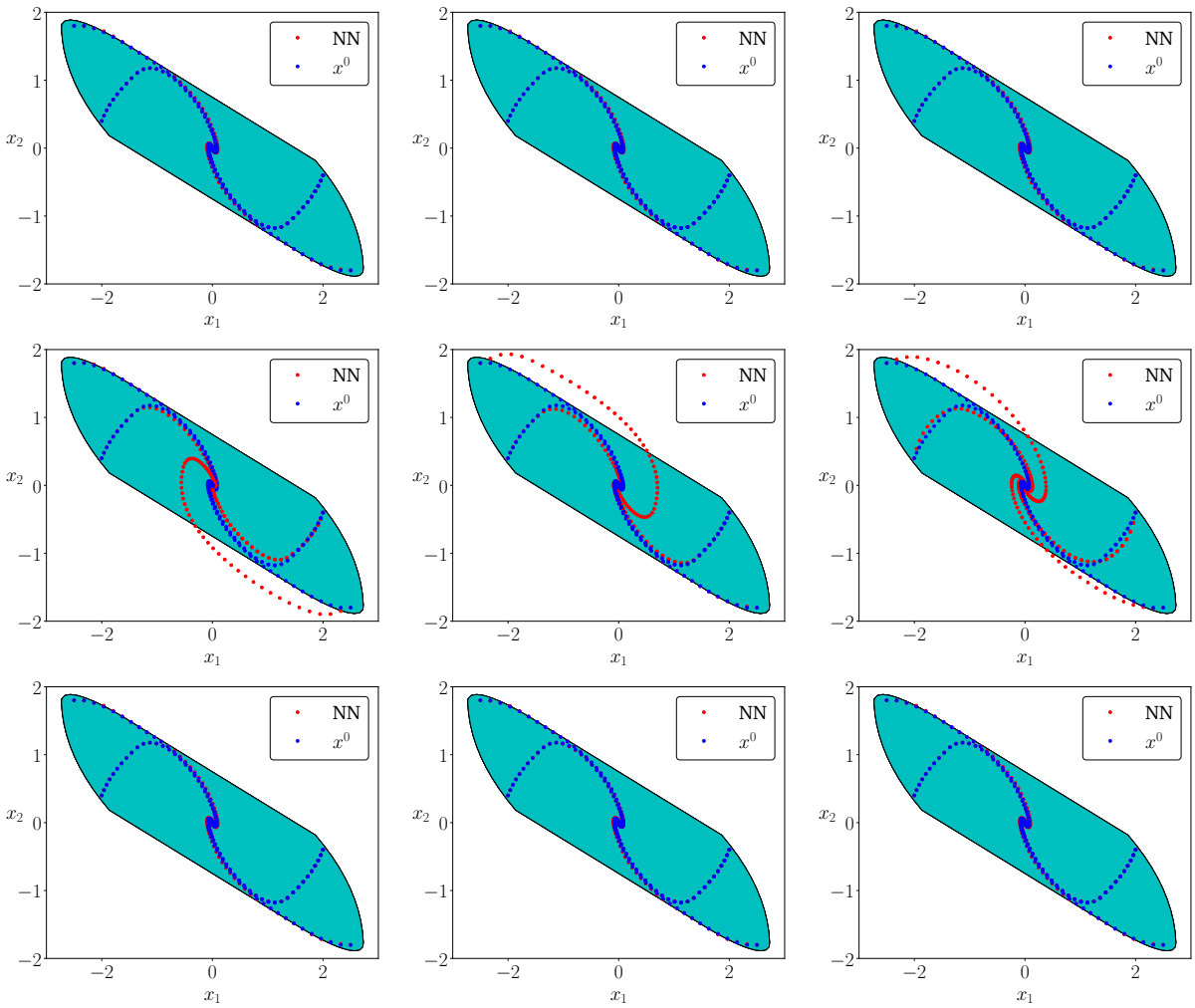

Figure 3.4: Comparison of closed-loop state trajectories obtained using the neural network solutions corresponding to Figure 3.3 with the optimal closed-loop trajectories.

trajectories obtained using the two controllers is barely noticeable.

We note that the quality of the MPC feedback law approximation by the NN can depend on some other factors as well. Such as the initial weights and biases used for training, the batch size in the stochastic gradient algorithm, the total number of epochs, etc. To illustrate that different approximate feedback laws (possibly worse) can also be obtained, we train nine more NNs with the same architecture ([2, 6, 1]) but with different initial guesses of the weights and biases in the NNs. All the other hyperparameters, such as the batch size, number of epochs, etc are kept the same in these different NN training problems. Figure 3.3 shows the MPC feedback approximation by these nine other trained NNs, and Figure 3.4 shows the corresponding closed-loop trajectories obtained with those NNs. We notice particularly that the NNs in the middle rows have a poor approximation of the optimal MPC feedback law, and also provide a noticeably poor closed-loop performance. The poor closed-loop performances by the middle row NNs caution that the NN controllers should be validated properly during or after the training phase before the final online deployment. In the simulation studies in Section 3.6, we also discuss how the trained NNs can be validated before the final deployment at the plant.

3.3 Linear offset-free model predictive control

In this section, we describe the linear offset-free MPC controller formulation used for the subsequent studies in this chapter to examine the NN controller design approach on large-scale application examples. We consider linear plant models augmented with an integrating disturbance model to achieve offset-free closed-loop performance (Pan-

nocchia and Rawlings, 2003; Morari and Maeder, 2012)

$$x^+ = Ax + Bu + B_d d \tag{3.8}$$

$$d^+ = d \tag{3.9}$$

$$y = Cx + C_d d \tag{3.10}$$

in which, $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input, $y \in \mathbb{R}^p$ is the measurement, and $d \in \mathbb{R}^d$ is the state in the disturbance model. The matrices (A, B, C) characterize the control input to measurement model, and (B_d, C_d) characterize the integrating disturbance model. The purpose of using the disturbance model is particularly to remove offset in some controlled measurements during the closed-loop operation, despite the presence of any unmeasured disturbances and plant-model mismatch. Given a time series of data collected from the plant, we construct a Kalman filter to estimate the state and disturbance in the two models as follows

$$\begin{bmatrix} \hat{x} \\ \hat{d} \end{bmatrix}^+ = \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{d} \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \left(y - \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{d} \end{bmatrix} \right) \tag{3.11}$$

Here, the variables $\hat{x} = \hat{x}(k|k-1)$, $\hat{d} = \hat{d}(k|k-1)$ denote the predicted state and disturbance estimates at the current time step k, given measurements up to the time step $k-1$. The variables $(\hat{x}, \hat{d})^+$ are defined similarly, and denote the predicted estimates at the next time step given measurements up to the current time step. We convert the predicted estimates to filtered estimates based the current measurement, which are then finally used in the MPC target selector and regulator formulations. The gains L_x and L_d are computed by solving a discrete algebraic Riccati equation (DARE) for the

steady-state Kalman filter (Rawlings et al., 2020, Pages 32-33) using the augmented linear model in (3.8) – (3.10). The noise covariances used in the DARE to obtain the filter gains are chosen heuristically for the simulation studies in this chapter. The gains computed using the DARE are appropriately converted to obtain the predictor form of the Kalman filter shown in (3.11).

Based on the disturbance estimate $\hat{d}$, and the setpoints for the input and controlled measurements (u_{sp}, r_{sp}), we solve the following target selector QP

$$\min_{x_s,\, u_s} \quad |u_{sp} - u_s|^2_{R_s} \tag{3.12}$$

$$\text{s.t} \quad \begin{bmatrix} I - A & -B \\ HC & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} B_d \hat{d} \\ r_{sp} - HC_d \hat{d} \end{bmatrix} \tag{3.13}$$

$$\underline{u} \leq u_s \leq \overline{u} \tag{3.14}$$

in which, the target steady-state pair (x_s, u_s) is the decision variable, $r = Hy$ are the controlled measurements that are typically chosen as some subset or linear combination of all the measurements, and $[\underline{u}, \overline{u}]$ are the input constraints. We note that the constraint $r_{sp} = H(Cx_s + C_d\hat{d})$ may be difficult to satisfy in real time based on the value of the setpoint and input constraints. To handle this case, we move this setpoint constraint in the stage cost and also minimize the offset in the controlled measurements when determining the target steady state. For the simulation studies in this chapter, we assume that the input setpoint u_{sp} is fixed at some steady state of the system. And only the controlled measurement setpoint r_{sp} changes in real time during the process operation.

The goal of the MPC regulator is to determine the control input to apply to the plant. We use the state estimate $(\hat{x})$ and the target steady-state pair (x_s, u_s) to solve

the following regulator QP

$$\min_{\tilde{\mathbf{u}}} \quad \sum_{k=0}^{N-1} \Big(|\tilde{x}(k)|_Q^2 + |\tilde{u}(k)|_R^2 \Big) + |\tilde{x}(N)|_P^2 \tag{3.15}$$

$$\tilde{x}^+ = A\tilde{x} + B\tilde{u} \tag{3.16}$$

$$\underline{u} \leq \tilde{u} + u_s \leq \overline{u} \tag{3.17}$$

$$\tilde{x}(0) = \hat{x} - x_s \tag{3.18}$$

in which, the state $\tilde{x}$ and control input $\tilde{u}$ are considered in deviation from the target steady state (x_s, u_s), and Q, R are the penalty matrices in the QP. The matrix P for the terminal cost is chosen as the optimal cost-to-go matrix of the optimal unconstrained LQR. The decision variable in the QP is control input sequence $\tilde{\mathbf{u}} = [\tilde{u}(0)', \tilde{u}(1)', ..., \tilde{u}(N-1)']'$. The first element of the optimal input sequence is applied to the plant. The function map from the parameters $(\hat{x}, x_s, u_s)$ to the first control input is the MPC feedback law defined as $\kappa_N(\hat{x}, x_s, u_s) = \tilde{\mathbf{u}}^0(0; \hat{x}, x_s, u_s) + u_s$. We can also implement a rate-of-change penalty in the QP by augmenting state system state with the control input at one previous time step (u_{-1}) (Rawlings et al., 2020, Exercise 1.25). The MPC feedback law in this case becomes also a function of the previous control input in addition to the state estimate and the target steady-state pair.

Several algorithms have been proposed in the literature to solve the above MPC regulator QP. The papers Kouzoupis et al. (2018) and Wright (2019) review the different convex optimization methods available to solve the linear MPC QP. In this chapter, we use a “dense" formulation of the regulator QP in which the state sequence is eliminated from the set of decision variables. And the future control input sequence $\tilde{\mathbf{u}}$ is the only decision variable in the problem. We use the QP solver CVXOPT (Vandenberghe, 2010) for the training data generation and online timing comparisons. The solver is tailored

for dense problems, similar to the MPC regulator QP considered in this chapter. We note that if a more efficient QP solver is available, then it is advantageous for both the online QP and approximate NN based MPC approaches, because a fast QP solver can be used to reduce the offline training data generation time for NNs.

3.4 Neural network controller design

The target selector QP discussed in the previous section is typically small compared to the regulator QP. The latter QP accounts for most of the online computation time for the MPC controller execution. Therefore, we focus on replacing only the MPC regulator QP using NNs by approximating the feedback law $\kappa_N(\cdot)$ for a set of operationally relevant states and steady-state targets.

3.4.1 Structured neural network

An intuitive approach to approximate the feedback law may be to develop a standard feedforward NN that takes the vector $[x', x_s', u_s']'$ as the input, and produces an approximate control input that is close to the optimal input. This approach has been proposed by several researchers (Karg and Lucia, 2020; Chen et al., 2018). But the scalability of the approach has not been demonstrated to large-scale systems. A major issue with the approach is that the NN controller has no insights about the structure of the optimal MPC feedback law. At a minimum, the MPC feedback law outputs a zero control at the origin. And for the offset-free setpoint tracking problem discussed in the previous section, the MPC feedback law satisfies $\kappa_N(x = x_s, u_s, u_s) = u_s$, i.e, the MPC controller uses the steady-state control input to maintain the system at a steady state. This structure should be incorporated in an approximate NN controller to improve the

training data requirements.

To incorporate the steady-state feedback law information in a NN controller, we introduce the following structured architecture

$$
\begin{aligned}
z_0 &= \begin{bmatrix} x', & x_s', & u_s', & x_s', & x_s', & u_s' \end{bmatrix}' \\
f_i(z_{i-1}) &= \begin{bmatrix} W_i & 0 \\ 0 & W_i \end{bmatrix} z_{i-1} + \begin{bmatrix} b_i \\ b_i \end{bmatrix} \\
z_i &= g(f_i(z_{i-1})), \text{for } i \in \mathbb{I}_{1:h} \\
u &= u_s + \begin{bmatrix} W_{h+1}, & -W_{h+1} \end{bmatrix} z_h
\end{aligned} \tag{3.19}
$$

in which, $g(a) = \max(0, a)$ is the rectified linear unit (ReLU) activation function applied element wise on the vector a, the subscript i is used to represent the different hidden layers, z_i is the output of the hidden layer i, z_0 is the input to the network, and h is the number of hidden layers. The unknown parameters in the NN are the weights and biases W_i, b_i, W_{h+1}, which are subsequently determined by solving the NN training optimization problem. At a steady state when $x = x_s$ in the input z_0, the NN (3.19) outputs u_s regardless of the choices of the weights and biases. The top and bottom half in the output z_i of each layer contain a repeated subvector and the term $\begin{bmatrix} W_{h+1}, & -W_{h+1} \end{bmatrix} z_h$ equals zero. We note that the structure architecture proposed above can also be viewed as follows

$$
u = u_s + f_N(x, x_s, u_s) - f_N(x_s, x_s, u_s) \tag{3.20}
$$

Here, f_N is a standard feedforward NN without any bias term in the last layer. The structured NN architecture (3.19) can be developed using the symbolic differentiation software Tensorflow (Abadi et al., 2015). And the training optimization problem can

be solving using customizable feedforward network classes available in the software.

When implementing a rate-of-change penalty on the control input, the optimal MPC feedback law becomes a function of the previous control input (u_{-1}) as well in addition to the state and steady-state targets. So we also provide the previous control input to the NN if a rate-of-change penalty on the control input is implemented in the MPC regulator. We modify the input to the NN (3.19) as follows

$$z_0 = \begin{bmatrix} x', & u'_{-1}, & x'_s, & u'_s, & x'_s, & u'_s, & x'_s, & u'_s \end{bmatrix}'$$

The weights and biases W_i, b_i, W_{h+1} are modified to be of appropriate dimensions.

We note that even after training with a sufficient amount of data, the obtained NN may not produce a control within the constraints $[\underline{u},\ \overline{u}]$. And to ensure that the NN outputs a control input within the constraints, we apply the saturation function ($\mathrm{sat}(u, \underline{u}, \overline{u})$) to the output of the NN. We use $\kappa_{NN}(\cdot)$ to denote the approximate feedback law represented by a NN in the subsequent discussion in this chapter.

3.4.2 Data generation

The feasible region of the MPC regulator QP (3.16) – (3.18) for a fixed steady state is $\mathbb{R}^n$, i.e, the entire n-dimensional state space. The target steady state for the QP also changes at almost every time step in industrial applications due to time varying disturbances and setpoints. The full domain of the MPC feedback law is therefore the combination of the state space $\mathbb{R}^n$ and the possible set of steady-state target pairs. Densely sampling this entire domain of the MPC feedback law is not possible for large applications due to the curse of dimensionality. We show in the simulation studies in Section 3.6 that sampling the entire domain is also not required for practical applications.

Large industrial chemical plants often operate in different operating regimes or

scenarios. Each operating scenario is driven by only a few controlled measurement setpoints and large magnitude disturbances that change frequently. The controlled measurement setpoints may change based on product goals, but the setpoints for a large number of measurements other than the controlled measurements remain constant for long periods of time. The states and steady-state targets actually visited during the closed-loop operation is thereby considerably less than the entire domain of the MPC feedback law. We use this insight about the typical operation of large chemical plants, and sample only the relevant states and steady-state targets for a particular operating mode of the plant. We generate the training data for NNs as follows

1. First, we determine the controlled measurements (r) and an anticipated range $[\underline{r}_{sp}, \overline{r}_{sp}]$ in which their setpoints may change during the plant operation.

2. Next, we determine the large magnitude disturbances (d), their effect on the plant dynamics via a disturbance model (B_d, C_d), and the range $[\underline{d}, \overline{d}]$ in which those disturbances can change during the operation.

3. We obtain pseudo random binary signals (PRBS) for the setpoints (r_{sp}) and disturbances (d) in their respective ranges.

4. We perform an offline closed-loop simulation using the plant model and MPC controller by solving the target selector and regulator QPs for all the transient states and steady-state targets encountered during the simulation. All these states, steady-state targets, and corresponding optimal control inputs are gathered as the training data set.

In the step (2) above, we assume that the disturbances have a physical meaning and assume that the matrices (B_d, C_d) are obtained from a prior disturbance model identification step. The simulation in the step (4) is performed offline using the linear plant

model, and we do not perform Kalman filtering for state and disturbance estimation. The known states of the linear model and disturbances are directly used in the target selector and MPC regulator.

During the plant operation, it is expected that the NN controller does not encounter the states and steady-state targets used in the training data. But similar values are expected because an adequate dynamic model, and setpoint and disturbances are used for the data generation. The NN controller after the training step produces approximate, interpolated control inputs in real time during the process operation.

For large MPC problems, the training data generation step can start to take impractical amounts of time if performed serially because of long time (e.g, minutes) required to solve each QP. We parallelize the data generation step by simultaneously performing multiple offline closed-loop simulations over different cores within a CPU and across multiple CPUs as well. The data generated from all these parallel closed-loop simulations are used for the NN controller training.

3.4.3 Training optimization problem

After the training data generation, the next step is to determine the unknown weights and biases such that the NN obtains an approximation of the optimal MPC feedback law. To achieve this goal, we solve the following optimization problem

$$\min_{\theta} \frac{1}{N_s} \sum_{j=1}^{N_s} [\kappa_{NN}(x_j, x_{sj}, u_{sj}; \theta) - \kappa_N(x_j, x_{sj}, u_{sj})]^2 \tag{3.21}$$

in which, the decision variable θ contains the weights and biases W_i, b_i, W_{h+1}, the subscript j is used to denote a particular training data sample, and N_s is the total number of training samples. The optimization problem is solved using the stochastic gradient

algorithm Adam. We do not consider any regularization penalty in the problem. The recent theoretical and empirical works in the machine learning literature have demonstrated that large NNs have good interpolation capabilities (Belkin et al., 2019; Zhang et al., 2017; Arora et al., 2019; Allen-Zhu et al., 2019) without any explicit form of regularization in the training problem.

3.5 Robustness of neural network controllers

In this section, we establish that under sufficiently small approximation errors of the MPC feedback law, the NN controllers satisfy a closed-loop inherent robustness property. The optimal MPC controller designed for a linear system is known to be robust to sufficiently small state estimation errors and disturbances (Heath and Wills, 2005; Pannocchia et al., 2011). The approximation error of the MPC feedback law by a NN can be treated as an additional disturbance to the system, and the existing input-to-state stability (Sontag and Wang, 1995) results available in the literature can be used to establish the robustness of NN controllers. We show that the allowable approximation error by a NN controller to achieve the closed-loop robustness property can also be established.

We first discuss some required preliminary results, then present a theorem to characterize the robustness property of approximate NN controllers.

3.5.1 Preliminaries

We use the following definitions of $\mathcal{K}$, $\mathcal{K}_\infty$, and $\mathcal{KL}$ functions (Rawlings et al., 2020, Page 695).

- A function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is of class $\mathcal{K}$ if it is continuous, zero at zero, and strictly

increasing.

- A function is of class $\mathcal{K}_\infty$ if it is of class $\mathcal{K}$ and unbounded ($\alpha(s) \to \infty$ as $s \to \infty$).
- A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{I}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is of class $\mathcal{KL}$ if it is continuous, for each $k \in \mathbb{I}_{\geq 0}$, $\beta(\cdot, k)$ is of class $\mathcal{K}$, and for each $s \geq 0$, $\beta(s, \cdot)$ is non-increasing and satisfies $\lim_{k\to\infty}\beta(s, k) = 0$.

We define the level set of a function as follows. Given $V : X \to \mathbb{R}_{\geq 0}$ and $\tau > 0$, $\text{lev}_\tau V = \{x \in X \mid V(x) \leq \tau\}$.

We use the definitions described next about (i) robust positive invariance of a set, (ii) input-to-state stability (ISS), (iii) ISS-Lyapunov function, and (iv) a known result that if a system admits an ISS-Lyapunov function then it is ISS (Jiang and Wang (2001); Rawlings et al. (2020), Appendix B).

Definition 3.1. (Robust positive invariance of a set). A closed set X is robustly positive invariant for the system $x^+ = f(x, w)$, $w \in \mathbb{W}$ if $x \in X$ implies that $f(x, w) \in X$ for all $w \in \mathbb{W}$.

Definition 3.2. (Input-to-state stability (ISS) of a system). Given that $\mathbb{W}$ is a compact set containing the origin and that X is a closed robustly positive invariant set for $x^+ = f(x, w)$, $w \in \mathbb{W}$. The system is ISS in X if there exist functions $\beta(\cdot) \in \mathcal{KL}$ and $\sigma(\cdot) \in \mathcal{K}$, such that for all $x \in X$ and $w(i) \in \mathbb{W}$, $i \in \mathbb{I}_{0:k-1}$, $|\psi(k; x, \mathbf{w}_k)| \leq \beta(|x|, k) + \sigma(||\mathbf{w}_k||)$. Here, $\psi(k; x, \mathbf{w}_k)$ is the solution of the system at time k, starting from an initial state x, and with the disturbance sequence $\mathbf{w}_k$ affecting the system.

Definition 3.3. (ISS-Lyapunov function). A function $V : X \to \mathbb{R}_{\geq 0}$ is an ISS-Lyapunov function in X for the system $x^+ = f(x, w)$, $w \in \mathbb{W}$ if there exist four functions $\alpha_1(\cdot), \alpha_2(\cdot), \alpha_3(\cdot) \in \mathcal{K}_\infty$ and $\sigma(\cdot) \in \mathcal{K}$ such that for all $x \in X$ and $w \in \mathbb{W}$, we have

the following

$$\alpha_1(|x|) \leq V(x) \leq \alpha_2(|x|)$$
$$V(f(x,w)) - V(x) \leq -\alpha_3(|x|) + \sigma(|w|)$$

Proposition 3.4. *(Existence of an ISS-Lyapunov function implies ISS). Given that $\mathbb{W}$ is a compact set containing the origin, and that X is a closed robustly positive invariant set for $x^+ = f(x,w)$, $w \in \mathbb{W}$. If the function $f(\cdot)$ is continuous and there exists a continuous ISS-Lyapunov function in X for $x^+ = f(x,w)$, $w \in \mathbb{W}$, then the system is ISS in X.*

We also use the following two propositions from Allan et al. (2017) and Rawlings and Ji (2012). First proposition is useful to bound values of continuous functions, and the next one is an inequality for $\mathcal{K}$ functions.

Proposition 3.5. *Given sets $C \subseteq D \subseteq \mathbb{R}^n$, C compact, D closed, and a continuous function $f : D \to \mathbb{R}^n$. Then there exists a function $\sigma(\cdot) \in \mathcal{K}_\infty$, such that for all $x \in C$ and $y \in D$, we have the following*

$$|f(x) - f(y)| \leq \sigma(|x - y|) \tag{3.22}$$

Proposition 3.6. *Given that $\sigma(\cdot) \in \mathcal{K}$, the following relation holds for all $a_i \in \mathbb{R}_{\geq 0}$, $i \in \mathbb{I}_{1:n}$*

$$\sigma(a_1 + a_2 + \cdots + a_n) \leq \sigma(na_1) + \sigma(na_2) + \cdots + \sigma(na_n) \tag{3.23}$$

3.5.2 Robustness property

We next use the results discussed above to establish the robustness property of approximate NN controllers. We consider the control problem of regulating the state

of a linear system to some fixed steady state (x_s, u_s). And the state and control input are defined in deviation from that steady state as

$$x := x - x_s, \quad u := u - u_s \tag{3.24}$$

We denote the optimal MPC feedback law using $\kappa_N(x)$, and the approximate NN feedback law as $\kappa_{NN}(x) = \kappa_N(x) + e_{NN}(x)$. Here, $e_{NN}(x)$ is the approximation error in the NN feedback law that may depend on the state of the linear system. We assume in the analysis below that the target steady state (x_s, u_s) is such that the optimal unconstrained LQR feedback law is a feasible controller within some neighborhood around the origin.

Nominal stability properties of an MPC controller have been well studied in the literature (Mayne et al., 2000). The optimal cost of the MPC regulator QP can be used as a Lyapunov function to establish nominal stability. When no hard terminal region constraint is used in the QP, the level set $\text{lev}_\tau\ x'Px$ can be used as an implicit terminal region ($\mathbb{X}_f$) to establish nominal stability. Here, $d > 0$ is a constant such that $\ell(x, u) \geq d$ for all $x \in \mathbb{R}^n \setminus \mathbb{X}_f$, and $\underline{u} \leq u + u_s \leq \overline{u}$. We choose the parameter τ such that the optimal LQR is a feasible controller for all $x \in \mathbb{X}_f$. The region of attraction of the closed-loop system $x^+ = Ax + B\kappa_N(x)$ under feedback with an MPC controller when no hard terminal constraint is used in the QP can be characterized as the level of the optimal cost function as follows $\mathcal{X}_N = \text{lev}_{Nd+\tau}\ V_N^0(x)$ (Limon et al., 2006). The optimal cost function also satisfies the following requirements of a Lyapunov function $c_1 |x|^2 \leq V_N^0(x) \leq c_2 |x|^2$ and $V_N^0(x^+) - V_N^0(x) \leq -c_1 |x|^2$ for some $c_2 \geq c_1 > 0$, in the region of attraction $\mathcal{X}_N$. This property is used to establish the nominal stability of the closed-loop linear system in feedback with the MPC controller.

For the robustness analysis, we assume that the NN controller uses a state estimate

($\hat{x}$) to compute the control input. We examine the following closed-loop system under feedback with the approximate NN controller to analyze robustness

$$\hat{x}^+ = A\hat{x} + B\kappa_N(\hat{x}) + Be_{NN}(\hat{x}) + w - Ae + e^+, \tag{3.25}$$

in which, $\hat{x} = x + e$ is the state estimate, e, e^+ are the state estimation errors at the current and next time steps, and w is a process disturbance. We use $\phi_d(k; \hat{x})$ to denote the solution of the above closed-loop system with disturbances at the time step k starting from an initial state $\hat{x}$. Additionally, we denote the solution of the nominal closed-loop system $x^+ = Ax + B\kappa_N(x) + Be_{NN}(x)$ at the time step k starting from an initial state x as $\phi(k; x)$.

The Theorem 3.7 characterizes the inherent robustness of approximate NN controllers and the allowable bound on the approximation error for which the NN is robust to disturbances. The result holds also for other approximate MPC approaches that use different parametric functions to approximate the feedback law such as polynomials and other piecewise affine functions.

We note that a related work Hertneck et al. (2018) also examines the robustness of approximate MPC controllers. The work considers nonlinear systems and also treats the feedback law approximation error as an additional disturbance to the system. The work, however, uses a robust MPC formulation using constraint tightening procedures to determine the allowable approximation error in the NN. The robustness of NN controllers trained using samples of a nominal MPC controller formulation with both state estimation errors and process disturbances has not been discussed in the literature.

Theorem 3.7. *For all* $\rho \in \Big(0,\ Nd + \tau\Big]$*, there exist constants* $\delta_1, \delta_2, \delta_3 > 0$*, functions* $\beta(\cdot) \in \mathcal{KL}$ *and* $\alpha_e(\cdot), \alpha_w(\cdot), \alpha_n(\cdot) \in \mathcal{K}$*, such that for all disturbance sequences satisfying* $||\mathbf{e}_{k+1}|| \leq \delta_1$*,* $||\mathbf{w}_k|| \leq \delta_2$*, and approximation error satisfying* $|e_{NN}(\hat{x})| \leq \delta_3$ *for all* $\hat{x}$ *in*

the set $S := \text{lev}_\rho\, V_N^0(\hat{x})$*, the closed-loop states of the system* (3.25) *satisfy the bound*

$$|\phi_d(k;\hat{x})| \leq \beta(|\hat{x}|\,, k) + \alpha_e(||\mathbf{e}_{k+1}||) + \alpha_w(||\mathbf{w}_k||) + \alpha_n(\bar{e}_{NN}) \tag{3.26}$$

Here, $\bar{e}_{NN}$ *is the largest approximation error in the set* S *defined as* $\bar{e}_{NN} = \max_{\hat{x}\in S} |e_{NN}(\hat{x})|$.

For the nominal case when both the state estimation error and process disturbance sequences $\mathbf{e}$ and $\mathbf{w}$ are zero, the solutions of the closed-loop system satisfy the bound

$$|\phi(k;x)| \leq \beta(|x|\,, k) + \alpha_n(\bar{e}_{NN}) \tag{3.27}$$

Any NN training with more data and large architectures reduces the approximation error $e_{NN}(\cdot)$ in the state space, which is beneficial to tighten the above bound on the closed-loop states.

Proof of Theorem 3.7: We establish the robustness property in the theorem in two steps. First, we show that there exist constants $\delta_1, \delta_2, \delta_3 > 0$ such that S is a robustly positive invariant set. Then, we show that $V_N^0(\hat{x})$ is an ISS Lyapunov function for the closed-loop system (3.25) on the set S. We denote $\tilde{x}^+ = A\hat{x} + B\kappa_N(\hat{x})$ as the nominal state evolution under feedback with the optimal MPC controller, and $d(k) = [e_{NN}(\hat{x}(k))'B', w(k)', e(k)', e(k+1)']'$ denotes the combination of all the disturbances affecting the system at a given time k.

To establish that the set S is robustly positive invariance, we show that if $V_N^0(\hat{x}) \leq \rho$, then $V_N^0(\hat{x}^+) \leq \rho$. The optimal cost function ($V_N^0(\hat{x})$) is continuous, and using $\mathbb{R}^n$ as the closed set ($\hat{x}^+ \in \mathbb{R}^n$) and S as the compact set ($\tilde{x} \in S$), we have the following using

Proposition 3.5 for some $\sigma_1(\cdot) \in \mathcal{K}_\infty$

$$\begin{aligned}\left|V_N^0(\hat{x}^+) - V_N^0(\tilde{x}^+)\right| &\leq \sigma_1(\left|\hat{x}^+ - \tilde{x}^+\right|) \\ &\leq \sigma_1(|Be_{NN}(\hat{x})| + |w| + |A|\,|e| + \left|e^+\right|) \\ &\leq \sigma_1(|d| + |d| + |A|\,|d| + |d|) := \sigma_2(|d|) \end{aligned} \tag{3.28}$$

Here, $\sigma_2(s) := \sigma_1(|A|\,s + 3s)$. The last inequality follows because $|a| \leq |[a', b']'|$ for two vectors a and b, and the function $\sigma_2(\cdot)$ is of type $\mathcal{K}_\infty$.

Next, we split the set S in outer and inner parts based on the value of the optimal cost function $V_N^0(\hat{x})$. And bound the disturbance d in each part such that $V_N^0(\hat{x}^+) \leq \rho$. For the subsequent analysis, we recall from Subsection 3.5.2 that the optimal cost function satisfies the following relations for the nominal state evolution from $\hat{x}$ to $\tilde{x}^+$ under feedback with the MPC controller

$$c_1\,|\hat{x}|^2 \leq V_N^0(\hat{x}) \leq c_2\,|\hat{x}|^2 \tag{3.29}$$

$$V_N^0(\tilde{x}^+) - V_N^0(\hat{x}) \leq -c_1\,|\hat{x}|^2 \tag{3.30}$$

Relation (3.29) holds for the state $\tilde{x}^+$ as well.

Case 1 (Outer part of the set S): $\rho/2 \leq V_N^0(\hat{x}) \leq \rho$. Using the upper bound on the optimal cost, we have $\rho/(2c_2) \leq |\hat{x}|^2$ for all states $\hat{x}$ in this outer part of S. To analyze the worst-case disturbance, the following can be obtained using equation (3.28): $V_N^0(\hat{x}^+) \leq V_N^0(\tilde{x}^+) + \sigma_2(|d|)$. Next, using the cost decrease (3.30) for the nominal state evolution $\tilde{x}^+ = A\hat{x} + B\kappa_N(\hat{x})$, we have $V_N^0(\hat{x}^+) \leq V_N^0(\hat{x}) - c_1\,|\hat{x}|^2 + \sigma_2(|d|)$. Therefore, if $\sigma_2(|d|) \leq \rho c_1/(2c_2)$ at all time steps in the outer part of the set S, the optimal cost satisfies $V_N^0(\hat{x}^+) \leq \rho$.

Case 2 (Inner part of the set S): $0 \leq V_N^0(\hat{x}) \leq \rho/2$. From equation (3.28) and

the cost decrease (3.30) for the nominal state evolution $\tilde{x}^+ = A\hat{x} + B\kappa_N(\hat{x})$, we have the following for the inner part of the set S: $V_N^0(\hat{x}^+) \leq \rho/2 + \sigma_2(|d|)$. Therefore, if $\sigma_2(|d|) \leq \rho/2$ at all times steps in the inner part of the set S, the optimal cost satisfies $V_N^0(\hat{x}^+) \leq \rho$.

Hence, if we choose $|d| \leq \min\{\sigma_2^{-1}(\rho c_1/(2c_2)), \sigma_2^{-1}(\rho/2)\} := \delta_{\text{max}}$ at all time steps, then the set S is robustly positive invariant for the closed-loop system (3.25). Since $c_1 \leq c_2$, we have $\delta_{\text{max}} = \sigma_2^{-1}(\rho c_1/(2c_2))$. From the definition of the disturbance d, we obtain $\delta_1 = \delta_2 = \delta_{\text{max}}/4$, and $\delta_3 = \delta_{\text{max}}/(4\,|B|)$.

The optimal MPC cost function satisfies the relation

$$V_N^0(\hat{x}^+) - V_N^0(\hat{x}) \leq -c_1\,|\hat{x}|^2 + \sigma_2(|d|)$$

in the robustly positive invariant set S and the disturbance set $|d| \leq \delta_{\text{max}}$. The optimal cost function also satisfies the upper and lower bounds (3.29). So from the Definition 3.3, it satisfies the requirements of an ISS-Lyapunov function. Hence, using the result in Proposition 3.4, the closed-loop system (3.25) is ISS (Definition 3.2). And there exist functions $\beta(\cdot) \in \mathcal{KL}$ and $\sigma(\cdot) \in \mathcal{K}$, such that the closed-loop states satisfy $|\phi_d(k;\hat{x})| \leq \beta(|\hat{x}|\,, k) + \sigma(||\mathbf{d}_k||)$. Using Proposition 3.6 and the definition of $d(k)$, we have

$$\begin{aligned}|\phi_d(k;\hat{x})| &\leq \beta(|\hat{x}|\,, k) + \sigma(|B|\,\bar{e}_{NN} + ||\mathbf{w}_k|| + 2\,||\mathbf{e}_{k+1}||)\\ &\leq \beta(|\hat{x}|\,, k) + \sigma(3\,|B|\,\bar{e}_{NN}) + \sigma(3\,||\mathbf{w}_k||) + \sigma(6\,||\mathbf{e}_{k+1}||)\end{aligned}$$

which establishes the bound in Theorem 3.7 with the functions $\alpha_e(s) := \sigma(6s)$, $\alpha_w(s) := \sigma(3s)$, $\alpha_n(s) := \sigma(3\,|B|\,s)$, and $\alpha_e(\cdot), \alpha_w(\cdot), \alpha_n(\cdot) \in \mathcal{K}$. The allowable approximation error in the NN controller is $\delta_3 = \delta_{\text{max}}/(4\,|B|) = \sigma_2^{-1}(\rho c_1/(2c_2))/(4\,|B|)$, over the robust positive invariant set S.

3.6 Large-scale application examples

In this section, we present simulation studies to demonstrate the scalability of the proposed NN design approach using two large application examples. We examine the data requirements, the closed-loop performances obtained by approximate NN controllers, and the online computation benefits of NNs in the simulation studies. After the training, we examine the performances of obtained NN controllers in two types of plant behaviors

1. The plant encounters setpoints and disturbances not present in the training data, but within the same range of values used in the data generation.

2. The plant encounters some extra setpoints and disturbances not used in the training data.

The first case examines the performance of NNs when the plant behaves in an “expected" manner, while the second case examines the performance when the plant behaves in an “unexpected" fashion.

We perform the validation simulations directly using the available plant model for the case studies in this section. In industrial applications, the NN training should be followed by an additional validation step before the final online controller deployment. This additional validation can be carried out as follows

1. The optimal MPC controller can be used to generate some extra test data. And a prediction error metric such as the mean squared error (MSE) of the trained NN can be computed on the test data to gauge the quality of the approximated feedback law.

2. Comparative closed-loop simulations can be performed using the linear plant model with the optimal MPC and the trained NN controllers. Closed-loop perfor-

mance metrics can be examined here to analyze the quality of the approximated feedback law by the NN.

Other probabilistic validation methods for NNs trained using a robust MPC controller formulation have been proposed in the papers Karg et al. (2019) and Hertneck et al. (2018).

Apart from examining the performances of the optimal MPC and trained NNs, we also analyze the closed-loop performances of the following controllers, which are equivalently fast as NNs

1. Steady-state controller (SS): We directly apply the steady-state control input $u = u_s$ computed from the target selector to the plant.

2. Saturated linear quadratic regulator (satK): We obtain the unconstrained LQR gain K by solving the DARE, and use the feedback law $u = u_s + K(x - x_s)$ to determine the control input.

3. Short horizon controller (SH): We solve an MPC problem with a shorter horizon length to determine the control input.

4. Unstructured NN (UNS): Rather than developing a NN using the structured architecture proposed in Section 3.4, we use a standard feedforward NN architecture to develop the NN controller.

To analyze the quality of approximated feedback law by NNs, the offline computational effort required for training, and the memory footprint required for online deployment, we compute the following metrics

- Controller performance index

$$\Lambda_k = \frac{1}{k}\sum_{t=1}^{k}\big(|x(t) - x_s(t)|_Q^2 + |u(t) - u_s(t)|_R^2 + |\Delta u(t)|_S^2 \big)$$

- Performance degradation compared to the optimal MPC defined as

$$\%\ \text{Performance loss} = 100(\Lambda_{N_t}^{\text{F}} - \Lambda_{N_t}^{\text{MPC}})/\Lambda_{N_t}^{\text{MPC}}$$

- Average and worst case online speedups compared to the optimal MPC
- Data generation and NN training times
- Memory required to store the weights and biases of the trained NNs

Here, N_t denotes the number of time steps in the closed-loop simulation, $\Lambda_{N_t}^{\text{F}}$ and $\Lambda_{N_t}^{\text{MPC}}$ denote the controller performance indexes at the end of the simulation period for the fast and optimal MPC controllers. The data generation and online timing comparisons are performed on a computing cluster. We use the cluster to parallelize the data generation on different cores within a CPU and across multiple CPUs as well. All the NN training are performed on a GPU with a 32 GB memory.

3.6.1 CSTRs in series with a flash separator

We consider a plant containing a series of CSTRs and a flash separator as the first example. The schematic of the plant is depicted in Figure 3.5. The feed stream to the reactors contains primarily the species A. Both the CSTRs facilitate the reactions below to produce a desired product B and an undesired side product C

$$\text{A} \xrightarrow{\text{r}_1} \text{B}, \quad \text{B} \xrightarrow{\text{r}_2} \text{C} \tag{3.31}$$

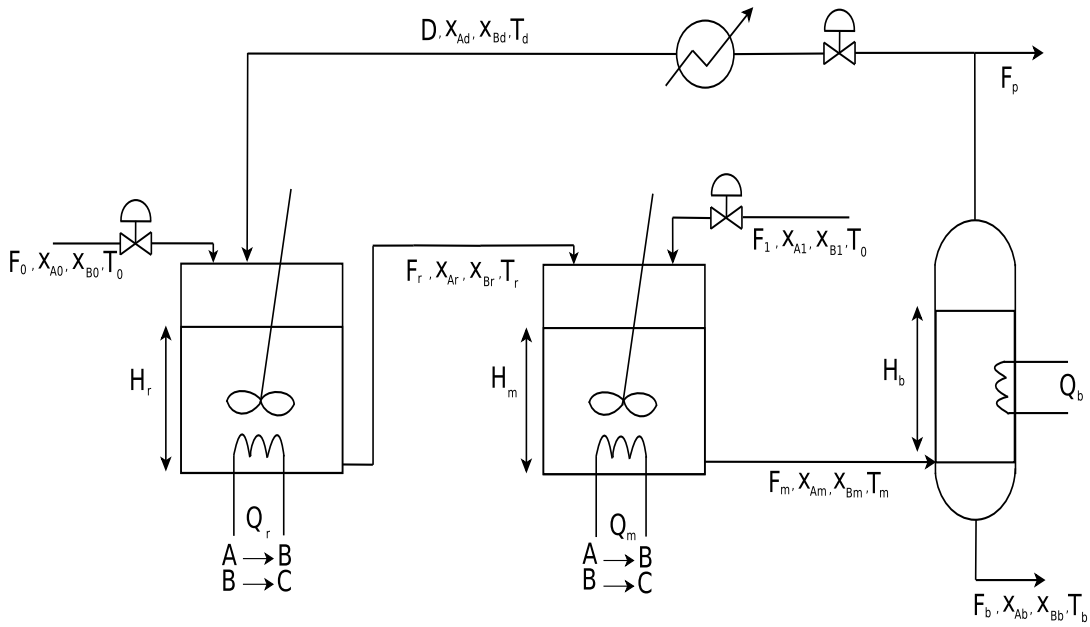

Figure 3.5: A diagram of the CSTRs in series with a flash separator system.

The outlet stream of the second CSTR is supplied to a non-adiabatic flash that separates the reaction mixture in two streams each containing a majority of the reactant A and product B. The vapor stream contains primarily the reactant A and is recycled back to the first CSTR. The dynamic model for the plant simulation is obtained from Venkat (2006) (Appendix), which are also provided in Appendix 3.7 of this chapter. The plant has 12 states ($H_r, x_{Ar}, x_{Br}, T_r, H_m, x_{Am}, x_{Bm}, T_m, H_b, x_{Ab}, x_{Bb}, T_b$), 6 control inputs ($F_0, F_1, D, Q_r, Q_m, Q_b$), and 5 disturbances ($x_{A0}, x_{A1}, x_{B0}, x_{B1}, T_0$). The controlled measurements for the MPC controller are the heights and temperatures in the two CSTRs and the flash separator. We assume that all the states are measured for the simulation studies, and the sample time for the measurements is 10 seconds. We also provide the parameters for the plant model and the range of controlled measurement setpoints and disturbances used for the NN training data generation and validation simulations in Table 3.3 in Appendix 3.7.

For the MPC controller design, we obtain a linear discrete time model of the plant

around the steady state given in Table 3.3. We use the sample time of 10 seconds to obtain this discrete time model. We convert all the control inputs and outputs in this model for the MPC controller such that the input constraints satisfy $\overline{u} - \underline{u} = \mathbf{2}$. The MPC regulator tuning parameter are chosen as $Q = 10^3 C'C$, $R = 0.1 I_6$, and $S = 0.1 I_6$. The forecasting horizon length used in the MPC is $N = 450$ (75 minutes). We generate a total of 1.5×10^5 training data samples by simulating the linear model using the MPC controller based on PRBS signals of setpoints and disturbances sampled in the range given in Table 3.3. The signals are sampled such that enough transient and steady-state data is contained in the training data. We collect samples of the variables $x, u_{-1}, x_s, u_s, \kappa_N(\cdot)$ for the NN training. The data generation step was parallelized in 100 separate offline simulations. We assume that the setpoint of the controlled measurement H_m, i.e, the height of the second CSTR, is fixed in the training data. We change this setpoint in the validation simulation when examining the performance of NN controllers in the unexpected plant behavior scenario.

We consider three structured NNs based on the generated training data. All the three NNs contain three hidden layers with 448, 480, 512 nodes in the layers. These NNs are denoted subsequently as NN-3-448, NN-3-480, and NN-3-512. For closed-loop validation simulations using the estimated controllers and the plant, we use two sets of PRBS setpoints and disturbance signals for a total 4320 timesteps (12 hours). The two sets of signals correspond to the expected and unexpected plant behavior scenarios. We develop a short horizon MPC controller with $N = 10$. We also train an unstructured NN that contains 3 hidden layers, and has 224 nodes in each layer. The details of all the NN controllers considered in the simulation study are also summarized in Table 3.1. The variables x and x_s are additionally scaled for the NN training as follows $x := 2x/(x_{\text{max}} - x_{\text{min}})$ and $x_s := 2x_s/(x_{\text{max}} - x_{\text{min}})$. Here, x_{max} and x_{min} are the maximum and minimum values of the states observed in the entire training data. The scaling is

performed during the online NN execution as well when computing the control input in real time.

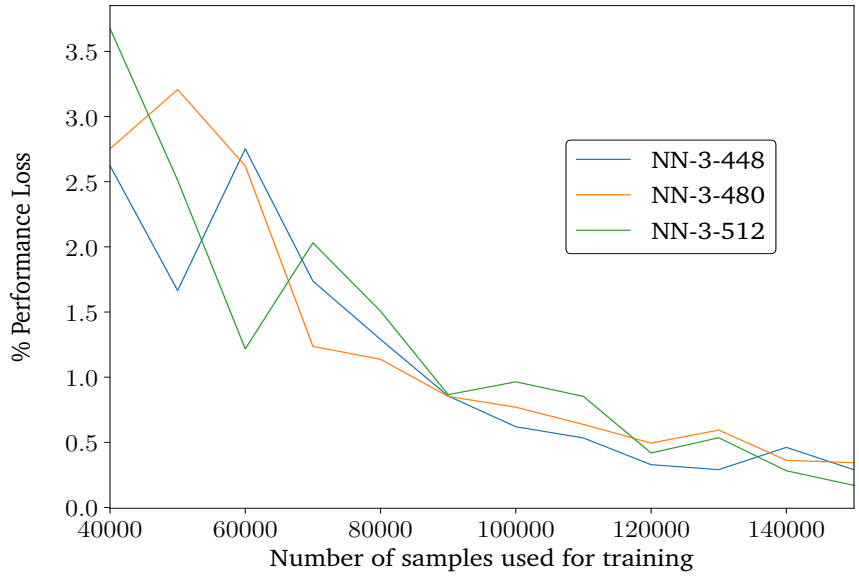

Figure 3.6: Percentage performance losses of the NN controllers with varying amounts of training data (CSTRs in series with a flash example).

The data requirements to obtain an accurate NN controller is studied first. We train structured NNs with the three chosen architectures for varying amounts of the training data starting from 4×10^4 to $1..5 \times 10^5$ samples, in increments of 10^4 samples. To train all the controllers, we use a batch size of 1024 samples in the algorithm Adam. Each training session of the estimated controllers is stopped after 2000 epochs. For each NN training performed, we use 10% of the training data as the "holdout" data set. The mean squared error (MSE) metric is examined on this holdout data set and the NN parameters that have the best loss metric on the holdout set are chosen for the final validation simulations. We use all the trained NNs and the optimal MPC controller in validation simulations using the setpoints and disturbance signals that were generated for the expected plant behavior case. Figure 3.6 shows the variation in the performance

loss of the trained NNs compared to the optimal MPC for varying amounts of training data. The trained NNs have large performance losses with fewer amounts of data, but the performances improve noticeably with an increase in the training data. After about 9×10^4 training samples, all the NNs provide less than 1% performance loss. We summarize the best loss metric obtained by the three NN architectures in Table 3.1.

Next, we use the NNs that have the best performance losses above in another validation simulation with setpiont and disturbance signals that represent an unexpected plant behavior. The performance losses obtained in these validation simulations are also summarized in Table 3.1. We observe that the performances of the NNs deteriorate significantly from less 1% to $5 - 8\%$. The performance degradation is particularly because of some unseen samples of the variables x, u_{-1}, x_s, u_s resulting from the setpoint changes in the height of the second CSTR, which was kept constant during the training data generation. This study illustrates that the NN controllers should not be expected to generalize beyond the data used for training. All the setpoints and disturbances that are anticipated to change during the online plant operation should be sampled in the training data.

We also visualize the transient closed-loop trajectories obtained with the estimated NNs compared to the optimal MPC. Figures 3.7 and 3.8 shows the closed-loop control input and controlled measurement trajectories obtained using the NN-3-448 and optimal MPC controllers. The trajectories are shown for the expected plant behavior case for a period of 2 hours, and we observe that the NN provides almost indistinguishable performance compared to the optimal MPC controller. We also plot the controller performance index Λ_k for the three trained NNs, optimal MPC, SS, and satK controllers in Figure 3.9 for the full simulation period of 12 hours. The performance index plots also highlight that the NNs provides almost the same performance as the optimal MPC controller. The degradation in the performance of the NN controllers can be quantified

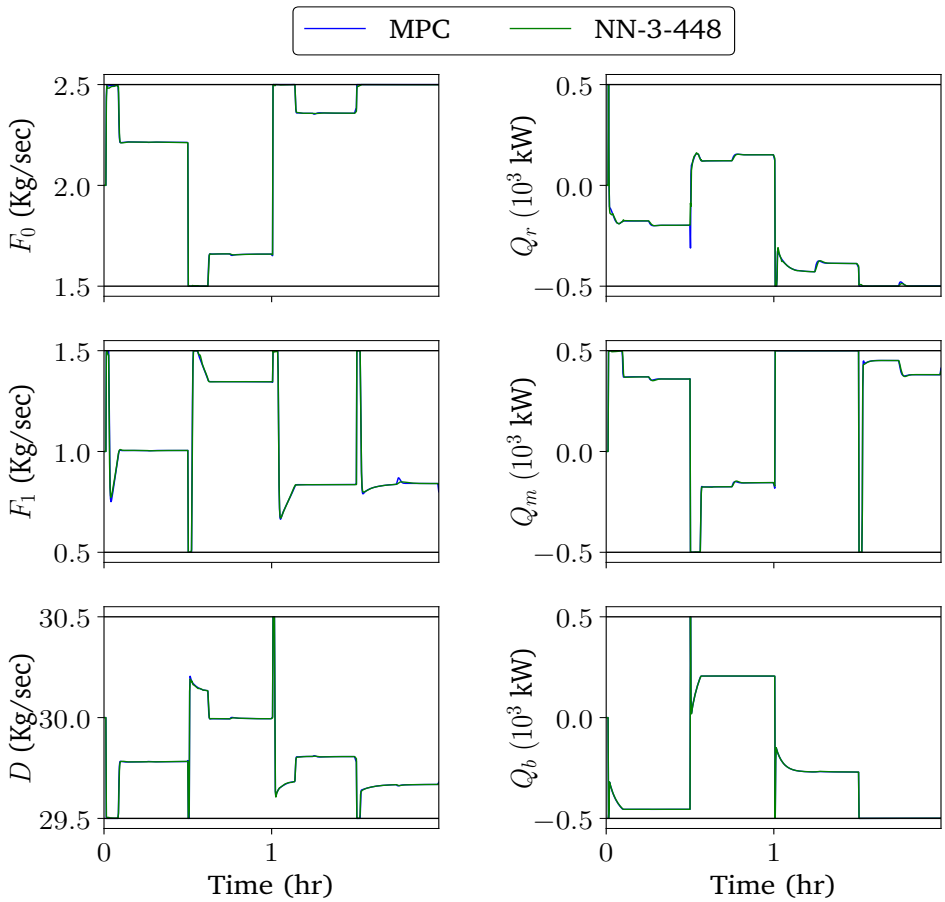

Figure 3.7: Closed-loop control input trajectories of the CSTRs with a flash separator plant under feedback with a trained NN and the optimal MPC controller. The performances of the NN and optimal MPC controllers are almost the same.

using the loss metric, which is reported in Table 3.1.

Further, we analyze the online computational benefits of the trained NNs over the online QP based MPC controller. We plot the histograms of the online computational times of the NNs and optimal MPC obtained in one validation simulation in Figure 3.10. In addition, we also compute the average and worst-case speedups over the online QP based MPC and report these metrics in Table 3.1. The QP solver requires around 8-13 seconds in this example. By contrast, the NNs require only around 0.1 to

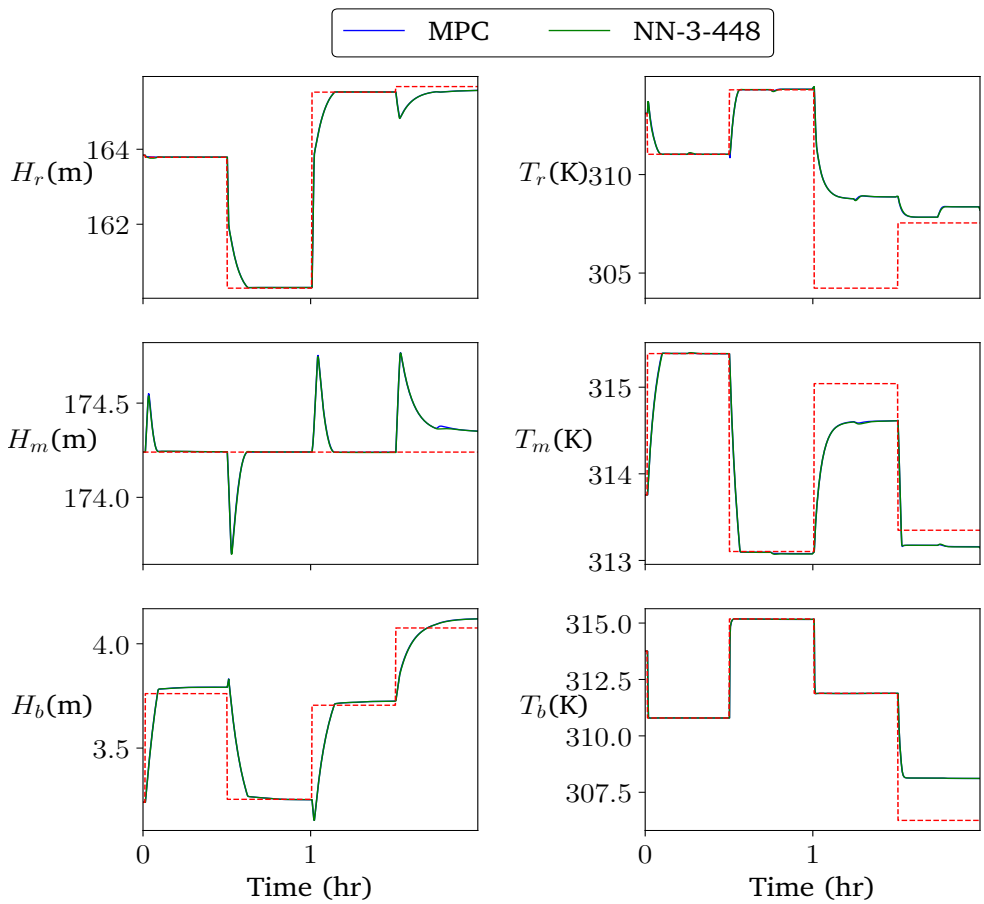

Figure 3.8: Closed-loop trajectories of the controlled measurements of the CSTRs with a flash separator plant under feedback with a trained NN and the optimal MPC controller. The performances of the NN and optimal MPC controllers are almost the same.

4 milliseconds to compute the control input in real time, and are easily deployable in the available sample time of 10 seconds. The NNs execute MPC around 4-5 orders of magnitude faster than the QP solver used for comparison.

In Table 3.1, we also summarize the performance loss metric obtained in the validation simulation (expected plant behavior case) using the other four heuristic controllers, their corresponding speedups compared to the optimal MPC, the number of parameters in all the NNs, and the memory required to store the weights and biases in

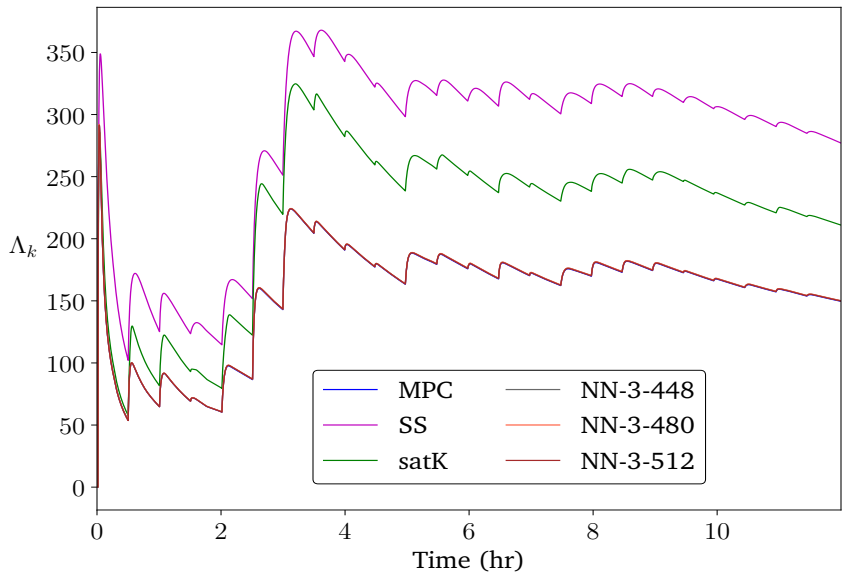

Figure 3.9: Controller performance index Λ_k obtained in the validation simulation using the three trained NNs, optimal MPC, SS, and satK controllers (CSTRs in series with a flash example).

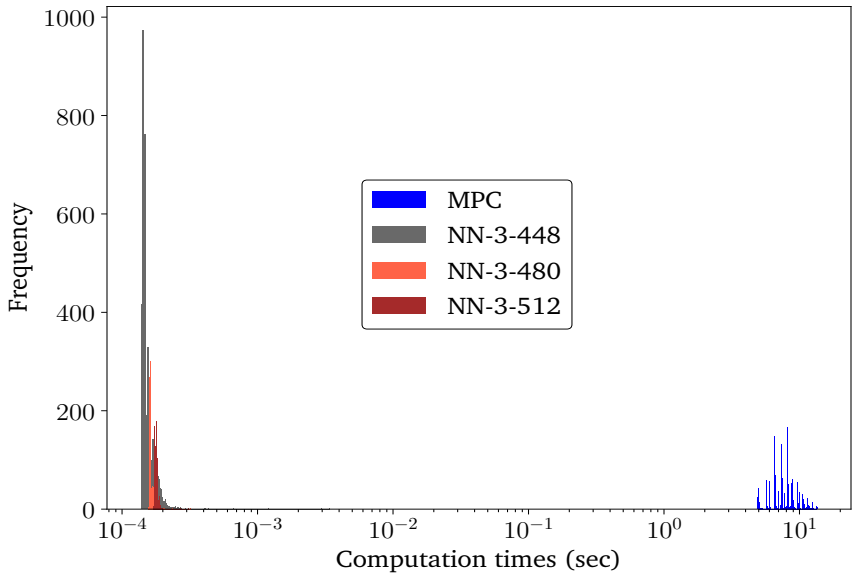

Figure 3.10: Histograms of the online computation times for the optimal MPC and the three NN controllers (CSTRs in series with a flash example).

Controller	Neural network architecture	Number of Parameters	Memory footprint (MB)	Training time (min)	% Performance losses Expected, Unexpected plant behaviors	Average speedup	Worst case speedup
SS					85.18%, 106.39 %		
satK					41.03%, 27.61 %	2.47×10^5	3.07×10^5
SH					1.61%, 2.46 %	3.75×10^3	8.4×10^1
NN-UNS	[36, 224, 224, 224, 6]	1.10×10^5			80.49%, 73.29 %	9.31×10^4	4.99×10^3
NN-3-448	[36, 448, 448, 448, 6]	1.10×10^5	0.84	13.02	0.28 %, 5.57 %	5.26×10^4	1.45×10^3
NN-3-480	[36, 480, 480, 480, 6]	1.26×10^5	0.96	12.51	0.34 %, 7.58%	5.09×10^4	1.53×10^4
NN-3-512	[36, 512, 512, 512, 6]	1.42×10^5	1.08	12.77	0.16 %, 5.90 %	4.62×10^4	1.58×10^4

Table 3.1: Metrics summarizing the simulation study performed for the CSTRs in series with a flash separator example.

the NNs. The memory required to store the NN parameters are approximately around 1 MB, which show that the NNs can be conveniently deployed on memory constrained hardware. The memory footprint can be crucial during deployment of NN controllers at fast low-level regulatory control layers where cheap hardware is used in applications.

We also observe in Table 3.1 that the unstructured NN provides a poor feedback control performance with a loss metric of 80%. Such a performance is unacceptable for an industrial application, and demonstrates that learning the feedback law without the structure proposed in Section 3.4 is a much more challenging problem. In the proposed architecture, the NN has the correct MPC feedback law at the steady-states and the feedback law approximation problem simplifies to improving the NN control at only the transient states.

3.6.2 Industrial crude distillation unit

We next demonstrate the scalability of the NN controller design approach on a large-scale industrial crude distillation unit example. The dynamic model of the plant was

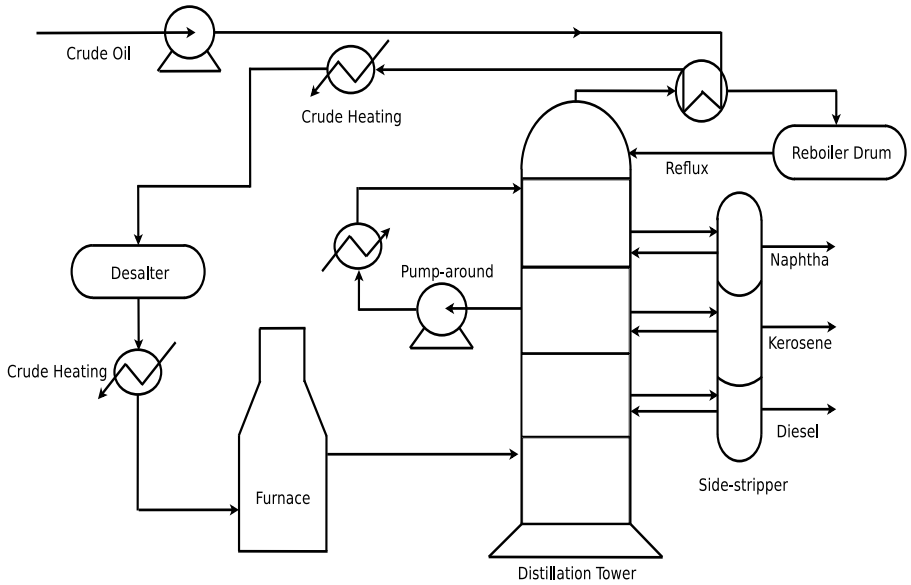

Figure 3.11: A diagram of the industrial crude distillation unit plant.

obtained from AspenTech, and was also used in Pannocchia et al. (2007) to examine the scalability of the partial enumeration approach for large MPC applications. The schematic of the plant is shown in Figure 3.11, which is a typical crude distillation unit found in petrochemical refineries. The plant model is linear, and we scale the control inputs and measurements for the MPC controller similar to the previous example. The model has 252 states, 32 control inputs, and 90 measurements. Only four out of all the measurements that represent the quality of the crude side products have setpoints. We use five disturbances on the crude composition, fuel gas quality, and steam header pressure. The sample time for the measurements is 1 minutes. The MPC regulator tuning parameters are chosen as $Q = 2C'C$, $R = 0.1I$, and the forecasting horizon length is $N = 140$ (around 2.5 hours). We generate a total of 3.6×10^6 training data samples, which are obtained after performing 149 parallel closed-loop simulations. The time consumed for the overall training data generation was 27.8 hours.

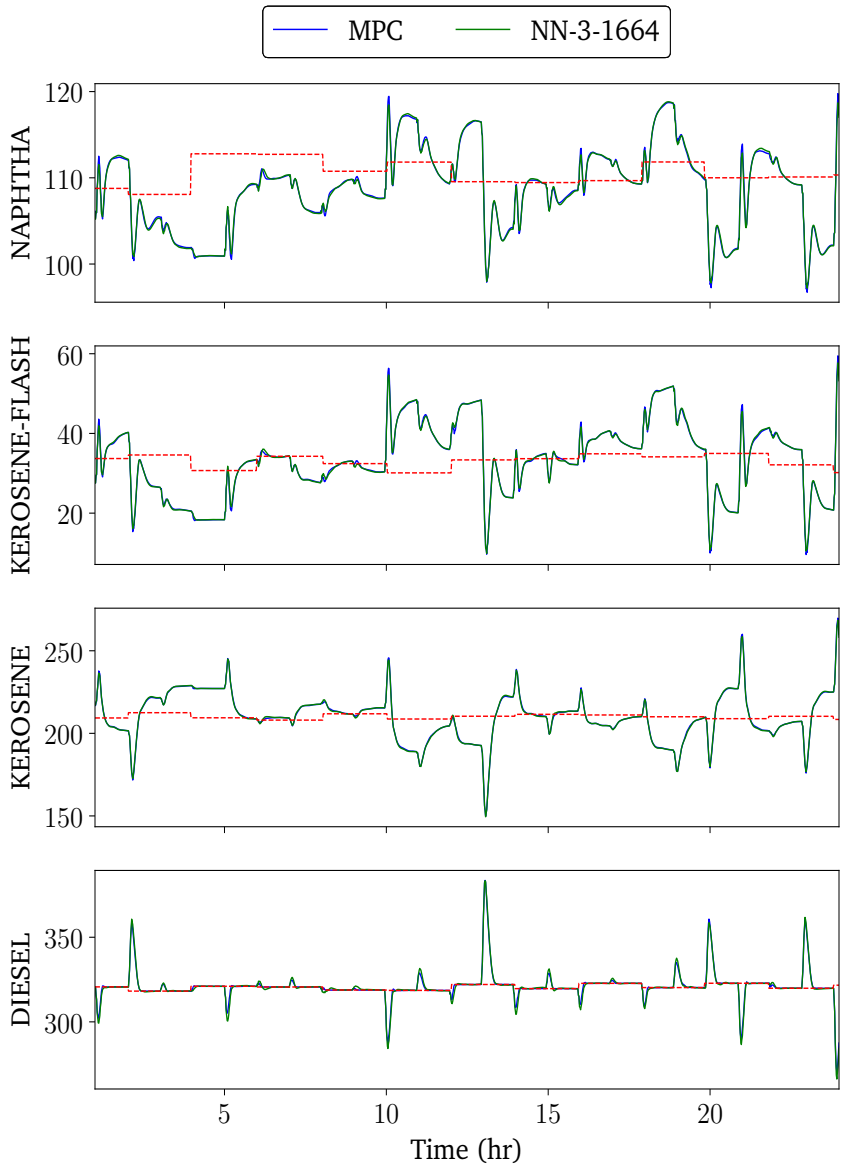

Figure 3.12: Closed-loop trajectories of the controlled measurements in the crude distillation unit plant under feedback with a trained NN and the optimal MPC controller.

We consider three structured NN architectures for this example. Each NN has three hidden layers, and we use 1664 (NN-3-1664), 1792 (NN-3-1792), and 1920 (NN-3-1792) nodes in the hidden layers. To perform validation simulations with the estimated

controllers, we generate one set of setpoint and disturbance signals containing a total 2880 timesteps (2 days). The short horizon MPC controller is developed using $N = 3$. The states and targets (x, x_s) are scaled for the NN controller training similar to the previous example.

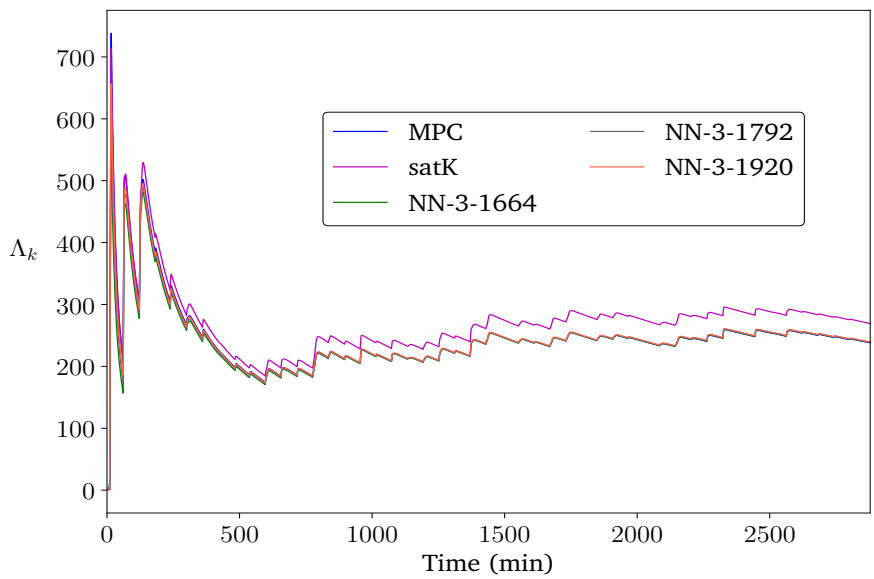

Figure 3.13: Controller performance index Λ_k of the NNs, optimal MPC, and satK controllers in the validation simulation (Crude distillation unit example).

First, we examine the closed-loop trajectories obtained with a trained NN and optimal MPC controllers in a validation simulation. Figure 3.12 shows the transient behavior of the four controlled measurements in the plant under feedback with a NN and the optimal MPC controller. The closed-loop simulation is shown for a period of 1 day. We observe that both the controllers obtain similar closed-loop profiles of the controlled measurements, and their performances are almost the same. We note that three controlled measurements in Figure 3.12 remain far away from their setpoints at most times. This behavior is particularly because we use large sized disturbances in the simulation. The large disturbances activate more input constraints, which deteriorates

the performance of the heuristic satK controller and illustrates the value of using NNs to approximate the complex MPC feedback law. We also plot the controller performance index (Λ_k) obtained with all the NNs, the optimal MPC, and the satK controller for the full simulation period of 2 days. Figure 3.13 shows this plot, and we observe that all the trained NNs provide a similar performance as the optimal MPC.

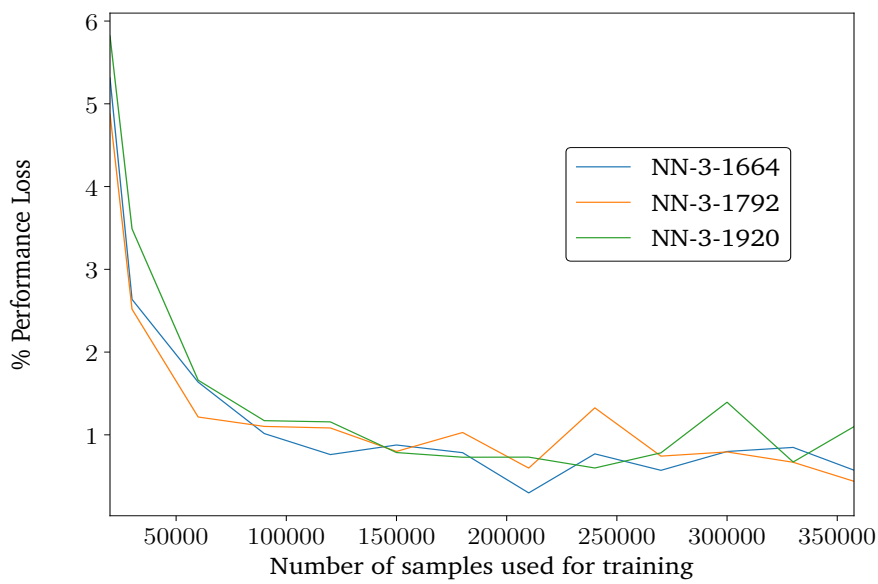

Figure 3.14: Percentage performance losses of the NN controllers with varying amounts of training data (Crude distillation unit example).

Further, we examine the data requirements to obtain an accurate NN controller and online computational benefits of using the NNs. Figures 3.14 and 3.15 examine these two aspects. For all the NN training performed for this example, we use a batch size of 2048 in the algorithm Adam, and the NNs were trained for a total of 1500 epochs. We use 5% of the overall training data as the holdout data set. In Table 3.2, we summarize all the metrics for the simulation study for this example such as the best performance loss obtained by the NNs in the validation simulation, the maximum time required to train the NNs, the number of parameters in the NNs, the memory footprint to store the

weights and biases, and the loss metrics of the other heuristic controllers.

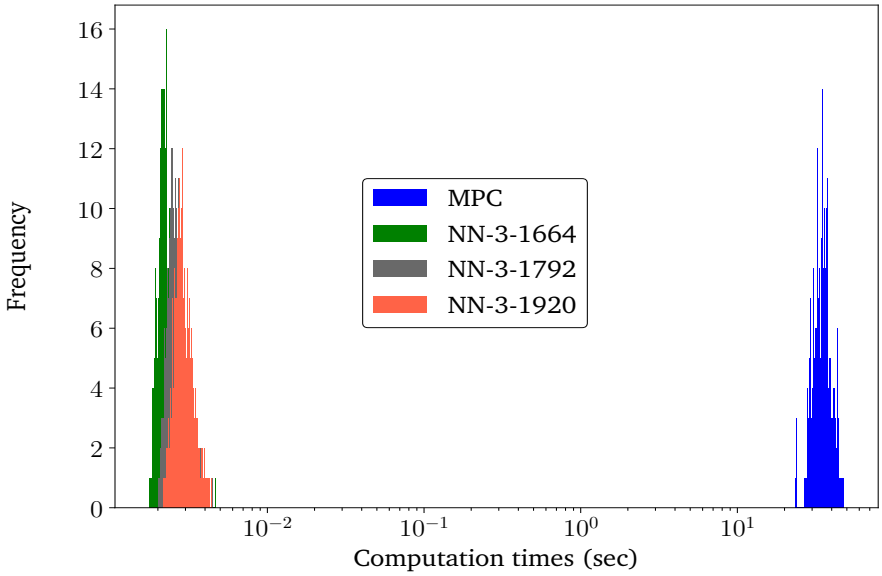

Figure 3.15: Histograms of the online computation times of the NNs and optimal MPC controllers (Crude distillation unit example).

The performance losses obtained by the NN controllers reported in Table 3.2 are in the range $0.29-0.62\%$, which illustrate that the NNs have obtained accurate approximation of the MPC feedback law in the operationally relevant state space of interest. The offline data generation and training times highlight that the NN controllers can be designed tractably for this large example. This particular example is challenging, because the QP solver CVXOPT requires an average 35 seconds, and 47 seconds in the worst case to solve each MPC optimization problem. In comparison, all the NNs execute MPC in just 2 to 7 milliseconds. The NNs provide around 4 orders of magnitude speedups compared to the QP solver. The performance of the NNs is almost indistinguishable compared to the optimal MPC controller. The main reasons for scalability of the NN design approach in this large example are (i) sampling the state space using only the relevant setpoint and disturbance signals, and (ii) using the structured architecture for

offset-free control.

Controller	Neural network architecture	Number of Parameters	Memory footprint (MB)	Training time (min)	% Performance loss	Average speedup	Worst case speedup
SS					120.59 %		
satK					13.07 %	4.83×10^5	1.51×10^5
SH					1.56 %	6.66×10^3	2.51×10^3
NN-3-1664	[1072, 1664, 1664, 1664, 32]	1.85×10^6	14.18	52.66	0.29 %	1.52×10^4	6.57×10^3
NN-3-1792	[1072, 1792, 1792, 1792, 32]	2.11×10^6	16.15	55.27	0.43 %	1.33×10^4	5.25×10^3
NN-3-1920	[1072, 1920, 1920, 1920, 32]	2.39×10^6	18.24	59.24	0.59 %	1.18×10^4	4.48×10^3

Table 3.2: Metrics summarizing the simulation study for the industrial crude distillation unit example.

3.7 Conclusions

In this chapter, we have presented a feedback controller design approach using neural networks to approximate the MPC feedback law for large-scale applications. The scalability of the proposed approach has been demonstrated using large application examples that may be challenging with available QP solvers.

We started with examining the MPC feedback law approximation approach on a small scale double integrator example in which both the optimal and NN feedback laws can be visualized. We illustrated in this example that NNs have the ability to obtain a good approximation of the optimal MPC feedback law.

The NN controller design approach was next applied to approximate the feedback law for the offset-free MPC formulation, in which the feedback law becomes also a function of the target steady-state pair in addition to the initial state. We introduced a new structured NN architecture that encodes the steady-state MPC feedback law regardless of the choice of weights and biases in the NN. To generate the training data,

we proposed that only the relevant state space based on anticipated setpoints and disturbances should be sampled for NN training. The application of the proposed NN design approach has been demonstrated on a large-scale industrial crude distillation unit example with upto 252 states, 32 control inputs, and an MPC forecasting horizon length of 140. The trained NN controllers execute MPC around 4 orders of magnitude faster than an available QP solver, with less than 1% closed-loop performance loss compared to the optimal MPC.

The key features for scalability to the large examples is sampling only the operationally relevant state space, and using the structured architecture for offset-free control. The proposed methods that avoid online optimization to date in the literature, such as storing all the regions in the MPC feedback law or sampling the entire state space for NN training have not been demonstrated to scale to the size of applications presented in this chapter.

The online optimization based approach has been adopted by practitioners over the years due to its ability to handle multivariable systems, process constraints, and has strong stability and robustness properties. The requirement of determining the anticipated setpoints and disturbances, and more computing hardware for offline training in large applications are additional complexities that must be considered for the design of NN controllers. The advantages of NNs are their fast online MPC execution times, and the suitability of deployment on memory constrained hardware. We discuss future research directions in Chapter 6.

This chapter concludes the contributions of this thesis on feedback controller design methods using machine learning. In the next two chapters, we develop approaches to combine neural networks with first principles process knowledge to develop hybrid dynamic models from data for use in real time optimization and control.

Appendix

CSTRs in series with a flash separator model details

The following nonlinear ODEs (Venkat (2006), Appendix) are used to simulate the plant in the CSTRs in series with a flash separator example presented in Subsection 3.6.1.

CSTR – 1

$$\frac{dH_r}{dt} = \frac{F_0 + D - F_r}{\rho A_r} \tag{3.32}$$

$$\frac{dx_{Ar}}{dt} = \frac{F_0(x_{A0} - x_{Ar}) + D(x_{Ad} - x_{Ar})}{\rho A_r H_r} - k_{1r} x_{Ar} \tag{3.33}$$

$$\frac{dx_{Br}}{dt} = \frac{F_0(x_{B0} - x_{Br}) + D(x_{Bd} - x_{Br})}{\rho A_r H_r} + k_{1r} x_{Ar} - k_{2r} x_{Br} \tag{3.34}$$

$$\frac{dT_r}{dt} = \frac{F_0(T_0 - T_r) + D(T_d - T_r)}{\rho A_r H_r} - \frac{(k_{1r} x_{Ar} \Delta H_1 + k_{2r} x_{Br} \Delta H_2)}{C_p} + \frac{Q_r}{\rho A_r C_p Hr} \tag{3.35}$$

CSTR – 2

$$\frac{dH_m}{dt} = \frac{F_r + F_1 - F_m}{\rho A_m} \tag{3.36}$$

$$\frac{dx_{Am}}{dt} = \frac{F_r(x_{Ar} - x_{Am}) + F_1(x_{A1} - x_{Am})}{\rho A_m H_m} - k_{1m} x_{Am} \tag{3.37}$$

$$\frac{dx_{Bm}}{dt} = \frac{F_r(x_{Br} - x_{Bm}) + F_1(x_{B1} - x_{Bm})}{\rho A_m H_m} + k_{1m} x_{Am} - k_{2m} x_{Bm} \tag{3.38}$$

$$\frac{dT_m}{dt} = \frac{F_r(T_r - T_m) + F_1(T_0 - T_m)}{\rho A_m H_m} - \frac{(k_{1m} x_{Am} \Delta H_1 + k_{2m} x_{Bm} \Delta H_2)}{C_p} + \frac{Q_m}{\rho A_m C_p H_m} \tag{3.39}$$

Flash Separator

$$\frac{dH_b}{dt} = \frac{F_m - F_b - D - F_p}{\rho A_b} \tag{3.40}$$

$$\frac{dx_{Ab}}{dt} = \frac{F_m(x_{Am} - x_{Ab}) - (D + F_p)(x_{Ad} - x_{Ab})}{\rho A_b H_b} \tag{3.41}$$

$$\frac{dx_{Bb}}{dt} = \frac{F_m(x_{Bm} - x_{Bb}) - (D + F_p)(x_{Bd} - x_{Bb})}{\rho A_b H_b} \tag{3.42}$$

$$\frac{dT_b}{dt} = \frac{F_m(T_m - T_b)}{\rho A_b H_b} + \frac{Q_b}{\rho A_b C_p H_b} \tag{3.43}$$

The mass fractions of the species A and B in the vapor phase in the flash separator are computing using the relations

$$x_{Ad} = \frac{\alpha_A x_{Ab}}{\alpha_A x_{Ab} + \alpha_B x_{Bb} + \alpha_C(1 - x_{Ab} - x_{Bb})} \tag{3.44}$$

$$x_{Bd} = \frac{\alpha_B x_{Bb}}{\alpha_A x_{Ab} + \alpha_B x_{Bb} + \alpha_C(1 - x_{Ab} - x_{Bb})} \tag{3.45}$$

The flow rates of the outlet streams from each unit and the purge stream are computed using

$$Fr = k_r\sqrt{H_r}, \quad F_m = k_m\sqrt{H_m} \tag{3.46}$$

$$Fb = k_b\sqrt{H_b}, \quad F_p = 0.01D \tag{3.47}$$

The rate constants for the two reactions depend on the temperatures in each reactor as follows

$$k_{1r} = k_1^\star e^{(-E/(RT_r))}, \quad k_{2r} = k_2^\star e^{(-E/(RT_r))} \tag{3.48}$$

$$k_{1m} = k_1^\star e^{(-E/(RT_m))}, \quad k_{2m} = k_2^\star e^{(-E/(RT_m))} \tag{3.49}$$

The parameters used for the plant ODEs and the above expressions are provided in Table 3.3.

Parameters used to simulate the ODEs					
Parameter	Value	Unit	Parameter	Value	Unit
α_A	3.5		k_m	2.5	m^2
α_B	1.1		k_b	1.5	m^2
α_C	0.5		ΔH_1	-40	kJ/kg
ρ	50	kg/m^3	ΔH_2	-50	kJ/kg
C_p	3	kJ/kg-K	E/R	150	K
A_r	0.3	m^2	$k_1^\star$	4×10^{-4}	sec^{-1}
A_m	2	m^2	$k_2^\star$	1.8×10^{-6}	sec^{-1}
A_b	4	m^2	T_d	313	K
k_r	2.5	m^2			
Actuator constraints $(\underline{u}, \overline{u})$					
F_0	(1.5, 2.5)	kg/sec	Q_r	(-500, 500)	kW
F_1	(0.5, 1.5)	kg/sec	Q_m	(-500, 500)	kW
D	(29.5, 30.5)	kg/sec	Q_b	(-500, 500)	kW
Setpoint bounds $(\underline{r}_{sp}, \overline{r}_{sp})$					
H_r	(158.8, 168.8)	m	T_r	(303, 323)	K
H_m	(169.2, 179.2)	m	T_m	(310, 316)	K
H_b	(2.2, 4.2)	m	T_b	(303, 323)	K
Disturbance bounds $(\underline{d}, \overline{d})$					
x_{A0}	(0.7, 0.85)		x_{B1}	(0, 0.15)	
x_{B0}	(0, 0.15)		T_0	(305, 321)	K
x_{A1}	(0.7, 0.85)				
Steady state used for linearization (x_s, u_s, d_s)					
H_r	163.8	m	F_0	2	kg/sec
x_{Ar}	0.40		F_1	1	kg/sec
x_{Br}	0.54		D	30	kg/sec
T_r	313.1	K	Q_r	0	kW
H_m	174.2	m	Q_m	0	KW
x_{Am}	0.37		Q_b	0	kW
x_{Bm}	0.58		x_{A0}	0.8	
T_m	313.7	K	x_{B0}	0.1	
H_b	3.24	m	x_{A1}	0.8	
x_{Ab}	0.15		x_{B1}	0.1	
x_{Bb}	0.73		T_0	313	K
T_b	313.7	K			

Table 3.3: Parameters used in the ODEs to simulate the plant in the CSTRs in series with a flash separator example, actuator constraints, setpoint and disturbance bounds, and the steady state used to obtain the linear model for the MPC controller.

Chapter 4

Grey-box modeling and disturbance forecasting in building systems

4.1 Introduction

In this current and next chapters, we present hybrid process modeling strategies that utilize both the available first principles knowledge and advantages of neural networks. Building energy systems affected by large, occupancy induced unmeasured heat disturbances are considered as the application of interest in this chapter. For such building systems, we develop a two step model identification framework to estimate an (i) actuator to measurement, and (ii) a disturbance forecasting model from data. We use a grey-box modeling approach to develop the actuator to measurement model, and a neural network (NN) to design the disturbance forecasting model. The intended purpose is to use the two models for improved real time energy cost optimization with an economic model predictive control (MPC) formulation.

Building systems comprise a major portion of the United States energy usage. Available survey in the literature reports almost 40% energy usage by commercial and residential buildings combined (Energy, 2018). Thus, developing advanced modeling and control approaches for buildings is an impactful research topic. For modern electricity

markets with time varying energy prices, several researchers have proposed the use of MPC for feedback control and real time energy cost optimization in buildings (Henze, 2005; Ma et al., 2012a; Oldewurtel et al., 2012; Mendoza-Serrano and Chmielewski, 2012; Patel et al., 2018; Ma et al., 2012b). A major reason of these proposals is because traditional rule-based and proportional-integral (PI) controllers cannot be used to obtain appreciable energy cost savings (Afram and Janabi-Sharifi, 2014).

To deploy an MPC controller in buildings for energy cost minimization, a mathematical model is first developed to describe the heat transfer dynamics between the actuators, measurements, and disturbances. Then, an optimization problem with an energy price cost (rather than a tracking cost considered so far in this thesis) as the objective function is formulated. The problem is solved in real time to determine the optimal actuator moves to apply to the plant. Future predictions of disturbances affecting the building system, desired comfort temperature constraints, and energy prices charged by utility providers are used in the optimization problem.

The closed-loop performance of an MPC controller can depend significantly on the quality of the dynamic model. Several model identification methods have been proposed in the literature to develop dynamic models for building systems (Privara et al., 2013; Drgoňa et al., 2020; Wang and Chen, 2019; Serale et al., 2018). The methods can be classified into white-box, grey-box, and black-box modeling approaches. In the white-box approach, the model is developed using completely first principles knowledge about the building. The parameters in the dynamic model are also determined using the known physical properties about the materials used during the building construction.

As we discussed in Subsection 1.1.1, the grey-box and black-box approaches first parameterize a dynamic model containing some unknown parameters. And a data-fitting problem is solved to estimate those unknown parameters in the model. For grey-box

modeling of buildings, resistance capacitance (RC) networks are often used to describe the temperature dynamics (Madsen and Holst, 1995; Kramer et al., 2012). The developed RC networks can also be viewed as analogies of the heat transfer relationships of the building temperature dynamics. The approach yields a reduced order model of the building, which is often suitable for energy cost optimization with an MPC controller using standard solvers (Patel et al., 2016). To develop a black-box model, one may estimate a NN from data to predict the future measurements based on a recent history of actuator moves and measurements (Shahnazari et al., 2019; Ferracuti et al., 2017). The approach is convenient for model development in applications. However, if a black-box NN is used in the actuator to measurement part of the dynamic model, then the resulting MPC optimization problem can become challenging to solve using standard solvers. In this chapter, we focus on developing a modeling approach that exploits the advantages of both the grey and black-box methods. While also ensuring that the final developed model is suitable for online optimization using standard solvers.

Disturbances such as ambient temperature, solar irradiation, and internal occupancy induced heat load regularly affect the temperature dynamics in building systems. The occupancy induced heat load is often generated by the electrical equipment usage or the human metabolism of occupants inside the building (Balvedi et al., 2018; Yan et al., 2015). All the above disturbances have a noticeable effect on the building temperature dynamics, so their feedforward predictions should be included in the MPC optimization problem when deploying an MPC controller. A grey-box model developed using the RC network approach is often linear between the actuator, disturbances, and measurements. For quadratic stage costs and linear constraints, a quadratic program (QP) is solved by the MPC controller. The MPC does not manipulate the disturbances and the decision variables in the QP are only the actuator moves. Nonlinear machine learning models can be used to predict the disturbances in the MPC optimization prob-

lem, while maintaining a QP for the controller. In this chapter, we use this structural insight and develop a model identification framework to estimate a grey-box actuator to measurement model and a NN to predict the disturbances in the MPC problem.

Among all the disturbances affecting a building system, the ambient temperature and solar irradiation are typically measured. And accurate predictions of these disturbances can also be obtained using meteorological or weather forecasting services (Florita and Henze, 2009). Contrary to these disturbances, the occupancy induced heat load is not measured in range of real building systems. For example, one may not measure the electrical equipment usage in *each zone* of a large building. Inside a classroom full of students in a university building, the heat generated by the combined human metabolism of all the students may not be measured. Thus, the occupancy induced heat load is an unmeasured disturbance that affects real building systems.

An unmeasured disturbance in process data is typically ignored in the model identification problem by standard system identification methods. And an application of such standard methods to a data set collected from building systems can give a poor dynamic model (Kim et al., 2018). Therefore, the unmeasured heat disturbance should be systematically treated in the building model identification step. Feedforward predictions of the disturbances over the future control horizon in the MPC problem should also be used to enhance the controller performance.

The heat disturbance in building systems is often strongly correlated with the occupancy schedules/patterns in the building. The occupancy schedule in turn is correlated with some auxiliary variables such as the time-of-day, type-of-day (workday/holiday), and time-of-year. A first principles based modeling of the heat disturbance can be challenging and cumbersome for different building systems. But a black-box NN can be conveniently developed based on the historical data collected at the building site to approximate the patterns in the heat disturbance. The NN can be trained to use the

auxiliary variables as inputs when predicting the heat disturbance. Finally, feedforward disturbance predictions provided by the NN can be used in an economic MPC controller to achieve higher energy cost savings.

Overview of contributions

With the above motivations, we propose a method to develop (i) a grey-box actuator to measurement model, and (ii) a NN to predict the heat disturbance. Both these models are estimated in two steps so that input excitation and historical operational data sets can be used to estimate the grey-box and NN models, respectively. We treat the unmeasured heat disturbance in the grey-box modeling step by also estimating a piecewise constant signal for the disturbance in the model identification problem. This piecewise constant signal is estimated by introducing additional decision variables in the modeling problem corresponding to some approximate values of the disturbance over a pre-specified zero order hold duration. After the grey-box modeling step, we present a method to determine approximate confidence intervals (CIs) on the physical parameters in the building dynamic model. The CIs can be used to gauge the reliability of the estimated grey-box model for subsequent use in the disturbance model identification and the MPC controller implementation.

Next, we use the estimated grey-box model and routine historical operational data to train a NN heat disturbance model. The NN takes the auxiliary measurements such as the time-of-day, etc as inputs to predict the disturbance. The NN model is estimated by solving a prediction error minimization problem based on the temperature measurements collected from the building system, and without any measurements of the heat disturbance. And the NN *infers* the disturbances required to accurately predict the temperatures during the optimization process. Both the grey-box and NN disturbance models are estimated by solving multistep ahead prediction error minimization

problems using symbolic differentiation based software (Andersson et al., 2019; Abadi et al., 2015).

Finally, we examine performance improvements when the NN disturbance model is used to provide feedforward heat disturbance predictions in an MPC controller. We highlight that an MPC that uses the heat disturbance predictions provides an improved performance compared to an MPC that does not use those predictions.

The effectiveness of the proposed model identification approach is demonstrated via simulation studies with a two time scale building model that has a single heat disturbance source. We note, however, that the proposed approach can also be applied to estimate higher order building models with multiple NNs to predict the disturbance for each zone. In this case, the grey-box building model should be parameterized accounting for all the inter zone heat transfer connections and the disturbance sources. Then, multistep ahead prediction error minimization problems similar to the ones proposed in this chapter may be solved to estimate the grey-box and NN disturbance models.

We next discuss some related work in the literature on building modeling under unmeasured disturbances and disturbance forecasting. We also highlight the novel features of our modeling framework compared to the existing work.

4.1.1 Related literature

Building modeling under unmeasured heat disturbances

Although grey-box modeling of buildings has been studied widely in the literature (Li and Wen, 2014; Harb et al., 2016), only a few works have considered treating the unmeasured heat disturbance in the building model identification step. Kim et al. (2016) uses a linear output disturbance model to account for the disturbance. The state in the disturbance model was used to explain the mismatch between the predic-

tions of the building model (considering no heat disturbance model) and measured temperatures. The work proposes to simultaneously estimate the parameters in both the building dynamic and output disturbance model from data.

Guo et al. (2021) and Coffman and Barooah (2018) introduce another state in the building dynamic model for the heat disturbance. Since this disturbance is typically occupancy generated and varies slowly in time in real building applications, a zero dynamics is assigned on the introduced state. A simultaneous state and parameter estimation problem is next solved to determine the building model parameters and heat disturbances. These works, however, require a heuristic tuning of the noise covariance (often unknown in applications) penalties in the model identification problem.

Ellis (2021) propose to use black-box NNs to model the heat disturbance in the building dynamic model. And simultaneously estimates the parameters in the building model and the NN. Zeng et al. (2021) present another approach to simultaneously estimate a dynamic model and disturbances in the training data. The work estimates a black-box linear discrete time model and a transformed version of the disturbance in a single optimization problem. Physical constraints on some combination of the linear model parameters and regularization penalty on the disturbances are used in the model identification problem.

In this chapter, we present a method to simultaneously estimate parameters in a *grey-box* building model and heat disturbances. We estimate only some approximate values of the heat disturbances over a pre-specified zero order hold duration in the model identification problem. Such an approach has not been previously proposed in the literature. Additionally, we also present a method to compute confidence intervals (CIs) on the physical building model parameters. We demonstrate that the building model can be estimated with reliable accuracy and uncertainties using the approach proposed in this chapter.

Disturbance modeling in building systems

As mentioned previously, a building system is affected by three main sources of disturbances: ambient temperature, solar irradiation, and occupancy induced heat disturbance. Several researchers have studied the advantages of including the information about all these disturbances in the MPC optimization problem. For example, Oldewurtel et al. (2012) and Oldewurtel et al. (2013) examine the benefits of including weather and occupancy information in an MPC controller to achieve improved energy cost savings. Different approaches to predict the weather disturbances have been developed in the literature. The approaches use (i) detailed physical (Thilker et al., 2021), (ii) data-driven machine learning (Papantoniou and Kolokotsa, 2016; Pang et al., 2020), or (iii) stochastic (Ren and Wright, 2002) models.

Most of the weather disturbance forecasting approaches in the literature assume that measurements of the disturbance are available to develop the prediction model. Sensors to measure the ambient temperature and solar irradiation in an area are often available. However, the occupancy induced heat disturbance is not measured in building systems. The size of the disturbance also varies between each building system. These issues pose a challenge when modeling and forecasting the occupancy induced heat disturbance for use in an MPC implementation. Available works that have studied the advantages of including the occupancy information in an MPC controller assume that the true occupancy patterns in the building are known (Oldewurtel et al., 2012), or that special sensors are available to infer the occupancy count in real time (Sangogboye et al., 2017).

For the disturbance modeling approach proposed in this chapter, we do not use any occupancy related information in the identification approach. We use only the estimated building model from the first grey-box modeling step, and the measurements of

building temperature, cooling duty, ambient temperature, and some auxiliary variables such as the time-of-day, etc. The NN infers the required heat disturbance to accurately predict the temperature measurements during the optimization process. The overall two step modeling approach can be conveniently applied to different building systems without any knowledge or measurements of occupancy patterns.

The rest of this chapter is organized as follows. In Section 4.2, we describe the building system and simulation framework used to demonstrate the effectiveness of the proposed modeling framework. Section 4.3 discusses the grey-box modeling approach to simultaneously estimate the physical parameters in the building model and piecewise constant signal for the unmeasured heat disturbance. We also present the approach to compute confidence intervals on the building model parameters in this section. Then in Section 4.4, we present the method to estimate a NN predictor for the unmeasured heat disturbance from historical operational data. The economic MPC formulations used to examine the advantages of including the NN disturbance predictions in an MPC problem are discussed in Section 4.5. Simulation studies to illustrate the effectiveness of the modeling approach, and the advantages of using the heat disturbance predictions in an MPC controller are presented in Section 4.6. Conclusions of this thesis chapter are provided in Section 4.4.

The results and methods in this chapter are to appear in Kumar et al. (2022). The mathematical notation used in this chapter are given in Section 1.3.

4.2 Building system and simulation framework

The building system shown in Figure 4.1 is considered for the simulation studies. The plant is simulated using the ordinary differential equations (ODEs) in equations (4.1) – (4.2). The system has two states (x): T_z and T_m. The states correspond to the

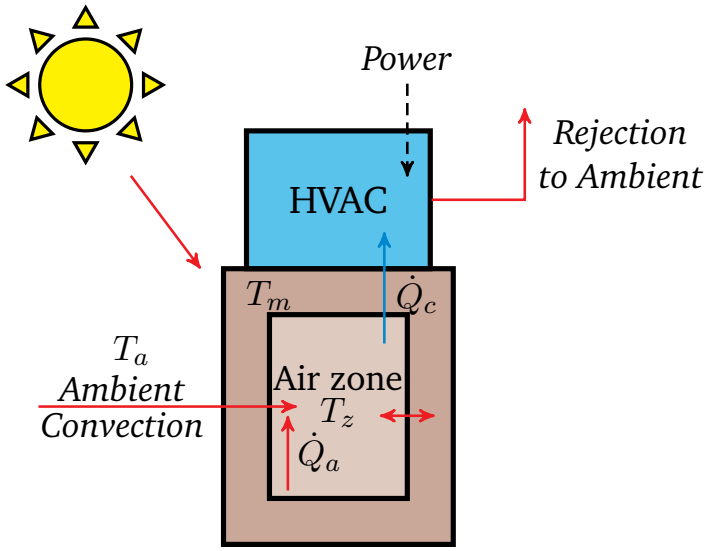

Figure 4.1: Schematic of the building system considered for grey-box and disturbance modeling in this chapter.

temperatures of the building zone and mass. The zone temperature is a representative of the air temperature, while the mass temperature represents a hypothetical temperature of the walls, furniture, etc in the building area. The zone temperature is the only measurement ($y = T_z$). The cooling duty provided to the zone is the manipulated control input ($u = \dot{Q}_c$). The physical parameters in the dynamic building model are, H_m/C_z, H_a/C_z, H_m/C_m, and $1/C_z$. The values of these parameters used to simulate the plant are given in Table 4.1. We choose the parameters such that the building mass has a higher heat capacity than the zone and $C_m > C_z$. The choice leads to the zone temperature evolving on a noticeably faster time scale than the mass temperature. The sample time for the measurements collected from the building system is 1 minutes.

$$\frac{dT_z}{dt} = \frac{H_m}{C_z}(T_m - T_z) + \frac{H_a}{C_z}(T_a - T_z) - \frac{\dot{Q}_c}{C_z} + \frac{\dot{Q}_a}{C_z} \tag{4.1}$$

$$\frac{dT_m}{dt} = \frac{H_m}{C_m}(T_z - T_m) \tag{4.2}$$

The plant has two disturbances, the ambient temperature (T_a) and the occupancy

induced heat load ($\dot{Q}_a$). The ambient temperature is measured while the occupancy induced heat load is an unmeasured disturbance. For simplicity in this chapter, we do not explicitly consider the solar irradiation. And assume that its effect is accounted in the plant dynamics via a high heat transfer coefficient H_a that controls the amount of heat transfer between the ambient and zone temperature. In real building systems, if the solar irradiation is measured and future predictions are also available, then the disturbance can be incorporated in both the model identification and control problems similarly as the ambient temperature.

Parameter	Value	Parameter	Value
H_m/C_z	3.47×10^{-4} sec^{-1}	H_a/C_z	3.70×10^{-4} sec^{-1}
H_m/C_m	5.55×10^{-5} sec^{-1}	$1/C_z$	1×10^{-3} °C/kJ
K_p	-0.2 kW/°C	τ_I	600 sec
$\underline{\dot{Q}_c}$	0 kW	$\overline{\dot{Q}_c}$	8 kW

Table 4.1: Parameters used for the data generation and closed-loop MPC simulations with the building system.

During the data generation process, we assume that a PI controller is installed to manipulate the cooling duty to regulate the zone temperature at some desired setpoints. We use pseudo random binary signals (PRBS) of the zone temperature setpoints when generating the training data. The tuning parameters (K_p, τ_I) in the PI controller and cooling duty constraints ($\underline{\dot{Q}_c}$, $\overline{\dot{Q}_c}$) used during the data generation process are also given in Table 4.1. For the closed-loop simulations performed to examine the advantages of including the occupancy induced disturbance forecasts in the MPC problem, we replace the PI controller and use an MPC controller to directly manipulate the cooling duty. The dry bulb temperature data of the Santa Barbara, California area obtained from NCEI (2019) is used as the ambient temperature in both the data generation and closed-loop MPC simulations.

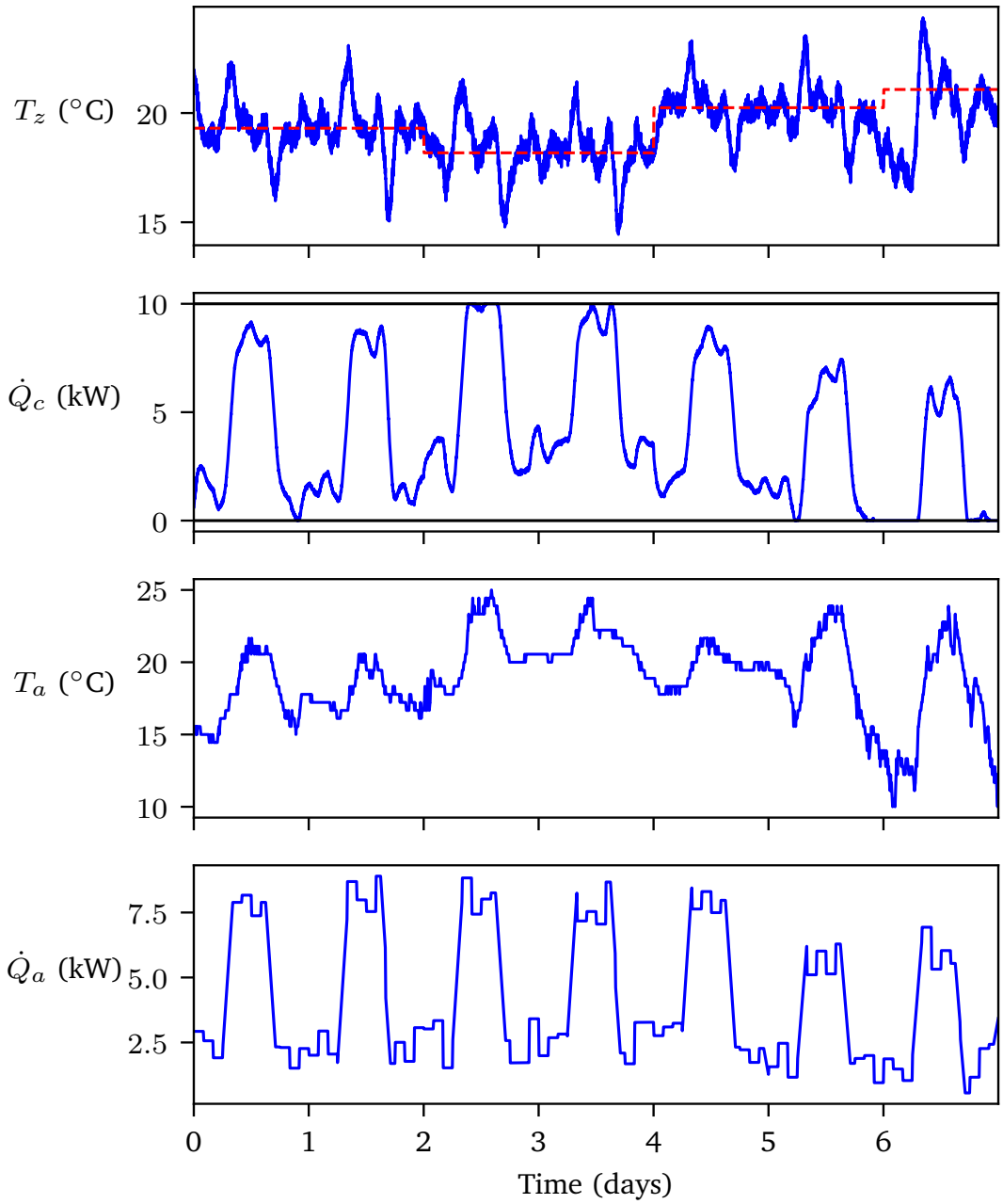

Figure 4.2: Sample one week of training data set generated from the building system to illustrate the weekly pattern in the unmeasured heat disturbance (last row).

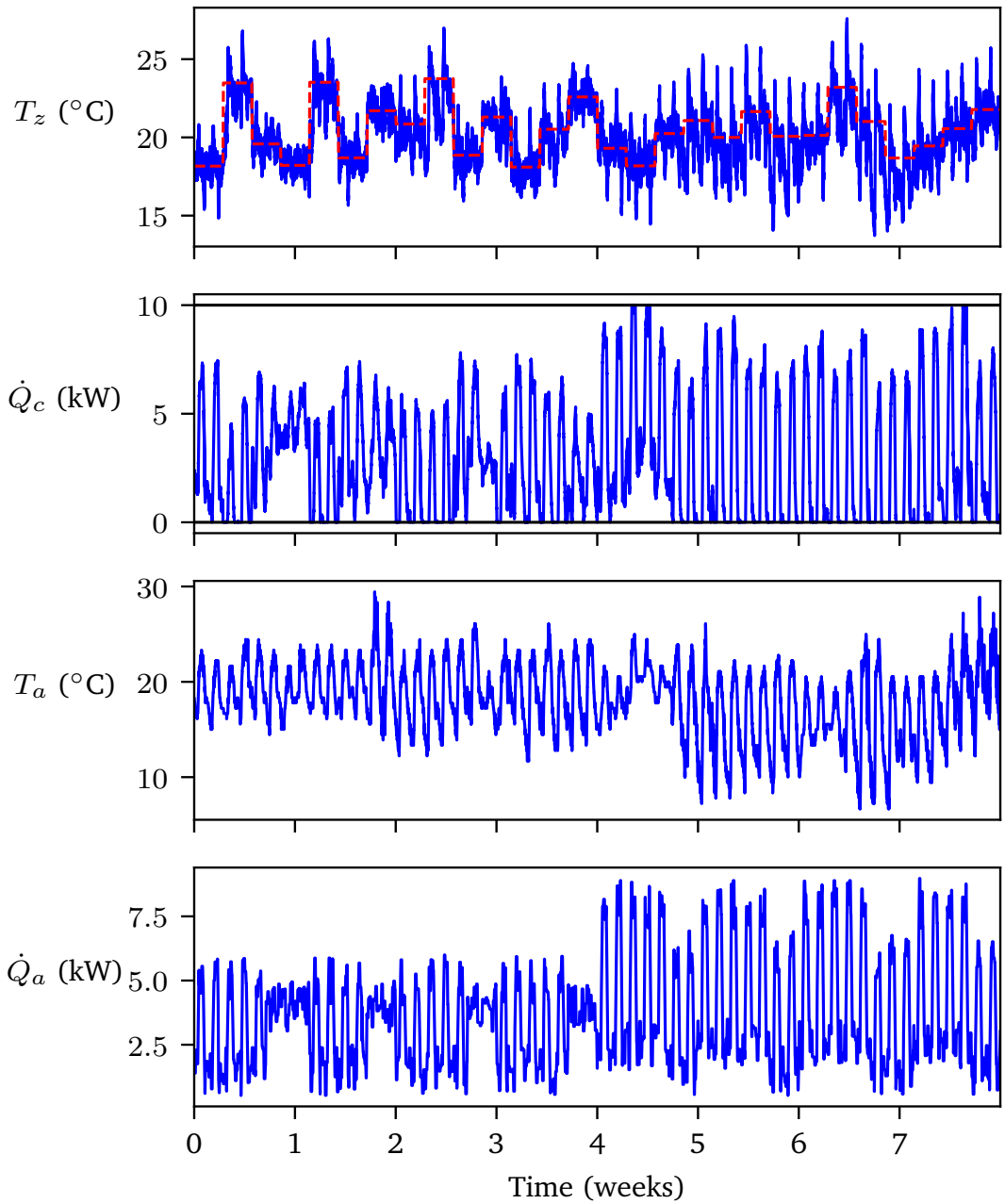

Figure 4.3: Sample eight week of training data set generated from the building system to illustrate the differences in the heat disturbance (last row) patterns between the summer (first four weeks) and regular (last four weeks) university session periods.

To generate the occupancy induced heat load in the case studies, we assume that the building system is a type of university building. And we generate the heat disturbances with some fixed patterns that are a representative of the disturbance in such buildings. The occupancy count in university buildings may depend on the type of semester session. More occupants may be present during a regular semester session, and less during a summer session. The number of occupants may also depend on the type-of-day, i.e, whether a particular day is a regular workday or holiday. Further, the number of occupants also depends on the time-of-day and varies between morning, afternoon, and nights.

With the above background, the heat disturbance in the case studies is generated as a function of three variables: type-of-session, type-of-day, and time-of-day. We first create the disturbance as a deterministic function of these variables. Then, a stochastic component is added to reflect the random nature of the disturbance in real building systems. We generate the heat disturbance profiles such that the difference between the highest and lowest values of the disturbance creates approximately an equal effect as a similar change in the ambient temperature. This approach ensures that in the building system environment studied to demonstrate the efficacy of the proposed model identification framework, both the disturbances have a noticeable contribution to the zone temperature.

Figure 4.2 shows a sample training data set generated from the building system to illustrate the weekly pattern in the heat disturbance. We show one week of training data from a regular university session period in that Figure. We notice that the heat disturbances are larger during the day times, and less during the evenings and nights. The disturbance is also smaller during the weekends compared to the workdays. Figure 4.3 shows an eight week of sample training data to illustrate the difference of the heat disturbance values between the regular and summer session periods. The first four

weeks of data corresponds to a summer session period, and the latter four weeks of data corresponds to a regular university session period. We notice that the disturbance values are lower in the summer compared to the regular session period, which are a representative of the lower and higher occupancy count in the two types of sessions.

4.3 Grey-box building model identification

To estimate a grey-box dynamic model of the building system, we first collect some training data by applying an exciting cooling duty signal to the system. Then, we use the obtained sequences of the zone temperature, cooling duty, and ambient temperature to solve a model identification problem. The goal of this modeling step is to estimate the physical parameters in the building model with a reliable accuracy and uncertainty.

We next discuss the proposed grey-box model identification problem and our approach to compute approximate confidence intervals on the building model parameters. We assume in the model identification problem that the structure of the dynamic model is known, and any unmodeled affect such as heat transfer with an adjacent zone is negligible.

4.3.1 Model identification problem

As mentioned previously, the heat disturbance is often directly correlated with the occupancy schedule of the building. And the occupancy schedule is many real buildings varies slowly in time (Coffman and Barooah, 2018; Zeng et al., 2021), so the true heat disturbance signal can be approximated to a reasonable accuracy using a piecewise constant signal. We estimate the physical parameters in the building dynamic model

and a piecewise constant heat disturbance signal by solving one optimization problem. Additionally, the data collected from real buildings or the building system considered in this chapter may be in the form of one or multiple trajectories. All the data trajectories should be used in the model identification problem to obtain the best possible estimates of the building model parameters. Each trajectory may be affected by a different heat disturbance signal. So we estimate multiple piecewise constant disturbance signals for each trajectory in the model identification problem.

We solve the following optimization problem for the grey-box building model identification

$$\min_{\theta=\{x_i(0),\theta_G,\dot{\mathcal{Q}}_{ia}^j\}} \phi(\theta) = \sum_{i=1}^{N_{tr}} \sum_{k=0}^{N_t-1} (y_i(k) - \hat{y}_i(k))^2 \tag{4.3}$$

$$\text{s. t.} \qquad x_i(k+1) = A(\theta_G)x_i(k) + B(\theta_G)u_i(k) \; + B_p(\theta_G) \begin{bmatrix} T_{ia}(k) \\ \dot{Q}_{ia}(k) \end{bmatrix} \tag{4.4}$$

$$\hat{y}_i(k) = Cx_i(k) \tag{4.5}$$

$$\dot{Q}_{ia}(k) = \dot{\mathcal{Q}}_{ia}^{\lceil k/N_{QaH} \rceil} \tag{4.6}$$

in which, the subscript i is used to refer to the ith trajectory, N_{tr} is the total number of data trajectories, and N_t is the number of time steps in each trajectory. The decision variables (θ) in the optimization problem are the initial model states of each trajectory ($x_i(0)$), the building model parameters (θ_G), and the heat disturbances ($\dot{\mathcal{Q}}_{ia}^j$) that characterize the piecewise constant disturbance signals. We use the subscript j to refer the jth disturbance value in the ith data trajectory. The decision variables for the building model parameters are considered in a logarithm scale, because the corresponding true parameters in the plant may be of significantly different orders of magnitude. The decision variables for these parameters in the logarithm scale facilitates the scaling in

the optimization problem. The other decision variables for the initial states and heat disturbances are considered in standard physical units.

The building model matrices $A(\theta_G)$, $B(\theta_G)$, and $B_p(\theta_G)$ to make the measurement predictions in the optimization problem are nonlinear functions of the decision variables θ_G. The matrices are constructed by first forming the corresponding continuous time matrices, then discretizing exactly (using matrix exponentials) at the sample time of the measurements. The heat disturbance sequence ($\dot{\mathbf{Q}}_{ia}$) in the optimization problem is constructed by extracting appropriate elements from the decision variables as shown in (4.6). And repeating for N_{QaH} number of time steps. The total number of the heat disturbance decision variables in the optimization problem are $N_{tr}N_t/N_{QaH}$. We also solve for the initial model states ($x_i(0)$) of each data trajectory in the problem because the mass temperature is not measured.

The number of time steps N_{QaH} for which each disturbance variable is repeated in the piecewise constant signals is a pre-specified parameter, which also determines the total number of decision variables in the model identification problem. The parameter should be tuned until a good fit to the data is achieved, and the building model parameters are estimated with reliable uncertainties. In the simulation studies presented in Section 4.6, we show that the parameter can be tuned based on the model fit and the computed confidence intervals on the building model parameter estimates.

The problem (4.3) is a multistep ahead prediction error minimization problem. The software CasADi is used to solve the model identification problem. To solve the optimization problem, the software constructs symbolic graphs so that analytical derivatives of the objective function can be computed. The size of the computation graph grows with the number of time steps (N_t) in each data trajectory. The training data collected from the building system should be organized into multiple trajectories so that the number of time steps per trajectory is fewer, which improves the solution time

of the optimization problem.

4.3.2 Confidence interval computation

To evaluate the reliability of the building model parameter estimates obtained by solving the problem (4.3), we now discuss an approach to compute confidence intervals (CIs) on the estimates. For a linear regression problem with a Gaussian random measurement noise, the α-level confidence region can be derived analytically as an ellipsoidal region. In higher dimensions, the ellipsoidal region becomes harder to analyze so the bounding box surrounding the ellipse or the marginal intervals on each parameter estimate can be reported as the CIs.

The problem (4.3), however, is a nonlinear regression problem. In this case, the analytical tractability to determine the statistically correct CIs on the parameter estimates is lost. However, we can compute approximate CIs on the estimates by forming a quadratic approximation of the objective function around the optimum of the nonlinear objective function (Rawlings and Ekerdt, 2020, Page 524-525). The quadratic approximation can be obtained by computing the Hessian of the objective function at the optimum of the nonlinear problem. The Hessian matrix can be used to construct an approximate normal distribution of the parameter estimates. The approximate normal distribution can then be used to derive the α-level ellipsoidal confidence region or the marginal CIs on the parameter estimates. This approach gives approximate CIs on estimates obtained by solving a nonlinear regression problem, and can be highly valuable to practitioners in applications. In this chapter, we employ this approach to compute approximate CIs on the parameter estimates obtained by solving the problem (4.3).

We focus on computing the marginal CIs and for only the building model parameters estimates because they characterize the main dynamic model for the MPC controller.

The α-level marginal CIs on the parameter estimates are computed using the relation

$$\theta_i = \hat{\theta}_i \pm c_i, \tag{4.7}$$

$$c_i = \Big(\frac{2\phi(\hat{\theta})}{n_d - n_p} F_F^{-1}(\alpha; 1, n_d - n_p) H_{ii}^{-1}\Big)^{1/2} \tag{4.8}$$

in which, $\hat{\theta}$ denotes the optimal solution of the optimization problem, $n_d = N_{tr} N_t$ is the number of training data samples used in the problem, and $n_p = 4 + 2N_{tr} + N_{tr} N_t / N_{QaH}$ is the total number of decision variables in the problem. The term $F_F^{-1}(\alpha; 1, n_d - n_p)$ is the inverse of the cumulative F-distribution, at the specified degrees of freedom, at the probability α. The matrix $H \in \mathbb{R}^{n_p \times n_p}$ is the Hessian of the objective function. The Hessian can sometimes become approximately singular, so we add a small positive definite term ϵI to the Hessian (with $\epsilon = 10^{-8}$) so that it is always numerically invertible. The term H_{ii}^{-1} denotes the element at the row and column i in the inverse of the Hessian. We compute the 95% confidence intervals, and $\alpha = 0.95$. Relation (4.7) does not give the CIs in the physical units for the building model parameters, because they are considered in the logarithm scale in the model identification problem. For the case studies in Section 4.6, we convert the CIs on the building model parameters back to the physical units and report the final CIs as percentage deviation from the optimal estimates.

4.4 Neural network disturbance model identification

In this section, we present the proposed approach to estimate a NN disturbance model from data, which can be subsequently used in an MPC controller to provide feedforward predictions of the heat disturbance. The estimation procedure of this NN is not a typical supervised learning problem, because the heat disturbance is not di-

rectly measured. We present an approach to use the measured variables (zone temperature, cooling duty, ambient temperature, and auxiliary information), and the building model estimated from the previous identification step to develop the NN model. We parameterize the NN heat disturbance model as follows

$$\dot{Q}_a = f_N(m; \theta_N) \tag{4.9}$$

in which, m denotes a vector of auxiliary measurements that can be used by the NN to predict the heat disturbance, and θ_N denotes the parameters such as the weights and biases in the NN. The time-of-day, type-of-day, and type-of session information are provided in the vector of auxiliary measurements to the NN. The time-of-day information is provided for each minute of the day as a continuous number between -1 to 1. The approach creates a discontinuity in the time-of-day variable at midnights. But we observe in the simulations that this discontinuity does not significantly affect the predictive capability of the NN model. The one hot encoding (Davis, 2010) approach is used to provide the type-of-day and type-of-session information to the NN model. In an industrial application, if any other variables are posited to have an affect on the heat disturbance, then they can also be included in the vector of auxiliary measurements to provide as input to the NN.

Based on the NN model parameterization, and the estimated building model from the previous identification step, we solve the following problem to estimate the unknown parameters in the NN

$$\min_{\theta_{T_{m0}},\, \theta_N} \sum_{i=1}^{N_{tr}} \sum_{k=0}^{N_t-1} (y_i(k) - \hat{y}_i(k))^2 \tag{4.10}$$

$$\text{s. t.} \qquad x_i^+ = Ax_i + Bu_i + B_p \begin{bmatrix} T_{ia} \\ f_N(m_i; \theta_N) \end{bmatrix} \tag{4.11}$$

$$\hat{y}_i = Cx_i \tag{4.12}$$

$$x_i(0) = \begin{bmatrix} y_i(0) \\ f_{T_{m0}}(z_i; \theta_{T_{m0}}) \end{bmatrix} \tag{4.13}$$

in which, we use the subscript i to refer to the ith data trajectory, N_{tr} is the total number of data trajectories, and N_t is the number of time steps in each trajectory. The linear model matrices A, B, B_p used in (4.11) are obtained from the previous grey-box model identification step.

In the model identification problem, we use another NN $f_{T_{m0}}(\cdot)$ to predict the initial mass temperature for each trajectory. We provide the vector z_i as an input to that NN, which contains a recent set of measurements of the zone temperature, cooling duty, ambient temperature, and auxiliary variables at the beginning of the ith trajectory. The decision variables (θ_N, $\theta_{T_{m0}}$) in the optimization problem are the unknown weights and biases in the disturbance predictor NN, and the other NN introduced to predict the initial mass temperatures at the beginning of each data trajectory. We note that the NN $f_{T_{m0}}(\cdot)$ is not finally used in an MPC controller implementation, so this NN is discarded after the disturbance modeling step.

A common approach in the grey-box parameter estimation literature to treat an unmeasured state in the model identification problem is to also estimate the initial states ($x_i(0)$) by introducing decision variables directly for these states. We choose the

approach described above of introducing another NN $f_{T_{m0}}(\cdot)$ (instead of estimating the initial states) because the stochastic gradient algorithm used to solve the problem does not perform effectively if some decision variables appear only in a few data trajectories. With the proposed approach, only one function ($f_{T_{m0}}(\cdot)$) that remains the same across the training data trajectories is estimated in the optimization problem.

The optimization problem (4.10) is also a multistep ahead prediction error minimization problem. The problem is solved using the algorithm Adam (Kingma and Ba, 2014) with the software TensorFlow (Abadi et al., 2015).

The goal of developing the NN disturbance model is to finally use the NN in an MPC controller. The MPC may be implemented for the operation of the building system for a period of months to year. Real building applications or the system considered in this chapter may be affected by different heat disturbance patterns in different times of year. And the NN should be able to accurately predict the disturbance in the different times of year as well. We collect the training data from the building system such that heat disturbance patterns from all the different times of year are contained in the temperature measurements. The building system is simulated in a routine operation mode to collect the entire training data, since input excitation data is generally not available in applications for longer periods. We show in the simulation studies in Section 4.6 that input excitation data is also not required to obtain a good NN disturbance model.

We scale all the variables to facilitate the scaling of the NN training optimization problem. All the measured variables are scaled using the mean and standard deviation of the variable. For example, we scale the zone temperature as follows

$$y = (y - \mu_y)/\sigma_y \tag{4.14}$$

in which, μ_y is the mean and σ_y is the standard deviation. These two quantities are

computed over the entire training data. The cooling duty and ambient temperature are also scaled similarly. The state evolution equation (4.11) assumes that all the variables are in the physical units. So the mass temperature and the heat disturbance predictions by the NNs in (4.13) and (4.11) should be in the physical units. The weights and biases in the NNs typically have small values during the optimization process. We also scale the outputs of the two NNs using the inverse of the relation (4.14) so that some appropriately scaled quantities are inferred by the NNs. For this additional scaling, we use the mean and standard deviation of the zone temperature measurements to scale the output of the NN $f_{T_{m0}}(\cdot)$. The mean and standard deviation of the cooling duty is used to scale the outputs of the heat disturbance predictor NN $f_N(\cdot)$.

4.5 Economic model predictive control

In this section, we discuss the economic MPC controller formulations used for the closed-loop simulation studies. The purpose of the closed-loop MPC simulations are as follows. First, to examine the achievable performance with an MPC controller that uses the building dynamic and NN disturbance models estimated with the frameworks presented in this chapter. Second, to gauge the performance improvements by an MPC controller that uses the NN heat disturbance forecasts compared to an MPC that does not use those predictions. This second analysis helps to elucidate the achievable benefits of systematically modeling and forecasting the unmeasured heat disturbance in an MPC controller formulation.

We implement and examine the performances of the following three MPC controllers

1. Perfect MPC controller (P-MPC): We use the true building system model and perfect future disturbance predictions in this controller.

2. Feedback MPC (FB-MPC): The building dynamic model estimated with the grey-box modeling step discussed in Section 4.3 is used. We do not use any feedforward predictions of the heat disturbance and the MPC uses only feedback (using an integrating disturbance model) to account for that disturbance.

3. Feedforward Feedback MPC (FFFB-MPC): We use both the building dynamic and NN disturbance models estimated with the approaches discussed in Sections 4.3 and 4.4 for this controller. We also use an integrating disturbance model to account for the plant-model mismatch and any inaccuracies in the disturbance predictions.

All three MPC controllers use perfect predictions of the ambient temperature, because reasonable forecasts of that disturbance for an area are often available via third-party weather forecasting services. We implement the P-MPC to establish the best achievable performance with an MPC controller. A comparison between the P-MPC and FFFB-MPC sheds light on the quality of closed-loop performance that can be attained when building and NN disturbance models estimated using the approach proposed in this chapter are used in an MPC controller. The difference in the performances of the FB-MPC and FFFB-MPC illustrates the economic benefits of using the NN heat disturbance predictions in an MPC controller.

For all the MPC controllers, we use the linear dynamic model shown in equations (4.15) – (4.17) below. The model considers feedforward disturbances via an additional term in the linear dynamics, and accounts for the unmeasured disturbances and plant-model mismatch using an integrating disturbance model (Pannocchia and Rawlings,

2003; Morari and Maeder, 2012).

$$x^+ = Ax + Bu + B_p p + B_d d \tag{4.15}$$

$$d^+ = d \tag{4.16}$$

$$y = Cx + C_d d \tag{4.17}$$

in which, $x \in \mathbb{R}^2$ is the state, $u \in \mathbb{R}$ is the control input (cooling duty), $p \in \mathbb{R}^{N_p}$ is the measured or predicted disturbance, $y \in \mathbb{R}$ is the measurement, $d \in \mathbb{R}$ is the state in the integrating disturbance model, and w_x, w_d, v are the noise sources in the linear dynamic model and measurements. The noise sources are modeled as Gaussian distributions with zero mean and covariances of Q_{wx}, Q_{wd}, and R_v corresponding to the noise terms w_x, w_d, and v, respectively. A Kalman Filter is constructed for state estimation during the closed-loop controller implementation. The noise covariances are tuned heuristically to develop the Kalman Filter in the simulation study. We note that for applications, however, data-driven methods are available in the literature (Odelson et al., 2006) to systematically estimate the noise covariances from data for the Kalman Filter tuning. The matrices A, B, B_p characterize the main linear dynamic model, and B_d, C_d denote the integrating disturbance model. We use an input disturbance model with $B_d = [0, 1]'$, and $C_d = 0$ for the MPC controller. Given the linear model matrices and noise covariances, we obtain filtered estimates of the state and disturbance $(\hat{x}, \hat{d})$ using the Kalman filter for subsequent use in the MPC regulator.

The goal of the economic MPC controller is to operate the building system with a minimum energy cost and comfort constraint violations. We solve the following economic optimization problem for the MPC regulator (Rawlings et al., 2012, 2018;

Patel et al., 2016)

$$\min_{\mathbf{u}} \quad \sum_{k=0}^{N-1} [p_e(k)u(k) + S_R(\Delta u(k))^2 + H_s s(k)] \tag{4.18}$$

$$\text{s. t.} \quad x^+ = Ax + Bu + B_p p + B_d \hat{d}, \tag{4.19}$$

$$x(0) = \hat{x}, \tag{4.20}$$

$$\underline{T}_z - s_2 \leq T_z \leq \overline{T}_z + s_1, \tag{4.21}$$

$$s \geq 0, \tag{4.22}$$

$$\underline{u} \leq u \leq \overline{u} \tag{4.23}$$

in which, p_e is the energy price that may be charged by electricity providers, ($\underline{u}$, $\overline{u}$) denote the available cooling duty limits, the matrix S_R is used to implement a rate-of-change penalty on the cooling duty, ($\underline{T}_z$, $\overline{T}_z$) denote the time varying comfort temperature constraints, H_s is a matrix used to penalize the comfort constraint violations, and N is the forecasting horizon length for the MPC controller. The optimization problem (4.18) is a type of quadratic program, since all the constraints and stage costs in the problem are linear or quadratic. We use $S_R = 1$, and $H_s = [10^2, 10^2]$ in the problem. The energy price, comfort temperature constraints, and the disturbance predictions are all time-varying parameters. The decision variable in the problem is the future control input sequence $\mathbf{u}$. The problem is solved after every measurement sampling instant, and the first move ($u^0(0)$) of the optimal input sequence is applied to the building system.

The feedforward disturbance model matrix B_p and future predictions $\mathbf{p}$ in the problem (4.18) are chosen based on the type of the MPC controller. For the P-MPC, we use the true disturbance model of the plant and the perfect disturbance predictions. In the FB-MPC, we use the ambient to the zone temperature component of the estimated

disturbance model matrix B_p. The exact predictions of the ambient temperature are used, and no heat disturbance predictions are included in the MPC problem. For the FFFB-MPC controller, we use the full disturbance model matrix B_p estimated from the grey-box modeling step. The estimated NN based on the approach proposed in Section 4.4 is used to provide the heat disturbance predictions, and we use the perfect ambient temperature predictions for this controller.

4.6 Simulation studies

We present case studies in this section to demonstrate the efficacy of the model identification frameworks to estimate accurate and reliable models, and the economic benefits of using the NN heat disturbance predictions in an MPC controller. We start with discussing the training data generation approach for the model identification frameworks. Then, we present the results that demonstrate the performances of the proposed grey-box and NN disturbance modeling methods. Finally, we present closed-loop simulations comparing the performances of the three MPC controllers discussed in the previous section.

4.6.1 Training data generation

We generate two types of data sets for the grey-box dynamic and NN disturbance model identification steps. We simulate the building system and PI controller setup discussed in Section 4.2 using pseudo random binary signals (PRBS) of the zone temperature setpoint to generate the data sets. For the grey-box model identification, we generate a total of 1 week of data. The heat disturbance pattern representing a regular university session period is used to generate this data set. The zone temperature set-

point values in the PRBS signal are sampled from a uniform distribution in the range $18 - 28°\mathrm{C}$, which are changed every 8 hours. The mass temperature dynamics in the building system is slow, and that frequency of setpoint changes ensures that the mass temperature is sufficiently excited in the training data.

For the NN disturbance model identification, we generate a total of 20 weeks of data. The zone temperature setpoints for this data set is also sampled from a uniform distribution in the range $18 - 24°\mathrm{C}$, and are changed every 2 days. The setpoints are changed with a slower frequency and in a narrow range to represent a routine operational type data set. The heat disturbance patterns from both the regular and summer session period are present for an equal proportion of 10 weeks in the entire data set. This approach ensures that sufficient amount of heat disturbances patterns from both the summer and regular session periods are contained in the training data.

4.6.2 Grey-box model estimation

The grey-box building model identification problem discussed in Section 4.3 is solved for three different choices of the zero order hold durations on the heat disturbance. These hold durations are chosen as 2, 4, and 6 hours. For all the three optimization problems, the entire training data set generated for the grey-box modeling is split in two trajectories. This step ensures that the number of time steps per trajectory is fewer and grey-box model identification problem can be solved faster. Each trajectory after the splitting contains a total of 84 hours of data. We use an initial guess of $23°\mathrm{C}$ for the initial model states, 10^{-2} for the physical building model parameters, and 5kW for the heat disturbance variables. In the subsequent discussion, we also highlight the robustness of the presented results for different initial guesses of the decision variables in the model identification problem. The initial guesses for the physical building

model parameters are provided to the optimizer in a logarithm scale, since the decision variables for those parameters are considered in a logarithm scale in the optimization problem.

We first examine the training data fit and the quality of the grey-box parameter estimates obtained by solving the three model identification problems. We compute the root mean squared error (RMSE) metric over the entire training data, and the relative errors in the parameter estimates. In Table 4.2, we report the RMSE metric, the parameter estimates, and the relative error in the estimates obtained for the three modeling problems solved. We observe that for a small 2-hour disturbance hold duration, the training data fit is the best with the lowest RMSE metric. The better fit is obtained because the optimizer has more decision variables to fit the data closely with this small hold duration. For the longer 6-hour disturbance hold duration, the training data fit is poor and the RMSE metric is the largest. For both the 2 and 6-hour cases, all the grey-box parameters have large estimation errors.

The parameter estimation errors with the 4-hour disturbance hold duration are the best and range between $4 - 38\%$. The quality of these parameter estimates are reasonable, however. The building modeling problem considered in this chapter is challenging due to the slow dynamics of the mass temperature and the significant amount of unmeasured heat disturbance in the training data. We show the fit to the training data obtained at the solution of the model identification problem solved with the 4-hour hold choice in Figure 4.4. The training fit is shown for one of the 84-hour training data trajectories. We observe in the plot that the fit to both the zone temperature and heat disturbances are reasonable, and the estimated quantities are close to the corresponding true values in the training data.

The heat disturbance estimates obtained after solving the model identification problem (4.3) are not subsequently useful for a validation of building dynamic model, be-

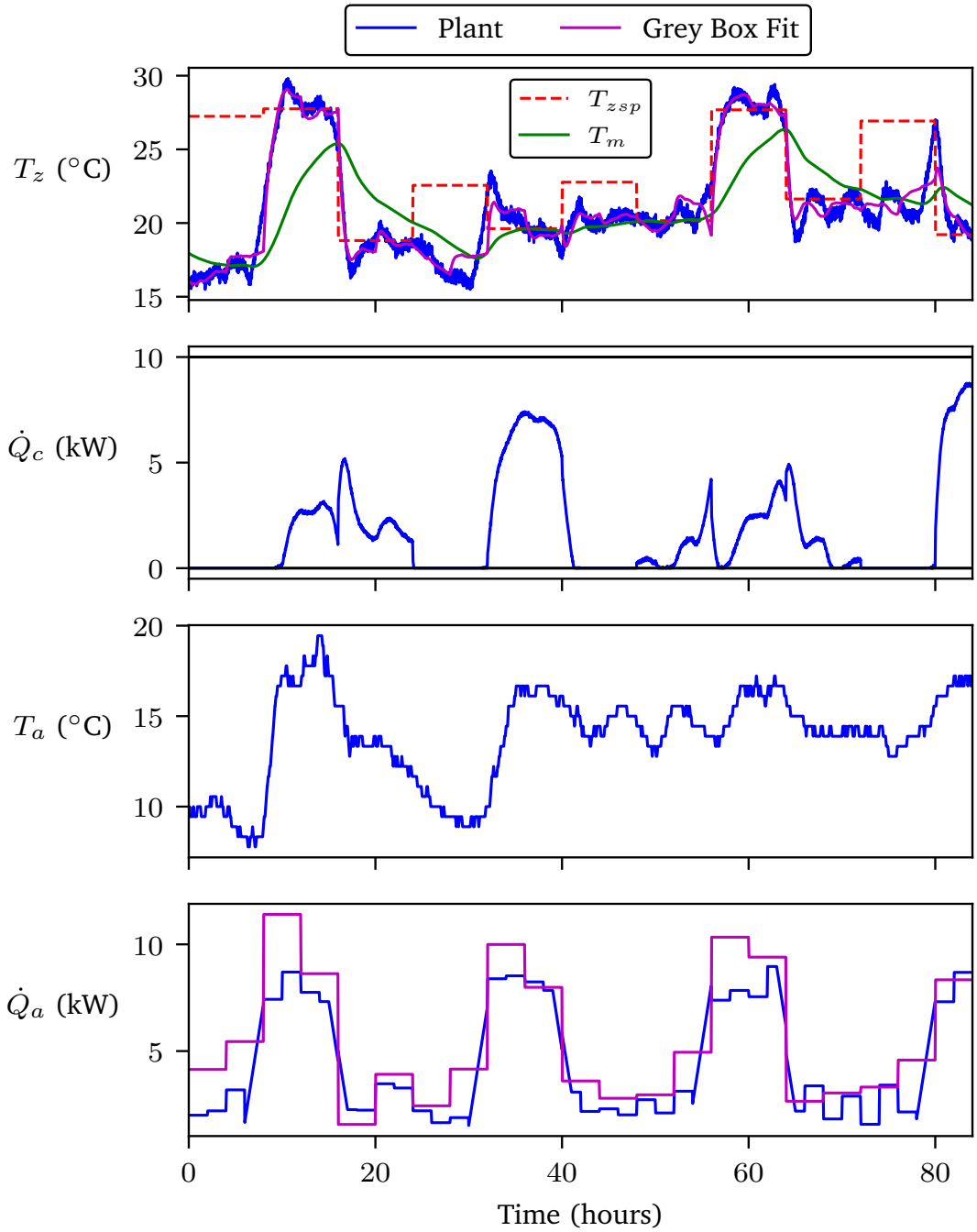

Figure 4.4: Training data fit to the zone temperature and heat disturbance obtained after solving the grey-box building model identification problem with the 4-hour choice of the disturbance hold duration.

cause any new data collected from the system would contain different heat disturbances. The relative errors on the parameter estimates cannot be computed as well, because the true building model parameters may not be known in applications. Therefore, we use the approximate confidence intervals (CIs) on the parameter estimates and the training fit as represented by the RMSE metric to guide the model selection.

We compute the CIs on the physical building model parameter estimates using the approach discussed in Subsection 4.3.2. In Table 4.2, we also report the CIs for the estimates obtained with the three model identification problems. The CIs in the Table are reported as percentage deviation from the parameter estimates. We notice that the CIs on the estimates are large for the model identification problems solved using the 2 and 6-hour disturbance hold durations. The large CIs are a result of more degrees of freedom in the optimization problem with the 2-hour hold, and an incorrect disturbance structure assumption with the 6-hour hold. The CIs on the parameter estimates obtained with the 4-hour hold duration are the tightest for most of the parameter estimates. For the subsequent NN disturbance model identification and the closed-loop MPC simulations, we choose the building model obtained with the 4-hour hold duration because this choice provides the tightest CIs on the parameter estimates and an adequate training data fit.

The times required to solve each model identification problem are also reported in Table 4.2, which are approximately around 2 minutes. The solution times show that all the model identification problems can be solved conveniently in industrial applications. The ease of solvability of the modeling problem in which we simultaneously estimate a grey-box dynamic model and piecewise constant disturbance signals is enabled by the symbolic differentiation based software CasADi.

Further, we also tested the robustness of the results presented in Table 4.2 for different initial guesses of the decision variables in the model identification problem. For

Zero order hold duration on $\dot{Q}_a$ (hours)	Parameter estimate % Relative error (% Confidence deviations)				Training fit (RMSE) (°C)	Model estimation time (minutes)
	H_m/C_z	H_a/C_z	H_m/C_m	$1/C_z$		
2	2.24×10^{-2} 6374.6% (−59.4%, +146.2%)	3.53×10^{-3} 855.2% (−54.3%, +118.7%)	1.62×10^{-3} 2823.0% (−11.7%, +13.2%)	7×10^{-3} 600.1% (−54.1%, +117.9%)	0.47	2.14
4	4.78×10^{-4} 37.9% (−10.8%, +12.1%)	3.83×10^{-4} 3.6% (−10.8%, +12.1%)	7.53×10^{-5} 35.7% (−17.4%, +21.1%)	7.83×10^{-4} 21.6% (−9.4%, +10.4%)	0.87	1.35
6	4.19×10^{-4} 20.7% (−34.1%, +51.8%)	7.73×10^{-16} 100% (−100%, +∞)	4.91×10^{-4} 784.6% (−14.7%, +17.2%)	1.74×10^{-4} 82.5% (−12.6%, +14.4%)	0.99	2.11

Table 4.2: Metrics summarizing the results obtained with the grey-box building modeling approach proposed in this chapter.

these tests, we fixed the guesses for the initial model states and heat disturbances. And the guesses for the grey-box building model parameters were simultaneously varied between 10^{-4} to 10^{-1}. For all the optimization problems solved with the different initial guesses, we observed that the training fit (represented by the RMSE metric) and the CIs on the parameter estimates can still be used to guide the model selection and choose an appropriate zero order hold duration on the heat disturbance. The optimization problem for the 4-hour hold duration converged to the same solution as shown in Table 4.2, with parameter estimation errors in the range 3% to 38% and the CIs between -9% to 22%. We observed that for initial guesses on the building model parameters outside the above-mentioned range, the estimation errors and the CIs started to get worse.

4.6.3 Disturbance model identification

We next use the grey-box building model estimated from the previous step to determine a NN heat disturbance model. First, the entire 20 weeks of data set generated for the disturbance model identification is split into 16 weeks of training, 2 weeks of holdout, and 2 weeks of validation data sets. This splitting is performed such that each data set contains an equal amount of disturbance patterns from both the regular and summer session periods. Further, each data set is also split into multiple trajectories of lengths 24 hours. We obtain these multiple data trajectories by starting from the beginning of each data set and extracting 24 hours of data in increments of 6 hours, until the last possible trajectory is extracted. As discussed in Chapter 1, the holdout data set is used to monitor the optimization problem loss on some unseen data at the end of every training epoch. And we choose the NN weights and biases that provide the best loss on the holdout set for further validation.

In the input vector z_i provided to the NN $f_{T_{m0}}(\cdot)$ for predicting the unmeasured mass temperature, we use a total of five measurements of the zone temperature, cooling duty, etc. The architecture of that NN is chosen as [40, 16, 1]. This notation of a feedforward NN architecture means that the NN has an input layer containing 40 nodes, a hidden layer with 16 nodes, and an output layer with one node. The architecture of the disturbance predictor NN is [5, 64, 64, 1]. We use the hyperbolic tangent as the activation function in both the networks. For the training, we use a batch size of 128 trajectories in the optimization algorithm Adam. And the NN model identification problem (4.18) is solved for a total of 1000 epochs.

After the training process, we evaluate the quality of the heat disturbance predictions by the estimated NN and the final zone temperature predictions of the composite grey-box and NN models. We compute the RMSE metrics for both these predictions

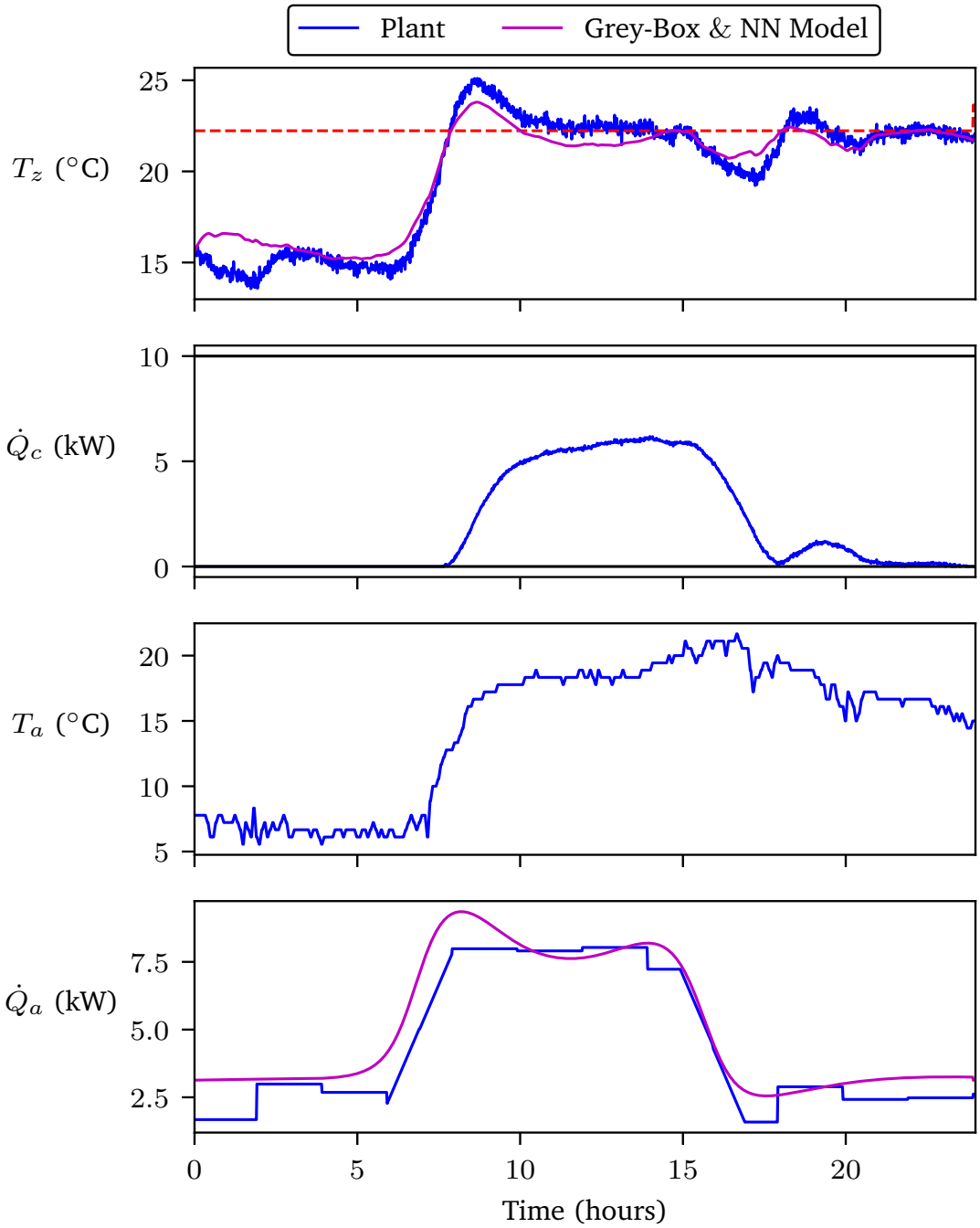

Figure 4.5: Predictions obtained using the estimated grey-box and NN disturbance models on one of the validation data trajectory.

over all the 24-hour trajectories in the validation data set. We report the RMSE metrics obtained for both the heat disturbance and zone temperature predictions in Table 4.3. The metrics show that both the variables are reasonably well predicted by the estimated models. We note that in industrial applications, the RMSE metric for the heat disturbance cannot be computed since the disturbance is not measured. The models must be validated using the CIs of the building model parameters and the RMSE metric for only the zone temperature predictions.

Figure 4.5 shows the heat disturbance predictions of the estimated NN and the final zone temperature predictions of the composite model on one of the 24-hour validation data trajectories. We notice from the Figure that good predictions of both the zone temperature and heat disturbances are provided by the estimated models. In Table 4.3, we also report the time taken for the NN disturbance model identification. The training time shows that the NN model identification problem can be conveniently solved offline for a disturbance model development in real building applications.

Metric	Value
RMSE – T_z	1.12 (°C)
RMSE – $\dot{Q}_a$	0.77 (kW)
NN Estimation Time	40.35 (Minutes)

Table 4.3: Metrics summarizing the performance of the NN disturbance model identification approach proposed in this chapter.

4.6.4 Closed-loop MPC simulations

Finally, we examine the closed-loop performances obtained with the three MPC controller formulations discussed in Section 4.5. We consider a total of 2 weeks of simulation period. The first 3 days of the closed-loop trajectories obtained under feedback with the MPC controllers are shown in Figure 4.6. For all the MPC controllers imple-

mented, we use the time varying energy prices and comfort temperature constraints as shown in that Figure. The energy prices are typical of the prices charged by electricity providers in real building applications. The prices are high during the day times and low during the nights. The comfort temperature constraints are narrow during the day times and wider the evenings and nights. The temperature constraints during the day times values reflect the tighter range of desired temperatures when occupants are present in the building. In the 2 weeks of simulation period, we use heat disturbance patterns from both the regular and summer sessions periods in equal amounts of 1 week each. The sample time for the plant measurements and MPC controller execution in this closed-loop simulation study is 5 minutes. The forecasting horizon length in all the MPC controllers is 1 day ($N = 288$).

In Figure 4.6, we observe that the P-MPC performs significant cooling during the nights when the energy prices are lower. And less cooling during the day times when the energy prices are high. The controller also successfully maintains the zone temperatures within the desired comfort constraints. The FFFB-MPC also performs qualitatively similar cooling as the P-MPC controller, with more cooling during the nights and less during the day times. The difference in the closed-loop trajectories between the P-MPC and FFFB-MPC is because we use the perfect dynamic model and disturbance forecasts for the P-MPC. Whereas, the estimated dynamic model and NN heat disturbance forecasts are used in the FFFB-MPC controller.

We also observe in Figure 4.6 that the FB-MPC performs noticeably less pre-cooling during the nights, and as a consequence ends up cooling more during the day times to maintain the zone temperature within the comfort constraints. The controller performs less cooling during the nights because it does not use any heat disturbance predictions. At the nights, the FB-MPC underpredicts the temperature evolution of the next day, and thus, it undercalculates the cooling requirements to maintain the temperatures in

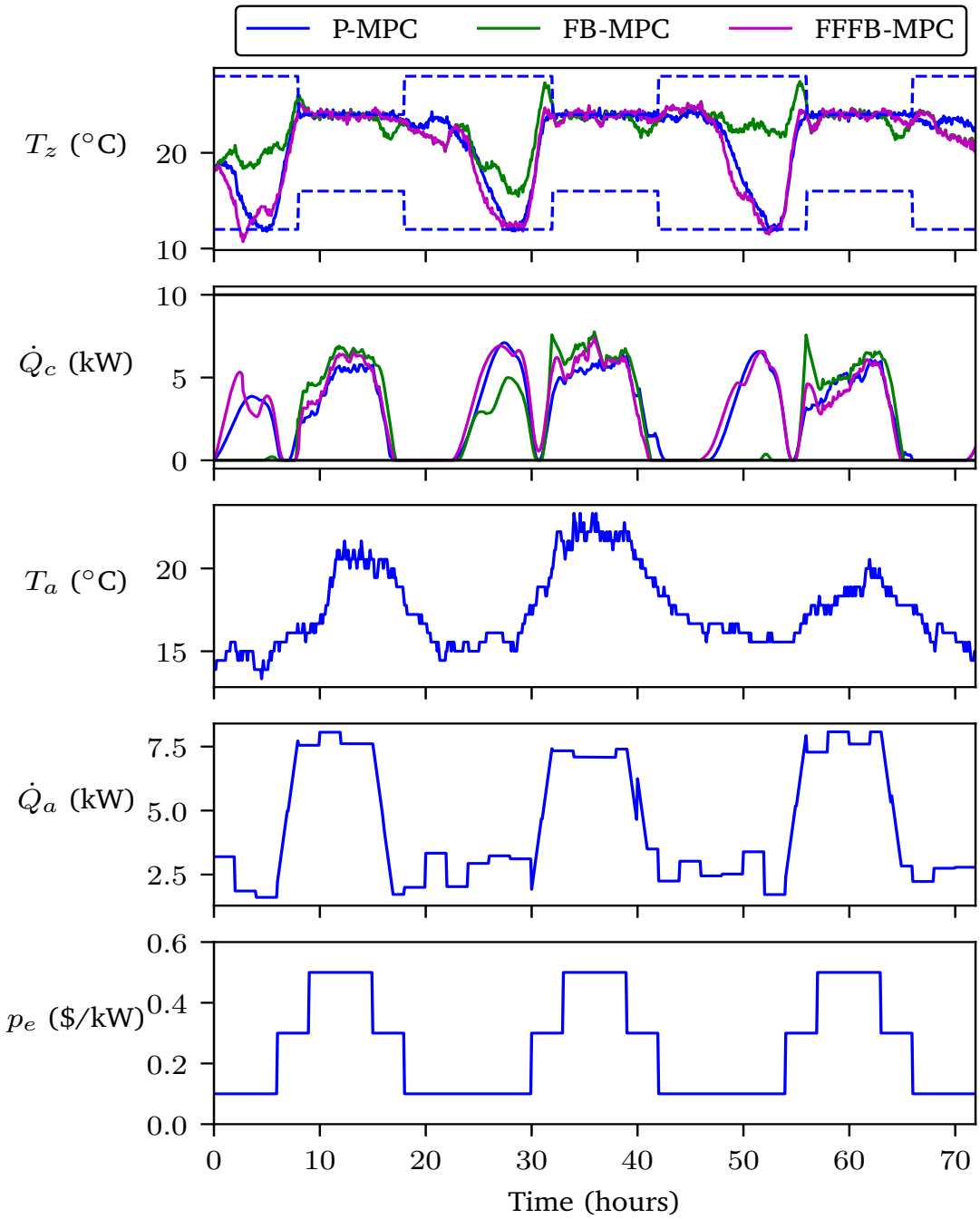

Figure 4.6: Closed-loop trajectories of the building system obtained under operation with the P-MPC, FB-MPC, and FFFB-MPC controllers. Significant amounts of cooling are performed during the nights by the P-MPC and FFFB-MPC controllers. But the FB-MPC performs less cooling during the nights and more during the day times when the energy prices are high.

the next day within the comfort constraints.

MPC controller	Temperature constraint violation cost (Λ_T)	Rate-of-change penalty cost (Λ_R)	Total energy cost (Λ_E)	Energy cost savings
FB-MPC	16603	81	2315	
FFFB-MPC	21171	84	2210	4.54 $\%$
P-MPC	12324	67	2160	6.7 $\%$

Table 4.4: Metrics summarizing the closed-loop performances of the three MPC controllers. The percentage energy cost savings shown in the last column for the FFFB-MPC and P-MPC are computed relative to the FB-MPC controller.

For the entire 2 weeks of closed-loop simulation performed using the three MPC controllers, we also compute the following three metrics

- Total zone temperature constraint violation: $\Lambda_T = \sum_{k=0}^{N_t} H_s s(k)$.
- Total rate-of-change penalty: $\Lambda_R = \sum_{k=0}^{N_t} S_R(\Delta u(k))^2$
- Total energy cost incurred: $\Lambda_E = \sum_{k=0}^{N_t} p_e(k)u(k)$

These three metrics are summarized in Table 4.4. We observe that the FB-MPC, FFFB-MPC, and P-MPC controllers incur the maximum to minimum energy costs. In Table 4.4, we also report the energy cost savings provided by the FFFB-MPC and P-MPC controllers compared to the FB-MPC. The percentage energy cost savings achieved by the FFFB-MPC is 4.54%, while the P-MPC controller provides savings of 6.7%. The improved performance of the FFFB-MPC over the FB-MPC demonstrates the economic value of treating the unmeasured heat disturbance in both the model identification and MPC problems.

4.7 Conclusions

In this chapter, we have presented a novel model identification approach for building energy systems affected by large unmeasured heat disturbances. The approach uti-

lizes both first principles knowledge and data-driven neural networks. The large heat disturbance is often generated in applications due to occupants inside the building. For such building systems, we proposed to develop a dynamic and disturbance model in two steps. In the first step, we estimate a grey-box dynamic model of the building system. We treat the presence of the unmeasured disturbance in the training data in this modeling step by also estimating an approximate piecewise constant signal for the disturbance. We further proposed a method to compute approximate confidence intervals on the estimated parameters in the grey-box model, which can be used to gauge the reliability of the model. In the second step, we use the estimated grey-box building model and historical operational data to estimate a NN heat disturbance predictor. The NN uses some auxiliary measurements as inputs to predict the heat disturbance. This model is estimated based on only the zone temperature measurements and without any known occupancy schedule or measurements of the disturbance. The composite grey-box and NN disturbance model retains a linear structure between the actuator and measurements. Thus, it can be conveniently used for an MPC implementation with standard online solvers.

We demonstrated the effectiveness of the proposed modeling framework via simulation studies with a two time scale based building model as the plant. We showed that the proposed grey-box model identification problem can be used to obtain an accurate and reliable building model from data, despite the presence of the large heat disturbance in the training data. The estimation errors on the physical parameters in the building model were in the range $3 - 38\%$, while the CIs on the estimates were between -9% to $+22\%$. We next demonstrated that the estimated grey-box building model from the first step can be used to obtain a NN disturbance model. The NN is suitable to provide feedforward predictions of the heat disturbance for different types of day and times of year.

Further, the estimated dynamic and disturbance models were used in closed-loop MPC simulations. We studied the performance of an MPC that uses the NN disturbance forecasts compared to an MPC that does not use those forecasts. We highlighted that using the feedforward NN disturbance predictions aids the MPC to achieve improved energy cost savings of around 4.5%. The result highlights the economic advantages of systematically treating the unmeasured heat disturbance in both the modeling and control problems.

Future directions for dynamic and disturbance modeling in building applications are discussed in Chapter 6. In the next chapter, we develop another class of hybrid dynamic modeling approaches for *nonlinear* chemical engineering processes. With the goal of using the estimated models in steady-state economic optimization at the RTO layer in the process industries.

Chapter 5

Hybrid process modeling with application to economic optimization

The use of dynamic models is common in industrial applications for process optimization and control. At the RTO layer in the process industries, a nonlinear model is typically used so that any nonlinearity in the process is accounted for when determining an economically profitable steady state (Cutler and Perry, 1983; Darby et al., 2011). The nonlinear model may be developed using either a fully first principles, a grey-box, or a black-box approach. In the first principles approach, rigorous knowledge about the process such as vapor-liquid equilibrium relationships, reaction kinetics, etc, are used to develop the model (Pantelides and Renfro, 2013). In the grey-box approach, a dynamic model is parameterized using some process knowledge and unknown parameters in the model are estimated from data (Zavala et al., 2008; Kravaris et al., 2013). Both the first principles and grey-box modeling approaches yield mathematical models that are often suitable for optimization based decision-making. However, these two approaches can require more process knowledge than available in applications and sometimes incorrect or incomplete knowledge may be used to result in a plant-model mismatch, and consequently a suboptimal economic performance.

In the black-box approach, no first principles based process knowledge is used, and

the dynamic model is developed solely using some training data collected from the plant and nonlinear function approximators such as neural networks (Sjöberg et al., 1995; MacMurray and Himmelblau, 1995; Wu et al., 2021). The issue with the black-box approach is that the developed models are not interpretable, and can often require excessive amounts of training data for the model development. A black-box model also may not be suitable for use in an optimization problem because of highly nonlinear and unstructured functions represented by NNs, which contain no physical insights about the plant.

To avoid the above issues with the first principles and black-box modeling approaches, several researchers have proposed the hybrid modeling approach to develop dynamic process models (Psichogios and Ungar, 1992; Von Stosch et al., 2014; Zendehboudi et al., 2018; Venkatasubramanian, 2019; Yang et al., 2020; Sansana et al., 2021). In this approach, both the basic first principles knowledge often available in applications, and the advantages of machine learning models are utilized. The motivation for the approach is that less process knowledge may be required than the first principles and grey-box modeling methods. The machine learning models can be used to represent some parts of the overall model for which no domain knowledge is available. The training data requirement for model development may still be less and the model may be more interpretable compared to a fully black-box model. The hybrid model can also be more suitable for use in an optimization problem because of process constraints encoded in the model, leading to only physically realistic objective function profiles for the optimizer.

In this chapter, we present a novel hybrid modeling approach to develop dynamic process models utilizing both the available first principles knowledge and the advantages of neural networks (NNs). We focus on developing models for nonlinear chemical engineering processes. We demonstrate that if the hybrid models are developed with

enough structural insights about the process, then they can also be successfully used in steady-state optimization and obtain high economic performance. We begin with a discussion about the related literature on hybrid and black-box process modeling using machine learning methods.

5.1 Literature review

The general idea of combining first principles knowledge and machine learning to develop hybrid process models has been around in the literature since the early 1990s. Psichogios and Ungar (1992) propose to use NNs to approximate some unknown, nonlinear functions in the hybrid model that may be challenging to model using the available process knowledge. The work considers a bioreactor example, and a NN was used to approximate the nonlinear substrate growth rate in the overall model. In the literature, this type of hybrid modeling approach is also referred as the series approach because the output of a NN is provided as an input to a dynamic model structured using the process knowledge. Rico-Martinez et al. (1994) use a NN to approximate an unknown reaction rate law in a continuous stirred tank reactor example. Thompson and Kramer (1994) propose compensate for inaccuracies in an available first principles model by adding a NN to the dynamics of that model. This approach is called the parallel hybrid model structure in the literature. The approach, however, can result in a hybrid model with a large black-box component explaining the difference between the available first principles model and the plant. The large black-box component can subsequently give a challenging optimization problem to solve at the RTO layer. Hence, we do not consider the parallel hybrid modeling approach in this chapter, and focus on developing a modeling framework using the series approach.

Many other researchers have applied the hybrid modeling approach over the years

to different applications such as polymerization systems (Tsen et al., 1996; Doyle III et al., 2003), chemical reactors (Su et al., 1993; Gupta et al., 1999; Luo et al., 2012; Qi et al., 1999), bio-processes (Schubert et al., 1994), designing fuel additives (Sundaram et al., 2001), etc. These earlier works typically used NNs of small sizes in the hybrid model. The recent advances in the NN training algorithms, and the availability of powerful symbolic differentiation based software (LeCun et al., 2015; Abadi et al., 2015; Lee et al., 2018; Andersson et al., 2019) encourage the development of sophisticated hybrid modeling approaches for process optimization and control.

Among the recent works, Lovelett et al. (2019) develop an approach to estimate hybrid process models for systems with only partial state measurements. The work uses a vector containing a recent history of past measurements and actuators to provide as inputs to the NNs that aim to approximate the unknown functions posited to depend on the unmeasured states in the plant. Bangi and Kwon (2020) use deep networks to develop hybrid dynamic models and applied their proposed approach to a hydraulic fracturing process. Chen and Ierapetritou (2020) estimate hybrid process models in two steps. First, a partial correlation coefficient analysis is used to determine the sources of plant-model mismatch in an available first principles based model. Then, the functions in that model leading to the mismatch are replaced with support vector machines (Steinwart and Christmann, 2008) or black-box NNs. Ghosh et al. (2019) develop an approach to account for the difference between an available first principles model and plant measurements using subspace identification (Qin, 2006). All these recent papers estimate the weights and biases in the NNs in the hybrid model by solving an onestep ahead prediction error minimization problem using a stochastic gradient algorithm (Kingma and Ba, 2014), or the Levenberg-Marquardt algorithm (Levenberg, 1944; Marquardt, 1963).

To design high-performance industrial automation systems, the estimated hybrid

models must be suitable for use in an optimization problem with standard solvers. Most recent papers focus particularly on the dynamic model development, and less effort is directed in the literature towards using the estimated models for optimization and control. Kahrs and Marquardt (2007) present simulation studies to examine the performance of hybrid models when they are used in steady-state optimization. The work considers an ethylene glycol process. The size of the NNs used in the hybrid model in that work were small, however. Schweidtmann and Mitsos (2019) recently developed a global optimization algorithm to optimize over hybrid and black-box models embedded with NNs. The work also demonstrates the efficacy of the algorithm to optimize over large hybrid and black-box models. In the paper Zhang et al. (2019), the performances of hybrid models are examined when they are used in a hierarchical RTO and MPC framework. The NNs in the hybrid models were estimated directly using the measurements of the unknown functions being approximated. This approach is impractical for deployment in applications. Zhang et al. (2019) study the performances of hybrid and black-box NN models when used in tracking and economic MPC controller formulations. Other researchers have investigated the use of genetic algorithm (Bhutani et al., 2006), and particle swarm algorithm (Bensingh et al., 2019) to optimize over hybrid dynamic models for process operations and control.

With the recent advances in NN training algorithms and high quality software, there is also a considerable interest in the literature to use black-box NNs for decision-making and control. The type of black-box NNs most relevant for dynamic process modeling are recurrent NNs (Rumelhart et al., 1986; Hochreiter and Schmidhuber, 1997). Wu et al. (2021) suggest the use of long short term memory networks (LSTMs) for process modeling and subsequently for use in an MPC controller formulation. Santander et al. (2022) propose to use recurrent NNs for process modeling, and then use the estimated NNs for economic optimization in a production planning layer. The work Jalanko et al.

(2021) studies the performance of black-box NNs for dynamic modeling of an industrial ethylene splitter. Despite all these recent works, the suitability of using fully black-box networks for effective process modeling and control is questionable. In the simulation studies in this chapter, we also examine the performances of black-box NN models when used in steady-state optimization.

Another dynamic model identification approach proposed in the recent literature is the Koopman operator based method. In this approach, the usual state is transformed into another high dimensional state, and a linear dynamic model is estimated in the newly defined state. The work Korda and Mezić (2018) presents the identification approach of such models, and also discusses how the models can be subsequently used in a linear MPC controller. Lusch et al. (2018) propose to use deep networks to identify the high dimensional state space in which the process dynamics are approximately linear. Several researchers are currently investigating the use of the Koopman operator based dynamic modeling approach in the process control literature (Schulze et al., 2022; Narasingam and Kwon, 2019).

Depending on the intended use, two types of Koopman operator dynamic model may be estimated from data. In the first approach (Korda and Mezić, 2018), a nonlinear function is determined to transform the usual state to the higher dimensional state, but a linear matrix is identified to transform that state back to the usual state. The approach gives a *linear* dynamic model between the control inputs and measurements. The model may be suitable for use in an MPC controller. But it cannot be used to determine an economically profitable steady state by systematically accounting for the nonlinearity in the process. In the second approach (Lusch et al., 2018), nonlinear functions are estimated from data to perform both the transformations (from the usual state to the higher dimensional state, and vice versa). The approach gives a model with similar characteristics as a black-box NN, and is highly nonlinear between the control

inputs and measurements. Due to the above issues with both the approaches, we do not consider the Koopman operator based approach for process modeling and optimization in this chapter.

Overview of contributions

In this chapter, we develop a hybrid process modeling approach that utilizes both the available first principles knowledge and neural networks estimated from data. The hybrid model is first structured using the first principles knowledge, and it contains some unknown functions that are parameterized using black-box NNs. We next estimate the parameters in the NNs by solving a multistep ahead prediction error minimization problem, which is one of the novel features of our proposed modeling framework. In the proposed approach, we treat both the full state and output measurement cases. And multiple NNs to approximate different unknown functions in the hybrid model can be conveniently incorporated in the model identification problem.

We present case studies using an illustrative chemical reactor and a styrene polymerization process. The effectiveness of the hybrid modeling approach to estimate accurate dynamic models is illustrated. We further demonstrate the suitability of the estimated hybrid models to determine an accurate optimum when they are used in a steady-state economic optimization problem. In the case studies, we also discuss the type of data that should be collected from the plant when the final goal is to use the estimated models in steady-state optimization. We also examine different plausible function parameterization choices of the NNs used to represent the unknown functions in the hybrid models. We emphasize that the NNs should be parameterized with enough structural insights about the functions being approximated to obtain a good economic performance.

The rest of this chapter is organized as follows. In the next section, we present the

proposed hybrid modeling framework. Section 5.3 discusses the approach used to estimate black-box NN models for comparisons in the simulation studies. In Section 5.4, we outline the steady-state optimization formulation used to examine the economic performances of the estimated models. In Section 5.5, we present detailed simulation studies to demonstrate the effectiveness of the hybrid modeling framework to estimate accurate dynamic models, and to highlight the suitability of the models to obtain accurate steady-state optimums. We discuss some conclusions of this thesis chapter in Section 5.6.

Portions of the methods and results presented in this chapter are to appear in Kumar and Rawlings (2023a). The mathematical notation used in this chapter are given in Section 1.3.

5.2 Hybrid process modeling

The goal of the process modeling methods presented in this chapter is that given a time series of data collected from a plant, estimate a continuous or discrete time representation of the dynamics of the plant. We assume that the plant dynamics may be governed by the nonlinear ordinary differential equations (ODEs)

$$\frac{dx}{dt} = f(x, u) \tag{5.1}$$

$$y = h(x) + v \tag{5.2}$$

Here, $x \in \mathbb{R}^n$ is the state, $y \in \mathbb{R}^p$ is the measurement, $u \in \mathbb{R}^m$ is the control input, and $v \in \mathbb{R}^p$ is a Gaussian random measurement noise. This noise term is of zero mean and covariance R_v. To generate the training data from the plant, we discretize the plant using the Runge-Kutta 4 (RK4) method (Butcher, 1996). We assume that any large

disturbances to the plant remain constant during the data generation step.

To design a dynamic process model of the plant (5.1) – (5.2), we first write down a set of ODEs from the available first principles knowledge. These ODEs define the main structure of the dynamic model and could be developed using chemical engineering fundamentals on mass and energy balances. These ODEs may contain some functions for which no domain knowledge is available. Those functions are parameterized using black-box NNs and unknown parameters in these NNs are subsequently estimated from data. We write the hybrid model in continuous time as follows

$$\frac{dx_g}{dt} = f_g(x_g, u, z; \theta_{NN}) \tag{5.3}$$

$$y = h_g(x_g) \tag{5.4}$$

in which $x_g \in \mathbb{R}^{N_g}$ denote the modeled states, and $f_g(\cdot)$ and $h_g(\cdot)$ define the modeled dynamic and measurement functions. This hybrid model contains a few black-box NNs that are used to represent some complex, unknown functions. The weights and biases in all those NNs are contained in the parameter θ_{NN}. We use the hyperbolic tangent as the activation function in all the NNs, so that standard gradient based optimization solvers can be subsequently applied to the hybrid models for use in steady-state optimization. The vector z in the hybrid model consists of a recent history of past measurements and control inputs. The vector is defined for every instant t as follows

$$z(t) = \Big[y(t - N_p\Delta)', .., y(t - \Delta)', .., u(t - N_p\Delta)', .., u(t - \Delta)'\Big]' \tag{5.5}$$

in which $t \geq 0$, Δ is the sample time of the measurements from the plant, and $N_p\Delta$ defines the amount of past information used in the vector. For discrete time, we use the past measurements and control inputs at the previous sample times to define the

vector z, similar to (5.5).

The modeled states x_g, and/or the vector z defined above are provided as inputs to the NNs used to represent the unknown functions in the hybrid model. We use the vector z for those situations when the unknown function being approximated by a NN can potentially depend on the unmeasured states in the plant. This approach of using a recent collection of measurements and control inputs has long been used in the system identification literature to develop dynamic models for systems with only output measurements (Ho and Kalman, 1966; Qin, 2006). The parameter N_p in (5.5) can be tuned heuristically to be a small integer that provides a good fit of the hybrid model to the collected training data.

The next step after developing the structure of the hybrid model is to obtain a discrete time representation of the model, so that it can be used for training. We use the RK4 method for this discretization step and obtain a state space form of the hybrid model. We then use the state space model in a multistep ahead prediction error minimization problem to determine the unknown parameters in the NNs from training data. The hybrid model is discretized as follows

$$k_1 = f_g(x_g, u, z; \theta_{NN}) \tag{5.6}$$

$$k_2 = f_g(x_g + (\Delta/2)k_1, u, z(t + \Delta/2); \theta_{NN}) \tag{5.7}$$

$$k_3 = f_g(x_g + (\Delta/2)k_2, u, z(t + \Delta/2); \theta_{NN}) \tag{5.8}$$

$$k_4 = f_g(x_g + \Delta k3, u, z(t + \Delta); \theta_{NN}) \tag{5.9}$$

$$x_g(t + \Delta) = x_g(t) + \frac{\Delta}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{5.10}$$

$$x_g^+ = f_{gd}(x_g, u, z; \theta_{NN}) \tag{5.11}$$

For this RK4 discretization step, the variable $z(t+\Delta)$ is straightforward to obtain based

on the definition of the vector z in (5.5). For the variable $z(t + \Delta/2)$, however, we require the values of the control inputs and measurements in between some sample times. For the training data generation in the case studies, the plant is simulated using a zero order hold assumption on the control inputs between any two time steps. So to construct the variable $z(t + \Delta/2)$ in the above discretization step, we also assume that the control inputs between any two sample times is the same as the value at the previous time step. To obtain the measurements between any two sample times for the variable $z(t + \Delta/2)$, we use linear interpolation on the measurements at the nearest two sample times. Depending on the modeling problem considered in the case studies, the vector z may also not be required as an input to the NNs if all the plant states are measured. In this case, the linear interpolation on the measurements is not required since the variable z would not be used in the hybrid model.

To use the hybrid model in a multistep ahead prediction error minimization problem, it is convenient to have a discrete time state space representation of the model. The variable z also changes at every time step. So to aid the prediction process, we also write the following dynamics for the variable z

$$
\begin{aligned}
z^+ &= \left[z'_{p+1:N_p p},\ h_g(x_g)',\ z'_{N_p p+m:N_p(p+m)},\ u' \right]' \\
z^+ &= f_z(x_g, u, z)
\end{aligned}
\tag{5.12}
$$

The combination of equations (5.11) and (5.12) completely define the state space form of the hybrid model. The full model uses the vector $\left[x'_g,\ z' \right]'$ as the state.

After developing the state space description of the hybrid model, we use it in a multistep ahead prediction error minimization problem to estimate the unknown parameters in the NNs. We assume in the training problem that the data from the plant may be collected in the form of one or multiple trajectories, so we minimize the pre-

diction errors of the hybrid model across the data in all the trajectories. We solve the following optimization problem

$$\min_{\theta_0,\, \theta_{NN}} \quad \sum_{i=1}^{N_{tr}} \sum_{k=0}^{N_t} |y_i(k) - \hat{y}_i(k)|^2 \tag{5.13}$$

$$\text{s. t.} \quad x_{gi}^{+} = f_{gd}(x_{gi}, u_i, z_i; \theta_{NN}) \tag{5.14}$$

$$z_i^{+} = f_z(x_{gi}, u_i, z_i) \tag{5.15}$$

$$\hat{y}_i = h_g(x_{gi}) \tag{5.16}$$

$$x_{gi}(0) = f_0(y_i(0), z_i(0); \theta_0) \tag{5.17}$$

in which, the subscript i denotes the trajectory number of a data sequence. The state space form of the hybrid model is used in (5.14) - (5.16) when making the measurement predictions over the future horizon in the optimization problem.

To make the future measurement predictions in each trajectory, we require the initial states $(x_{gi}(0), z_i(0))$. The states $z_i(0)$ can be constructed using the recent history of measurements and control inputs at the beginning of each data trajectory. The states $x_{gi}(0)$ are developed using the function $f_0(\cdot)$, which is chosen based on the information about the plant measurements. In the case studies in this chapter, we consider two cases: full and partial state measurements from the plant. For the full state measurement case, we directly use all the plant measurements (noisy) to construct the states $x_{gi}(0)$, and $f_0(y_i(0), z_i(0); \theta_0) = y_i(0)$. For the output measurement case, we use another NN to predict the unmeasured states. The recent history of information contained in the vector $z_i(0)$ are provided as inputs to this NN. The weights and biases in this NN are contained in the parameter θ_0. The unmeasured states predicted by this NN are then appropriately concatenated with the measurements $y_i(0)$ to give the function $f_0(\cdot)$ and

construct the full initial state vector $x_{gi}(0)$. The decision variables in the optimization problem are unknown weights and biases in all the NNs (θ_{NN}) used to approximate the unknown functions in the hybrid model, and the another introduced NN (θ_0) used to predict the initial unmeasured states.

As we also discussed in the previous chapter in Section 4.4, the typical approach in the grey-box parameter estimation literature to handle unmeasured states is to introduce decision variables for all the initial states directly in the model identification problem. We choose the approach describe above of introducing another NN in the optimization problem instead of solving for the initial states because the stochastic gradient algorithm does not perform effectively if some decision variables appear only in some data trajectories when solving a problem of the type 5.13. With the proposed approach, we estimate only one function to handle the unmeasured states, and this function remains the same across different data trajectories.

The hybrid model training optimization problem is solved using the software TensorFlow, using the stochastic gradient algorithm Adam. The model identification problem can be constructed to solve symbolically using customizable recurrent network classes available in the software. We note that to solve the problem 5.13, the software constructs symbolic graphs so that analytical derivatives of the objective function can be computed during the optimization problem. The size of the symbolic graphs can grow large if the future prediction horizon in the problem is too large. Thus, the training data collected from the plant should be organized into multiple trajectories so that the optimization problem is tractable to solve.

All the variables for the model identification problem are scaled as follows. The measured variables such as the control inputs and measurements are scaled using the mean and standard deviation of the variable. For example, the measurements are

scaled using the relation

$$y = (y - \mu_y)/\sigma_y \tag{5.18}$$

in which, μ_y and σ_y are the mean and standard deviation of the measurement. These two quantities for scaling are computed over the entire training data collected from the plant. The NNs used to approximate the unknown functions in the hybrid model ODEs should give predictions of the functions in the physical units. The weights and biases in the NNs during the training process typically take some small values. So the outputs of the NNs should also be scaled so that some scaled quantities are actually inferred by the NNs during the training process. This additional scaling should be performed based on how the outputs of the NNs enter the hybrid model. We perform this scaling based on the modeling problem at hand, and we mention our approach for the example systems considered in the case studies in Section 5.5.

5.3 Black-Box modeling

In the case studies in this chapter, we also consider a black-box NN for comparison with the hybrid modeling approach. We next describe the model formulation and the training approach used to develop black-box NNs in the case studies in this chapter. We do not use any first principles based process knowledge to estimate a black-box NN, and the model is developed directly in discrete time using training data.

First, we parameterize a standard feedforward NN to predict the current measurement based on the recent observations of measurements and control inputs contained in the vector z. Then, we write down a state space description of the black-box NN model and use it in a multistep ahead prediction error minimization problem to deter-

mine the unknown parameters in the network.

The black-box NN used to predict the current measurement is parameterized as follows

$$y = h_N(z; \theta_{NN}) \tag{5.19}$$

in which, $h_N(\cdot)$ is the measurement function parameterized by a black-box NN, and θ_{NN} contains the weights and biases in that NN. We use the hyperbolic tangent as the activation function in the NN so that standard gradient based solvers can be applied subsequently over the model for steady-state optimization.

Further, we write the following dynamics for the state z to make the measurement predictions over a future horizon using the black-box model

$$\begin{aligned} z^+ &= \left[z'_{p+1:N_p p},\ h_N(z; \theta_{NN})',\ z'_{N_p p+m:N_p(p+m)},\ u' \right]' \\ z^+ &= f_z(z, u; \theta_{NN}) \end{aligned} \tag{5.20}$$

The equation (5.20) defines the state space form of the black-box model. The vector z is the state in the black-box model. We next use this state space description in the following optimization problem to determine the unknown parameters in the black-box network

$$\min_{\theta_{NN}} \quad \sum_{i=1}^{N_{tr}} \sum_{k=0}^{N_t} |y_i(k) - \hat{y}_i(k)|^2 \tag{5.21}$$

$$\text{s. t.} \quad z_i^+ = f_z(z_i, u_i; \theta_{NN}) \tag{5.22}$$

$$\hat{y}_i = h_N(z_i; \theta_{NN}) \tag{5.23}$$

Here, the subscript i is used to enumerate the training data trajectories. The problem is

also a multistep ahead prediction error minimization problem. The decision variables in the optimization problem are the weights and biases in the measurement function NN (θ_{NN}). The optimization problem is also solved using the software TensorFlow and the stochastic gradient algorithm Adam.

We note that the black-box NN model formulation and the training approach discussed in this section can also be viewed as a recurrent NN with a predefined state (z) and dynamics. We scale all the variables (measurements and control inputs) for the training problem using their mean and standard deviations computed over the entire training data.

5.4 Steady-state economic optimization

The intended purpose of developing the process models is to finally use them for process optimization. The RTO layer in the process industries uses a nonlinear process model to determine the economically profitable steady states in real time during the process operation. The RTO layer is thus a suitable and impactful place to use the developed hybrid models for process operations.

In the case studies, we solve the following steady-state optimization problem typically solved at the RTO layer

$$\min_{x_s, u_s} \quad \ell(x_s, u_s; p) \tag{5.24}$$

$$\text{s. t.} \quad f(x_s, u_s) = 0 \tag{5.25}$$

$$g(x_s; p) \leq 0 \tag{5.26}$$

$$\underline{u} \leq u_s \leq \overline{u} \tag{5.27}$$

in which, p denotes some parameters characterizing the raw material cost, product

price, or some specific product goal. The decision variable in this optimization problem is the steady-state pair (x_s, u_s). The steady-state constraint (5.25) in the optimization problem is implemented based on the type of the process model. For the plant and hybrid models that do not use the variable z, we implement that constraint in continuous time as shown in the problem. For the black-box and hybrid models that use the variable z, the steady-state constraint is implemented in discrete time using the equation $f(x_s, u_s) = x_s$.

The optimization problem (5.24) is solved using the software CasADi (Andersson et al., 2019). For the case studies, we examine the following two aspects to gauge the suitability of the estimated hybrid models for use in steady-state optimization

- The nature of the cost curve defined by the function $\ell(\cdot)$ at the steady states of the model for a fixed parameter p.
- The solutions of the optimization problem for a range of economically relevant parameters $p \in [\underline{p},\ \overline{p}]$.

For both the types of analysis, we compare the solutions obtained with the estimated models with that of the plant.

5.5 Application examples

In this section, we present case studies using two nonlinear chemical process examples to demonstrate the ability of the hybrid process modeling approach to yield accurate dynamic models, which can also achieve good economic performance when subsequently used in steady-state optimization. We highlight the type of training data that should be collected from the plant if the goal is to use the model subsequently in steady-state optimization. We also consider multiple hybrid models with different

parameterization choices of the NNs used to represent the unknown functions, and examine the effect of the choices on the final economic performance.

5.5.1 Illustrative chemical reactor

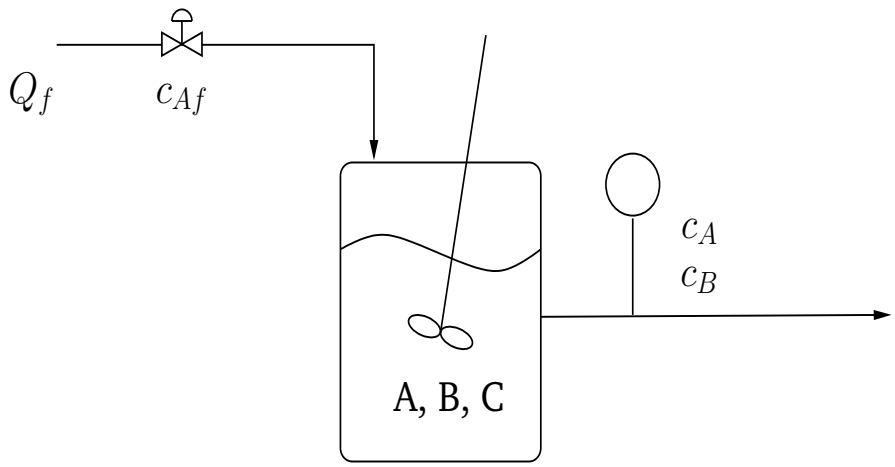

Figure 5.1: Diagram of the simple continuous stirred tank reactor (CSTR).

A simple continuous stirred tank reactor (CSTR) shown in Figure 5.1 is studied for the first example. A single feed stream enters the reactor, which operates at a constant temperature and volume. The feed stream consists of primarily the species A. The CSTR facilitates the following two reactions to form a desired species B and an undesired species C

$$\mathrm{A} \xrightarrow{\mathrm{r_1}} \mathrm{B} \tag{5.28}$$

$$3\,\mathrm{B} \xleftrightarrow{\mathrm{r_2}} \mathrm{C} \tag{5.29}$$

The second reaction that produces the undesired species C is reversible. We use the following equations to simulate the plant

$$\frac{dc_A}{dt} = \frac{F(c_{Af} - c_A)}{V} - r_1 \tag{5.30}$$

$$\frac{dc_B}{dt} = \frac{-Fc_B}{V} + r_1 - 3r_2 \tag{5.31}$$

$$\frac{dc_C}{dt} = \frac{-Fc_C}{V} + r_2 \tag{5.32}$$

$$r_1 = k_1 c_A, \quad r_2 = k_{2f} c_B^3 - k_{2b} c_C \tag{5.33}$$

The concentration of the three species are the states in the plant with $x = [c_A, c_B, c_C]'$. The measurements are the concentrations of only the species A and B with $y = [c_A, c_B]'$. The feed flow rate F to the reactor remains constant. And the concentration of the species A in the feed stream is the manipulated control input $u = c_{Af}$. The sample time of the measurements from the plant is 1 minutes. The other model details such as the parameters used in the ODEs (5.30) – (5.32), the covariance of the measurement noise, etc, used to simulate the plant are given in the Appendix 5.6 in Table 5.4.

We develop a hybrid model for the chemical reactor by first writing down the structure of the dynamic model as set of ODEs using mass balances. We assume that the two reaction rates in the ODEs are unknown. We parameterize the reaction rates using black-box NNs, and the unknown weights and biases in these NNs are subsequently estimated using training data collected from the plant. We consider two choices of the NN parameterization to approximate the reaction rates and develop two hybrid models. For the first model, we assume that the knowledge about the existence of the species C is known during the process modeling step. So we write down the mass balances for all the three concentrations and the reaction rates are parameterized using two separate

NNs as follows

$$r_1 = f_{r1}(c_A; \theta_{r1}) \tag{5.34}$$

$$r_2 = f_{r2}(c_B, c_C; \theta_{r2}) \tag{5.35}$$

in which, θ_{r1} and θ_{r2} are the parameters in the NNs used to approximate the reaction rates. This hybrid model has the full structural knowledge about the plant dynamics, and we subsequently refer to it as the Hybrid-F model.

For the second model, we assume that the knowledge about the existence of the species C is not known during the process modeling step. We write down the mass balances for only the concentrations of the species A and B. The first reaction is parameterized similarly as the Hybrid-F model. The second reaction is parameterized as a function of the concentration of B and the vector z. We use two NNs for the reaction rates as follows

$$r_1 = f_{r1}(c_A; \theta_{r1}) \tag{5.36}$$

$$r_2 = f_{r2}(c_B, z; \theta_{r2}) \tag{5.37}$$

This hybrid model uses the recent history of information contained in the vector z to *infer* the dependence of the species C on the second reaction rate r_2. This second model uses only some partial knowledge about the plant dynamics, so we refer to it subsequently as the Hybrid-P model.

To develop both the hybrid models, we assume that all the other parameters that describe the mass balances such as the feed flow rate and the volume of the reactor are known. The outputs of the NNs that predict the first reaction rate in the hybrid models are scaled using the standard deviation of the concentration of A. And the output of

the NNs that predict the second reaction rate are scaled using the standard deviation of the concentration of B. Both these two standard deviations for this scaling step are computed over the entire training data.

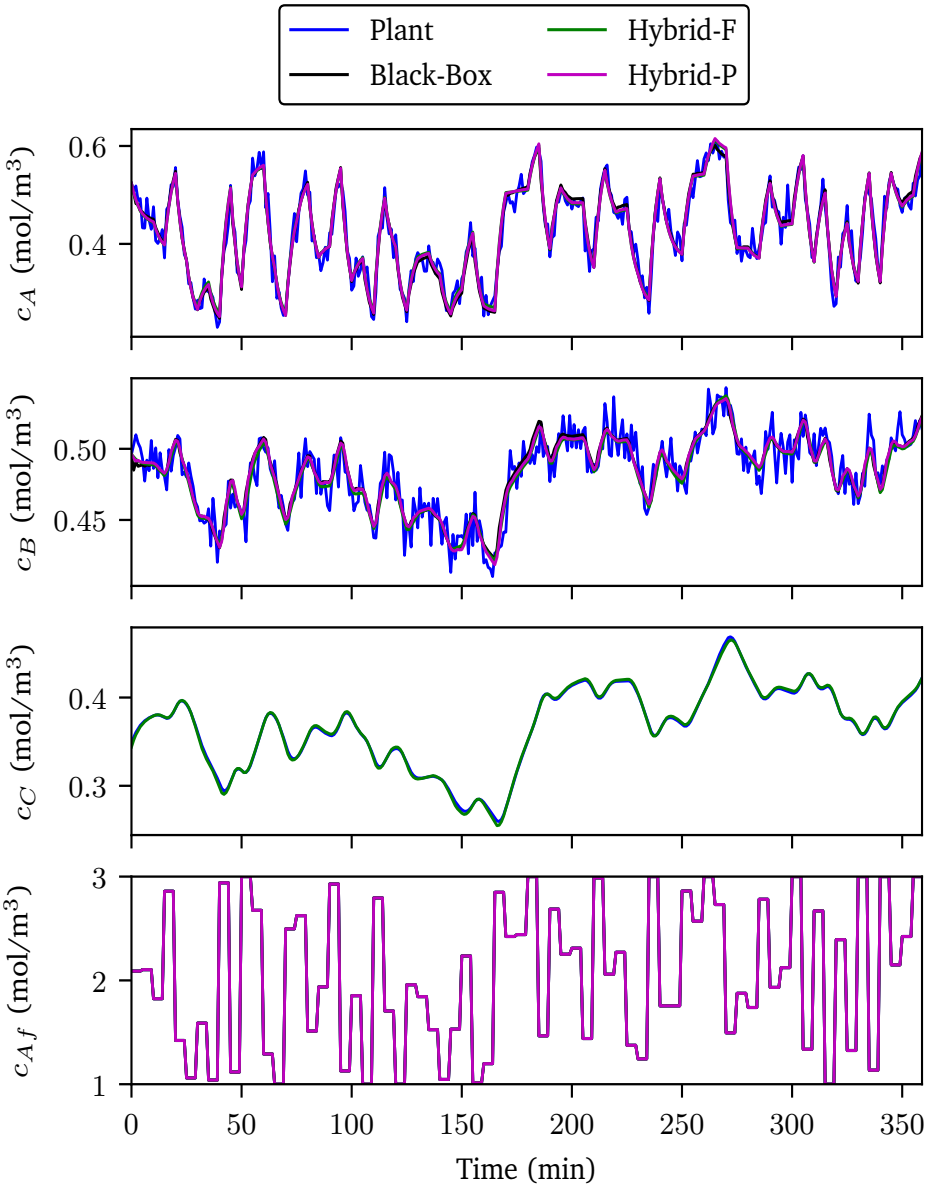

Figure 5.2: Plant measurements and model predictions on the validation data that contains almost no steady-state information about the plant.

First, we illustrate the type of training data that should be collected from the plant

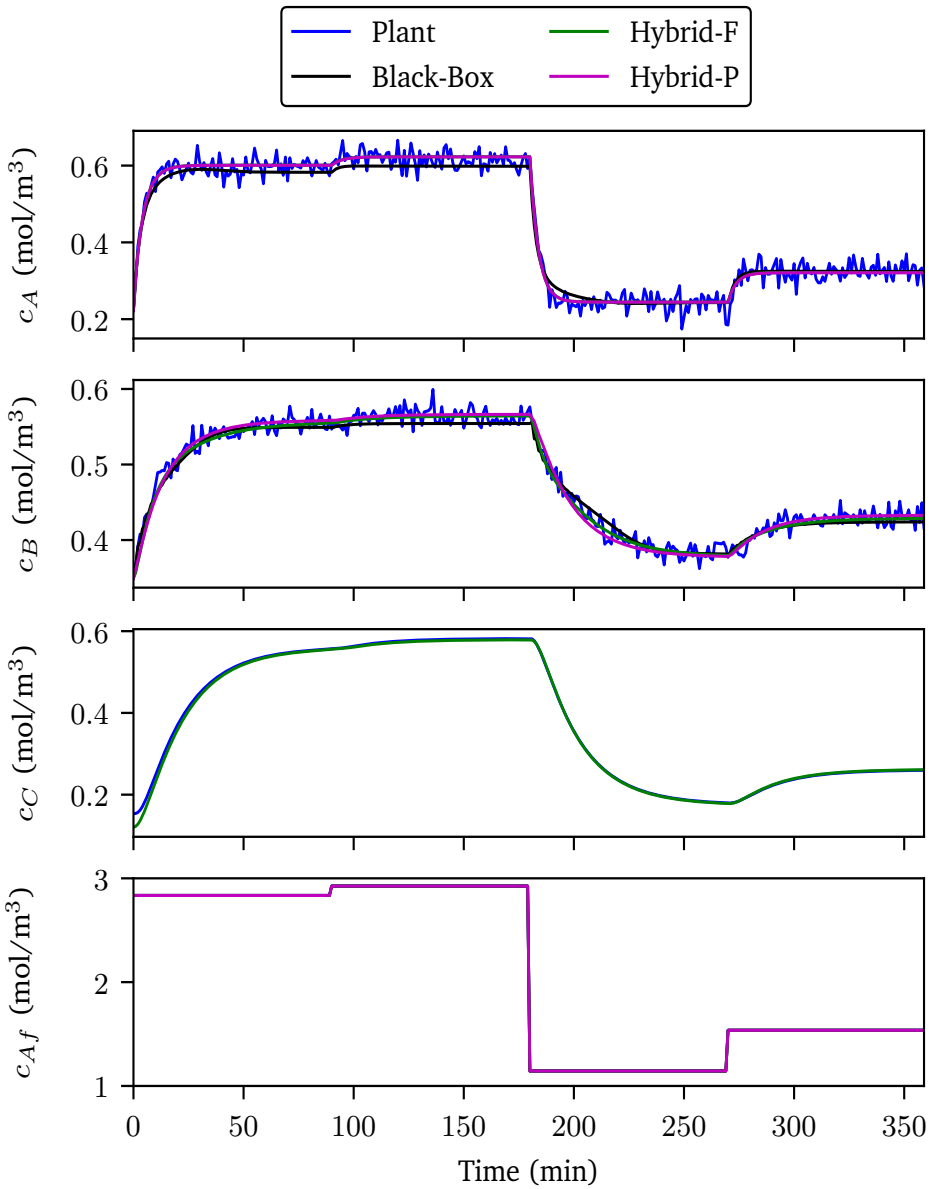

Figure 5.3: Plant measurements and model predictions on the validation data that contains a sufficient steady-state information about the plant.

if the final goal is to use the estimated hybrid models in steady-state optimization. In industrial applications, dynamic data is typically abundant due to time varying disturbances and operating conditions. The abundant availability of dynamic data may encourage practitioners to develop hybrid models from such data, but then use the

estimated models in steady-state optimization. We highlight that to obtain good economic performance using the hybrid models, the plant data used for training should contain enough steady-state information about the plant. With this purpose, we generate two types of data set from the plant containing almost none and a plenty of steady-state information, respectively. We simulate the plant using pseudo random binary signals (PRBS) of the control input. The values of the control input are sampled from a uniform distribution in the constraint set $[\underline{u},\ \overline{u}]$. And we change the sampled values at different hold durations to specify whether any steady-state information is captured in the training data. These hold durations are chosen as 5 and 90 minutes to generate data sets with none and enough steady-state information, respectively. The values of the range ($[\underline{u},\ \overline{u}]$) in which the control inputs are sampled are given in Table 5.4. Both the types of data sets are generated for a total of 36 hours. We use 24 hours for training, 6 hours as the holdout set, and the remaining 6 hours of data for model validation.

Based on the two generated data sets, we train three models: Hybrid-F, Hybrid-P and a Black-Box model. To develop the models, we choose $N_p = 2$ to construct the vector z of recent history of past measurements and control inputs. After the model training, we examine the predictions of the estimated models on the 6 hours of validation data kept from the two types of data sets. Figure 5.2 (no steady-state information) and 5.3 (sufficient steady-state information) show the model predictions on the two validation data sets. We observe that all the three estimated models accurately predict the plant measurements for their respective data sets. In Table 5.1, we report the architectures of the NNs used in the three models, the training time, and the root mean squared error (RMSE) metric obtained by the models on the validation data. The latter two metrics in the table are shown for the case when the models are trained with the data containing enough steady-state information about the plant. The RMSE met-

rics show that all the estimated models predict the plant measurements almost equally well. The training times highlight that the Black-Box model is the fastest to train, but the two hybrid models can also be developed in a reasonable amount of time.

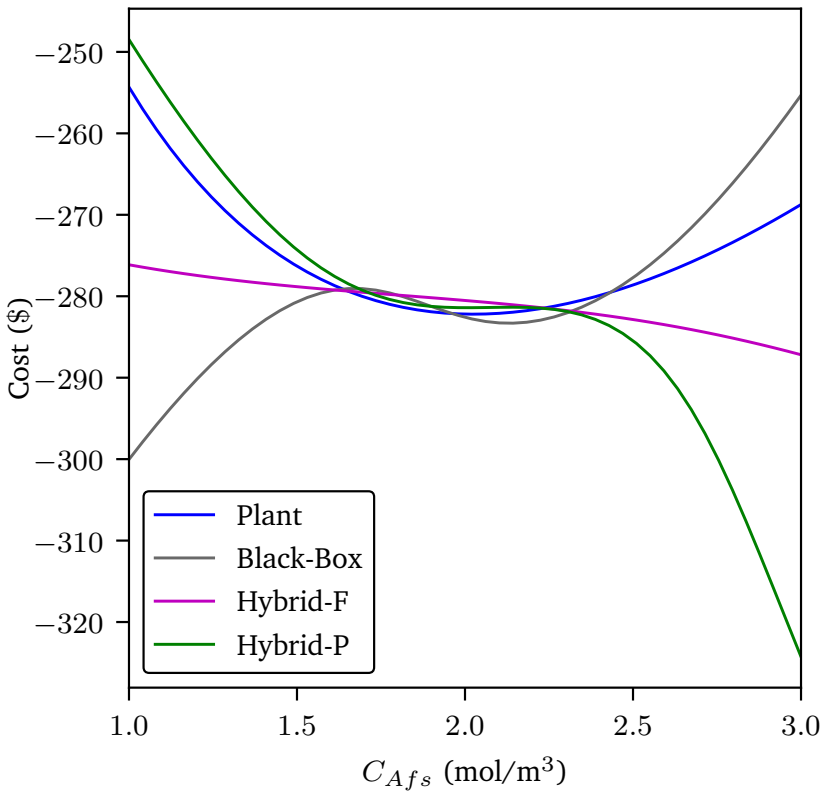

Figure 5.4: Cost curves of the plant and the estimated models when developed using the data set that contains almost no steady-state information.

We next examine the steady-state optimization solutions of the models estimated using the two data sets. We consider an economic optimization problem of the form discussed in the previous Section 5.4. The objective function in the problem accounts for the raw material cost of the species A and selling price of the product B as follows

$$\ell(x_s, u_s; p) = p_A c_{Af} - p_B c_B \tag{5.38}$$

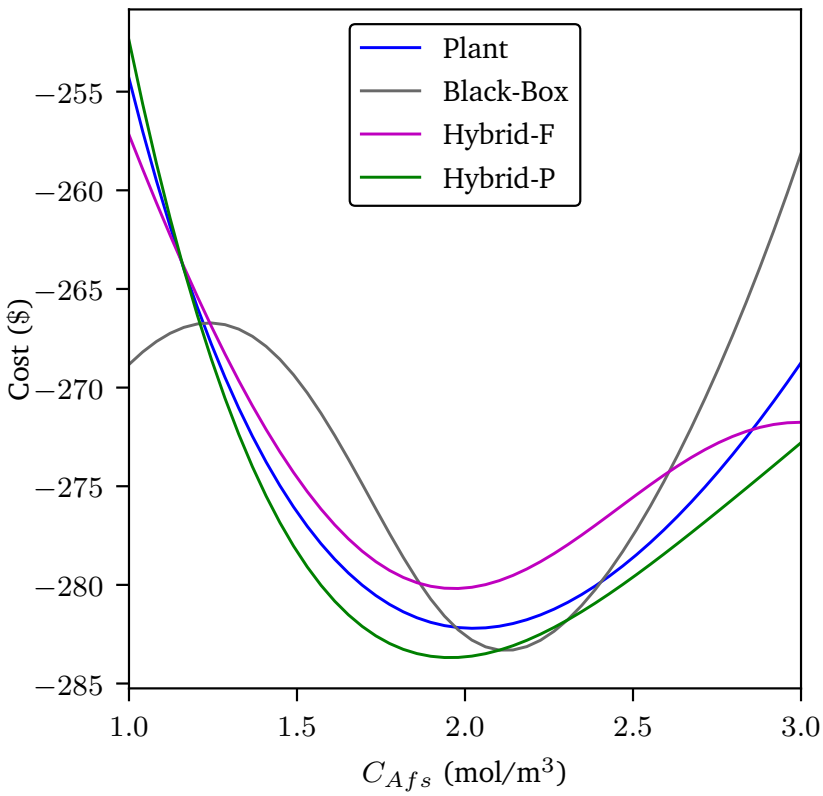

Figure 5.5: Cost curves of the plant and the estimated models when developed using the data set that contains a sufficient steady-state information.

The control inputs constraints ($[\underline{u},\ \overline{u}]$) used for the problem are the same as bounds used to generate the training data, and are given Table 5.4. We do not consider any state constraint $g(\cdot)$ for this example. The parameter vector $p = [p_A,\ p_B]'$ defines the economic cost and characterizes the optimization problem.

Figures 5.4 and 5.5 show the cost curves of the estimated models obtained after training with the two types of data sets. The cost curves are computed by first solving the steady-state equation of the models at some control input values, which are chosen by discretizing the constraint set $[\underline{u},\ \overline{u}]$. And then evaluating the cost function at the computed steady states. The parameter vector in the cost function is chosen as $p = [100, 1000]'$. Figure 5.4 shows the cost curves of the models after training with the

data that contains almost no steady-state information. We notice that although the estimated models obtained a good fit to that type of data (as seen from the model predictions in Figure 5.2), the models have an inaccurate representation of the steady-state plant cost curve. And using the optimal solutions of these estimated models would yield a highly suboptimal economic performance. Figure 5.5 shows the cost curves of the models obtained after training using the data that contains a sufficient steady-state information about the plant. We observe that the cost curves of the hybrid models are similar to the plant curve. The optimal solutions of these two models are also close the plant optimum. The Black-Box model, however, has two optimal solutions. And it is still not reliable after training with the data containing a plenty of steady-state information.

We also examine the quality of the reaction rate approximations by the estimated NNs in the two hybrid models. For this purpose, we compute the relative error ($e = |r_1 - r_{1NN}|/|r_1|$) in the predictions of the first reaction rate in a range of the concentrations of the species A (c_A). The error is computed using the NNs that approximate the first reaction rate in the Hybrid-F and Hybrid-P models, obtained after training with the data set that contains a sufficient steady-state information about the plant. We plot the computed errors in Figure 5.6. We observe that the errors in this first reaction rate are small (below 0.1) in the majority of the state space. Next, we examine the errors in the predictions of the second reaction rate. The second reaction is reversible, and at equilibrium, the reaction rate is zero. So rather than computing the relative error, we use a modified error defined as $e = |r_2 - r_{2NN}| - \epsilon_1|r_2| - \epsilon_2$ to examine the quality of prediction errors in this second reaction. We choose $\epsilon_1 = \epsilon_2 = 5 \times 10^{-2}$, and a value of $e < 0$ means that the reaction rate is well predicted by the NN. The modified error is computed in the state space of the concentrations of the species B and C. We compute the errors for the NN estimated in only the Hybrid-F model, because the NN used to

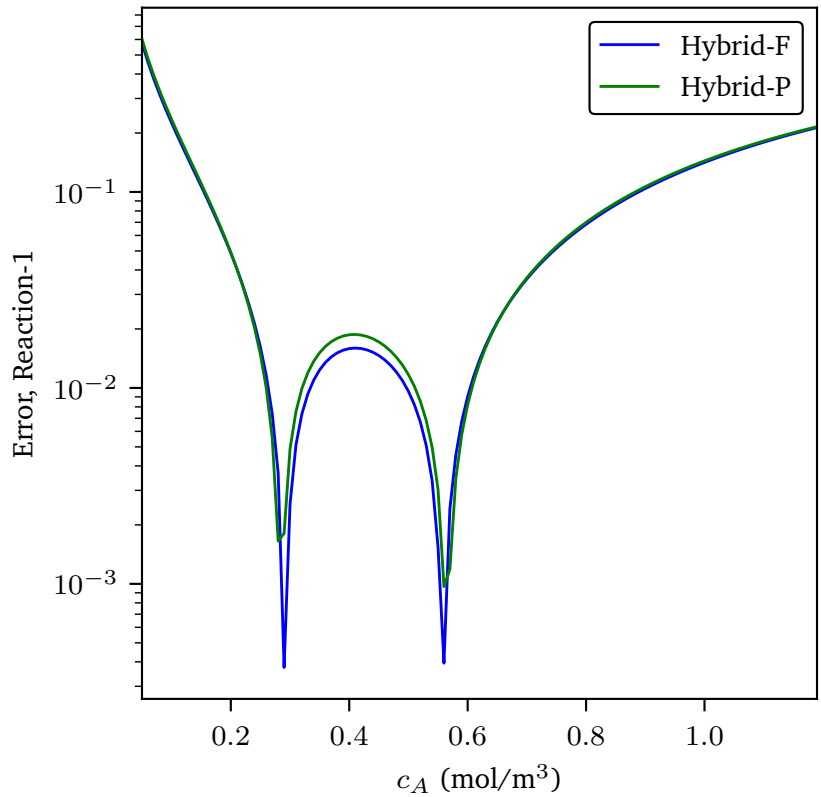

Figure 5.6: Relative errors ($|r_1 - r_{1NN}|/|r_1|$) in the first reaction rate by the estimated NNs in the hybrid models in the state space of the concentration of the species A.

approximate the second reaction rate in the Hybrid-P takes the vector z as the input rather than the concentration c_C. The modified error for the Hybrid-F model is plotted in Figure 5.7. The modified error value in a majority of the state space is less than zero, which show that the second reaction rate is accurately predicted by the NN in the Hybrid-F model.

The cost curves of the estimated models and the plant in Figure 5.5 are shown for a fixed value of the parameter p in the optimization problem. We next examine the solutions of the optimization problem for different values of the parameter p in a range $[\underline{p}, \overline{p}]$. This study is performed to comprehensively gauge the quality of the optimums

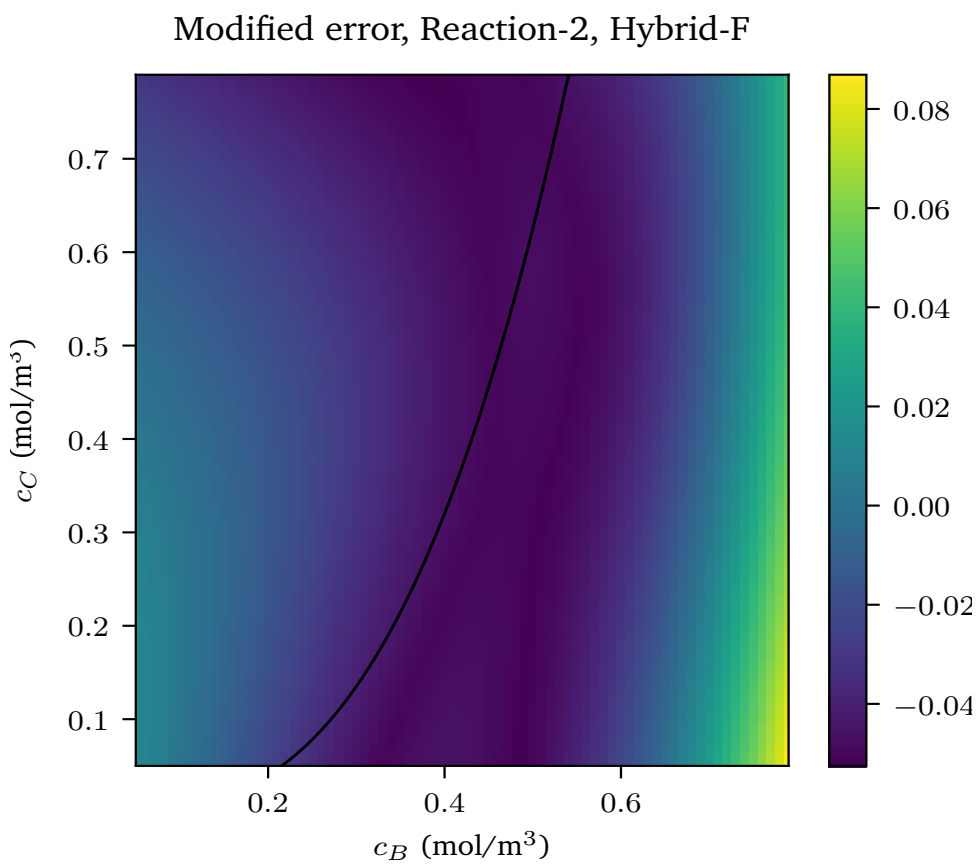

Figure 5.7: Modified errors ($|r_2 - r_{2NN}| - \epsilon_1|r_2| - \epsilon_2$) in the second reaction rate by the estimated NN in the Hybrid-F model in state space of the concentrations of the species B and C. The black line shows the equilibrium curve ($k_{2f}c_B^3 = k_{2b}c_C$) of the second reaction.

of the estimated models for a range of operating conditions. The parameter range $[\underline{p}, \overline{p}]$ (given in Table 5.4) is chosen such that the control input constraint set is thoroughly explored at the optimum solutions of the plant. We sample 100 parameter values in the range using a uniform distribution. For each sampled parameter, we solve the steady-state optimization problem for the plant and the estimated models. Then, we compute the following three metrics to examine the quality of the optimums of the estimated models.

- Optimal input error defined as $|u_s - u_s^*|/|u_s^*|$. In this metric, u_s^* is the optimal

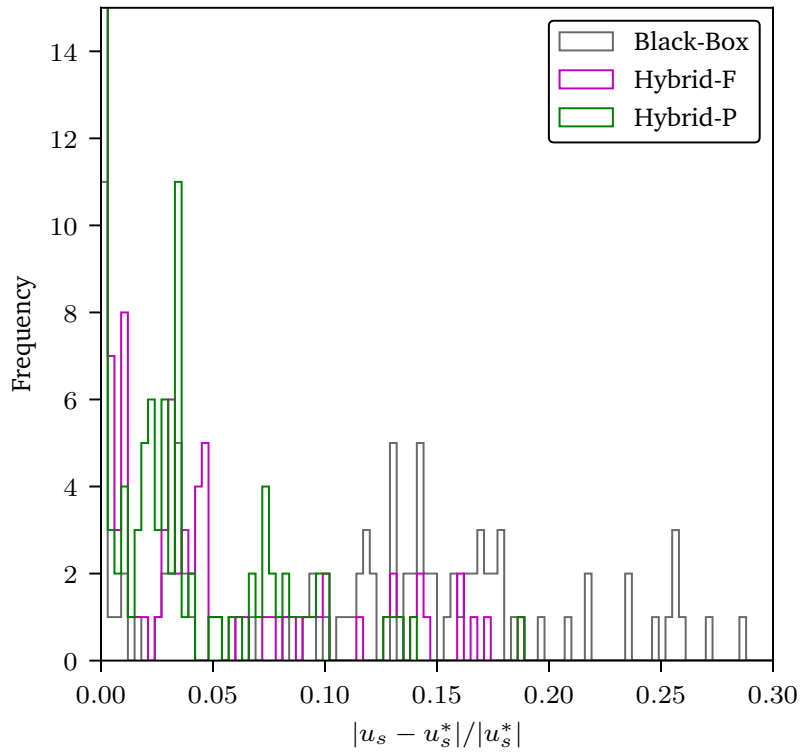

Figure 5.8: Histograms of the optimal control input errors obtained after solving the optimization problems with the Hybrid-F, Hybrid-P, and Black-Box models.

control input of the plant, and u_s is the optimal control input of the estimated model.

- Performance loss of the model defined as $|V_s - V_s^*|/|V_s^*|$. Here, V_s^* is the cost value obtained by solving the optimization problem with the plant model. And V_s is the cost obtained when the plant is operated at the steady state corresponding to the optimum of the estimated model.

- Time required to solve each optimization problem.

We initialize all the optimization problems at the steady state of the model corresponding to the midpoint of the input constraint set. Figures 5.8 and 5.9 show the histograms of the optimal input errors and the performance losses obtained after solv-

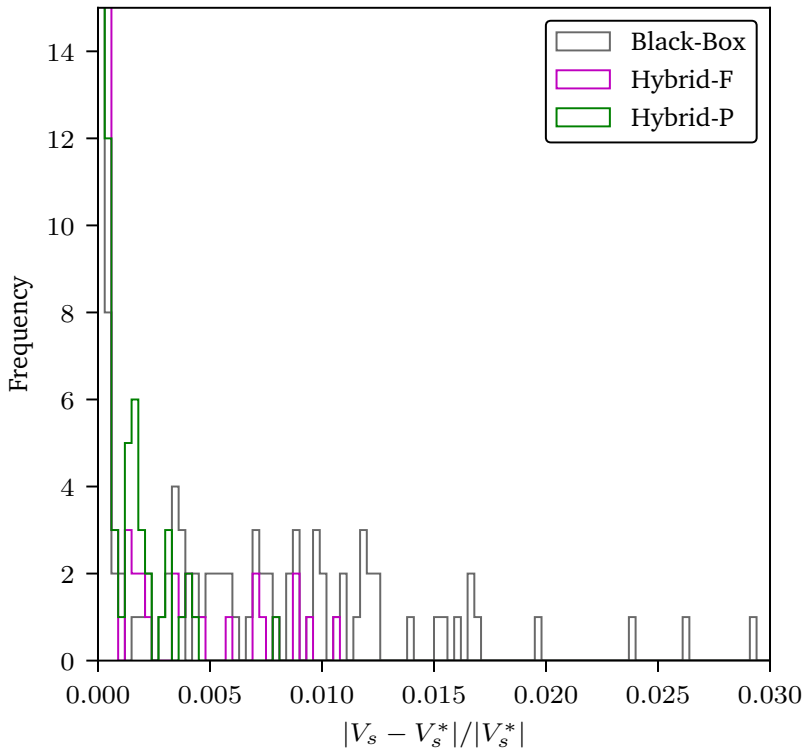

Figure 5.9: Histograms of the performance losses obtained after solving the optimization problems with the Hybrid-F, Hybrid-P, and Black-Box models.

ing all the optimization problems for the plant and the estimated models. The means of the above three metrics are also reported in Table 5.1. We observe that all the models provide good economic performance with the loss metric between $0.08 - 0.61\%$. The Black-Box model has larger optimal control input errors than the hybrid models with a mean of 11.6%. We note, however, that the performance of the Black-Box model can deteriorate significantly for a poor initial guess in the steady-state problem. Because we also observe from the cost curve in Figure 5.5 that the Black-Box model has two optimums. The optimization problem solution times reported in Table 5.1 are less than 1 seconds for all the models, which show that the steady-state problem can be conveniently solved in applications with all the estimated models.

Model	Neural network architectures	Prediction error (RMSE)	Training time (min)	Average Optimal input error	Average Performance loss	Average Optimization time (sec)
Hybrid-F	$f_{r1} = [1, 8, 1]$ $f_{r2} = [2, 32, 32, 1]$ $f_0 = [2, 16, 1]$	0.16	14.97	4.24 %	0.13 %	0.50
Hybrid-P	$f_{r1} = [1, 8, 1]$ $f_{r2} = [7, 32, 32, 1]$	0.16	14.51	3.77 %	0.08 %	0.68
Black-Box	$h_N = [6, 64, 64, 2]$	0.18	2.89	11.65 %	0.61 %	0.52

Table 5.1: Metrics summarizing the simulation study performed for the chemical reactor example.

5.5.2 Styrene polymerization process

Polymerization reactors have a large-scale application in the process industries, and they also pose significant challenges for dynamic modeling and control due to their nonlinear dynamics (Congalidis and Richards, 1998; Ray, 1985). We next consider a styrene polymerization example to demonstrate the effectiveness of the hybrid modeling approach to yield an accurate model that can also be used to obtain good steady-state economic performance.

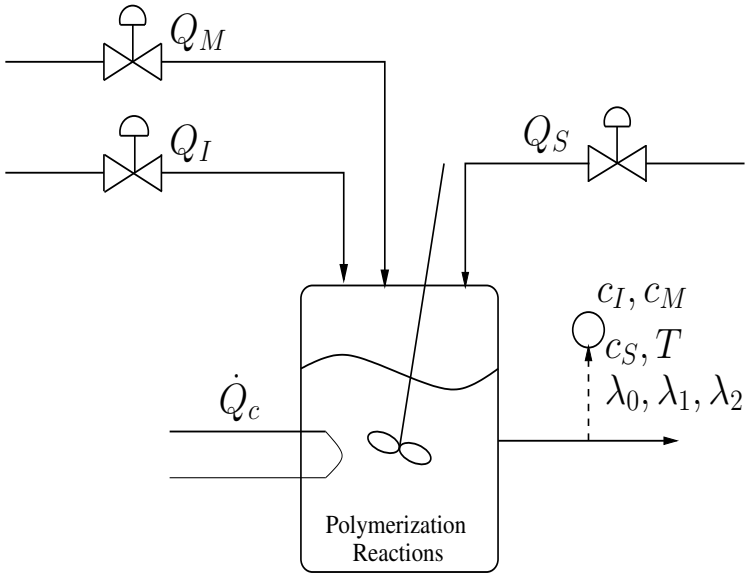

Figure 5.10: A schematic of the styrene polymerization system.

The polymerization reactor system shown in Figure 5.10 is considered. Three feed streams enter the reactor, containing the initiator, the monomer, and the solvent. The outlet stream contains the polystyrene product and unreacted monomer. We use the ODEs shown in equations (5.39) – (5.45) to simulate the plant. The dynamic model is obtained from Ray (1972); Hidalgo and Brosilow (1990); Prasad et al. (2002). The plant has four manipulated control inputs $u = [Q_I, Q_M, Q_S, \dot{Q}_c]'$, which correspond to the flow rates of the three feed streams and the cooling duty supplied to the reactor to remove the heat generated by the polymerization reactions. The states (x) in the plant dynamics are the concentrations of the initiator (c_I), the monomer (c_M), the solvent (c_S), the reactor temperature (T), and the first three moments characterizing the polymer molecular weight distribution ($\lambda_0, \lambda_1, \lambda_2$).

$$\frac{dc_I}{dt} = \frac{Q_I c_{If} - Q_o c_I}{V} - r_I(c_I, T) \tag{5.39}$$

$$\frac{dc_M}{dt} = \frac{Q_M c_{Mf} - Q_o c_M}{V} - r_M(c_I, c_M, T) \tag{5.40}$$

$$\frac{dc_S}{dt} = \frac{Q_S c_{Sf} - Q_o c_S}{V} \tag{5.41}$$

$$\frac{dT}{dt} = \frac{Q_o}{V}(T_f - T) + \frac{r_M(c_I, c_M, T)\Delta H_r}{\rho C_p} - \frac{\dot{Q}_c}{\rho C_p V} \tag{5.42}$$

$$\frac{d\lambda_0}{dt} = g_0(c_I, c_M, c_S, T) - \frac{Q_o \lambda_0}{V} \tag{5.43}$$

$$\frac{d\lambda_1}{dt} = g_1(c_I, c_M, c_S, T) - \frac{Q_o \lambda_1}{V} \tag{5.44}$$

$$\frac{d\lambda_2}{dt} = g_2(c_I, c_M, c_S, T) - \frac{Q_o \lambda_2}{V} \tag{5.45}$$

The reaction rates of the initiator and monomer (r_I, r_M), and the functions characterizing the polymer moment dynamics (g_0, g_1, g_2) are all nonlinear functions of the quantities shown in the ODEs. The expressions for these five functions and other model details used to simulate the plant are given in Appendix 5.6 and Table 5.5. We non-

dimensionalize the ODEs to simulate the plant. We assume that all the states in the plant are measured in the training data to develop the hybrid and black-box models. And the sample time of the measurements is chosen as 1 minutes.

When developing a dynamic model for this styrene polymerization system, parameterizing the reaction rates (r_I, r_M) and the moment functions (g_0, g_1, g_2) using the available first principles knowledge can be a challenging task in applications. We apply the hybrid modeling approach for the polymerization system and use black-box NNs to approximate these functions using training data collected from the plant. First, we write down the dynamics for all the states using mass and energy balances and assume that those five functions are unknown. We parameterize the functions using NNs. We consider two plausible parameterization choices of the NNs to approximate the unknown functions and develop two hybrid models.

For the first model, we assume that the functional dependencies of the reaction rates and polymer moment functions are known exactly as in the plant. We parameterize the functions using five separate NNs, and the known functional dependencies of each function are provided as inputs to the NNs. The five NNs are parameterized as follows

$$r_I = f_{rI}(c_I, T; \theta_{rI}) \tag{5.46}$$

$$r_M = f_{rM}(c_I, c_M, T; \theta_{rM}) \tag{5.47}$$

$$g_0 = f_{g0}(c_I, c_M, c_S, T; \theta_{g0}) \tag{5.48}$$

$$g_1 = f_{g1}(c_I, c_M, c_S, T; \theta_{g1}) \tag{5.49}$$

$$g_2 = f_{g2}(c_I, c_M, c_S, T; \theta_{g2}) \tag{5.50}$$

in which, $\theta_{rI}, \theta_{rM}, \theta_{g0}, \theta_{g1}$, and θ_{g2} denote the parameters in the five networks. The hybrid model developed using this approach utilizes the full structural knowledge about

the plant dynamics, and the dependencies of the unknown functions. We refer to the model developed with this approach as the Hybrid-F model.

For the second hybrid model, the five unknown functions are parameterized using two NNs (vector-valued) as follows

$$\begin{bmatrix} r_I \\ r_M \end{bmatrix} = f_r(c_I, c_M, c_S, T; \theta_r) \tag{5.51}$$

$$\begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = f_g(c_I, c_M, c_S, T; \theta_g) \tag{5.52}$$

in which, θ_r and θ_g denote the parameters in the two networks. The first NN $f_r(\cdot)$ has some extra noisy inputs to predict the raection rates. And both the NNs in this parameterization have the potential to learn some unrelated correlations across the functions due to the limited amount of available training data. We examine the performance of this hybrid model particularly to study the impact on the final economic performance when some mismatch in the NN function parameterization are made during the process modeling step. The hybrid model developed with the above NN parameterization has only partial knowledge about the dependencies of the functions being approximated. So we refer to the model developed with this approach as the Hybrid-P model. For the training optimization problem, we scale the NN outputs of the functions r_I, r_M, g_0, g_1, g_2 using the standard deviations of the variables $c_I, c_M, \lambda_0, \lambda_1, \lambda_2$, respectively. The standard deviations of these variables for the scaling step are computed using their measurements over the entire collected training data.

We generate a total of 5 days of training data set by simulating the styrene polymerization plant using a PRBS control input sequence. From the 5 days of data, we

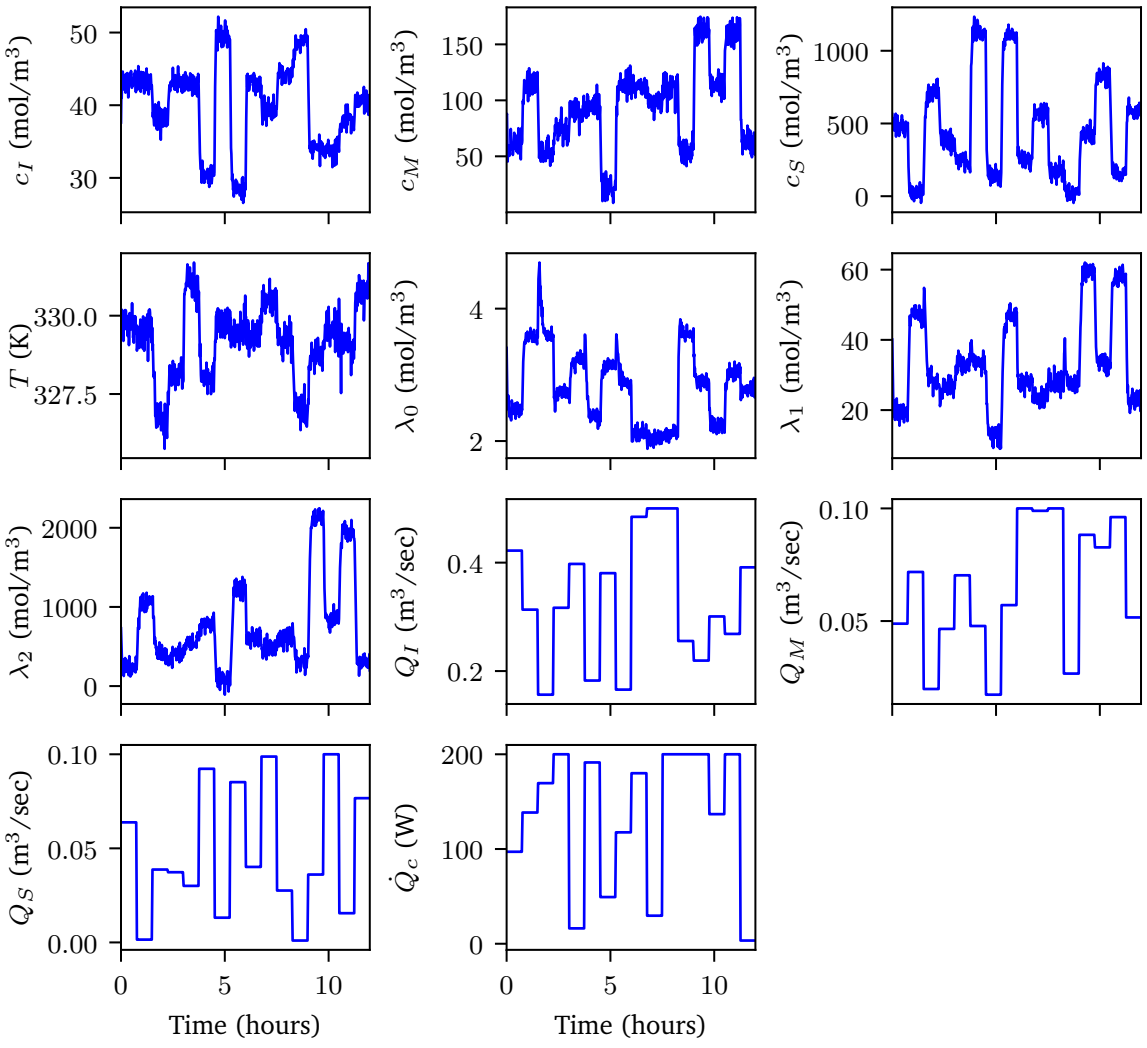

Figure 5.11: Twelve hours of sample training data set used to develop the hybrid and black-box models for the styrene polymerization system.

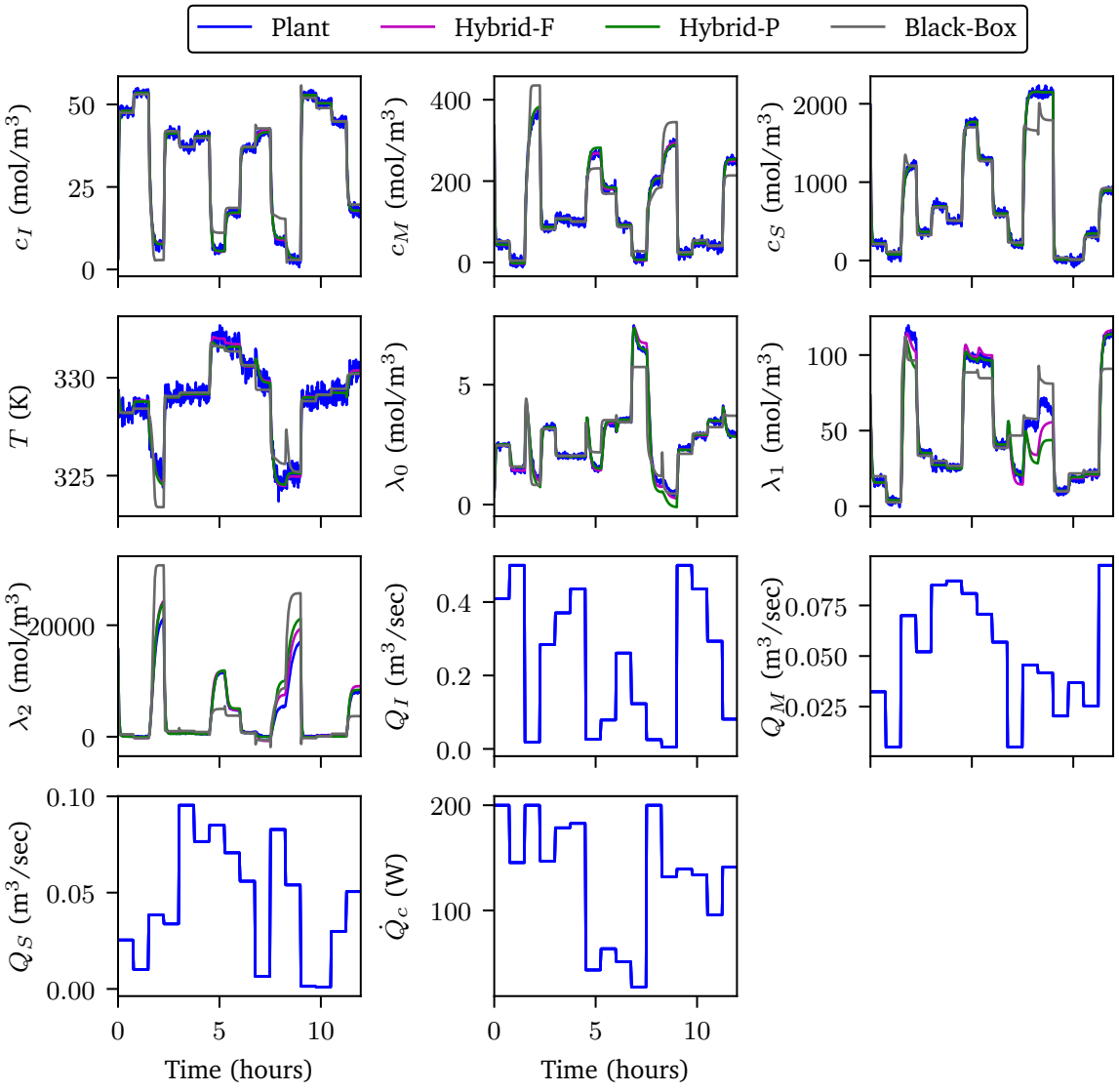

Figure 5.12: Predictions of the estimated hybrid and black-box models compared to the plant measurements on the validation data set. The hybrid models provide accurate predictions of the plant measurements, but the black-box model has a noticeably poor predictions at many time steps.

use 4 days for training, 12 hours as the holdout set, and keep 12 hours of data for validation. The first 12 hours of measurements and control inputs in the training data is shown in Figure 5.11. Based on the generated data, we train three models: Hybrid-F, Hybrid-P, and a Black-Box model. The architectures of the NNs used in the three models are given in Table 5.3. The training data for the multistep ahead prediction error minimization problem is split into multiple trajectories each of length 1 hour. So that the size of the computation graph constructed to solve the problem is reasonable, and the model identification problem can be solved tractably.

After the model training step, we examine the predictions of the estimated models on the 12 hours of data kept for validation. Figure 5.12 shows the predictions of the estimated models on the validation data. We observe that both the Hybrid-F and Hybrid-P models accurately predict the plant measurements. The Black-Box model, however, has a noticeably poor predictions at many time steps. We also compute the RMSE metric of the estimated models on the 12 hours of validation data, which are reported in Table 5.3. The metric is around 0.12 and 0.15 for the two hybrid models, whereas 0.34 for the Black-Box NN model. The metrics also highlight that the Black-Box model has poor predictions of the plant measurements compared to the hybrid models.

We also report the time required to solve the training optimization problem for the three models in Table 5.3. The Black-Box model is the fastest to train, and requires around 7.5 minutes. The hybrid models require around 38 and 31 minutes for training. The training optimization problem would be typically solved offline in applications. And the times required to solve the training problem for all the models show that they can be conveniently developed in applications after the training data is collected from the plant.

We next examine the quality of function approximations obtained by the estimated

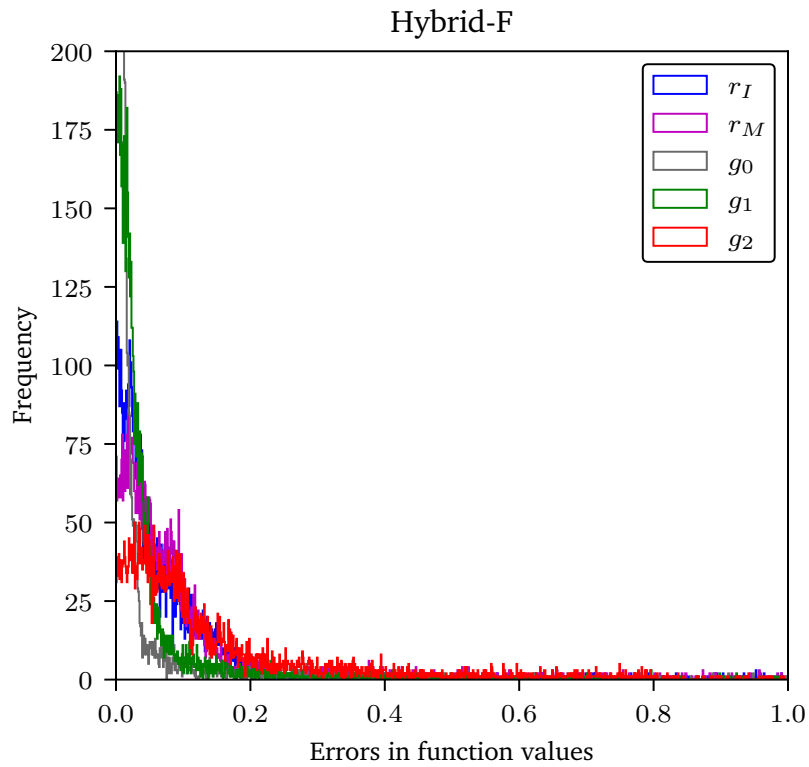

Figure 5.13: Histograms of the approximation errors of the reaction rates and polymer moment functions by the estimated NNs in the Hybrid-F model.

NNs in the hybrid models. Unlike the previous CSTR example, visualizing the errors over the state space is not possible for all the functions in this example because some of them depend on more than two variables. So we compute the errors on the training data set samples and examine the approximation quality on only the relevant state space. For each training data sample, we compute the error metric $e = |f - f_{NN}| \,/\, |f|$, for each approximated function. Here, f denotes the actual function value in the training data, and f_{NN} is the NN approximation of that function. We compute the error metric for all the five functions using the NNs estimated in both the hybrid models. Figures 5.13 and 5.14 show the histograms of the errors in the fives functions for the Hybrid-F and Hybrid-P models, respectively. We also compute the median of the error metrics for all the functions in both the hybrid models. The median errors are reported

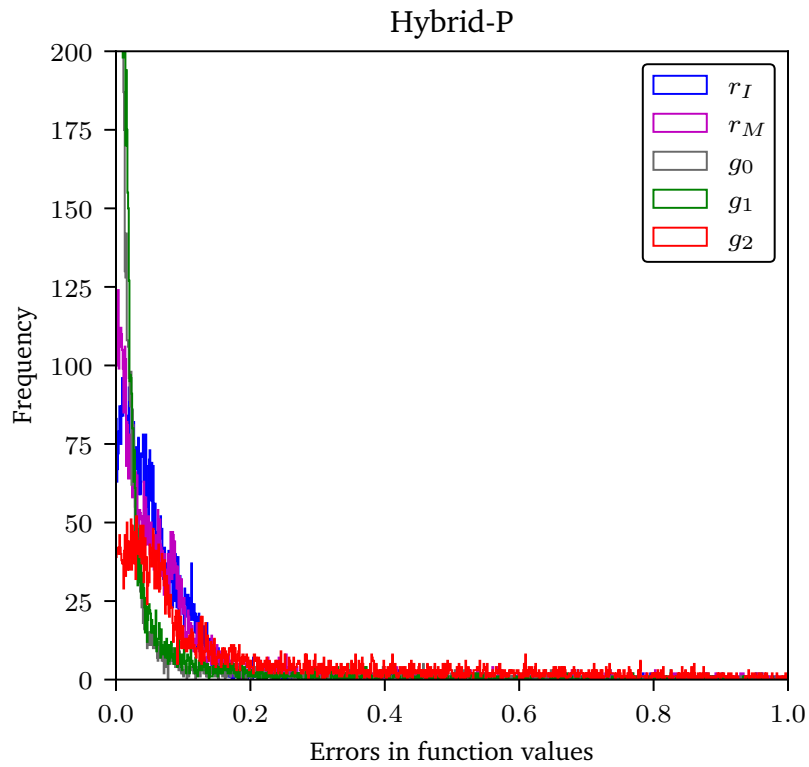

Figure 5.14: Histograms of the approximation errors of the reaction rates and polymer moment functions by the estimated NNs in the Hybrid-P model.

in Table 5.2. We observe that the median errors range between $0.9 - 12.1\%$ across all the functions in both the hybrid models. The histograms in Figures 5.13 and 5.14 and the median errors reported in Table 5.2 both illustrate that all the estimated NNs in the hybrid models have obtained a good approximation of the unknown functions in the styrene polymerization system.

The performances of the estimated models when they are used in an economic optimization problem are examined next. We consider a steady-state optimization problem of the form discussed in Section 5.4. We use the following economic cost and state

Function	Hybrid-F	Hybrid-P
r_I	4.47%	4.86%
r_M	6.03%	4.91%
g_0	0.96%	1.13%
g_1	2.23%	1.49%
g_2	10.41%	12.05%

Table 5.2: Median of the function approximation errors obtained by the estimated NNs in both the hybrid models.

constraint in the problem

$$\ell(x_s, u_s; p) = p_I \rho_I Q_{Is} + p_M \rho_M Q_{Ms} + p_S \rho_S Q_{Ss} + p_E Q_{cs} - p_{PR} M_m (Q_{Is} + Q_{Ms} + Q_{Ss}) \lambda_{1s} \tag{5.53}$$

$$g(x_s; p) = p_{MW} - M_m \frac{\lambda_{2s}}{\lambda_{1s}} \tag{5.54}$$

The stage cost $\ell(\cdot)$ considers the raw material costs of the initiator, monomer, solvent, the energy cost incurred for reactor cooling, and the polystyrene product selling price. The last term in the stage cost uses the production rate from the reactor $M_m(Q_{Is} + Q_{Ms} + Q_{Ss})$, to compute the total product selling price. The state constraint $g(\cdot)$ imposes a desired molecular weight constraint to be achieved in the polymer product during the reactor operation. The vector $p = [p_I, p_M, p_S, p_E, p_{PR}, p_{MW}]'$ contains all the parameters that characterize the optimization problem. The control input constraints $[\underline{u},\ \overline{u}]$ used for the problem are given in Table 5.5.

We sample a total of 500 values of the parameter vector p using a uniform distribution in a range $[\underline{p},\ \overline{p}]$. And the steady-state optimization problem is solved for each estimated models and the plant with all the sampled parameter values. The values of the bounds on the parameter p are given in Table 5.5. We choose these bounds such that the optimal solutions of the plant thoroughly span the input constraint set. For

each sampled parameter value and the optimization problem solved with the estimated models, we compute the following four metrics

- Optimal input error: $|u_s - u_s^*|/|u_s^*|$
- Economic performance loss: $|V_s - V_s^*|/|V_s^*|$
- Time required to solve each optimization problem.
- Value of the molecular weight constraint $g(\cdot)$ at the steady state of the plant corresponding to optimal control input of the estimated model.

The first three metrics are the same as the ones evaluated for the previous CSTR example. In addition, we also compute the molecular weight constraint metric to gauge whether the desired molecular weight is achieved when the plant is operated at the optimal solution of the estimated models.

Figures 5.15, 5.16, and 5.17 show the histograms of the optimal control input errors, performance losses, and the molecular weight constraint metric for all the optimization problems solved using the sampled parameters and the estimated models. The average of the four metrics discussed above are also computed, which are summarized in Table 5.3. We compute the average of the molecular weight constraint metric for only those problems that give an optimal solution such that the desired molecular weight constraint is not satisfied during the plant operation. This computed mean is reported as the average constraint violation in the Table. In Figure 5.16, we notice that the distribution of the economic performance loss metric for the Hybrid-F model is nonlinear. And it has two modes with an uneven number of samples at the modes. Such a distribution is challenging to characterize using just one metric, so we also report the median of the performance loss metric in Table 5.3.

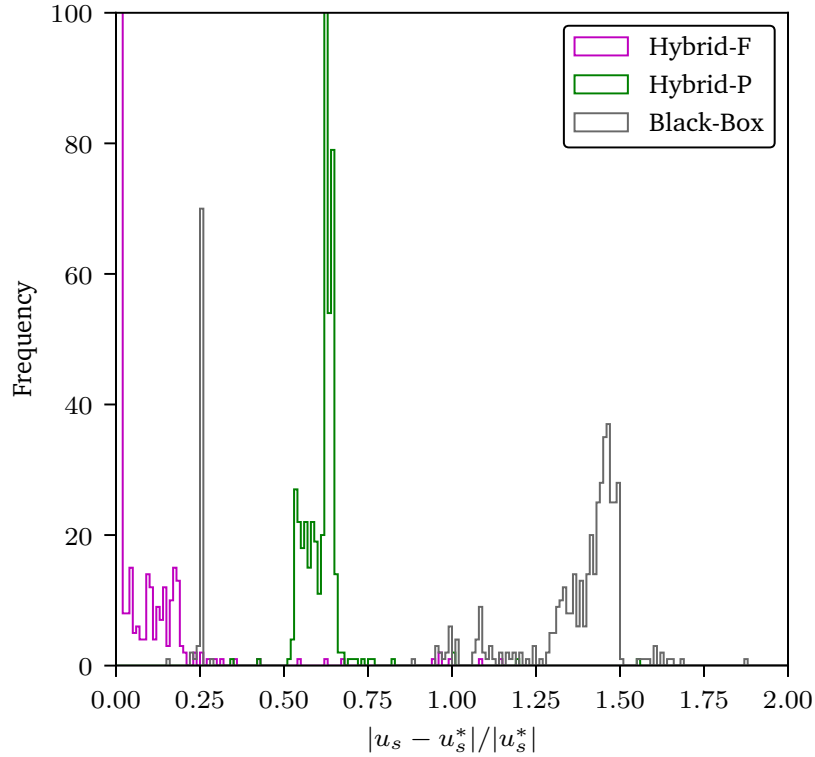

Figure 5.15: Histograms of the optimal control input errors obtained by solving the steady-state optimization problems for the Hybrid-F, Hybrid-P, and Black-Box models.

We observe from the metrics in the Table that the Hybrid-F model provides the best performance. The model achieves the least optimal control input error, performance loss, and constraint violation. The medians of the economic performance loss metric are 6.4%, 17.6%, and 93.7% for the Hybrid-F, Hybrid-P, and Black-Box models, respectively. The Hybrid-P model provides a poor performance compared to the Hybrid-F, whereas, the Black-Box model gives an unacceptable performance for an industrial application.

The average time required to solve the steady-state optimization problem for the Black-Box model is around 22 seconds. And the two hybrid models require around 7 and 12 seconds. The RTO layer optimization problem in the process industries is typically solved online on a slow time scale (hours). The optimization problem solution

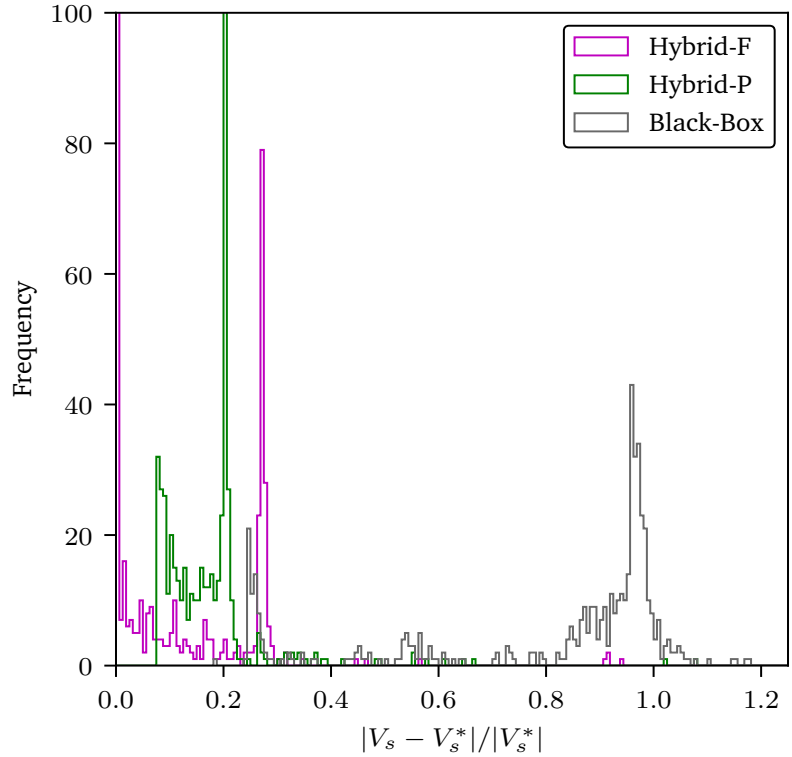

Figure 5.16: Histograms of the economic performance loss metric obtained by solving the steady-state optimization problems for the Hybrid-F, Hybrid-P, and Black-Box models.

times of the estimated models show that they can be conveniently implemented at the RTO layer for steady-state economic optimization.

The improved economic performance of the Hybrid-F model for this styrene polymerization example highlights that hybrid models should be developed be enough structural insights to attain high performance. The performance of the Hybrid-P shows that the economic performance of the estimated hybrid models has some robustness to any mismatch in the NN parameterization during the process modeling step. However, the steady-state economic performance is expected to deteriorate further if any more mismatch in the NN parameterization are made in the process modeling step.

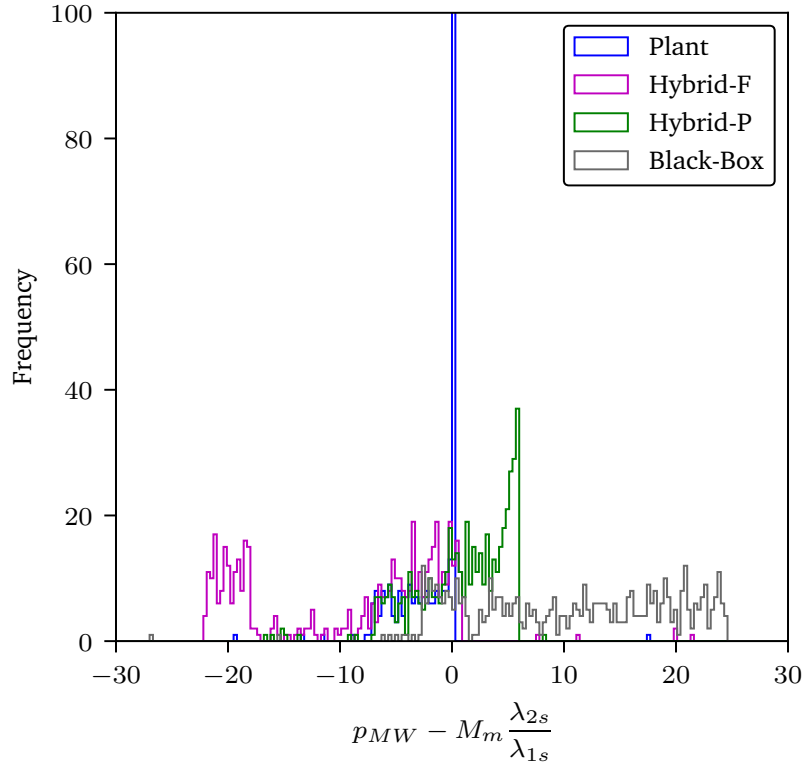

Figure 5.17: Histograms of the molecular weight constraint metric computed by evaluating the state constraint $g(\cdot)$ at the steady state of the plant corresponding to the optimum solutions of the Hybrid-F, Hybrid-P, and Black-Box models.

Model	Neural network architectures	Prediction error (RMSE)	Training time (min)	Average Optimal input error	% Performance loss (Mean, Median)	Average constraint violation	Average Optimization time (sec)
Hybrid-F	f_{rI} = [2, 32, 1] f_{rM} = [3, 32, 1] f_{g0} = [4, 192, 192, 1] f_{g1} = [4, 192, 192, 1] f_{g2} = [4, 192, 192, 1]	0.12	38.11	6.29%	12.20%, 6.41%	2.2	12.14
Hybrid-P	f_r = [4, 64, 2] f_g = [4, 256, 256, 3]	0.15	31.68	61.65%	17.44%, 17.60%	3.56	7.09
Black-box	h_N = [11, 384, 384, 7]	0.34	7.28	122.02%	81.7%, 93.74%	12.99	22.84

Table 5.3: Metrics summarizing the simulation study performed for the nonlinear styrene polymerization example.

5.6 Conclusions

In this chapter, we have presented a hybrid process modeling approach for nonlinear dynamical processes. The modeling approach utilizes the basic available first principles knowledge often available in applications. And combines that knowledge with neural networks (NNs) that approximate some unknown functions in the dynamic model. The parameters in the networks are estimated by solving a multistep ahead prediction error minimization problem, which is one of the novel features of the proposed modeling approach. We treated both the full state and output measurement cases, and multiple NNs to approximate different unknown functions in the hybrid model may be conveniently included in the dynamic model. The output measurement case was treated by using a vector of recent history of measurements and control inputs to predict the unknown functions in the model that may potentially depend on the unmeasured states.

We demonstrated the effectiveness of the hybrid modeling approach via simulation studies with two nonlinear chemical process examples. In the first chemical reactor example, we highlighted that to obtain good steady-state economic performance with the hybrid models, the training data used for model development should contain sufficient steady-state information about the plant. We also examined the quality of the approximated reaction rates in the state space of the concentrations of the different species in the reactor.

We next considered a challenging styrene polymerization example, in which black-box NNs were used to approximate highly nonlinear reaction rates and some polymer moment functions. We considered two different parameterization choices of the NNs and developed two hybrid models. After the training step, the NNs in the hybrid model obtain good approximations of the unknown nonlinear functions. And the overall mod-

els are suitable to accurately predict the plant measurements. We then examined the performances of the estimated hybrid models when they are used in steady-state optimization. We showed that to achieve good performance, the hybrid model should be developed with enough structural insights about the dynamic model and functional dependencies of the unknown functions. For the styrene polymerization example, a hybrid model that utilizes the most possible structural information about the plant achieved 6.7% median performance loss when compared to the plant.

This chapter concludes the contributions of thesis on developing dynamic process models using both the available first principles knowledge and machine learning. In the next final chapter, we summarize the contributions and conclusions from each chapter in this thesis. And outline some future work directions for both hybrid process modeling and feedback controller design using machine learning.

Appendix

Model details for the chemical reactor example

We use the parameters provided in the Table below to simulate the plant and solve steady-state optimization problems in the chemical reactor example considered in Subsection 5.5.1.

Parameters
k_1 = 2×10^{-1} mol/(m^3·min)
k_{2f} = 5×10^{-1} m^6/(mol^2·min)
k_{2b} = 1×10^{-1} mol/(m^3·min)
F = 0.6 m^3/min
V = 15 m^3
R_v = diag([5×10^{-4}, 1×10^{-4}])
$\underline{u}$, $\overline{u}$ = 1, 3 mol/m^3
$\underline{p}$, $\overline{p}$ = [100, 500] , [100, 1500]

Table 5.4: Parameters used for the chemical reactor example.

Model details for the styrene polymerization example

We use the following expressions to compute the reaction rates and polymer moment functions (Ray, 1972; Prasad et al., 2002) in the plant model (5.39) – (5.45) for the styrene polymerization example considered in Subsection 5.5.2.

$$r_I = k_d c_I \tag{5.55}$$

$$r_M = 2f k_d c_I + (k_p + k_f) c_M c_P \tag{5.56}$$

$$g_0 = (k_{fs} k_f c_S c_M + k_{td} c_P)\alpha c_P + (k_{tc} c_P^2/2) \tag{5.57}$$

$$g_1 = \frac{c_P}{1-\alpha}\Big[(k_{fs}c_S + k_f c_M + k_{td}c_P)(2\alpha - \alpha^2) + k_{tc}c_P\Big] \tag{5.58}$$

$$g_2 = \frac{c_P}{(1-\alpha)^2}\Big[(k_{fs}c_S + k_f c_M + k_{td}c_P)(4\alpha - 3\alpha^2 + \alpha^3) + k_{tc}c_P(2+\alpha)\Big] \tag{5.59}$$

Here, c_P characterizes the overall concentration of the polymer radical and it is computed using the expression

$$c_P = \sqrt{\frac{2fk_d c_I}{k_{td} + k_{tc}}} \tag{5.60}$$

The parameter α is the probability of propagation, and it is computed using the expression

$$\alpha = \frac{k_p c_M}{(k_p c_M + k_{fs}c_S + k_f c_M + (k_{td} + k_{tc})c_P)} \tag{5.61}$$

The reaction mechanism for the styrene polymerization process is outlined in Hidalgo and Brosilow (1990). The rate constants in all the expressions above depend on the reactor temperature, and they are computed as follows

$$k_d = k_{d0}e^{-E_d/RT},\ k_p = k_{p0}e^{-E_p/RT},\ k_{fs} = k_{fs0}e^{-E_{fs}/RT}$$

$$k_f = k_{f0}e^{-E_f/RT},\ k_{tc} = k_{tc0}e^{-E_{tc}/RT},\ k_{td} = k_{td0}e^{-E_{td}/RT}$$

To generate the training data and solve the steady-state optimization problem for the plant, we non-dimensionalize the ODES (5.39) – (5.45) by redefining some of the

Parameters	Parameters
f = 0.6	k_{d0} = 2.38×10^{16} 1/sec
ΔH_r = 2×10^3 J/mol	k_{p0} = 4.24×10^6 m^3/(mol·sec)
ρC_P = 6×10^4 J/m^{3}K	k_{fs0} = 3.65×10^4 m^3/(mol·sec)
V = 50 m^3	k_{f0} = 2.12×10^4 m^3/(mol·sec)
c_{If} = 58.8 mol/m^3	k_{tc0} = 2.5×10^8 m^3/(mol·sec)
c_{Mf} = 869 mol/m^3	k_{td0} = 2.5×10^8 m^3/(mol·sec)
c_{Sf} = 4000 mol/m^3	E_d = 1.23×10^5 J/mol
T_f = 330K	E_p = 2.95×10^4 J/mol
M_m = 0.10415 kg/mol	E_{fs} = 9.14×10^4 J/mol
ρ_I = 1100 kg/m^3	E_f = 5.30×10^4 J/mol
ρ_M = 909 kg/m^3	E_{tc} = 7.01×10^3 J/mol
ρ_S = 862 kg/m^3	E_{td} = 7.01×10^3 J/mol
R = 8.314 J/(K·mol)	Q_{os} = 0.15 m^3/sec
Q_{cs} = 100 J/sec	
$\underline{u} = [0.005\ \mathrm{m}^3/\mathrm{sec},\ 0.005\ \mathrm{m}^3/\mathrm{sec},\ 0.001\ \mathrm{m}^3/\mathrm{sec},\ 0\ \mathrm{W}]'$	
$\overline{u} = [0.5\ \mathrm{m}^3/\mathrm{sec},\ 0.1\ \mathrm{m}^3/\mathrm{sec},\ 0.1\ \mathrm{m}^3/\mathrm{sec},\ 2 \times 10^2\ \mathrm{W}]'$	
$\underline{p} = [0,\ 0,\ 0.1,\ 0.1,\ 50,\ 1]'$	
$\overline{p} = [50,\ 50,\ 0.1,\ 0.1, 10^5,\ 25]'$	

Table 5.5: Parameters used for the plant simulation and steady-state optimization in the styrene polymerization example.

variables as follows

$$c_I = \frac{c_I}{c_{Mf}},\ c_M = \frac{c_M}{c_{Mf}},\ c_S = \frac{c_S}{c_{Mf}},\ T = \frac{T - T_f}{T_f}$$
$$\lambda_0 = \frac{\lambda_0}{c_{Mf}},\ \lambda_1 = \frac{\lambda_1}{c_{Mf}},\ \lambda_2 = \frac{\lambda_2}{c_{Mf}},\ Q_I = \frac{Q_I}{Q_{os}}$$
$$Q_M = \frac{Q_M}{Q_{os}},\ Q_S = \frac{Q_S}{Q_{os}},\ \dot{Q}_c = \frac{\dot{Q}_c}{Q_{cs}},\ t = \frac{Q_{os}t}{V}$$

Chapter 6

Concluding remarks

In this thesis, we have developed new methods and approaches to use machine learning algorithms to design feedback controllers and dynamic process models for large multivariable processes. Machine learning algorithms offer convenient and powerful methods to harness the large-scale data collected in industries due to the growing digitization, and utilize the tremendous computing power available in this modern age. These algorithms can be leveraged to upgrade the existing automation and control methods implemented in industrial applications.

The keys aspects for the suitability of a controller design algorithm for an industrial deployment are that (i) it must be sufficiently robust to noise in the data, and (ii) it should be able to handle unmeasured disturbances and plant-model mismatch. In a typical model predictive control (MPC) implementation, the noise in the data is conveniently handled during the model identification step. And an integrating disturbance model is used to leverage feedback during the process operation and account for unmeasured disturbances and plant-model mismatch. The MPC uses the main dynamic and integrating disturbance models subsequently in an online optimization to manipulate the process actuators. The approach has been deployed in a range of industries in the past few decades. Recently, however, there has been a significant interest in the control systems research literature to use model-free reinforcement learning (RL)

to directly learn a controller from data. And avoid both the model identification and online optimization steps for the controller design. Although appealing, the RL algorithms had not been demonstrated to even handle noise in the training data collected for controller design.

In Chapter 2, we developed a new model-free approach to estimate unconstrained feedback controllers from noisy process data. We extended an available Q-learning algorithm that was developed for linear systems with Gaussian process noise of known covariance. We first treated the case of an unknown noise covariance, then proposed to use the extended algorithm for the case of only output measurements and both process and measurement noise. Simulation studies were presented with a heating, ventilation, and air-conditioning example to demonstrate the effectiveness of the proposed algorithm to give a reliable controller from a viable amount of training data. The developed approach is a step towards an industrially implementable model-free controller design algorithm.

The linear MPC feedback law is a complex, piecewise affine function defined over the state space of parameters in the quadratic program (QP). The number of regions in the feedback law grow *exponentially* with the model dimensions and the forecasting horizon length in the MPC problem. Due to this reason, online optimization has been the preferred method of choice for an MPC implementation in industrial applications. Since storing and traversing all the regions in the MPC feedback law becomes intractable for even moderate MPC problem sizes. It has been pointed out recently in the literature that neural networks with the rectified linear unit as the activation function also represent piecewise affine functions. And the complexity of representable functions can grow exponentially with the addition of more hidden nodes and layers in the network. With the availability of the powerful computational resources for data generation and network training, it can be possible to approximate the MPC feedback

law for large-scale problems.

In Chapter 3, we developed a NN controller design algorithm to approximate the MPC feedback law offline, so that NNs can be used online to execute MPC faster than QP solvers. We presented a structured NN architecture to achieve offset-free setpoint tracking during the online implementation. We also proposed to generate the training data for NN training by sampling the state space for only the relevant plant operational scenarios. The suitability of the NN controller design approach was demonstrated on large-scale application examples. The NN design approach presented in this chapter can be suitable to address large MPC applications that may be out of reach with QP solvers.

Both the real time optimization (RTO) and MPC layers in the process operations hierarchy rely heavily on dynamic process models. The development of new model identification approaches is beneficial to improve performances of these two layers in applications. The second half of this thesis focused on developing methods to estimate dynamic process models from data using machine learning. Despite the successes of machine learning algorithms in several domains, the use of fully black-box models for process modeling and optimization should be strongly cautioned. Hybrid process models can utilize the basic first principles knowledge usually available in applications, and combine that knowledge with machine learning models. The approach leverages the advantages of both the domain knowledge and machine learning based function approximators. Black-box NNs can be used in the approach to approximate some portions of the overall model that can be challenging to parameterize/model with the available process knowledge. The final model can be interpretable, data efficient, and suitable for process optimization.

In Chapter 4, we considered building systems affected by large unmeasured heat disturbances. We developed a novel two step model identification approach to esti-

mate hybrid dynamic models for such building systems. First principles based domain knowledge was used to develop a grey-box dynamic model in the first identification step. And NNs were used to model the heat disturbance patterns in the building system in the second identification step. The NN disturbance predictions can be used subsequently in an MPC problem for improved energy cost optimization. We presented simulation studies using a two time scale building system to demonstrate the economic advantages of using the NN disturbance forecasts in an MPC controller. The disturbance modeling and forecasting approach presented in this chapter may be suitable to improve performances of MPC implementations in both commercial or residential building applications.

In Chapter 5, we focused on developing hybrid models for nonlinear chemical engineering processes. We used NNs to approximate some complex, unknown functions such as reaction kinetics in the hybrid model. The parameters in the NNs were estimated using training data by solving a multistep ahead prediction error minimization problem. Multiple NNs to approximate different unknown functions in the hybrid model can be conveniently incorporated in the identification framework. We demonstrated the effectiveness of the estimated hybrid models to obtain an accurate optimum when they are used in a steady-state optimization problem. We emphasized that structural insights about the dynamic model such as the functional dependencies of the unknown functions being approximated are crucial to obtain good economic performance. The hybrid modeling approach presented in this chapter is a step towards an industrially implementable approach for dynamic model development and process optimization.

Overall, we have presented new approaches to develop feedback controllers and dynamic process models by leveraging the opportunities presented by machine learning algorithms. And the methods are developed with an eye towards their suitability for

potential deployment in industrial applications.

6.1 Directions for future research

The methods and results presented in this thesis on using machine learning for process modeling and control are promising. The results warrant a future investigation to make the methods more suitable for deployment in applications. We next outline the possible future work directions for the methods and research topics considered in each chapter of this thesis.

Model-free controller design

The model-free Q-learning approach proposed in Chapter 2 is suitable to estimate linear, unconstrained feedback controllers from noisy data. The model-based MPC approach, however, still has a few crucial advantages that make it a preferred choice for an industrial deployment. We discuss these advantages and future work directions for the model-free approaches below

- Industrial processes are often affected by large unmeasured disturbances, and a typical control objective is to maintain some primary measurements at their setpoints in the presence of those disturbances. In an MPC implementation, we can use integrating disturbance models to account for plant-model mismatch and unmeasured disturbances, and achieve zero tracking error in the primary measurements. Future work in the model-free controller design area should address the issue of achieving offset-free control performance.

- For noise filtering during the online implementation, a Kalman filter can be developed using the estimated dynamic model and noise covariances with a model-

based controller. The model-free approach loses this ability of using a Kalman filter for noise filtering during the closed-loop implementation. So future work may also be directed towards developing appropriate noise filtering methods for implementation with a model-free controller.

- Most industrial processes have constraints on the actuators, which can be conveniently accounted for in the online MPC optimization problem. Developing a model-free controller design approach for input (and state) constrained systems should also be a focus of future research.

Approximate model predictive control using neural networks

The offline NN controller design approach proposed in Chapter 3 is tractable for large applications, and it is suitable to improve the online MPC execution times by orders of magnitude compared to QP solvers. To make the approach more competitive to the online optimization based MPC approach, future work can be directed on the following topics

- The approximate MPC based NN controller does not have any closed-loop stability properties, whereas, the online optimization based MPC approach enjoys strong nominal stability and robustness properties. Future work may be directed to develop approaches that guarantee the stability of NN controllers by *design*.

- We only discussed the replacement of the MPC regulator QP by NNs in Chapter 3, and the target selector QP was still solved online in the case studies. Although the target selector QP is typically small, replacing this QP also can lead to a technology with no optimization involved during the online implementation. So, future work may be directed to developing approaches to replace the target selector QP using NNs.

- We considered applications that require solutions to a linear setpoint tracking MPC problem. Future work can investigate the use of NNs to approximate the feedback law for nonlinear, economic, and mixed-integer MPC formulations.

Dynamic modeling of building systems

In the area of developing dynamic models for building systems, future work can be focused on the following

- For the case studies in Chapter 4, we did not consider any occupancy measurements in the NN heat disturbance model identification step. Approaches can be developed to also include occupancy measurements (if available) to potentially improve the quality of the NN heat disturbance predictions.

- We considered a simple two time scale based building model to examine the efficacy of the proposed grey-box and NN disturbance model identification approaches. While the dynamics of some buildings can be described with the two time scale model, many large building systems can be more complex and require complex, higher order models to represent the dynamics. The two step grey-box dynamic and NN disturbance modeling approach may be extended to such higher order building models.

- The performance of the proposed two step model identification approach can be tested on real data collected from buildings affected by occupancy induced disturbances.

Hybrid nonlinear process modeling

The hybrid modeling framework proposed in Chapter 5 can be useful in applications where less first principles knowledge is available for dynamic modeling and process

optimization. The following issues should be addressed in future work

- We assumed that the plant is not affected by any large disturbances in the training data collected for model identification. Industrial processes, however, are almost always affected by unmeasured nonzero mean disturbances. Methods should be developed to treat such large disturbances in the training data in the hybrid model identification framework.

- The application of the estimated hybrid models was investigated for only steady-state optimization in Chapter 5. The performances of the hybrid models can also be examined when they are used in a dynamic MPC formulation.

- Hybrid models can also be used for the task for process monitoring in industrial applications. In the case studies, we observed that after training, the estimated hybrid models can accurately predict the measurements obtained from the non-linear plant. An avenue for using the hybrid models can be to predict some critical variables of interest during process operation for which no fast online sensor is available. For example, the polymer molecular weight (function of the modeled states) in the styrene polymerization example.

Bibliography

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

A. Afram and F. Janabi-Sharifi. Theory and applications of HVAC control systems—A review of model predictive control (MPC). *Build. Environ.*, 72:343–355, Feb 2014.

D. A. Allan, C. N. Bates, M. J. Risbeck, and J. B. Rawlings. On the inherent robustness of optimal and suboptimal nonlinear MPC. *Sys. Cont. Let.*, 106:68 – 78, 2017. ISSN 0167-6911. doi: 10.1016/j.sysconle.2017.03.005.

Z. Allen-Zhu, Y. Li, and Y. Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6158–6169, 2019.

J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi—a software framework for nonlinear optimization and optimal control. *Math. Prog. Comp.*, 11 (1):1–36, Mar 2019. doi: 10.1007/s12532-018-0139-4.

S. Arora, S. S. Du, W. Hu, Z. Li, and R. Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019.

B. F. Balvedi, E. Ghisi, and R. Lamberts. A review of occupant behaviour in residential buildings. *Energ. Buildings*, 174:495–505, 2018.

M. S. F. Bangi and J. S.-I. Kwon. Deep hybrid modeling of chemical process: Application to hydraulic fracturing. *Comput. Chem. Eng.*, 134:106696, 2020.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

A. Bemporad, A. Oliveri, T. Poggi, and M. Storace. Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations. *IEEE Trans. Auto. Cont.*, 56(12):2883–2897, 2011.

R. J. Bensingh, R. Machavaram, S. R. Boopathy, and C. Jebaraj. Injection molding process optimization of a bi-aspheric lens using hybrid artificial neural networks (anns) and particle swarm optimization (pso). *Measurement*, 134:359–374, 2019.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, Belmont, MA, 1995.

N. Bhutani, G. Rangaiah, and A. Ray. First-principles, data-based, and hybrid modeling and optimization of an industrial hydrocracking unit. *Ind. Eng. Chem. Res.*, 45(23): 7807–7816, 2006.

S. J. Bradtke, B. E. Ydstie, and A. G. Barto. Adaptive linear quadratic control using policy iteration. In *American Control Conference, 1994*, volume 3, pages 3475–3479. IEEE, 1994.

L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Rev. Control*, 46: 8–28, 2018.

J. C. Butcher. A history of runge-kutta methods. *Applied numerical mathematics*, 20(3): 247–260, 1996.

L. Cavagnari, L. Magni, and R. Scattolini. Neural network implementation of nonlinear receding-horizon control. *Neural computing & applications*, 8(1):86–92, 1999.

K. J. Chan, J. A. Paulson, and A. Mesbah. Deep learning-based approximate nonlinear model predictive control with offset-free tracking for embedded applications. In *Proc. Ame. Contro. Con.*, pages 3475–3481. IEEE, 2021.

S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari. Approximating explicit model predictive control using constrained neural networks. In *2018 Annual American Control Conference (ACC)*, pages 1520–1527. IEEE, 2018.

S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari. Large scale model predictive control with neural networks and primal active sets. *Automatica*, 135:109947, 2022.

Y. Chen and M. Ierapetritou. A framework of hybrid model development with identification of plant-model mismatch. *AIChE J.*, 66(10):e16996, 2020.

A. R. Coffman and P. Barooah. Simultaneous identification of dynamic model and occupant-induced disturbance for commercial buildings. *Build. Environ.*, 128:153–160, 2018.

J. P. Congalidis and J. R. Richards. Process control of polymerization reactors : An industrial perspective. *Polymer Reac. Eng.*, 6(2):71–111, 1998.

C. R. Cutler and R. T. Perry. Real time optimization with multivariable control is required to maximize profits. *Comput. Chem. Eng.*, 7:663–667, 1983.

M. L. Darby, M. Nikolaou, J. Jones, and D. Nicholson. RTO: An overview and assessment of current practice. *J. Proc. Cont.*, 21(6):874 – 884, 2011.

M. J. Davis. Contrast coding in multiple regression analysis: Strengths, weaknesses, and utility of popular coding structures. *Journal of data science*, 8(1):61–73, 2010.

O. Dogru, K. Velswamy, F. Ibrahim, Y. Wu, A. S. Sundaramoorthy, B. Huang, S. Xu, M. Nixon, and N. Bell. Reinforcement learning approach to autonomous pid tuning. *Comput. Chem. Eng.*, 161:107760, 2022.

F. J. Doyle III, C. A. Harrison, and T. J. Crowley. Hybrid model-based approach to batch-to-batch control of particle size distribution in emulsion polymerization. *Comput. Chem. Eng.*, 27(8-9):1153–1163, 2003.

J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen. Approximate model predictive building control via machine learning. *Applied Energy*, 218:199–216, 2018.

J. Drgoňa, J. Arroyo, I. C. Figueroa, D. Blum, K. Arendt, D. Kim, E. P. Ollé, J. Oravec, M. Wetter, D. L. Vrabie, et al. All you need to know about model predictive control for buildings. *Annual Rev. Control*, 50:190–232, 2020.

M. J. Ellis. Machine learning enhanced grey-box modeling for building thermal modeling. In *2021 American Control Conference (ACC)*, pages 3927–3932. IEEE, 2021.

D. o. Energy. 2018 renewable energy data book, 2018. URL: `https://www.energy.gov/eere/analysis/articles/2018-renewable-energy-data-book`.

M. Fazel, R. Ge, S. Kakade, and M. Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning*, pages 1467–1476. PMLR, 2018.

F. Ferracuti, A. Fonti, L. Ciabattoni, S. Pizzuti, A. Arteconi, L. Helsen, and G. Comodi. Data-driven models for short-term thermal behaviour prediction in real buildings. *Appl. Energ.*, 204:1375–1387, 2017.

A. R. Florita and G. P. Henze. Comparison of short-term weather forecasting models for model predictive control. *HVAC&R Res.*, 15(5):835–853, 2009.

D. Ghosh, E. Hermonat, P. Mhaskar, S. Snowling, and R. Goel. Hybrid modeling approach integrating first-principles models with subspace identification. *Ind. Eng. Chem. Res.*, 58(30):13533–13543, 2019.

E. G. Gilbert and K. T. Tan. Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Trans. Auto. Cont.*, 36(9):1008–1020, Sep 1991.

M. D. Graham and J. B. Rawlings. *Modeling and Analysis Principles for Chemical and Biological Engineers*. Nob Hill Publishing, Santa Barbara, CA, 2nd, paperback edition, 2022. 560 pages, ISBN 978-0-9759377-6-1.

Z. Guo, A. R. Coffman, J. Munk, P. Im, T. Kuruganti, and P. Barooah. Aggregation and data driven identification of building thermal dynamic model and unmeasured disturbance. *Energ. Buildings*, 231:110500, 2021.

S. Gupta, P.-H. Liu, S. A. Svoronos, R. Sharma, N. Abdel-Khalek, Y. Cheng, and H. El-Shall. Hybrid first-principles/neural networks model for column flotation. *AIChE J.*, 45(3):557–566, 1999.

B. Hambly, R. Xu, and H. Yang. Policy gradient methods for the noisy linear quadratic regulator over a finite horizon. *SIAM J. Cont. Opt.*, 59(5):3359–3391, 2021.

H. Harb, N. Boyanov, L. Hernandez, R. Streblow, and D. Müller. Development and validation of grey-box models for forecasting the thermal response of occupied buildings. *Energy and Buildings*, 117:199–207, 2016.

W. Heath and A. Wills. The inherent robustness of constrained linear model predictive control. *IFAC Proceedings Volumes*, 38(1):71–76, 2005.

G. P. Henze. Energy and cost minimal control of active and passive building thermal storage inventory. *J. Solar Ener. Eng.*, 127(3):343–351, 2005.

M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3):543–548, 2018.

P. M. Hidalgo and C. B. Brosilow. Nonlinear model predictive control of styrene polymerization at unstable operating points. *Comput. Chem. Eng.*, 14(4/5):481–494, 1990.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

B. L. Ho and R. E. Kalman. Efficient construction of linear state variable models from input/output functions. *Regelungstechnik*, 14:545–548, 1966.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

C. D. Hubbs, C. Li, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick. A deep reinforcement learning approach for chemical production scheduling. *Computers & Chemical Engineering*, 141:106982, 2020.

M. Jalanko, Y. Sanchez, V. Mahalec, and P. Mhaskar. Adaptive system identification of industrial ethylene splitter: A comparison of subspace identification and artificial neural networks. *Comput. Chem. Eng.*, 147:107240, 2021.

Z. Jiang, M. J. Risbeck, V. Ramamurti, S. Murugesan, J. Amores, C. Zhang, Y. M. Lee, and K. H. Drees. Building hvac control with reinforcement learning for reduction of energy cost and demand charge. *Energ. Buildings*, 239:110833, 2021.

Z.-P. Jiang and Y. Wang. Input-to-state stability for discrete-time nonlinear systems. *Automatica*, 37:857–869, 2001.

O. Kahrs and W. Marquardt. The validity domain of hybrid models and its application in process optimization. *Chem. Eng. Proc. PI*, 46(11):1054–1066, 2007.

B. Karg and S. Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, 2020.

B. Karg, T. Alamo, and S. Lucia. Probabilistic performance validation of deep learning-based robust nmpc controllers. *arXiv preprint arXiv:1910.13906*, 2019.

D. Kim, J. Cai, K. B. Ariyur, and J. E. Braun. System identification for building thermal systems under the presence of unmeasured disturbances in closed loop operation: Lumped disturbance modeling approach. *Build. Environ.*, 107:169–180, 2016.

D. Kim, J. Cai, J. E. Braun, and K. B. Ariyur. System identification for building thermal systems under the presence of unmeasured disturbances in closed loop operation: Theoretical analysis and application. *Energ. Buildings*, 167:359–369, 2018.

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

J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

D. Kouzoupis, G. Frison, A. Zanelli, and M. Diehl. Recent advances in quadratic programming algorithms for nonlinear model predictive control. *Vietnam Journal of Mathematics*, 46(4):863–882, 2018.

R. Kramer, J. Van Schijndel, and H. Schellen. Simplified thermal and hygric building models: A literature review. *frontarch*, 1(4):318–325, 2012.

K. Krauth, S. Tu, and B. Recht. Finite-time analysis of approximate policy iteration for the linear quadratic regulator. *Advances in Neural Information Processing Systems*, 32, 2019.

C. Kravaris, J. Hahn, and Y. Chu. Advances and selected recent developments in state and parameter estimation. *Comput. Chem. Eng.*, 51:111–123, 2013.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017.

P. Kumar and J. B. Rawlings. Hybrid process modeling approaches using neural networks and application to process optimization. Comput. Chem. Eng., 2023a. In preparation.

P. Kumar and J. B. Rawlings. Unconstrained feedback controller design using Q-learning from noisy process data. Comput. Chem. Eng., 2023b. Under review.

P. Kumar, J. B. Rawlings, and S. J. Wright. Industrial, large-scale model predictive control with structured neural networks. *Comput. Chem. Eng.*, 150:107291, 2021. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2021.107291.

P. Kumar, J. B. Rawlings, M. J. Wenzel, and M. J. Risbeck. Grey-box model and neural network disturbance predictor identification for economic MPC in building energy systems. Energ. Buildings, 2022. Under review.

S. J. Kuntz and J. B. Rawlings. Maximum likelihood estimation of linear disturbance models for offset-free model predictive control. In *American Control Conference*, pages 3961–3966, Atlanta, GA, June 8–10, 2022.

M. Kvasnica, J. Löfberg, and M. Fikar. Stabilizing polynomial approximation of explicit mpc. *Automatica*, 47(10):2292–2297, 2011.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *J. Mach. lear. Res.*, 4: 1107–1149, 2003.

W. E. Larimore. Canonical variate analysis in identification, filtering, and adaptive control. In *Proceedings of the 29th Conference on Decision and Control*, pages 596–604, 1990.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

J. H. Lee, J. Shin, and M. J. Realff. Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, 114:111–121, 2018.

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Q. Appl. Math*, 2(2):164–168, 1944.

S. Levine and V. Koltun. Learning complex neural network policies with trajectory optimization. In *Int. Conf. Mach. Learn.*, pages 829–837. PMLR, 2014.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. lear. Res.*, 17(1):1334–1373, 2016.

F. L. Lewis and K. G. Vamvoudakis. Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):14–25, 2011.

X. Li and J. Wen. Review of building energy modeling for control and operation. *Ren. Sust. Energ. Rev.*, 37:517–537, 2014.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

D. Limon, T. Alamo, F. Salas, and E. F. Camacho. On the stability of MPC without terminal constraint. *IEEE Trans. Auto. Cont.*, 51(5):832–836, May 2006.

L. Ljung. *System Identification: Theory for the User*. Prentice Hall, New Jersey, second edition, 1999.

R. J. Lovelett, J. L. Avalos, and I. G. Kevrekidis. Partial observations and conservation laws: Gray-box modeling in biotechnology and optogenetics. *Industrial & Engineering Chemistry Research*, 59(6):2611–2620, 2019.

R. J. Lovelett, F. Dietrich, S. Lee, and I. G. Kevrekidis. Some manifold learning considerations toward explicit model predictive control. *AIChE Journal*, 66(5):e16881, 2020.

N. Luo, W. Du, Z. Ye, and F. Qian. Development of a hybrid model for industrial ethylene oxide reactor. *Ind. Eng. Chem. Res.*, 51(19):6926–6932, 2012.

B. Lusch, J. N. Kutz, and S. L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nat. Commun.*, 9(1):1–10, 2018.

J. Ma, J. Qin, T. Salsbury, and P. Xu. Demand reduction in building energy systems based on economic model predictive control. *Chem. Eng. Sci.*, 67(1):92 – 100, 2012a.

Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves. Model predictive control for the operation of building cooling systems. *IEEE Ctl. Sys. Tech.*, 20(3): 796–803, 2012b.

J. C. MacMurray and D. Himmelblau. Modeling and control of a packed distillation column using artificial neural networks. *Comput. Chem. Eng.*, 19(10):1077–1088, 1995.

H. Madsen and J. Holst. Estimation of continuous-time models for the heat dynamics of a building. *Energ. Buildings*, 22(1):67–79, 1995.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11(2):431–441, 1963.

D. Q. Mayne and S. V. Raković. Optimal control of constrained piecewise affine discrete-time systems. *Comp. Optim. Appl.*, 25(1-3):167–191, 2003. ISSN 0926-6003. doi: {10.1023/A:1022905121198}.

D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

D. I. Mendoza-Serrano and D. J. Chmielewski. HVAC control using infinite-horizon economic MPC. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 6963–6968, 2012.

T. M. Mitchell et al. *Machine learning. WCB*. McGraw-Hill Boston, MA, 1997.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, Red Hook, NY, 2014. Curran Associates, Inc.

M. Morari and U. Maeder. Nonlinear offset-free model predictive control. *Automatica*, 48(9):2059–2067, 2012.

J. E. Morinelly and B. E. Ydstie. Dual MPC with reinforcement learning. *IFAC–P. Online*, 49(7):266–271, 2016.

A. Narasingam and J. S.-I. Kwon. Koopman lyapunov-based model predictive control of nonlinear chemical process systems. *AIChE J.*, 65(11):e16743, 2019.

NCEI. National centers for environmental information, 2019. URL: `https://www.ncei.noaa.gov/`.

R. Nian, J. Liu, and B. Huang. A review on reinforcement learning: Introduction and applications in industrial process control. *Comput. Chem. Eng.*, 139:106886, 2020.

B. J. Odelson, M. R. Rajamani, and J. B. Rawlings. A new autocovariance least-squares method for estimating noise covariances. Technical Report 2003-04, TWMCC, Department of Chemical Engineering, University of Wisconsin–Madison, September 2003.

B. J. Odelson, M. R. Rajamani, and J. B. Rawlings. A new autocovariance least-squares method for estimating noise covariances. *Automatica*, 42(2):303–308, Feb 2006.

F. Oldewurtel, A. Parisio, C. N. Jones, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and M. Morari. Use of model predictive control and weather forecasts for energy efficient building climate control. *Energ. Buildings*, 45:15–27, 2012.

F. Oldewurtel, D. Sturzenegger, and M. Morari. Importance of occupancy information for building climate control. *Appl. Energ.*, 101:521–532, 2013.

Z. Pang, F. Niu, and Z. O'Neill. Solar radiation prediction using recurrent neural network and artificial neural network: A case study with comparisons. *Renewable Energy*, 156:279–289, 2020.

G. Pannocchia and J. B. Rawlings. Disturbance models for offset-free MPC control. *AIChE J.*, 49(2):426–437, 2003.

G. Pannocchia, J. B. Rawlings, and S. J. Wright. Fast, large-scale model predictive control by partial enumeration. *Automatica*, 43:852–860, 2007.

G. Pannocchia, J. B. Rawlings, and S. J. Wright. Conditions under which suboptimal nonlinear MPC is inherently robust. *Sys. Cont. Let.*, 60:747–755, 2011.

C. C. Pantelides and J. G. Renfro. The online use of first-principles models in process operations: Review, current status and future needs. *Comput. Chem. Eng.*, 51:136–148, 2013.

S. Papantoniou and D.-D. Kolokotsa. Prediction of outdoor air temperature using neural networks: Application in 4 european cities. *Energ. Buildings*, 114:72–79, 2016.

N. R. Patel, M. J. Risbeck, J. B. Rawlings, M. J. Wenzel, and R. D. Turney. Distributed economic model predictive control for large-scale building temperature regulation. In *American Control Conference*, pages 895–900, Boston, MA, July 6–8, 2016.

N. R. Patel, M. J. Risbeck, J. B. Rawlings, C. T. Maravelias, M. J. Wenzel, and R. D. Turney. A case study of economic optimization of HVAC systems based on the Stanford university campus airside and waterside systems. 5th International High Performance Buildings Conference at Purdue University, West Lafayette, IN, July 2018.

J. A. Paulson and A. Mesbah. Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction. *IEEE Control Systems Letters*, 4 (3):719–724, 2020.

K. M. Powell, D. Machalek, and T. Quah. Real-time optimization using reinforcement learning. *Comput. Chem. Eng.*, 143:107077, 2020.

V. Prasad, M. Schley, L. P. Russo, and B. W. Bequette. Product property and production rate control of styrene polymerization. *J. Proc. Cont.*, 12(3):353–372, 2002.

S. Privara, J. Cigler, Z. Váňa, F. Oldewurtel, C. Sagerschnig, and E. Žáčeková. Building modeling as a crucial part for building predictive control. *Energ. Buildings*, 56:8–22, 2013.

D. C. Psichogios and L. H. Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE J.*, 38(10):1499–1511, 1992.

H. Qi, X.-G. Zhou, L.-H. Liu, and W.-K. Yuan. A hybrid neural network-first principles model for fixed-bed reactor. *Chem. Eng. Sci.*, 54(13-14):2521–2526, 1999.

S. J. Qin. An overview of subspace identification. *Comput. Chem. Eng.*, 30:1502–1513, 2006.

S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Eng. Pract.*, 11(7):733–764, 2003.

N. S. Raman, A. M. Devraj, P. Barooah, and S. P. Meyn. Reinforcement learning for control of building hvac systems. In *Proc. Ame. Contro. Con.*, pages 2326–2332. IEEE, 2020.

J. B. Rawlings and J. G. Ekerdt. *Chemical Reactor Analysis and Design Fundamentals*. Nob Hill Publishing, Santa Barbara, CA, 2nd, paperback edition, 2020. 664 pages, ISBN 978-0-9759377-4-7.

J. B. Rawlings and L. Ji. Optimization-based state estimation: Current status and some new results. *J. Proc. Cont.*, 22:1439–1444, 2012.

J. B. Rawlings and C. T. Maravelias. Bringing new technologies and approaches to the operation and control of chemical process systems. *AIChE J.*, 65(6), 2019. doi: 10.1002/aic.16615.

J. B. Rawlings, D. Angeli, and C. Bates. Fundamentals of economic model predictive control. In *IEEE Conference on Decision and Control (CDC)*, pages 3851–3861, Maui, HI, December 2012.

J. B. Rawlings, N. R. Patel, M. J. Risbeck, C. T. Maravelias, M. J. Wenzel, and R. D. Turney. Economic MPC and real-time decision making with application to large-scale HVAC energy systems. *Comput. Chem. Eng.*, 114:89–98, 2018.

J. B. Rawlings, D. Q. Mayne, and M. M. Diehl. *Model Predictive Control: Theory, Design, and Computation*. Nob Hill Publishing, Santa Barbara, CA, 2nd, paperback edition, 2020. 770 pages, ISBN 978-0-9759377-5-4.

W. H. Ray. On the mathematical modeling of polymerization reactors. *JMS - Rev. Macromol. Chem.*, pages 1–56, 1972.

W. H. Ray. Polymerization reactor control. In *Proc. Ame. Contro. Con.*, pages 842–848. IEEE, 1985.

M. J. Ren and J. A. Wright. Adaptive diurnal prediction of ambient dry-bulb temperature and solar radiation. *HVAC&R Res.*, 8(4):383–401, 2002.

R. Rico-Martinez, J. Anderson, and I. Kevrekidis. Continuous-time nonlinear signal processing: a neural network based approach for gray box identification. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pages 596–605. IEEE, 1994.

S. A. A. Rizvi and Z. Lin. Output feedback reinforcement Q-learning control for the discrete-time linear quadratic regulator problem. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 1311–1316. IEEE, 2017.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

F. C. Sangogboye, K. Arendt, A. Singh, C. T. Veje, M. B. Kjærgaard, and B. N. Jørgensen. Performance comparison of occupancy count estimation and prediction with common versus dedicated sensors for building model predictive control. In *Build. Sim.*, volume 10, pages 829–843. Springer, 2017.

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. *Comput. Chem. Eng.*, 151: 107365, 2021.

O. Santander, V. Kuppuraj, C. A. Harrison, and M. Baldea. Integrated deep learning-production planning-economic model predictive control framework for large-scale processes. a fluid catalytic cracker-fractionator case study. *Comput. Chem. Eng.*, 167: 107977, 2022.

J. Schubert, R. Simutis, M. Dors, I. Havlik, and A. Lübbert. Bioprocess optimization and control: Application of hybrid modelling. *Journal of biotechnology*, 35(1):51–68, 1994.

J. C. Schulze, D. T. Doncevic, and A. Mitsos. Identification of mimo wiener-type koopman models for data-driven model reduction using deep learning. *Comput. Chem. Eng.*, 161:107781, 2022.

A. M. Schweidtmann and A. Mitsos. Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory and Applications*, 180 (3):925–948, 2019.

D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. Doyle. *Process Dynamics and Control*. John Wiley and Sons, New York, fourth edition, 2017.

G. Serale, M. Fiorentini, A. Capozzoli, D. Bernardini, and A. Bemporad. Model predictive control (mpc) for enhancing building and hvac system energy efficiency: Problem formulation, applications and opportunities. *Energies*, 11(3):631, 2018.

M. M. Seron, G. C. Goodwin, and J. A. De Doná. Characterisation of receding horizon control for constrained linear systems. *Asian Journal of Control*, 5(2):271–286, Jun 2003.

H. Shahnazari, P. Mhaskar, J. M. House, and T. I. Salsbury. Modeling and fault diagnosis design for HVAC systems using recurrent neural networks. *Comput. Chem. Eng.*, 126: 189–203, 2019.

J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee. Reinforcement learning–overview of recent progress and implications for process control. *Comput. Chem. Eng.*, 127: 282–294, 2019.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724, 1995.

E. D. Sontag and Y. Wang. On the characterization of the input to state stability property. *Sys. Cont. Let.*, 24:351–359, 1995.

S. Spielberg, A. Tulsyan, N. P. Lawrence, P. D. Loewen, and R. Bhushan Gopaluni. Toward self-driving processes: A deep reinforcement learning approach to control. *AIChE J.*, 65(10):e16689, 2019.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

I. Steinwart and A. Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

H.-T. Su, N. Bhat, P. Minderman, and T. McAvoy. Integrating neural networks with first principles models for dynamic modeling. In *Dynamics and Control of Chemical Reactors, Distillation Columns and Batch Processes*, pages 327–332. Elsevier, 1993.

A. Sundaram, P. Ghosh, J. M. Caruthers, and V. Venkatasubramanian. Design of fuel additives using neural networks and evolutionary algorithms. *AIChE Journal*, 47(6): 1387–1406, 2001.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

C. A. Thilker, H. Madsen, and J. B. Jørgensen. Advanced forecasting and disturbance modelling for model predictive control of smart energy systems. *Appl. Energ.*, 292: 116889, 2021.

M. L. Thompson and M. A. Kramer. Modeling chemical processes using prior knowledge and neural networks. *AIChE J.*, 40(8):1328–1340, 1994.

A. Y.-D. Tsen, S. S. Jang, D. S. H. Wong, and B. Joseph. Predictive control of quality in batch polymerization using hybrid ann models. *AIChE J.*, 42(2):455–465, 1996.

S. Tu and B. Recht. Least-squares temporal difference learning for the linear quadratic regulator. In *International Conference on Machine Learning*, pages 5005–5014. PMLR, 2018.

L. Vandenberghe. The cvxopt linear and quadratic cone program solvers. *Online: http://cvxopt. org/documentation/coneprog. pdf*, 2010.

A. N. Venkat. *Distributed Model Predictive Control: Theory and Applications*. PhD thesis, University of Wisconsin–Madison, October 2006. URL `https://sites.engineering.ucsb.edu/~jbraw/jbrweb-archives/theses/venkat.pdf`.

V. Venkatasubramanian. The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE J.*, 65(2):466–478, 2019.

M. Von Stosch, R. Oliveira, J. Peres, and S. F. de Azevedo. Hybrid semi-parametric modeling in process systems engineering: Past, present and future. *Comput. Chem. Eng.*, 60:86–101, 2014.

Y. Wang, K. Velswamy, and B. Huang. A novel approach to feedback control with deep reinforcement learning. *IFAC–P. Online*, 51(18):31–36, 2018.

Z. Wang and Y. Chen. Data-driven modeling of building thermal dynamics: Methodology and state of the art. *Energ. Buildings*, 203:109405, 2019.

C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

C. Wen, X. Ma, and B. E. Ydstie. Analytical expression of explicit mpc solution via lattice piecewise-affine function. *Automatica*, 45(4):910–917, 2009.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

S. J. Wright. Efficient convex optimization for linear mpc. In *Handbook of Model Predictive Control*, pages 287–303. Springer, 2019.

Z. Wu, D. Rincon, J. Luo, and P. D. Christofides. Machine learning modeling and predictive control of nonlinear processes using noisy data. *AIChE J.*, 67(4):e17164, 2021.

F. A. Yaghmaie, F. Gustafsson, and L. Ljung. Linear quadratic control using model-free reinforcement learning. *IEEE Trans. Auto. Cont.*, 2022.

D. Yan, W. O'Brien, T. Hong, X. Feng, H. B. Gunay, F. Tahmasebi, and A. Mahdavi. Occupant behavior modeling for building performance simulation: Current state and future challenges. *Energ. Buildings*, 107:264–278, 2015.

S. Yang, P. Navarathna, S. Ghosh, and B. W. Bequette. Hybrid modeling in the era of smart manufacturing. *Comput. Chem. Eng.*, 140:106874, 2020.

H. Yoo, B. Kim, J. W. Kim, and J. H. Lee. Reinforcement learning based optimal control of batch processes using monte-carlo deep deterministic policy gradient with phase segmentation. *Comput. Chem. Eng.*, 144:107133, 2021.

M. Zanon and S. Gros. Safe reinforcement learning using robust mpc. *IEEE Trans. Auto. Cont.*, 66(8):3638–3652, 2020.

V. M. Zavala, C. D. Laird, and L. T. Biegler. A fast moving horizon estimation algorithm based on nonlinear programming sensitivity. *J. Proc. Cont.*, 18:876–884, 2008.

S. Zendehboudi, N. Rezaei, and A. Lohi. Applications of hybrid models in chemical, petroleum, and energy systems: A systematic review. *Appl. Energ.*, 228:2539–2566, 2018.

T. Zeng, J. Brooks, and P. Barooah. Simultaneous identification of linear building dynamic model and disturbance using sparsity-promoting optimization. *Automatica*, 129:109631, 2021.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Z. Zhang, Z. Wu, D. Rincon, and P. D. Christofides. Real-time optimization and control of nonlinear processes using machine learning. *Mathematics*, 7(10):890, 2019.