Hybrid and machine learning based fault detection and control

Hybrid and machine learning based modeling, fault detection, and control of complex processes

by

Hesam Hassanpour

A Thesis Submitted to the School of Graduate Studies in the Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy

Doctor of Philosophy (2022)	McMaster University
(Chemical Engineering)	Hamilton, Ontario, Canada

TITLE:	Hybrid and machine learning based modeling, fault detection, and control of complex processes
AUTHOR:	Hesam Hassanpour
	(McMaster University, Hamilton, ON)
SUPERVISOR:	Dr. Prashant Mhaskar
NUMBER OF PAGES:	xxi, **??**

ABSTRACT

The operation of industrial processes to achieve economic and safety objectives significantly depends on the use of advanced fault diagnosis and process control strategies. To design model-based fault diagnosis and control frameworks, developing sufficiently accurate models is an essential task. With the recent advance in computational power and data storage technologies, there have been growing interests in the use of machine learning techniques for process modeling and control. The performance of these techniques, however, depends on the use of a sufficient amount of high-quality data. The presence of uninformative and redundant data can hinder the model's ability to accurately predict dynamic behaviors. In this situation, the over-fitting also remains a challenging problem. As a result, it is necessary to use appropriate pre-processing tools, data mining techniques, and first-principles knowledge (if applicable) to achieve a reliable model. Motivated by the above considerations, this thesis focuses on the problem of hybrid and machine learning based modeling, fault detection, and control when collected data are not sufficient or there exist correlations between data samples. The first part of this thesis addresses the problem of system identification for heating, ventilation, and air conditioning (HVAC) systems when an insufficient amount of data is available. To this end, a hybrid machine learning based approach is developed where a pre-trained recurrent neural network (RNN) model (trained on a large amount of data from a representative zone) is leveraged to build a model for the zone in question. In the next phase, first-principles knowledge is integrated with data to develop a fault detection mechanism for HVAC systems using principal component analysis (PCA). The superior performance of the proposed approach, over the individual first-principles and data-driven based methods is shown. Finally, the problem of handling correlated data for the RNN-based model predictive control (MPC) implementations is addressed. To this end, PCA and autoencoder (AE)-based strategies are used to recognize the correlations that exist between data samples in both the input and output spaces. The constrained RNN-based MPC is then formulated by

adding a PCA-based squared prediction error (SPE) constraint. PCA and AE-based optimization problems are also defined to calculate the achievable set-points. The efficacy of the proposed approaches over the nominal (standard) RNN-based MPC is demonstrated using different set-point tracking scenarios for a chemical reactor example.

ACKNOWLEDGEMENTS

First and foremost, I am deeply grateful to my supervisor, Dr. Prashant Mhaskar, for his excellent guidance, ceaseless encouragement, support, and patience during the entire course of my research. His in-depth knowledge, invaluable experience, and innovative ideas have encouraged me in all the time of my academic research. Dr. Mhaskar has always been caring and supportive of his students, and I have been extremely fortunate in pursuing my graduate studies under his supervision. I would also like to thank the members of my committee, Dr. Brandon Corbett and Dr. Shahin Sirouspour for their insightful advice, invaluable questions, and suggestions that enhanced the quality of this research.

I would like to thank my friends in McMaster Advanced Control Consortium (MACC) for their support and invaluable discussions during my studies. It has been a great experience working in such a brilliant group. Also, I would also like to thank my friends outside school for supporting and encouraging during difficult moments.

Last but not least, I would like to thank my parents Hamid and Farzaneh. I am grateful beyond measure for their unconditional love, endless supports, encouragement, and sacrifices. Without them, I would not be where I am today. I would also like to thank my sister, Sahar, for always being supportive and encouraging.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and motivation

The importance of automation in the industries has significantly increased in recent years. Automated control systems are designed to improve process efficiency and profitability while maintaining safety. The increase in the level of automation, however, necessitates the use of techniques that not only allow for achieving the optimal control objectives, but also handling the abnormalities. This has motivated the research community to develop advanced fault diagnosis and process control (such as model predictive control (MPC)) techniques. Successful implementations of many of the proposed approaches require a process model with sufficient predictive ability. In this regard, different techniques including first-principles and data-driven (and machine learning) modeling are used. While first-principles models have good extrapolation capabilities, they are generally difficult to develop and maintain due to uncertainty in the physical parameters and unmeasured physical states.

With the recent advance in computing power and data storage technology, there have been growing interests in utilizing data-driven and machine learning based methods for dynamic modeling. These models, however, require a sufficient amount of high-quality data (Note that in the area of dynamic modeling, high-quality data refers to data collected under normal operating conditions that represent the true dynamic relationships between input and output variables without redundancies and correlations). Several linear statistical-based modeling approaches are developed such as autoregressive moving average exogenous (ARMAX) models (Mustafaraj et al. (2010); Shardt and Huang (2011)), principal component analysis (PCA) (Misra et al. (2002); Yoon and MacGregor (2004)), projection to latent structures (PLS) (MacGregor et al. (1994); MacGregor and Cinar (2012)), and subspace identification (SubID) (Moonen et al. (1989); Qin and Ljung (2003); Huang et al. (2005)). The simplicity (linearity) and predictive ability of these data-driven models make them attractive for a variety

of fault diagnosis (Chen and Lan (2010); Ferkl and Široký (2010); Li and Wen (2014); Shahnazari et al. (2018)) and MPC applications (Hale and Qin (2002); Golshan et al. (2010); Kheradmandi and Mhaskar (2018)). While the linear data-driven models have shown successful performance when used for fault diagnosis and MPC implementations, they cannot be used in general due to the complex and nonlinear nature of the chemical processes.

Among a number of machine learning techniques, artificial neural networks (ANN) have received significant attention due to their ability to capture nonlinearities. Different neural network architectures are developed, such as feedforward neural networks (FNN), recurrent neural network (RNN), convolutional neural networks (CNN), long short-term memory (LSTM) neural networks, for being employed in different fault diagnosis (Wu and Zhao (2018); Zhao et al. (2018); Shahnazari et al. (2019); Ge et al. (2021); Gravanis et al. (2022)) and MPC algorithms (Nikravesh et al. (2000); Kittisupakorn et al. (2009); Ferreira et al. (2012); Sadeghassadi et al. (2018); Wong et al. (2018); Alhajeri et al. (2022)). To develop ANN-based models, a significant amount of data is required to avoid over-fitting due to the presence of a large number of parameters. The negative impact of the over-fitting issue can get exacerbated when the training data are not informative, or they contain redundant information (this can usually happen when data are collected from a system under closed-loop conditions). In addition, although the data availability has generally increased, this is not necessarily the case for many real-world processes because of a general lack of instrumentation and data cleansing problems (unavailability of clean data). Therefore, techniques that not only have advantage of ANN to capture nonlinearities, but also can deal with insufficient and correlated data are sought-after.

Hybrid modeling approaches, those that integrate first-principles knowledge with data (hybrid first-principles data-driven models) or those that combine different data-driven techniques (hybrid data-driven models) to enhance the performance of an

individual approach, have been gaining attention in recent years , and could bring potential value to the problem of fault diagnosis and control designs. The hybrid first-principles data-driven models can balance the advantages and disadvantages of individual first-principles and data-driven based modeling techniques with the aim of improving prediction accuracy, extrapolation capabilities, and interpretability. The improved performance of these models (over purely first-principles and data-driven models) is shown in a number of fault diagnosis and control implementations (Psichogios and Ungar (1992); Anderson et al. (2000); Liang and Du (2007); Hosen et al. (2011); Ghosh et al. (2019); Bangi and Kwon (2020); Ghosh et al. (2021)). On the other hand, the hybrid data-driven models are developed to enhance the performance of individual techniques when data samples are not sufficiently informative (by taking advantages of individual approaches to extract useful features, recognize correlations, and capture nonlinearities). The applicability of these techniques is also evaluated in several fault diagnosis and control schemes (Chen and Liao (2002); Li and Wen (2014); Zhang et al. (2013); Zhou et al. (2014); Hassanpour et al. (2020)).

Transfer learning-based approaches have shown another promising potential to overcome challenges associated with limited data availability (see e.g., Pan and Yang (2009)). The main objective of these approaches is to transfer the knowledge from source tasks to a target task, where insufficient and uninformative data are available. Transfer learning is commonly used for training a deep neural network (DNN)-based model, where a large number of parameters need to be identified. The first step is to pre-train a model on a large dataset available for a related task. The pre-trained DNN model is then used (as a starting point) for developing a model for the target task. All or some parts of the pre-trained model are then fine-tuned using limited data available for the new task. Transfer learning offers several advantages in the context of DNN training, such as reducing training time, handling over-fitting, and improving model performance in the absence of large amounts of data. Examples of transfer learning algorithms can be found in a variety of applications in process

modeling, fault diagnosis, and control (Grubinger et al. (2017); Wang et al. (2020); Wu and Zhao (2020); Gao et al. (2021); Deng and Chen (2021); Lu et al. (2021); Li and Rangarajan (2022)).

While there exist several approaches to address the problem of modeling, fault diagnosis, and MPC designs in the presence of insufficient and uninformative data, there is still some gap in the literature to find out how to integrate different techniques to achieve better performance and avoid undesirable effects (such as over-fitting) that may occur during the training process. This has motivated us to propose some novel methods to address these issues.

1.2 Thesis objectives and outline

As mentioned earlier, general access to sufficient and informative data may not be possible in many practical situations due to limitations of measuring instruments and unavailability of clean data. Therefore, the main focus of the thesis is to address the problem of handling insufficient and uninformative/correlated data for dynamic modeling, fault detection, and control by utilizing/integrating appropriate machine learning-based tools and first-principles knowledge. The thesis is organized as follows:

In Chapter 2, the problem of system identification in the presence of a relatively small amount of data, for the unit in consideration, is addressed. A hybrid machine learning-based approach is proposed where a pre-trained RNN model of a representative unit (developed using a large amount of data) is leveraged to build models for the units with small amounts of data. This is done by integrating the pre-trained RNN model with the models built using the SubID technique to predict the residuals, defined as the differences between the actual outputs and the predictions obtained using the pre-trained model for the other zones. The efficacy of the proposed approach is

shown using real data collected from a multi-zone building heating, ventilation, and air conditioning (HVAC) systems.

In Chapter 3, a hybrid PCA-based approach is proposed by integrating first-principles knowledge with data to detect faults in HVAC systems. To this end, the temperature measurements, together with the residuals (first-principles knowledge), defined as the discrepancies between expected and observed behaviors, are used to develop multiple hybrid PCA models for normal operating conditions. Hotelling's T^2 and square prediction error (SPE) statistics corresponding to the new observations are then calculated to monitor the process behavior and detect deviations from the expected behavior. The efficacy of the proposed approach is compared with the individual first-principles-based and purely data-driven PCA (developed using raw temperature measurements) techniques.

In Chapter 4, the problem of handling situations where the training input data are not sufficiently rich (or they are correlated) for MPC implementations is addressed. Two approaches are proposed to address this issue. In the first approach, PCA on the input data is performed to recognize the existing correlation. The scores are then used to build a model using RNN (PCA-RNN model). This model is used within the MPC framework to compute the optimal scores (and subsequently manipulated inputs). In the second approach, an alternative solution is proposed by adding a PCA-based SPE constraint into the RNN-based MPC that uses a purely RNN model. This is done in order to make prescribed inputs follow the same correlation as that of the training input data. Finally, an approach is presented that allows to break the correlation in the MPC implementation while maintaining model validity (in order to improve possible economic performance of the MPC controller that can be achieved by using the full space of movements of the manipulated inputs). The efficacy of the proposed approaches is shown using a chemical reactor example.

In Chapter 5, the problem of implementing achievable set-points for the RNN-based

MPC, where collected output data contain correlated information (due to control structures or process itself), is addressed (to ensure that achievable set-points are prescribed to the MPC). The key idea is to perform PCA on the output data to recognize the existing correlation. The upper confidence limit of SPE is then determined to define a constraint in an optimization problem to compute the achievable set-points. The efficacy of the proposed approach is illustrated via implementations on a chemical reactor example.

Chapter 6 integrates the approaches, proposed in Chapter 4 and Chapter 5, to address the problem of handling correlated (input and output) data for implementing the RNN-based MPC. In this situation, if the correlation is not properly addressed, independently calculated manipulated inputs, by the controller, and arbitrarily prescribed set-points may require predictions in regions where the model is not trained. To address these problems, a PCA-based SPE constraint, developed using training input data, is added to the standard RNN-based MPC (constrained RNN-based MPC). Next, an optimization problem subject to the PCA-based SPE constraint, developed using training output data, is formulated to compute the achievable set-points. In addition, due to the superior ability of autoencoder (AE) to capture nonlinear correlations, an AE-based approach is proposed to compute the achievable set-points by replacing the PCA-based SPE constraint with the AE-based SPE constraint in the optimization problem to calculate the achievable set-points. The efficacy of the proposed approach is compared with the standard RNN-based MPC using different set-point tracking scenarios.

Finally, the main contributions of the thesis, together with the key findings and recommendations for future work, are presented in Chapter 7.

Bibliography

Alhajeri, M. S., Luo, J., Wu, Z., Albalawi, F., and Christofides, P. D. (2022). Process structure-based recurrent neural network modeling for predictive control: A comparative study. *Chemical Engineering Research and Design*, 179:77–89.

Anderson, J. S., McAvoy, T. J., and Hao, O. J. (2000). Use of hybrid models in wastewater systems. *Industrial & Engineering Chemistry Research*, 39(6):1694–1704.

Bangi, M. S. F. and Kwon, J. S. I. (2020). Deep hybrid modeling of chemical process: Application to hydraulic fracturing. *Computers & Chemical Engineering*, 134:106696.

Chen, J. and Liao, C. M. (2002). Dynamic process fault monitoring based on neural network and pca. *Journal of Process control*, 12(2):277–289.

Chen, Y. and Lan, L. (2010). Fault detection, diagnosis and data recovery for a real building heating/cooling billing system. *Energy Conversion and Management*, 51(5):1015–1024.

Deng, Z. and Chen, Q. (2021). Reinforcement learning of occupant behavior model for cross-building transfer learning to various HVAC control systems. *Energ. Buildings*, 254:110860.

Ferkl, L. and Široký, J. (2010). Ceiling radiant cooling: Comparison of ARMAX and subspace identification modelling methods. *Building and Environment*, 45(1):205–212.

Ferreira, P. M., Ruano, A. E., Silva, S., and Conceicao, E. Z. E. (2012). Neural networks based predictive control for thermal comfort and energy savings in public buildings. *Energy and Buildings*, 55(6):238–251.

Gao, N., Shao, W., Rahaman, M. S., Zhai, J., David, K., and Salim, F. D. (2021). Transfer learning for thermal comfort prediction in multiple cities. *Build. Environ.*, 165:107725.

Ge, X., Wang, B., Yang, X., Pan, Y., Liu, B., and Liu, B. (2021). Fault detection and diagnosis for reactive distillation based on convolutional neural network. *Computers & Chemical Engineering*, 145:107172.

Ghosh, D., Hermonat, E., Mhaskar, P., Snowling, S., and Goel, R. (2019). Hybrid modeling approach integrating first-principles models with subspace identification. *Industrial & Engineering Chemistry Research*, 58(30):13533–13543.

Ghosh, D., Moreira, J., and Mhaskar, P. (2021). Model predictive control embedding a parallel hybrid modeling strategy. *Ind. Eng. Chem. Res.*, 60(6):2547–2562.

Golshan, M., MacGregor, J. F., Bruwer, M. J., and Mhaskar, P. (2010). Latent variable model predictive control (LV-MPC) for trajectory tracking in batch processes. *Journal of process control*, 20(4):538–550.

Gravanis, G., Dragogias, I., Papakiriakos, K., Ziogou, C., and Diamantaras, K. (2022). Fault detection and diagnosis for non-linear processes empowered by dynamic neural networks. *Computers & Chemical Engineering*, 156:107531.

Grubinger, T., Chasparis, G. C., and Natschläger, T. (2017). Generalized online transfer learning for climate control in residential buildings. *Energ. Buildings*, 139:63–71.

Hale, E. T. and Qin, S. J. (2002). Subspace model predictive control and a case study. *In Proceedings of the American Control Conference*, 6:4758–4763.

Hassanpour, H., Corbett, B., and Mhaskar, P. (2020). Integrating dynamic neural network models with principal component analysis for adaptive model predictive control. *Chemical Engineering Research and Design*, 161:26–37.

Hosen, M. A., Hussain, M. A., and Mjalli, F. S. (2011). Control of polystyrene batch reactors using neural network based model predictive control (NNMPC): An experimental investigation. *Control Engineering Practice*, 19(5):454–467.

Huang, B., Ding, S. X., and Qin, S. J. (2005). Closed-loop subspace identification: an orthogonal projection approach. *Journal of Process Control*, 15(1):53–66.

Kheradmandi, M. and Mhaskar, P. (2018). Model predictive control with closed-loop re-identification. *Computers & Chemical Engineering*, 109:249–260.

Kittisupakorn, P., Thitiyasook, P., Hussain, M. A., and Daosud, W. (2009). Neural network based model predictive control for a steel pickling process. *Journal of Process Control*, 19(4):579–590.

Li, B. and Rangarajan, S. (2022). A conceptual study of transfer learning with linear models for data-driven property prediction. *Computers & Chemical Engineering*, 157:107599.

Li, S. and Wen, J. (2014). A model-based fault detection and diagnostic methodology based on PCA method and wavelet transform. *Energy and Buildings*, 68:63–71.

Liang, J. and Du, R. (2007). Model-based fault detection and diagnosis of HVAC systems using support vector machine method. *International Journal of refrigeration*, 30(6):1104–1114.

Lu, Y., Tian, Z., Zhou, R., and Liu, W. (2021). A general transfer learning-based framework for thermal load prediction in regional energy system. *Energy*, 217:119322.

MacGregor, J. and Cinar, A. (2012). Monitoring, fault diagnosis, fault-tolerant control and optimization: data driven methods. *Computers & Chemical Engineering*, 47:111–120.

MacGregor, J. F., Jaeckle, C., Kiparissides, C., and Koutoudi, M. (1994). Process monitoring and diagnosis by multiblock PLS methods. *AIChE Journal*, 40(5):826–838.

Misra, M., Yue, H. H., Qin, S. J., and Ling, C. (2002). Multivariate process monitoring and fault diagnosis by multi-scale PCA. *Computers & Chemical Engineering*, 26(9):1281–1293.

Moonen, M., De Moor, B., Vandenberghe, L., and Vandewalle, J. (1989). On-and off-line identification of linear state-space models. *International Journal of Control*, 49(1):219–232.

Mustafaraj, G., Chen, J., and Lowry, G. (2010). Development of room temperature and relative humidity linear parametric models for an open office using BMS data. *Energy and Buildings*, 42(3):348–356.

Nikravesh, M., Farell, A. E., and Stanford, T. G. (2000). Control of nonisothermal CSTR with time varying parameters via dynamic neural network control (DNNC). *Chemical Engineering Journal*, 76(1):1–16.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Psichogios, D. C. and Ungar, L. H. (1992). A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511.

Qin, S. J. and Ljung, L. (2003). Closed-loop subspace identification with innovation estimation. *IFAC Proceedings Volumes*, 36(16):861–866.

Sadeghassadi, M., Macnab, C. J., Gopaluni, B., and Westwick, D. (2018). Application of neural networks for optimal-setpoint design and MPC control in biological wastewater treatment. *Computers & Chemical Engineering*, 115:150–160.

Shahnazari, H., Mhaskar, P., House, J. M., and Salsbury, T. I. (2018). Heating, ventilation and air conditioning systems: fault detection and isolation and safe parking. *Computers & Chemical Engineering*, 108:139–151.

Shahnazari, H., Mhaskar, P., House, J. M., and Salsbury, T. I. (2019). Modeling and fault diagnosis design for HVAC systems using recurrent neural networks. *Computers & Chemical Engineering*, 126:189–203.

Shardt, Y. A. and Huang, B. (2011). Closed-loop identification condition for ARMAX models using routine operating data. *Automatica*, 47(7):1534–1537.

Wang, Y., Wu, D., and Yuan, X. (2020). LDA-based deep transfer learning for fault diagnosis in industrial chemical processes. *Computers & Chemical Engineering*, 140:106964.

Wong, W. C., Chee, E., Li, J., and Wang, X. (2018). Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. *Mathematics*, 6(11):242.

Wu, H. and Zhao, J. (2018). Deep convolutional neural network model based chemical process fault diagnosis. *Computers & Chemical Engineering*, 115:185–197.

Wu, H. and Zhao, J. (2020). Fault detection and diagnosis based on transfer learning for multimode chemical processes. *Computers & Chemical Engineering*, 135:106731.

Yoon, S. and MacGregor, J. F. (2004). Principal-component analysis of multiscale data for process monitoring and fault diagnosis. *AIChE Journal*, 50(11):2891–2903.

Zhang, Z., Wang, Y., and Wang, K. (2013). Intelligent fault diagnosis and prognosis approach for rotating machinery integrating wavelet transform, principal component analysis, and artificial neural networks. *The international journal of advanced manufacturing technology*, 68(1):763–773.

Zhao, H., Sun, S., and Jin, B. (2018). Sequential fault diagnosis based on lstm neural network. *IEEE Access*, 6:12929–12939.

Zhou, J., Guo, A., Celler, B., and Su, S. (2014). Fault detection and identification spanning multiple processes by integrating PCA with neural network. *Applied Soft Computing*, 14:4–11.

Chapter 2

A hybrid machine learning approach integrating recurrent neural networks with subspace identification for modelling HVAC systems

The contents of this chapter have been published in the *Canadian Journal of Chemical Engineering*.

- A hybrid machine learning approach integrating recurrent neural networks with subspace identification for modelling HVAC systems. Hesam Hassanpour, Prashant Mhaskar, and Michael J. Risbeck, *The Canadian Journal of Chemical Engineering*, 2022. DOI: 10.1002/cjce.24392

Abstract

This paper addresses the problem of system identification for heating, ventilation, and air conditioning (HVAC) systems using a relatively small amount of data for the zone in consideration, by leveraging larger datasets for similar zones. To this end, a hybrid machine learning approach is developed where a pre-trained recurrent neural network (RNN) model, trained on a large amount of data from a representative zone, is leveraged to build models for the other zones using a smaller amount of data. This is achieved by developing a hybrid model that integrates the pre-trained RNN model with the models built using the subspace identification (SubID) technique to predict the residuals (differences between the real outputs and the predicted outputs from the pre-trained RNN model) in the other zones. The effectiveness of the proposed hybrid approach is shown using real data collected from a multi-zone fitness centre. The results demonstrate the superior performance of the hybrid approach over the cases where individual RNN and SubID models are directly developed using only the data from the zones in question.

2.1 Introduction

Heating, ventilation, and air conditioning (HVAC) systems are the key components of modern buildings to maintain thermal comfort and indoor air quality. In recent decades, there has been a significant increase in the energy consumed by HVAC systems (Pérez-Lombard et al. (2008)). This has motivated research on energy-saving control and fault diagnosis strategies for these systems. In this regard, dynamic modelling of HVAC systems has received significant attention due to its application in model-based optimal control and fault diagnosis designs (Afroz et al. (2018)).

Modelling of HVAC systems can generally be divided into three categories: first-

principles models, data-driven models, and hybrid modelling techniques, where different methods are integrated to enhance the model performance. In first-principles models, the governing laws of physics are used to derive a set of mathematical equations of the system (Tashtoush et al. (2005); Nassif et al. (2008); Satyavada and Baldi (2016); Baldi et al. (2019)). These models are also utilized in the model-based control strategies such as model predictive control (MPC) (Ganesh et al. (2021)). Although first-principles models have good extrapolation capabilities, they are generally difficult to develop and maintain because of uncertainty in the physical parameters and unmeasured physical states.

The recent development of data storage and data-acquisition devices, along with the improvement in the computing technologies, has made data-driven modelling techniques highly attractive. These methods require the availability of a significant amount of process data. A number of approaches consider utilizing mathematical techniques such as statistical regression to establish the relationship between input/output variables. In this direction, several statistical-based methods such as Box-Jenkins (BJ), autoregressive exogenous (ARX), and autoregressive moving average exogenous (ARMAX) models have been developed and applied to predict different process variables in HVAC systems (Ríos-Moreno et al. (2007); Mustafaraj et al. (2010)). Latent variable-based methods such as principal component analysis (PCA) and projection to latent structures (PLS) have been developed for monitoring, fault diagnosis, and optimization (Misra et al. (2002); MacGregor and Cinar (2012)). These techniques have also been used for monitoring and fault diagnosis of HVAC systems (Chen and Lan (2010); Li and Hu (2019); Hassanpour et al. (2020b)). Subspace identification (SubID) models have also been used in the area of HVAC modelling and fault diagnosis (Ferkl and Široký (2010); Shahnazari et al. (2018)). In addition, machine learning techniques such as artificial neural networks (ANN) (Du et al. (2014); Huang et al. (2015); Coccia et al. (2021)) and support vector machines (SVM) (Liang and Du (2007)) have been utilized for modelling, control, and fault diagnosis of HVAC

systems.

Among several machine learning techniques, recurrent neural networks (RNN) have widely been used for the dynamic modelling due to their capabilities to capture nonlinearities (Bhat and McAvoy (1990); Wang et al. (2017); Sadeghassadi et al. (2018); Hassanpour et al. (2020a); Wu et al. (2020); Khadem and Rey (2021)), in general. RNN models have also been used for dynamic modelling of HVAC systems, in particular. In this direction, the nonlinear autoregressive network with exogenous inputs (NARX) modelling technique is used to develop the predictive models for HVAC systems in order to minimize energy consumption (Kusiak and Xu (2012)). The NARX-based models are also developed for being used in the MPC implementations (Yang et al. (2020)). In addition, other RNN architectures such as long short-term memory (LSTM) neural networks are developed to predict the power consumed by a building's HVAC system (Sendra-Arranz and Gutiérrez (2020)). An encoder-decoder LSTM-based modelling approach is also developed for the economic model predictive control (EMPC) design (Ellis and Chinde (2020)). The encoder model is used for state estimation, providing an initial condition for the decoder model embedded in the EMPC as the predictive model. RNN-based models are also used to build dynamic models and develop fault diagnosis strategies for HVAC systems (Shahnazari et al. (2019)).

In order to successfully develop a model-based control and fault diagnosis strategies, developing a model, which can predict the dynamic behaviour of the zone with acceptable accuracy, is a critical task. A reasonably good model of the zone must predict the dynamic behaviours within the zone in the presence of different internal and external loads. In this direction, several ANN-based modelling techniques have been used to model dynamics of multiple interacting zones (Garnier et al. (2014); Huang et al. (2015)). NARX neural network models are used to develop a model for a multi-zone building in order to predict indoor temperature associated with each zone (Huang

et al. (2015); Kim (2020)). LSTM networks have also been used to model a multi-zone building (Mtibaa et al. (2020)). In the context of ANN modelling, due to the presence of a large number of parameters, the availability of a significant amount of data is required to avoid the over-fitting problem and achieve a model with reasonable prediction accuracy. While the availability of data has grown in general, this is not necessarily the case with building systems due to a general lack of instrumentation and data cleansing issues (to ensure clean data are available for system identification). Techniques that can harness the ability of RNNs to capture process non-linearities, but yet work with relatively smaller amounts of data, are therefore particularly sought after in the area of building control.

In recent years, transfer learning-based approaches have been developed to handle the data shortage problem for HVAC model training (Gao et al. (2021); Deng and Chen (2021)). In general, the goal of transfer learning is to transfer the knowledge from source tasks to a target task, where a limited amount of training data is available (in the context of HVAC modelling, this can be considered as using an existing model, trained with a sufficient amount of data from source buildings, to assist in predicting a target building behaviour). Using the idea of transfer learning, some approaches have been proposed for cross-building energy prediction (Ribeiro et al. (2018); Fang et al. (2021)). In addition, some transfer learning-based methods have been proposed to predict the thermal responses of the building (Grubinger et al. (2017)). In Chen et al.'s study (Chen et al. (2020)), a transfer learning-based method is developed by re-training a small number of parameters of a pre-trained deep multilayer perceptron (MLP) neural network model, trained using a large amount of data associated with the source room, for prediction of the indoor air temperature and relative humidity of a target room with an insufficient amount of data. In Lu et al.'s study (Lu et al. (2021)), a transfer learning-based framework, which uses LSTM networks, is proposed to achieve the predictive model for thermal load prediction of a regional energy station under the condition of limited data. To handle the over-fitting problem of the RNN-

based models using a limited amount of data in transfer learning-based methods, it becomes useful to utilize the data-driven techniques that allow for more direct handling of this issue.

Motivated by the above considerations, this work addresses the problem of system identification for multi-zone building HVAC systems while handling the lack of availability of large amounts of data by integrating SubID-based models with RNN. The application of the proposed approach to a multi-zone fitness centre is demonstrated. The approach leverages an already pre-trained RNN model built using a large amount of data from a representative zone in order to develop models for the other zones using a smaller amount of data. In this approach, the residuals, which are the differences between the real outputs and the predictions obtained using the pre-trained RNN model, are used to develop models using SubID. These SubID-based residual models are then integrated with the pre-trained RNN model to predict the outputs of the other zones. The rest of the manuscript is organized as follows: Section 6.2 describes the multi-zone fitness centre, followed by a brief overview of the RNN and SubID techniques. Section 5.3 presents the proposed hybrid modelling approach for multi-zone HVAC systems. The application of the proposed approach to the fitness centre data is then presented in Section 5.4. Finally, concluding remarks are presented in Section 6.5.

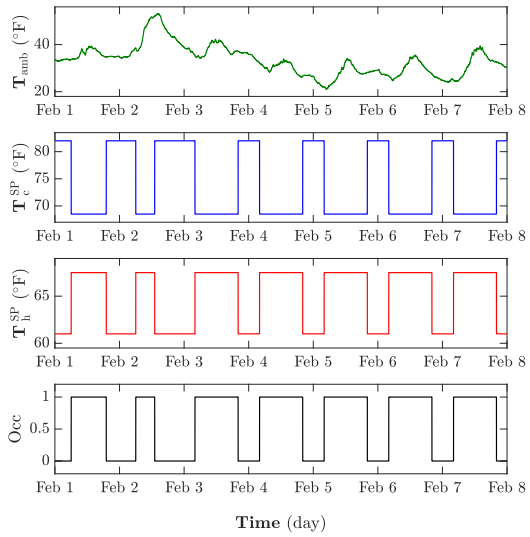
2.2 Preliminaries

A brief description of the case study, which is a multi-zone fitness centre, is provided in this section. Next, a brief overview of the RNN and SubID modelling approaches is given. In addition, the limitation of using these approaches in the presence of a data shortage problem is shown to provide the motivation for the proposed approach.

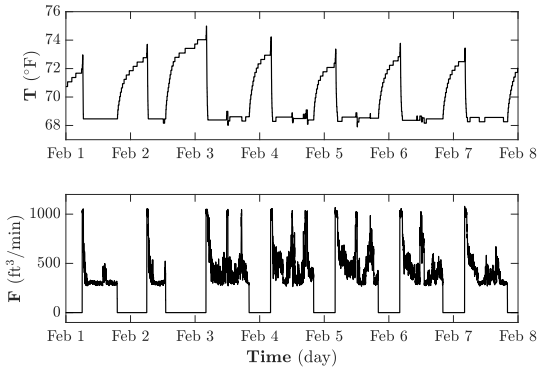
2.2.1 Case study: Multi-zone fitness centre

The datasets are collected from a fitness centre in an office building in the Midwest US (provided by Johnson Controls), which has six independently controlled zones (each dataset corresponding to each individual zone contains information about 35 days from Jan 26, 2020, to Feb 29, 2020). Separate heating and cooling set-points are used by the thermostats to maintain the temperature of the zones. The zone will be actively heated if the zone temperature (T) is less than the heating set-point (T_h^{SP}) ($T < T_h^{SP}$). In addition, the zone will be actively cooled if the zone temperature is above the cooling set-point (T_c^{SP}) ($T > T_c^{SP}$). When the zone temperature is in between T_h^{SP} and T_c^{SP} (e.g., during unoccupied hours when the set-point range is much larger), the temperature is free to float based on natural heat transfer, with no active heating or cooling from the HVAC system (except for minimum flows required for ventilation during occupied hours). The system has a fairly standard cascaded architecture where the thermostat determines the airflow set-point based on the current temperature set-point and measurement. A local controller within each variable air volume (VAV) box is then responsible for controlling flow (F) to that set-point. In summary, each zone has four inputs, including ambient temperature, heating set-point, cooling set-point, and occupancy mode, and two outputs, which are the zone temperature and flow. The values corresponding to the process inputs and outputs for a representative zone are shown in Figure 2.1, for the first week of February. The zone occupancy flags are a system setting equal to one when the building is considered occupied and zero when unoccupied. These values are scheduled by the building manager and thus do not necessarily represent the true occupancy cycle of the space. However, due to the almost perfect correlation between the occupancy flag and the temperature set-points, it might not be very useful to consider the occupancy flag as an input for the models. In addition, the values are sampled every minute at the indicated timestamps. Note, however, that the data are passed through a COV filter, which

means that new measurements are only transmitted if the measured value differs by at least a given amount from the previous measurement. This filtering manifests similarly to quantization in that the data appear staircase-like rather than smooth. The overall goal of the dynamic models is to predict the outputs (zone temperature and zone flow) based on the inputs (heating/cooling set-points, occupancy flag, and ambient temperature).



(a)



(b)

Figure 2.1: (a) Process inputs: ambient temperature (T_{amb}), cooling set-point (T_c^{SP}), heating set-point (T_h^{SP}), and occupancy mode, (b) Process outputs: zone temperature (T) and zone flow (F)

Remark 1. *Note that the inputs are defined in the context of modelling, which can include the manipulated inputs and measurable disturbances, such as ambient temperature. In addition, there are several unmeasurable disturbances for each zone, such as internal heat gains (from lighting and plug loads), which are obviously not considered as the inputs of the model.*

2.2.2 Recurrent neural networks (RNN)

An RNN is a class of ANNs that have been commonly used for dynamic modelling of non-linear systems (Shahnazari et al. (2018); Wu et al. (2019)). The existence of feedback loops in the RNN architecture provides the network with a memory that enables capturing dynamic behaviours in a way similar to non-linear state-space models. Various types of RNN such as long short-term memory (LSTM) and gated recurrent unit (GRU) have been developed for different applications such as time series prediction and natural language processing. A schematic of an RNN, together with its unfolded structure, is shown in Figure 2.2. As shown in this figure, the previous information (internal states) derived from preceding inputs is fed back into the network, allowing for storing the information from the previous sampling time. This information is then utilized to calculate the system outputs at the current time. The mathematical description of the RNN model can be defined as follows:

$$\begin{aligned} x_k &= f_h(W_u u_k + W_x x_{k-1}) \\ y_k &= f_o(W_y x_k) \end{aligned} \tag{2.1}$$

where x_k, u_k, and y_k represent the vectors of current states, inputs, and outputs, respectively. As can be seen in Equation (2.1), the current state depends on the

current input and previous state (x_{k-1}), and the current output depends on the current state. In addition, W_u, W_x, and W_y represent the weight matrices connecting the inputs to the states, connecting the states from the previous time step to the states in the following time step, and connecting the states to the outputs, respectively. $f_h(\cdot)$ and $f_o(\cdot)$ also represent the activation functions associated with hidden and output layers, respectively.

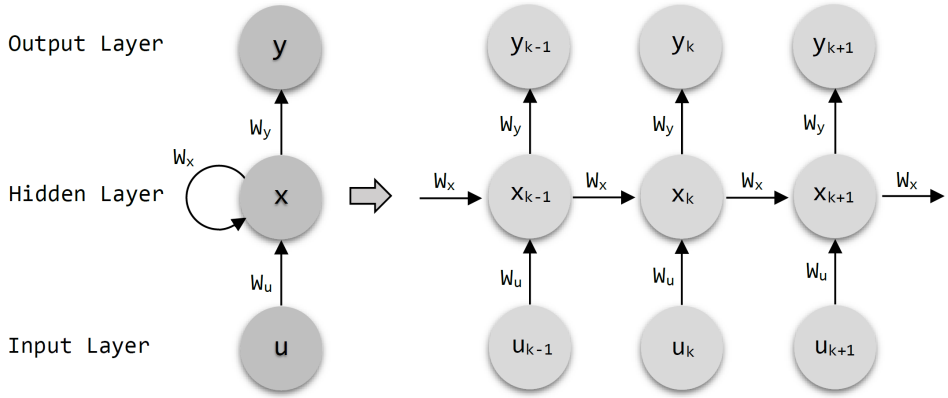


Figure 2.2: Schematic of a recurrent neural network and its unfolded structure

The existence of a large number of parameters provides neural networks with large degrees of freedom that can fit a variety of complex non-linear systems. However, a neural network-based model usually requires a significant amount of high-quality data in order to get the most adequate training and achieve high generalization ability. A larger size of data helps handle over-fitting issues. Therefore, in this work, we consider a representative zone, where a sufficient amount of data is available, and as such, a well-trained RNN model is available. The effort is to leverage or utilize this model for another zone where the data availability may be limited. To build a model using a limited amount of data, other statistical-based modelling methods that allow for more direct handling of the over-fitting problems are known to be useful. This motivates us to use the SubID technique to develop the residual models for the other zones where the training data samples are insufficient. Before proceeding to present the proposed hybrid approach (and to contrast with the proposed approach), the performance of

the RNN-based models developed using a limited amount of data is shown in the next section.

2.2.3 Subspace identification (SubID)

A brief overview of the SubID method is presented in this section (see Huang et al. (2005); Qin (2006) for more details). In this approach, the objective is to use input-output data in order to identify a discrete time linear time-invariant state-space model of the following form:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{2.2}$$

where $x \in \mathbb{R}^{n_x}$ represents the vector of state variables, $u \in \mathbb{R}^{n_u}$ represents the vector of inputs, and $y \in \mathbb{R}^{n_y}$ represents the vector of outputs. $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $C \in \mathbb{R}^{n_y \times n_x}$, and $D \in \mathbb{R}^{n_y \times n_u}$ are the system matrices (parameters) identified by this identification technique. The system matrices are directly computed using the input-output data Hankel matrices without any iteration, compared with the iterative algorithms utilized in the prediction error methods (PEM), which are based on optimization techniques (similar to how the RNN models are trained). In the subspace-based identification approaches, matrix decomposition techniques, such as QR decomposition and singular value decomposition (SVD), are used in order to identify a model. Identification also consists of two steps, which are finding the state sequence that captures the time varying dynamics and calculating the system matrices subsequently using an appropriate regression. Several methods have been developed for SubID techniques such as canonical variate analysis (CVA) (Larimore (1990)),

multivariable output error state-space (MOESP) (Verhaegen and Dewilde (1992)), and the numerical algorithms for subspace state-space system identification (N4SID) (Van Overschee and De Moor (1994)). In this work, N4SID approach (in MATLAB) is used to identify the residual models.

2.2.4 RNN and SubID-based models using a limited amount of data

In this section, the problem of developing RNN and SubID models for the case study, where a limited amount of data is available, is presented.

RNN model identification

In order to show the RNN modelling performance in the presence of a data shortage problem, we assume that the data samples corresponding to three days for zones a and b are available. Two scenarios are considered for each zone: (1) the first scenario considers using the data of three days in January (Jan 26–28), and (2) the second scenario uses the data of three days in February (Feb 24–26). These cases are listed in Table 2.1. Note that, for each case, the number indicates the month from which the data are used, and the letter represents the zones.

Table 2.1: Different cases which are considered to develop the models using a limited amount of data

Case	Training data	Validation data	Testing data
1.a	Jan 26	Jan 27	Jan 28
2.a	Feb 24	Feb 25	Feb 26
1.b	Jan 26	Jan 27	Jan 28
2.b	Feb 24	Feb 25	Feb 26

In order to train an RNN model for each case, the data samples corresponding to the input variables (heating set-points, cooling set-points, and ambient temperature)

and output variables (zone temperature and zone flow) are first divided into training, validation, and testing data. Therefore, in the first scenario, the data samples of Jan 26, Jan 27, and Jan 28, are used to train, validate, and test the model, respectively ($\approx$ 33% of data samples are used in each modelling stage). Similarly, the data samples of Feb 24, Feb 25, and Feb 26, are, respectively, used for training, validation, and testing phases in the second scenario. In the next step, the training data samples, corresponding to each case, are first normalized (mean-centred and scaled to unit variance), and then fed to an RNN network in MATLAB Machine Learning and Deep Learning toolbox (layer recurrent network (LRN) which has a standard RNN architecture). Note that the trained model predicts the normalized values of the outputs in the validation and testing phases. To calculate the outputs, the normalized values are re-scaled and shifted based on the standard deviation and mean values of the training data. In the RNN network, different hyperparameters exist that must be tuned such as the number of hidden layers, the number of neurons in each hidden layer, activation function types, and the number of epochs. To this end, different architectures with 1 hidden layer, including different numbers of neurons (2, 3, 4, and 5), are examined (note that adding more neurons or hidden layers results in the over-fitting problem due to the use of a small amount of data). In addition, *tanh* and linear activation functions are selected for the hidden layers and the output layer, respectively. The number of epochs is also chosen in a way to avoid over-fitting (the initial value is 20 but is modified as needed). Bayesian regularization backpropagation algorithm that updates the model parameters (weights and biases) based on Levenberg-Marquardt optimization is chosen. As mentioned, four RNN models using different numbers of neurons (RNN_n: n represents the number of neurons) are trained for each case mentioned above. In order to quantify the prediction error of the models, scaled root mean square error (RMSE) metric, given by Equation (6.5), is used:

$$\text{Scaled RMSE} = \sum_{i=1}^{l} \frac{\sqrt{\frac{\sum_{j=1}^{n}(y_{i,j}-\hat{y}_{i,j})^2}{n}}}{\sigma_i} \tag{2.3}$$

where $y_{i,j}$ and $\hat{y}_{i,j}$ are the j^{th} observation corresponding to the i^{th} output and its prediction, respectively, and σ_i represents the standard deviation of the i^{th} output. l and n also represent the number of outputs and observations, respectively. Due to different initializations of the parameters and non-convex nature of the optimization problem to calculate the optimal values of theses parameters, the results can be different each time a neural network model is trained. This also raises two related challenges: (1) the lack of guarantee of finding a global solution and (2) with the same architecture, the difficulty of ensuring that the same solution is found in every implementation, making it difficult to compare a neural network with another implementation. To try to get the global minima (or at least a good local minima), random initializations of the parameters can be utilized. While it may promote the achievement of the global minima, it is not guaranteed. This work specifically addresses the second concern. In particular, in developing and testing the RNN models for each neural network with specific architecture, multiple trainings with random re-initializations are performed (to account for the fact that different local minima may be found in a particular run), and all comparisons are based on averages over these multiple runs. Thus, the RNN model, for each specific architecture mentioned above, is trained 50 times for each case, and then, the scaled RMSE value is calculated in the validation phase. Figures 2.3 and 2.4 summarize the results obtained for the two cases corresponding to zones a and b, respectively. Based on the scaled RMSE values of the validation data, the best model for each case (RNN_4 for case 1.a, RNN_5 for case 2.a, RNN_5 for case 1.b, and RNN_4 for case 2.b) is then selected to evaluate prediction performance of the model on the unseen (testing) data. The results are shown in Figures 2.5 and 2.6 for zones a and b, respectively (dash-dotted red lines). As can be seen, these models are not able

to predict the zone outputs well due to the use of a small amount of data for training the RNN models. The scaled RMSE values obtained for each case in the testing phase are also listed in the first row of Table 2.3.

Remark 2. *Note that a model's ability to perform multi-step ahead predictions with reasonable accuracy is critical for being successfully used in different applications such as MPC. Therefore, the models are developed in this work in a way to predict the zone behaviour multi-step ahead (for an entire day). It should also be noted that the value of the scaled RMSE, for each model in the validation or testing phases, is calculated based on multi-step ahead prediction of the model.*

Remark 3. *In order to make sure that the trained model is working properly (with reasonable accuracy), it is necessary to evaluate the model performance on unseen (new) data. Thus, in the process of model building using machine learning or statistical-based approaches, the dataset should be divided into three categories including training data, validation or hold-out data, and testing or unseen data. The training data are only used to train the model (calculating the best set of parameters). The validation data are then used to assess the progress and draw conclusions in order to improve the model capability (providing an unbiased assessment of the model fit on the training data while tuning the model hyperparameters). After analyzing the model performance in the validation step, the best model, obtained based on a specific metric, will be chosen for being tested on the unseen data. This enables evaluating the prediction performance of the final model on new data.*

SubID model identification

The four cases defined in Section 2.2.4 are used again to build the subspace-based models. In the SubID approach, the number of states must be determined in order to develop a model. One of the advantages of this approach is the uniqueness of

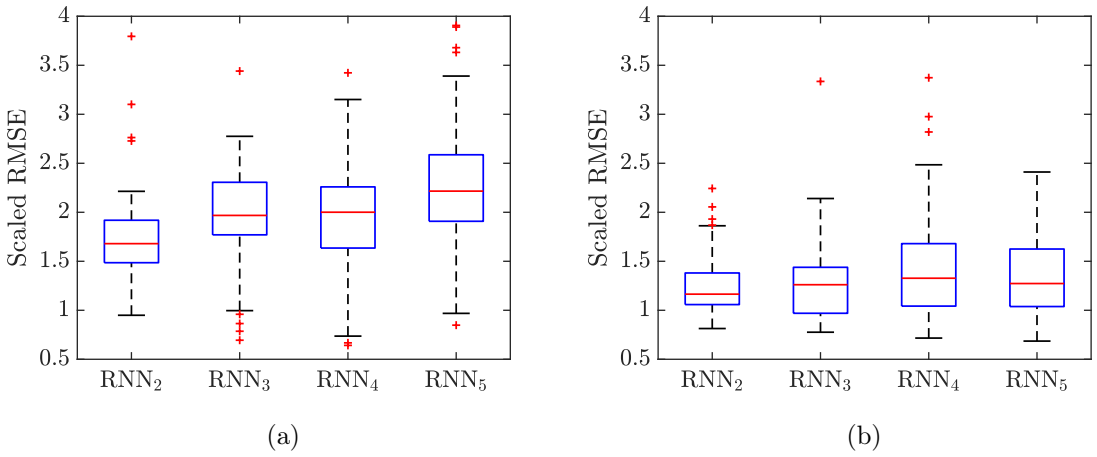


Figure 2.3: Scaled root mean square error (RMSE) of different recurrent neural network (RNN)-based models trained for (a) case 1.*a* and (b) case 2.*a*, in the validation phase

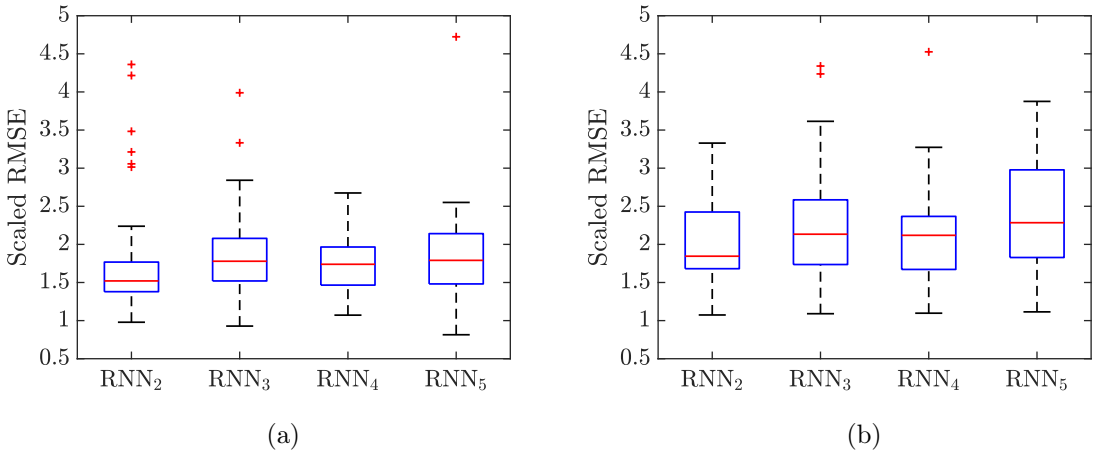


Figure 2.4: Scaled root mean square error (RMSE) of different recurrent neural network (RNN)-based models trained for (a) case 1.*b* and (b) case 2.*b*, in the validation phase

solution, unlike the neural network-based modelling methods where the results can be different in each run due to the different parameters initializations. Thus, the SubID method does not require multiple models when a certain number of states is considered. Similar to the RNN modelling, the training data, shown in Table 2.1, are first normalized and then used to train a model for each case. Since the trained model predicts the normalized outputs, the standard deviation and mean values of

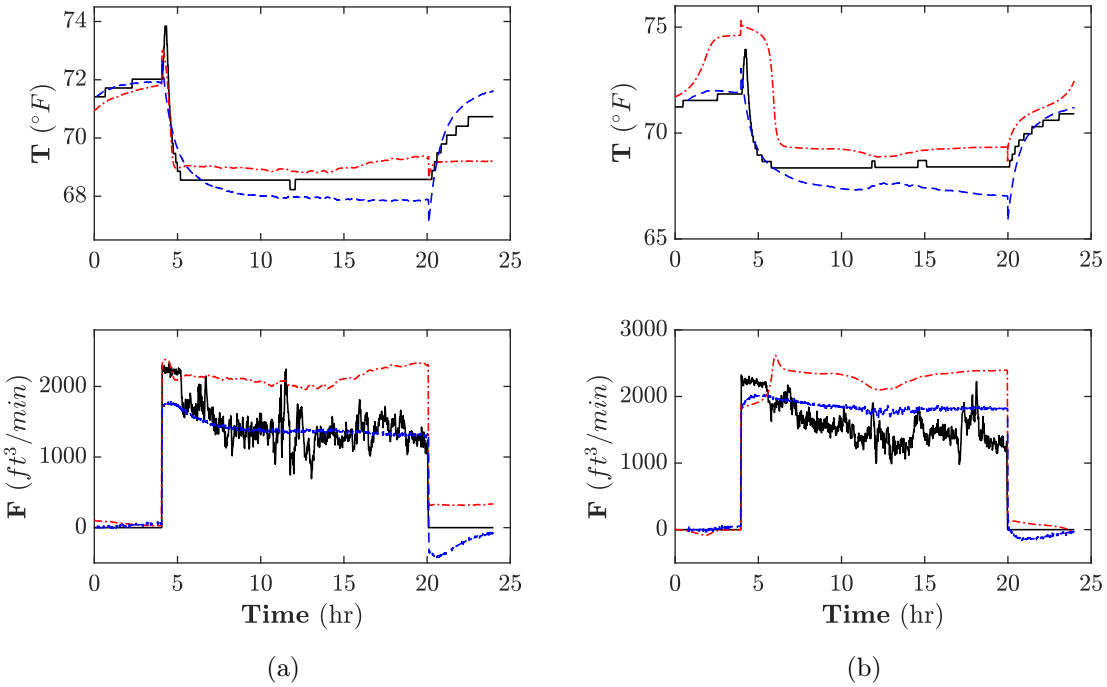


Figure 2.5: Comparison of the prediction results obtained using the best recurrent neural network (RNN) and subspace identification (SubID)-based models for (a) case 1.*a* and (b) case 2.*a* (in the testing phase): Measured outputs (solid black lines), SubID model prediction (dashed blue lines), and RNN model prediction (dash-dotted red lines)

the training data are used to calculate the predicted outputs. Different numbers of states (2–10) are used, and then, the best model is selected based on the lowest scaled RMSE value in the validation phase. The results show that the best models for cases 1.*a*, 2.*a*, and 1.*b* use 4 states. In addition, for case 2.*b*, the use of 5 states results in the best performance of the model on the validation data. The prediction results obtained using these models in the testing phase are shown in Figures 2.5 and 2.6 for zones *a* and *b*, respectively (dashed blue lines). In addition, the scaled RMSE values obtained by applying the SubID models, in the testing phase, are compared with those values obtained based on the RNN models and listed in the second row of Table 2.3. The results reveal the superior performance of the SubID models over the RNN models when a limited amount of data is available for system identification.

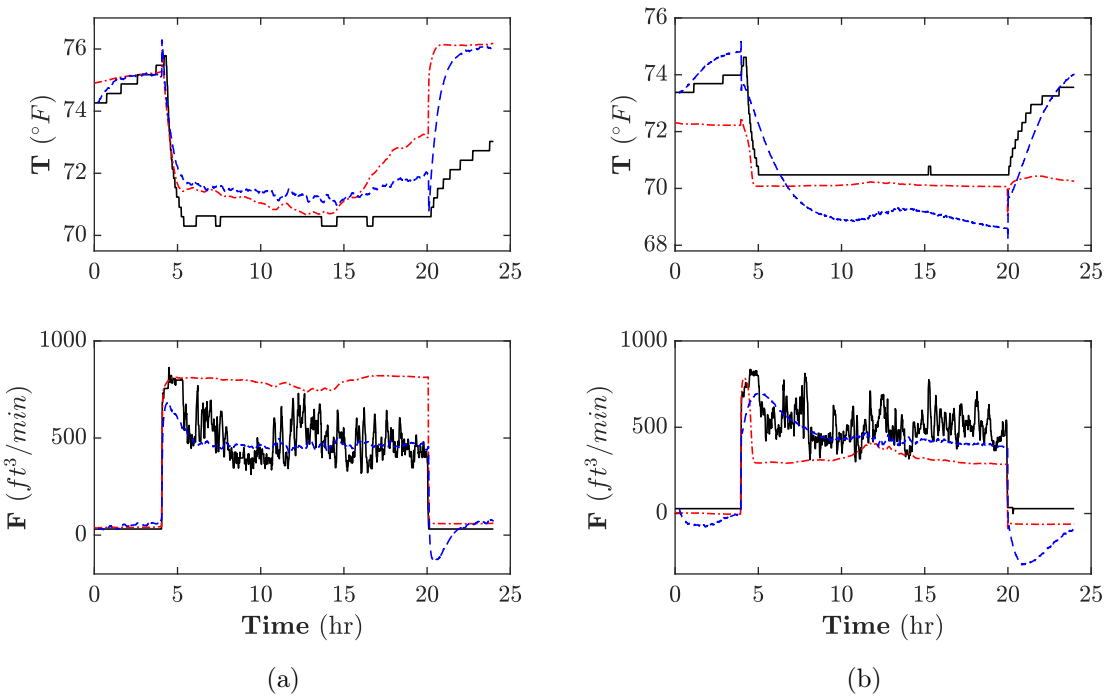


Figure 2.6: Comparison of the prediction results obtained using the best recurrent neural network (RNN) and subspace identification (SubID)-based models for (a) case 1.*b* and (b) case 2.*b* (in the testing phase): Measured outputs (solid black lines), SubID model prediction (dashed blue lines), and RNN model prediction (dash-dotted red lines)

While the SubID model provides good predictions, the question remains as to whether a well-trained RNN model that can capture the key characteristics of the unit in question (or a similar unit elsewhere) can be utilized to improve the overall model. This is addressed in the next section.

Remark 4. *Note that normalized and mean-centred values of the variables are used in training and prediction. This is consistent with the dynamic model structure (like SubID), which implies that* $x = 0$ *and* $u = 0$ *are the steady state of the system. That property often holds for process data operated near a particular steady state, but is not necessarily guaranteed for operational data. For instance, in the present case, a single model is developed to predict the behaviour of each zone in both the occupied and unoccupied conditions, which naturally have different dynamics, and there is no*

guarantee that mean-centred and scaled values of the variables will represent a steady state. As such, the efficacy of the modelling approach is primarily verified using validation/prediction capability.

2.3 Proposed hybrid modelling approach

The proposed approach for system identification is presented in this section. The main idea is leveraging an existing well-trained model, developed for one problem, in order to build models for other related problems. This is the key idea of any transfer learning method, as mentioned in Section 6.1. In the area of transfer learning for dynamic modelling, usually, a small number of parameters corresponding to a pre-trained RNN model, built for one problem, are selected to be re-trained on new data samples related to a different but related problem (Chen et al. (2020)) (available data from the new problem is usually limited). This not only makes the training process faster (compared to building a model from scratch) but also enables better handling of the over-fitting problem in the presence of a small amount of data from the new case. Recall that, as shown in Section 6.2, using a relatively small amount of data usually does not result in a model with good accuracy. When a pre-trained model (already built for a related problem using a large amount of data) is available, it can be utilized to predict the behaviour of a new problem. The prediction performance of the pre-trained model would be sufficiently good if the dynamic behaviour of the new problem is almost the same as the behaviour of the previous problem for which the pre-trained model exists, otherwise there will be some deviations.

The key idea with the proposed approach is to train a model to predict these deviations (or residuals, which are differences between the real values and predictions obtained by applying the pre-trained model), which can be used to correct the pre-trained model prediction. Since small amount of the residual data are available, this requires

using approaches allowing for better handling of the over-fitting issue, such as linear statistical-based methods, as shown in Section 2.2.4. The layout of the model building steps, along with the idea of the hybrid approach is shown in Figure 2.7. The top left figure shows the input–output data collected from a new problem. The same input is then fed to the pre-trained model to predict the behaviour of the new problem based on this model (y_n^{pt}). Finally, the residuals ($r = y_n - y_n^{pt}$), together with the inputs, are used to develop a residual model. The right figure shows the integration of the pre-trained model and the residual model to predict the behaviour of the new case (y_n^p).

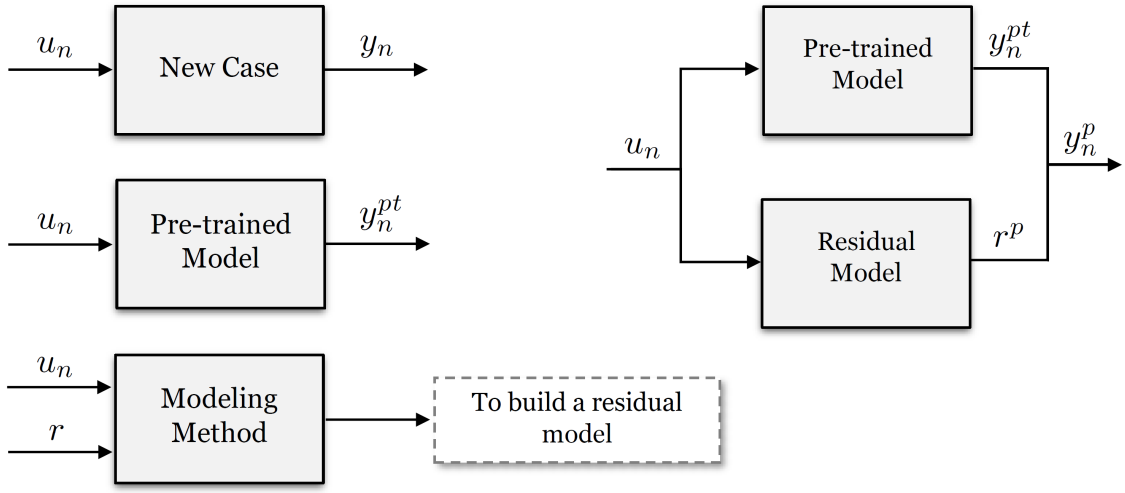


Figure 2.7: Schematic illustrating the hybrid modelling method. Left figure shows the steps for developing the residual model. Right figure shows the integration of the pre-trained model and the residual model

Consider that a limited amount of data from the new case is available, and it is divided into training, validation, and testing data. The steps to develop and utilize a residual model (in the hybrid model) are described below:

1. Training and validation input data are fed to the pre-trained model, and the prediction is calculated (y_n^{pt}).

2. Residuals, differences between the pre-trained model predictions and the real

values, are then calculated ($r = y_n - y_n^{pt}$).

3. Residuals, corresponding to the training part, are normalized (mean-centred and scaled to unit variance).

4. Normalized training residuals, together with the inputs, are used to build a residual model (because the residual model predicts the normalized residual, the standard deviation and mean values corresponding to the training residuals are used to calculate the predicted residuals)

5. Validation residuals are used to evaluate the residual model performance based on different hyperparameters, and then, the best model is selected.

6. The residual model is then integrated with the pre-trained model to predict the behaviour of the new problem. To this end, the input data are fed to both the pre-trained and residual models, and then, their predictions are added to make the final prediction (in this stage, the testing data are used to evaluate the performance of the hybrid model).

Remark 5. *Note that the existing model developed using a large amount of data does not have to be from the same set of zones or even the same building, but the set of input–output variables utilized to develop this model must contain the input and output variables that will be used to build the models based on a small amount of data.*

Remark 6. *Note that the main idea of the proposed approach is to integrate the residual model, developed in the presence of the data shortage problem for one problem, with the existing well-trained model of the different but related problem. The choice of an appropriate modelling strategy, utilized to develop models for the pre-trained model and the residual model, generally depends on the amount of data and the dynamic characteristics of the system. Due to the availability of a large amount of data and non-linear nature of the problem, the RNN modelling approach is utilized to build the model for the pre-trained model in this work (this will be shown in Section 2.4.1).*

On the other hand, because of the availability of a small amount of residual data, the SubID approach is selected to build the residual models due to its superior performance in handling the data shortage problem (as shown in Section 2.2.4). However, in general, other modelling techniques can be utilized to develop the pre-trained and residual models as long as their performance is satisfactory.

Remark 7. *Recall that the residuals are the differences between the pre-trained model prediction and the real value of the outputs for a new system. Thus, in the case where the new system is similar in structure to the system that was used to train the model, except that the outputs are simply scaled differently, a direct implementation of the proposed approach will not be useful. Instead, alternate hybrid modelling approaches will have to be considered and remain the focus of future work.*

2.4 Application of the hybrid approach to the multi-zone fitness centre

In this section, the proposed hybrid methodology is applied to the case study, multi-zone building HVAC system. A large amount of data corresponding to a representative zone is first used to develop a model. This model is then considered as the pre-trained model, available for the representative zone. Note that the pre-trained model is developed using both the RNN and SubID techniques. However, due to the superior performance of the RNN model over the subspace-based model in the case where a significant amount of data is available (this will be shown in the next section), the pre-trained RNN model is selected for developing the hybrid approach. Assuming that a limited amount of data corresponding to the other zones is available, the subspace-based residual models are then developed. Finally, the pre-trained RNN model is integrated with the subspace-based residual models to predict the behaviour of the other zones.

2.4.1 A representative zone modelling: A pre-trained model

In this section, the RNN and SubID techniques are used to model the representative zone (zone c), considered as a representative zone in the multi-zone building, using a large amount of data. The best model is then selected in order to be employed in the proposed hybrid method. In order to train an RNN model for zone c, a large amount of data corresponding to 34 days (Jan 26–Feb 28) is used. The data are first divided into the training data (Jan 26–Feb 22) ($\approx 80\%$ of data), validation data (Feb 23–25) ($\approx 10\%$ of data), and testing data (Feb 26–28) ($\approx 10\%$ of data). The training data samples are then normalized and used to develop an RNN model. Similar to the steps used in Section 2.2.4 to develop the RNN models, different architectures are considered to tune the hyperparameters. It should be mentioned that the presence of a large amount of training data allows for increasing the number of hidden layers and neurons in comparison with the previous models developed in Section 2.2.4. Again, several models with different architectures are trained multiple times, and then, their performances are evaluated on the validation phase to obtain the best model for zone c (the details are not presented here, for brevity). The best model, which uses 1 hidden layer with 10 neurons, is used in the testing phase to assess its performance on the unseen data. The results are shown in Figure 2.8.

Table 2.2: Comparison of the scaled root mean square error (RMSE) values obtained based on the subspace identification (SubID) and recurrent neural network (RNN)-based models for zone c (in the testing phase)

Model type	**Scaled RMSE**
SubID	0.77
RNN	0.56

In addition, a subspace-based model is developed for zone c using the same set of training, validation, and testing datasets to compare its performance with the RNN model. Several SubID models based on different numbers of states are developed using the normalized training data. Then, the prediction performance of each model

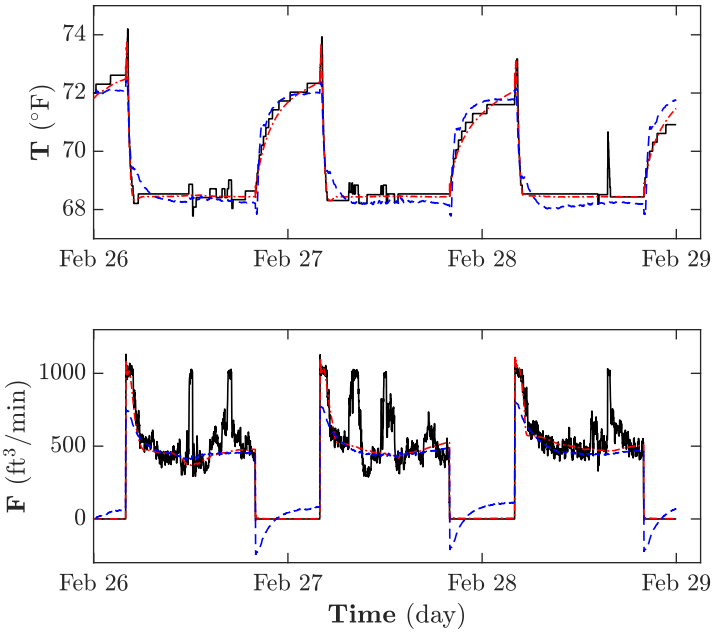


Figure 2.8: Comparison of the prediction results obtained using the recurrent neural network (RNN) and subspace identification (SubID)-based models for zone c (in the testing phase): Measured outputs (solid black lines), SubID model prediction (dashed blue lines), and RNN model prediction (dash-dotted red lines)

is evaluated in the validation phase to choose the final model. Based on the scaled RMSE values, the best result is obtained using a model with 10 states. The prediction result of this model in the testing phase is shown in Figure 2.8. The scaled RMSE values obtained based on the SubID and RNN models are also listed in Table 2.2. The results show the superior performance of the RNN model due to the fact that the RNN model is able to capture non-linearity more accurately, while not over-fitting, when a large amount of the training data is available for system identification.

2.4.2 Residual models

Consider that the pre-trained RNN model of zone c is available, which is employed to predict the behaviour of the four cases as described in Table 2.1. The steps, presented

in Section 5.3, are then applied to develop a residual model for each case. For example, for case 1.*a*, training and validation input data (from Jan 26 and Jan 27) are first fed to the pre-trained model, and the residuals are calculated. The training residuals, related to Jan 26, are then normalized and used with the training input to develop a model using the SubID. The reason for selecting the SubID technique is its superior performance when a relatively small amount of data is available, as shown in Section 2.2.4. Different numbers of states are used to develop different residual models, and the final model is chosen based on the lowest scaled RMSE value in the validation phase. This process is also repeated to build the subspace-based residual models for the other cases.

2.4.3 Hybrid modelling of the multi-zone building

Having obtained the residual models, these models are integrated with the pre-trained RNN model of zone c to predict the outputs of the four cases, mentioned in Section 2.2.4. The results obtained by applying the hybrid approach for cases 1.*a* and 2.*a* in the testing phase are shown in Figure 2.9. The results are also compared with the predictions obtained using only the pre-trained RNN model of zone c. As shown, although the pre-trained RNN model is able to predict the zone temperature fairly well in both cases, significant deviations from the real values can be observed for the zone flow predictions. However, applying the hybrid modelling approach enables the prediction of the zone behaviour with improved accuracy. This technique is also used to predict the outputs for cases 1.*b* and 2.*b*. Figure 2.10 shows the results obtained in the testing phase. Unlike the previous cases, applying the pre-trained RNN model leads to significant differences between the real and predicted values of the zone temperature. However, the use of the hybrid technique again results in a considerable improvement in the prediction of the zone temperature. The scaled RMSE values, obtained by applying the pre-trained and hybrid models, for all four

cases in the testing phase, are also listed in the third and fourth rows of Table 2.3, respectively. The results in this table show the superior performance of the hybrid models over the case where the RNN and SubID models, built using a limited amount of data, are used for prediction.

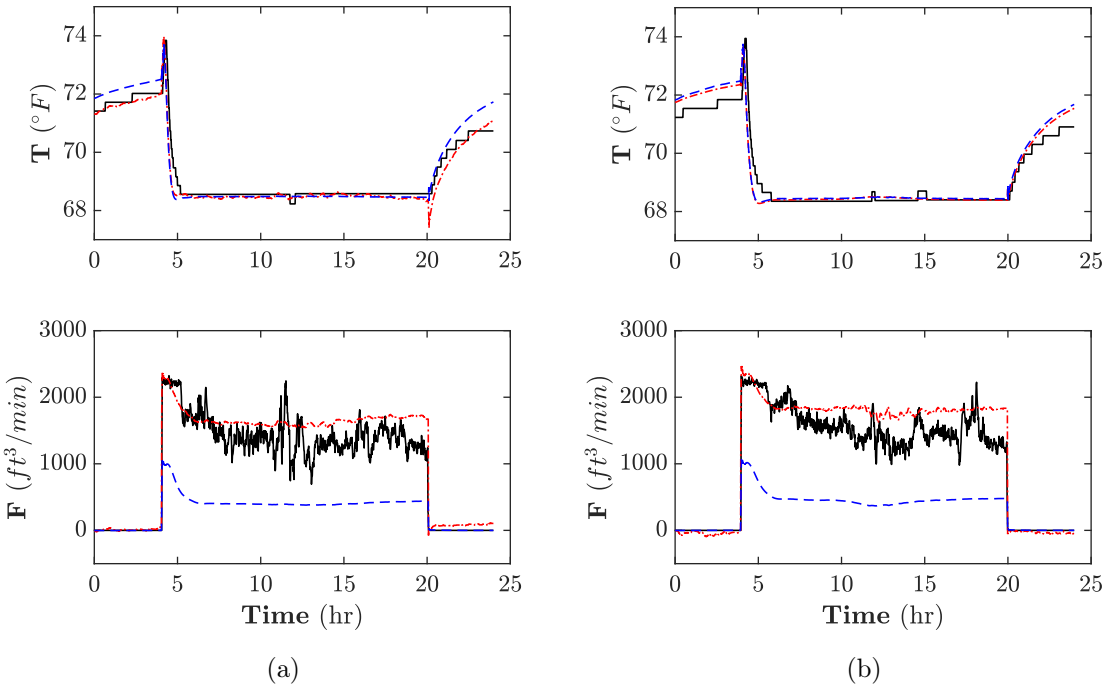


Figure 2.9: Comparison of the prediction results obtained using the pre-trained recurrent neural network (RNN) model of zone c and the hybrid approach for (a) case 1.a and (b) case 2.a (in the testing phase): Measured outputs (solid black lines), the pre-trained RNN model prediction (dashed blue lines), and the hybrid model prediction (dash-dotted red lines)

Table 2.3: Comparison of the scaled root mean square error (RMSE) values obtained based on different models for each case (in the testing phase)

Model type	**(1.a)**	**(2.a)**	**(1.b)**	**(2.b)**
RNN (limited data)	1.18	2.03	1.86	1.63
SubID (limited data)	0.65	0.98	1.03	1.40
Pre-trained RNN	1.31	1.49	1.59	2.08
Hybrid	0.52	0.63	0.75	0.83

Remark 8. *Note that a key step to evaluate the prediction performance of a state-*

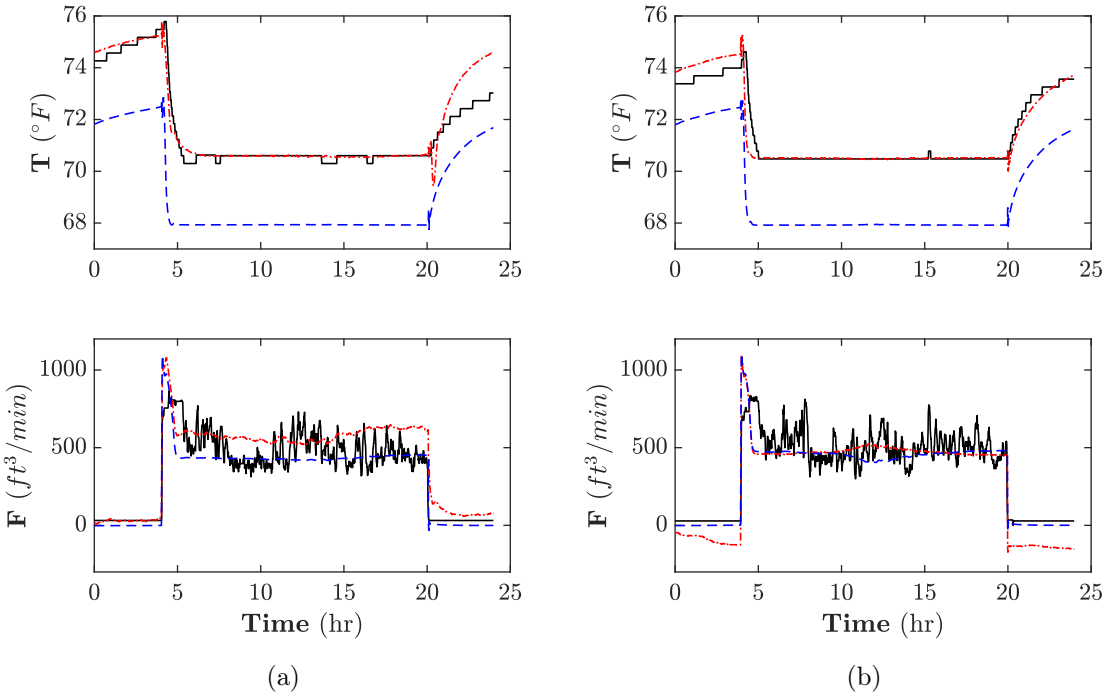


Figure 2.10: Comparison of the prediction results obtained using the pre-trained recurrent neural network (RNN) model of zone c and the hybrid approach for (a) case 1.b and (b) case 2.b (in the testing phase): Measured outputs (solid black lines), the pre-trained RNN model prediction (dashed blue lines), and the hybrid model prediction (dash-dotted red lines)

space data-driven model, on the unseen data, is to design an observer (e.g., Luenberger observer, Kalman filter, and extended versions for the non-linear systems) to estimate the state values where the process and model outputs are converged. The converged states are then considered to be the initial states of the identified model for prediction purposes (Ghosh et al. (2019); Alhajeri et al. (2021)). However, in this work, to assess the performance of the hybrid approach on the unseen data, a portion of the validation data (last 1000 validation data samples), in addition to the testing data, are fed to the model in order to ensure the output convergence (convergence must take place while using the validation data samples, which are available in advance).

Remark 9. *Note that the hybrid approach is developed for a multi-zoned building*

where the different zones follow a similar dynamic behaviour, albeit with different parameters (internal and external loads). In this situation, when a well-trained data-driven model of a representative zone is available (like the pre-trained RNN model in this work), it can be used to generate the residual. The residual model is then developed with the aim of being used in the hybrid approach to correct the prediction error between the pre-trained model of the representative zone and the zone in question. In the situation where the dynamic behaviour of the present unit is very different from the original pre-trained model, using this model (in both the residual development stage and integration stage) may lead to adding unnecessary information (non-linearity). Therefore, in the presence of data shortage problem, developing an individual data-driven model may outperform the hybrid approach. In addition, in order to decide when to use the hybrid approach and when to not, one can compare the prediction performance of the standalone model with the hybrid model, in the testing phase, and find which one is better.

Remark 10. *As shown in Table 2.3, not surprisingly, the scaled RMSE values obtained by applying the RNN and SubID models, developed using the limited data, are smaller than those obtained using the pre-trained RNN model, for some cases (pre-trained model was not trained on the data corresponding to the zones in question). This may raise a question of why not directly apply the SubID-based residual models to the RNN and SubID models trained with the limited data. Thus, for further analysis, we did try to use the RNN and SubID models, trained with the limited data, to develop the residual, and subsequently, hybrid models. However, the results indicate poorer performance of the hybrid models, developed in this way, compared to the proposed hybrid method. The reason can be explained by evaluating the residual values in the training phase. These values are close to zero when the RNN and SubID models, built using the limited data, are used due to the good fit between the model predictions and the training data, and therefore, they do not provide useful information for developing the residual model.*

Remark 11. *The idea of the hybrid modelling approach proposed in this work can be utilized in advanced control strategies such as MPC implementations. Approaches exist for utilizing a hybrid model integrating a first-principles model with a neural network-based residual model (Anderson et al. (2000)). The use of the hybrid models, which have at least one non-linear component (neural network), in the MPC implementations results in non-convex optimization problems. Therefore, in situations where control actions need to be calculated sufficiently fast (in the presence of computing power limitation or complex optimization problem) for real-time applications, it becomes useful to develop and utilize the linear hybrid model in the MPC framework (Ghosh et al. (2021)). In the present work, this can be achieved using a linear pre-trained model such as a subspace-based model of a representative unit instead of the pre-trained RNN model. However, this depends on the existence of a reliable linear pre-trained model and its superior performance when used in the hybrid approach, compared to a case where only a linear model is developed using a limited amount of data.*

Remark 12. *Note that the main focus of this work is to address the problem of developing the models when a limited amount of data is available for the other zones such as zones a and b. However, in the case where a significant amount of data is available for these zones, one can develop models trained on the full datasets to obtain better prediction accuracy.*

Remark 13. *Note that in the neural network-based transfer learning methods, a pre-trained model, which is usually a deep neural network model, developed for the source problem, is selected to be re-tuned/re-trained using the data from the new related problem. This is not being done in the present work. This is for two reasons. First, the existing pre-trained model of the source problem (representative zone), in this work, is a shallow neural network. Therefore, the transfer learning method in the sense of re-training some of the parameters (weights and biases) corresponding to some layers of the pre-trained model may not be applicable here. Second, due to the previous reason and the fact that a limited amount of data is available for the zones in question, the*

objective of this work is to propose an alternative approach, to the existing transfer-learning techniques, by integrating the shallow pre-trained RNN model with the residual models developed based on the subspace techniques. The residual models are developed in order to correct the predictions of the pre-trained model for the zones in question.

2.5 Conclusion

In this work, a hybrid machine learning approach, integrating a pre-trained (existing) RNN model with SubID-based residual models, is proposed to handle the case where a limited amount of data is available for system identification. The application of the hybrid approach to predict the behaviour of different zones in a multi-zone fitness centre is shown. To that end, the pre-trained RNN model, built using a large amount of data for one representative zone, is integrated with the SubID-based residual models, developed using a significantly smaller amount of data for the other zones. The key point is to take advantages of both the RNN model (capturing non-linearities while handling the over-fitting in the presence of a significant amount of data) and the SubID model (allowing for a more direct handling of the over-fitting issue when a limited amount of data is available). The effectiveness of the proposed hybrid approach was illustrated for a multi-zone building HVAC system, where real data are used to develop the models. The results show the superior performance of the hybrid modelling approach in handling the data shortage problem over the purely RNN and subspace-based models.

Bibliography

Afroz, Z., Shafiullah, G. M., Urmee, T., and Higgins, G. (2018). Modeling techniques used in building HVAC control systems: A review. *Renew. Sust. Energ. Rev.*, 83:64–84.

Alhajeri, M. S., Wu, Z., Rincon, D., Albalawi, F., and Christofides, P. D. (2021). Machine-learning-based state estimation and predictive control of nonlinear processes. *Chem. Eng. Res. Des.*, 167:268–280.

Anderson, J. S., McAvoy, T. J., and Hao, O. J. (2000). Use of hybrid models in wastewater systems. *Ind. Eng. Chem. Res.*, 39(6):1694–1704.

Baldi, S., Zhang, F., Le Quang, T., Endel, P., and Holub, O. (2019). Passive versus active learning in operation and adaptive maintenance of heating, ventilation, and air conditioning. *Appl. Energ.*, 252:113478.

Bhat, N. and McAvoy, T. J. (1990). Use of neural nets for dynamic modeling and control of chemical process systems. *Comput. Chem. Eng.*, 14(4-5):573–582.

Chen, Y. and Lan, L. (2010). Fault detection, diagnosis and data recovery for a real building heating/cooling billing system. *Energ. Convers. Manag.*, 51(5):1015–1024.

Chen, Y., Tong, Z., Zheng, Y., Samuelson, H., and Norford, L. (2020). Transfer learning with deep neural networks for model predictive control of HVAC and natural ventilation in smart buildings. *J Clean. Prod.*, 254:119866.

Coccia, G., Mugnini, A., Polonara, F., and Arteconi, A. (2021). Artificial-neural-network-based model predictive control to exploit energy flexibility in multi-energy systems comprising district cooling. *Energy*, 222:119958.

Deng, Z. and Chen, Q. (2021). Reinforcement learning of occupant behavior model for

cross-building transfer learning to various HVAC control systems. *Energ. Buildings*, 254:110860.

Du, Z., Fan, B., Jin, X., and Chi, J. (2014). Fault detection and diagnosis for buildings and HVAC systems using combined neural networks and subtractive clustering analysis. *Build. Environ.*, 73:1–11.

Ellis, M. J. and Chinde, V. (2020). An encoder–decoder LSTM-based EMPC framework applied to a building HVAC system. *Chem. Eng. Res. Des.*, 160:508–520.

Fang, X., Gong, G., Li, G., Chun, L., Li, W., and Peng, P. (2021). A hybrid deep transfer learning strategy for short term cross-building energy prediction. *Energy*, 215:119208.

Ferkl, L. and Široký, J. (2010). Ceiling radiant cooling: Comparison of ARMAX and subspace identification modelling methods. *Build. Environ.*, 45(1):205–212.

Ganesh, H. S., Seo, K., Fritz, H. E., Edgar, T. F., Novoselac, A., and Baldea, M. (2021). Indoor air quality and energy management in buildings using combined moving horizon estimation and model predictive control. *J. Build. Eng.*, 33:101552.

Gao, N., Shao, W., Rahaman, M. S., Zhai, J., David, K., and Salim, F. D. (2021). Transfer learning for thermal comfort prediction in multiple cities. *Build. Environ.*, 165:107725.

Garnier, A., Eynard, J., Caussanel, M., and Grieu, S. (2014). Low computational cost technique for predictive management of thermal comfort in non-residential buildings. *J. Process Contr.*, 24(6):750–762.

Ghosh, D., Hermonat, E., Mhaskar, P., Snowling, S., and Goel, R. (2019). Hybrid modeling approach integrating first-principles models with subspace identification. *Ind. Eng. Chem. Res.*, 58(30):13533–13543.

Ghosh, D., Moreira, J., and Mhaskar, P. (2021). Model predictive control embedding a parallel hybrid modeling strategy. *Ind. Eng. Chem. Res.*, 60(6):2547–2562.

Grubinger, T., Chasparis, G. C., and Natschläger, T. (2017). Generalized online transfer learning for climate control in residential buildings. *Energ. Buildings*, 139:63–71.

Hassanpour, H., Corbett, B., and Mhaskar, P. (2020a). Integrating dynamic neural network models with principal component analysis for adaptive model predictive control. *Chem. Eng. Res. Des.*, 161:26–37.

Hassanpour, H., Mhaskar, P., House, J. M., and Salsbury, T. I. (2020b). A hybrid modeling approach integrating first-principles knowledge with statistical methods for fault detection in HVAC systems. *Comput. Chem. Eng.*, 142:107022.

Huang, B., Ding, S. X., and Qin, S. J. (2005). Closed-loop subspace identification: an orthogonal projection approach. *J. Process Contr.*, 15(1):53–66.

Huang, H., Chen, L., and Hu, E. (2015). A neural network-based multi-zone modelling approach for predictive control system design in commercial buildings. *Energ. Buildings*, 97:86–97.

Khadem, S. A. and Rey, A. D. (2021). Nucleation and growth of cholesteric collagen tactoids: A time-series statistical analysis based on integration of direct numerical simulation (DNS) and long short-term memory recurrent neural network (LSTM-RNN). *J. Colloid Interfac. Sci.*, 582:859–873.

Kim, Y. J. (2020). A supervised-learning-based strategy for optimal demand response of an HVAC system in a multi-zone office building. *IEEE T. Smart Grid.*, 11(5):4212–4226.

Kusiak, A. and Xu, G. (2012). Modeling and optimization of HVAC systems using a dynamic neural network. *Energy*, 42(1):241–250.

Larimore, W. E. (1990). Canonical variate analysis in identification, filtering, and adaptive control. *In 29th IEEE Decis. Contr. P*, pages 596–604.

Li, G. and Hu, Y., . (2019). An enhanced PCA-based chiller sensor fault detection method using ensemble empirical mode decomposition based denoising. *Energ. Buildings*, 183:311–324.

Liang, J. and Du, R. (2007). Model-based fault detection and diagnosis of HVAC systems using support vector machine method. *Int. J. Refrig.*, 30(6):1104–1114.

Lu, Y., Tian, Z., Zhou, R., and Liu, W. (2021). A general transfer learning-based framework for thermal load prediction in regional energy system. *Energy*, 217:119322.

MacGregor, J. and Cinar, A. (2012). Monitoring, fault diagnosis, fault-tolerant control and optimization: data driven methods. *Comput. Chem. Eng.*, 47:111–120.

Misra, M., Yue, H. H., Qin, S. J., and Ling, C. (2002). Multivariate process monitoring and fault diagnosis by multi-scale PCA. *Comput. Chem. Eng.*, 26(9):1281–1293.

Mtibaa, F., Nguyen, K. K., Azam, M., Papachristou, A., Venne, J. S., and Cheriet, M. (2020). LSTM-based indoor air temperature prediction framework for HVAC systems in smart buildings. *Neural Comput. Appl.*, 32(23):17569–17585.

Mustafaraj, G., Chen, J., and Lowry, G. (2010). Development of room temperature and relative humidity linear parametric models for an open office using BMS data. *Energ. Buildings*, 42(3):348–356.

Nassif, N., Moujaes, S., and Zaheeruddin, M. (2008). Self-tuning dynamic models of HVAC system components. *Energ. Buildings*, 40(9):1709–1720.

Pérez-Lombard, L., Ortiz, J., and Pout, C. (2008). A review on buildings energy consumption information. *Energ. Buildings*, 40(3):394–398.

Qin, S. J. (2006). An overview of subspace identification. *Comput. Chem. Eng.*, 30(10-12):1502–1513.

Ribeiro, M., Grolinger, K., ElYamany, H. F., Higashino, W. A., and Capretz, M. A. (2018). Transfer learning with seasonal and trend adjustment for cross-building energy forecasting. *Energ. Buildings*, 165:352–363.

Ríos-Moreno, G., Trejo-Perea, M., Castañeda-Miranda, R., Hernández-Guzmán, V. M., and Herrera-Ruiz, G. (2007). Modelling temperature in intelligent buildings by means of autoregressive models. *Automat. Constr.*, 16(5):713–722.

Sadeghassadi, M., Macnab, C. J., Gopaluni, B., and Westwick, D. (2018). Application of neural networks for optimal-setpoint design and MPC control in biological wastewater treatment. *Comput. Chem. Eng.*, 115:150–160.

Satyavada, H. and Baldi, S. (2016). An integrated control-oriented modelling for HVAC performance benchmarking. *J. Build. Eng.*, 6:262–273.

Sendra-Arranz, R. and Gutiérrez, A. (2020). A long short-term memory artificial neural network to predict daily HVAC consumption in buildings. *Energ. Buildings*, 216:109952.

Shahnazari, H., Mhaskar, P., House, J. M., and Salsbury, T. I. (2018). Heating, ventilation and air conditioning systems: fault detection and isolation and safe parking. *Comput. Chem. Eng.*, 108:139–151.

Shahnazari, H., Mhaskar, P., House, J. M., and Salsbury, T. I. (2019). Modeling and fault diagnosis design for HVAC systems using recurrent neural networks. *Comput. Chem. Eng.*, 126:189–203.

Tashtoush, B., Molhim, M., and Al-Rousan, M. (2005). Dynamic model of an HVAC system for control analysis. *Energy*, 30(10):1729–1745.

Van Overschee, P. and De Moor, B. (1994). N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75–93.

Verhaegen, M. and Dewilde, P. (1992). Subspace model identification, part i: The output-error state-space model identification class of algorithms. *Int. J. Control*, 56:1187–1210.

Wang, Y., Velswamy, K., and Huang, B. (2017). A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems. *Processes*, 5(3):46.

Wu, Z., Rincon, D., and Christofides, P. D. (2020). Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. *J. Process Contr.*, 89:74–84.

Wu, Z., Tran, A., Rincon, D., and Christofides, P. D. (2019). Machine learning-based predictive control of nonlinear processes. part i: Theory. *AIChE J.*, 65:e16729.

Yang, S., Wan, M. P., Chen, W., Ng, B., and Dubey, S. (2020). Model predictive control with adaptive machine-learning-based model for building energy efficiency and comfort optimization. *Appl. Energ.*, 271:115147.

Chapter 3

A hybrid modeling approach integrating first-principles knowledge with statistical methods for fault detection in HVAC systems

The contents of this chapter have been published in the *Computers & Chemical Engineering* Journal.

- A hybrid modeling approach integrating first-principles knowledge with statistical methods for fault detection in HVAC systems. Hesam Hassanpour, Prashant Mhaskar, John M. House, and Timothy I. Salsbury, *Computers & Chemical Engineering*, 2020, 142, 107022. DOI: 10.1016/j.compchemeng.2020.107022

Abstract

This work presents a hybrid modeling technique that combines first-principles knowledge with principal component analysis (PCA) to detect faults in heating, ventilation, and air conditioning (HVAC) systems. Residuals, defined as the discrepancies between expected and observed behaviors, along with temperature measurements are used to develop multiple hybrid PCA models. Each model describes the normal behavior of the system in a particular operating state of air-handling units (AHUs). Hotelling's T^2 and square prediction error (SPE) statistics corresponding to the new observations are calculated using the hybrid PCA model in order to monitor the process and detect deviations from the expected behavior. The efficacy of the proposed approach to detect faults is evaluated and compared with two benchmark approaches: (1) residual analysis (based on first-principles models) and (2) a data-driven method (based on PCA) applied to raw temperature measurements. The superior performance of the proposed methodology, over the benchmarks, is shown via simulation tests with commonly occurring fault scenarios.

3.1 Introduction

Heating, ventilation and air conditioning (HVAC) systems, which are widely used in modern buildings to maintain thermal comfort and indoor air quality, account for more than 40% of total energy consumed in commercial and residential buildings. These systems contain components such as actuators, sensors, air dampers, valves, and controllers which are all susceptible to faults. The occurrence of various types of faults in HVAC systems can cause indoor thermal discomfort and excessive energy consumption. This has motivated research on the design of fault detection and isolation (FDI) strategies for HVAC systems to ensure efficient operations to save energy and ensure comfort. Existing studies on building FDI can be divided into three main

categories: (1) qualitative model-based methods, (2) quantitative model-based methods, and (3) process history-based methods (see Venkatasubramanian et al. (2003c), Venkatasubramanian et al. (2003a), and Venkatasubramanian et al. (2003b)).

Qualitative model-based techniques can be used for fault diagnosis when data samples are not available to develop an appropriate model because of data collection costs. A *priori* knowledge of the process (fundamental understanding of the process) is required to develop these approaches and they can be divided into two categories including rule-based FDI and qualitative physics-based FDI techniques. A large set of `if-then-else` rules and inference mechanism are commonly used in the rule-based approaches to detect fault symptoms (see e.g., House et al. (2001), Schein et al. (2006), Song et al. (2008), and Yang et al. (2008)). In the qualitative physics-based methods (see e.g., Muller et al. (2013), equations are developed by analyzing the relationships between variables in the process and fundamental behaviors of the system. The predicted qualitative values are compared with actual observations to detect faults.

In quantitative model-based methods, an explicit mathematical model of the system, developed by laws of physics that govern the process behavior, is utilized to create analytical redundancy. The analytical redundancies enable generation of residuals that capture the difference between observed and expected behaviors. Faults can be detected if the residuals breach certain thresholds (see e.g., Zhang et al. (2010) and Du and Mhaskar (2014)). Robustness of FDI filters can be improved by considering model uncertainties in computing residuals. In addition, the efficacy of the models used in these approaches, is required to be validated for both normal and faulty operating conditions. Several studies utilized these techniques in order to design FDI strategies for HVAC systems (see e.g., Seem and House (2009), Thumati et al. (2011), Provan (2011), and O'Neill et al. (2014)). In Seem and House (2009), first-principles knowledge is utilized to derive model-based residuals, assuming that steady-state conditions are imposed by the sequencing control logic. A detailed physics-based model

is used in Thumati et al. (2011) to diagnose faults in HVAC systems using generated residuals. Basic thermodynamic principles are used in Provan (2011) to develop a reduced-order hybrid system for HVAC systems and the model is transformed to generate a diagnosis model. Because the quantitative model-based FDI approaches are developed based on knowledge of the physical principles, the physical impacts of different types of faults can be ascertained. However, developing and maintaining first-principles models remains challenging. This has led to efforts to devise data-driven FDI frameworks.

In process history-based methods, only the availability of a considerable amount of historical data is needed to model the process, in contrast to the qualitative and quantitative model-based methods, where a *priori* knowledge of the process is required. These methods, known as data-driven or machine learning techniques, have received significant attention due to the existing challenges for developing and maintaining first-principles models and increasing the availability of data. The use of these approaches has also been increasing significantly to design FDI schemes for HVAC systems. In this direction, several techniques such as subspace identification methods (see e.g., Qin (2006) and Shahnazari et al. (2018)), multivariate statistical-based approaches such as principal component analysis (PCA) (see e.g., Du et al. (2007), and Li and Wen (2014)), artificial neural networks (ANNs) (see e.g., Fan et al. (2010) and Shahnazari et al. (2019)), and support vector machines (SVM) (see e.g., Liang and Du (2007) and Ren et al. (2008)) have been used to develop models and FDI methodologies for HVAC systems.

Among many machine learning approaches, principal component analysis (PCA) has been widely employed for process monitoring and fault detection and diagnosis (see e.g., Joe Qin (2003), MacGregor and Cinar (2012), Du and Du (2018), and Qin and Chiang (2019)). This multivariate statistical technique has also been used to develop FDI strategies for HVAC systems. In Chen and Lan (2010), PCA is employed to

extract the correlation of measured variables and reduce the dimension of measured data in building heating/cooling system. Square prediction error (SPE) statistic is then used in order to detect sensor faults in the system. In Wang and Xiao (2004b) and Wang and Xiao (2004a), PCA-based strategies are developed to detect and isolate sensor faults in air handling units (AHUs) and the SPE statistic and SPE contribution plot supplemented by expert rules are then used to detect and isolate the faults. In Wang and Qin (2005), a PCA-based approach is proposed for flow sensor fault detection and validation of variable air volume (VAV) terminals. Sensor faults are detected using both the Hotelling's T^2 and the SPE statistics. PCA-based strategies have been developed to detect faults in HVAC chiller systems (see e.g., Beghi et al. (2016) and Li and Hu (2019)). In Du et al. (2017), a combination of subtractive clustering and PCA is used to assess the reliability of sensor measurements and detect faults in a vapor compression system.

The idea of hybrid modeling methods, those that combine various modeling techniques to enhance the performance of an individual approach, has been gaining attention in recent years (see e.g., Hosen et al. (2011) and Ghosh et al. (2019)). Hybrid-based FDI strategies have also been developed for HVAC systems to improve diagnostic performance. In Wang and Cui (2006), a qualitative model is integrated with a PCA model to diagnose faults in centrifugal chillers. In Liang and Du (2007), a first-principles model is combined with a support vector machine (SVM) classifier to diagnose faults in HVAC systems. The first-principles model is also integrated with the PCA technique in Wu and Sun (2011) to develop an FDI strategy to detect faults in HVAC units such as AHU and VAV systems. PCA is employed for dimensionality reduction and the energy consumption is analyzed to detect faults in VAV systems. In situations where both the residual information (obtained using first-principles knowledge) and sufficient process data are available, it becomes useful to synergize the residuals directly with measurements through an appropriate data-driven fault detection mechanism; this idea has not been used for fault detection.

Motivated by the above considerations, the present manuscript addresses the problem of fault detection in HVAC systems by integrating first-principles knowledge with a statistical data-driven PCA approach. The rest of the manuscript is organized as follows: In Section 3.2, a brief description of the simulation test bed, air handling unit and sequencing control strategy, is presented. The method of the residual analysis (developed using first-principles knowledge) proposed in Seem and House (2009) is then reviewed. Note that while the method presents residuals that can be used for fault detection, the focus in the previous work was not on developing a fault detection methodology, but on providing qualitative results indicating what residuals were most affected by each fault. Subsequently, a purely data-driven FDI method using PCA on measured temperatures in different locations of the AHU is implemented and considered as a benchmark for comparison. In Section 3.3, the proposed hybrid approach that integrates first-principles knowledge with principal component analysis is presented. The efficacy of the proposed method, compared with the residual and data-driven based techniques, is then illustrated via simulation tests with commonly occurring fault scenarios. Finally, concluding remarks are presented in Section 6.5.

3.2 Preliminaries

3.2.1 Air handling unit system description

Air-handling units (AHUs) in HVAC systems are used to condition and distribute air within a building. A schematic diagram of a single duct AHU is shown in Fig. 3.1. Fresh air enters the AHU through the outdoor air damper and is mixed with recirculated air passing through the recirculation air damper. Mixed air is drawn by the supply fan through the heating and cooling coils to be conditioned before being distributed to the building. The recirculated air, drawn from the building by the

return fan, may be either exhausted or recirculated depending on the position of the mixing-box dampers (recirculation, exhaust, and outdoor air dampers). The airflow is also controlled by the supply and return fans. Feedback control is used in variable-air-volume (VAV) AHUs to manipulate mixing-box dampers, heating coil valve, and cooling coil valve to satisfy the supply air temperature set-point.

Sequencing control strategy

Sequencing control logic is used in an AHU to choose the proper components in order to control the supply air temperature at any given time (see Seem et al. (1999)). To control the supply air temperature, an AHU uses four operating states. The active operating state is determined by the sequencing control logic which is illustrated by a state transition diagram in Fig. 3.2. The conditions that must be satisfied to move from one state to another are specified next to each arrow.

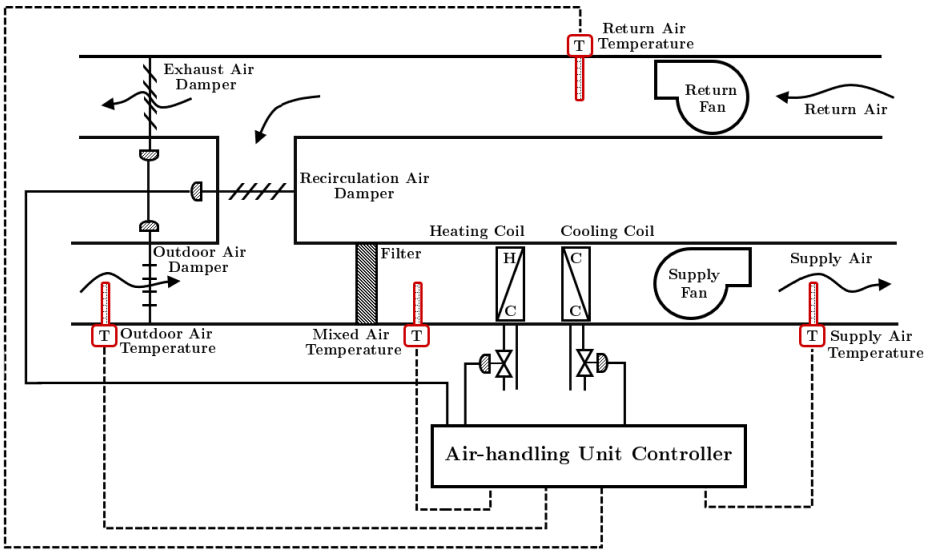


Figure 3.1: Schematic of a single-duct air-handling unit.

In State 1, the heating coil valve is modulated to maintain the supply air temperature at the set-point for cold outdoor air. The cooling coil valve is closed, and the mixing-

box dampers are positioned to allow the minimum outdoor air required for ventilation. The transition to State 2 occurs after the control signal for the heating coil valve is saturated at the no-heating position. Note that a control signal is considered saturated at a specific position when it has been at that position continuously for a predefined length of time. In State 2, the mixing-box dampers are modulated to satisfy the set-point value for the supply air temperature and both the heating and cooling coil valves are closed. After the control signal for the dampers is saturated at the minimum outdoor air position, transition to State 1 occurs. Transition to State 3 occurs after the control signal for the dampers is saturated at the maximum outdoor air position. In State 3, the chilled water flow rate to the cooling coil is modulated to achieve the supply air temperature set-point. The heating coil valve is closed, and the mixing-box dampers are positioned to allow the maximum outdoor air to enter (outdoor air damper is fully open). The transition to State 2 occurs after the control signal for the cooling coil valve is saturated at the no-cooling position. In addition, when the outdoor air temperature is greater than the changeover temperature plus the deadband temperature, transition to State 4 occurs. In State 4, the operating condition is the same as State 3, except that the mixing-box dampers are positioned to provide the minimum outdoor air needed for ventilation. Transition to State 3 occurs when the outdoor air temperature is less than the changeover temperature. As illustrated in Fig. 3.2, the values of the changeover temperature and deadband must be specified to define the conditions necessary for transitions between States 3 and 4. The changeover temperature is usually considered equal to the return air temperature and the deadband is around 1 °F (0.56 °C). Note that a deadband is used to avoid frequent transitions between States 3 and 4 due to noisy sensor readings.

The active operating states of the AHU, determined by the sequencing control logic and depending on the prevailing weather conditions, are shown in Fig. 3.3 for one year of normal operation and can be listed as: (1) State 1 (S_1), (2) State 2 (S_2), (3) State 3 (S_3), (4) State 4 (S_4), (5) Transitions from State 1 to State 2 ($S_{1\rightarrow 2}$), (6)

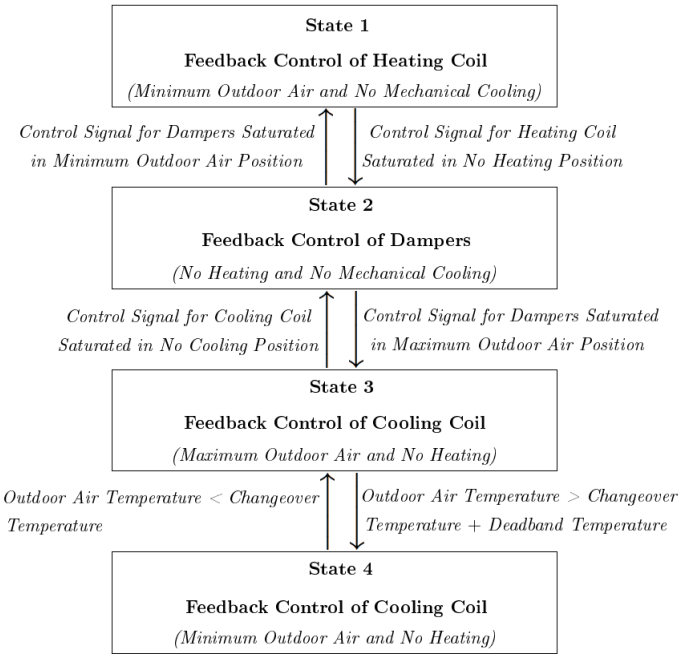


Figure 3.2: State transition diagram for the sequencing control strategy to control air-handling units.

Transitions from State 2 to State 1 ($S_{2\to1}$), (7) Transitions from State 2 to State 3 ($S_{2\to3}$), (8) Transitions from State 3 to State 2 ($S_{3\to2}$), (9) Transitions from State 3 to State 4 ($S_{3\to4}$), and (10) Transitions from State 4 to State 3 ($S_{4\to3}$). An arrow indicates the transition from one state to another.

In addition, temperatures in different locations of the AHU are shown in Fig. 3.4. Note that the supply air temperature set-point is fixed at 55 °F (12.78 °C), for April 1 to October 31, and 60 °F (15.56 °C) for the remaining months of the year. It should also be noted that the data in Figs. 3.3 and 3.4 are simulated for Chicago climate conditions.

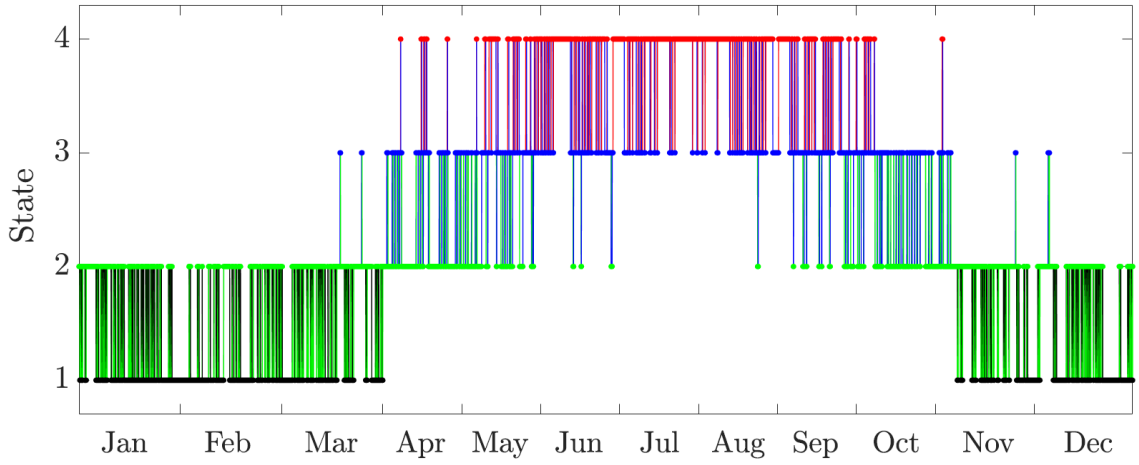


Figure 3.3: Active operating states of the air-handling unit (AHU) in the normal conditions.

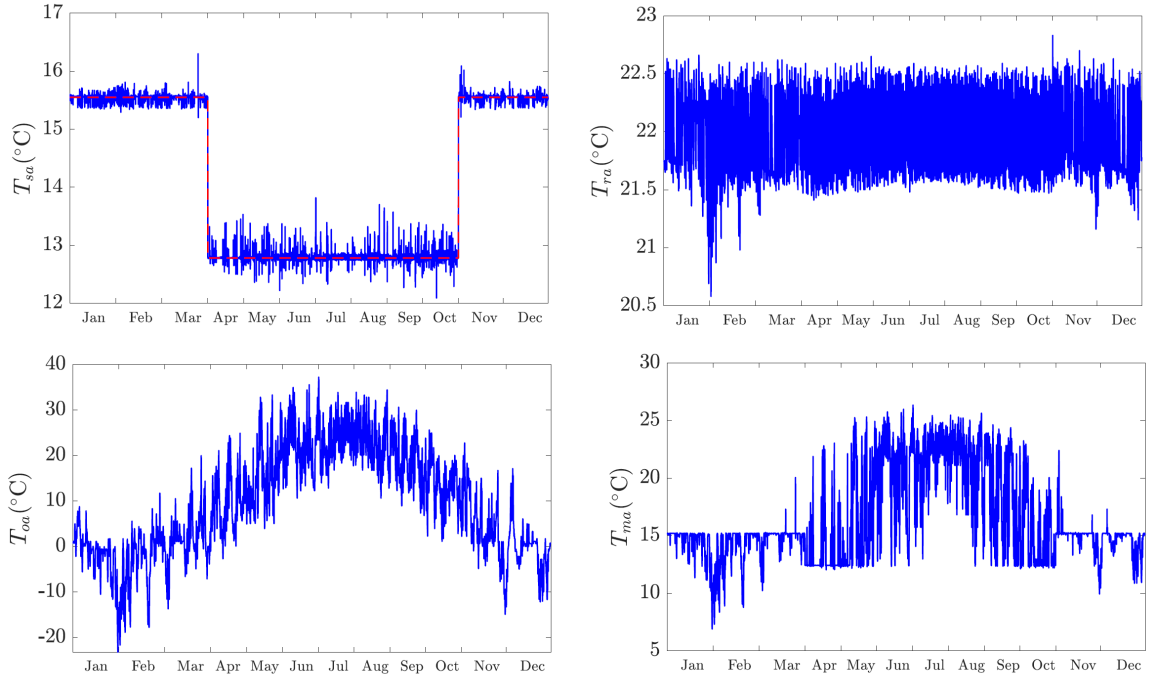


Figure 3.4: Supply air temperature (T_{sa}) and its set-point (dashed line), return air temperature (T_{ra}), outdoor air temperature (T_{oa}), and mixed air temperature (T_{ma}) for the normal operating conditions.

3.2.2 Review of the model-based residuals (first-principles knowledge)

In this section, we review the methodology presented in Seem and House (2009) for developing the residuals that can be used for fault detection of AHUs. Note that the residual information is one of the components of the hybrid approach proposed for fault detection in Section 3.3. Recall that the supply, return, outdoor, and mixed air temperatures are measured using sensors as shown in Fig. 3.1. Model-based residuals are developed assuming that steady-state conditions prevail at specific times and locations in the system due to well-controlled conditions. Mass and energy balances are performed over appropriate control volumes, depending on each particular operating state of the AHU. Finally, 13 residual models are obtained which are all limited to certain operating states (either within a certain state or at a certain transition) in the AHU. The summary of all 13 residuals and the conditions (operational states or transitions) where they are applicable is presented in Table 3.1. To evaluate the approach, various faults including supply air, return air, mixed air, and outdoor air temperature sensor faults, stuck or leaking heating and cooling coil valve faults, and stuck or leaking recirculation damper faults were simulated (see Seem and House (2009) for further details).

The fault detection results for two illustrative cases are discussed: (1) Return air temperature sensor offset of 2 °C (Fault 1) and (2) Mixed air temperature sensor offset of 2 °C (Fault 2) . These temperature sensor faults are introduced by linearly increasing the offsets from 0 °C to 2 °C over a three-month period. A positive offset causes an artificially low sensor reading. Thus, if the actual temperature is α °C and the value of offset is 2 °C, the sensed temperature is $(\alpha - 2)$ °C. Note that these are sensor faults for variables that are applicable in all operating states. The values of all 13 residuals were calculated. As shown in Seem and House (2009), the residual r_9

Table 3.1: Summary of the residuals and the conditions (states and transitions) when they are applicable

Residual	**Conditions when Applicable**
$r_1 = T_{s,2\to3} - T_{o,2\to3} - \frac{\hat{\dot{W}}_{fan}}{\hat{\dot{m}}_s \hat{c}_p}$	Transitions from State 2 to State 3 ($S_{2\to3}$)
$r_2 = T_{s,3\to2} - T_{o,3\to2} - \frac{\hat{\dot{W}}_{fan}}{\hat{\dot{m}}_s \hat{c}_p}$	Transitions from State 3 to State 2 ($S_{3\to2}$)
$r_3 = f_{design} - \frac{T_{s,1\to2} - T_{r,1\to2} - \frac{\hat{\dot{W}}_{fan}}{\hat{\dot{m}}_s \hat{c}_p}}{T_{o,1\to2} - T_{r,1\to2}}$	Transitions from State 1 to State 2 ($S_{1\to2}$)
$r_4 = f_{design} - \frac{T_{s,2\to1} - T_{r,2\to1} - \frac{\hat{\dot{W}}_{fan}}{\hat{\dot{m}}_s \hat{c}_p}}{T_{o,2\to1} - T_{r,2\to1}}$	Transitions from State 2 to State 1 ($S_{2\to1}$)
$r_5 = f_{design} - \frac{T_{m,1} - T_{r,1}}{T_{o,1} - T_{r,1}}$	State 1 (S_1)
$r_6 = T_{s,2} - T_{m,2} - \frac{\hat{\dot{W}}_{fan}}{\hat{\dot{m}}_s \hat{c}_p}$	State 2 (S_2)
$r_7 = 1 - \frac{T_{m,3} - T_{r,3}}{T_{o,3} - T_{r,3}}$	State 3 (S_3)
$r_8 = T_{o,3} - T_{m,3}$	State 3 (S_3)
$r_9 = f_{design} - \frac{T_{m,4} - T_{r,4}}{T_{o,4} - T_{r,4}}$	State 4 (S_4)
$r_{10} = T_{s,2\to3} - T_{m,2\to3} - \frac{\hat{\dot{W}}_{fan}}{\hat{\dot{m}}_s \hat{c}_p}$	Transitions from State 2 to State 3 ($S_{2\to3}$)
$r_{11} = T_{o,2\to3} - T_{m,2\to3}$	Transitions from State 2 to State 3 ($S_{2\to3}$)
$r_{12} = T_{s,3\to2} - T_{m,3\to2} - \frac{\hat{\dot{W}}_{fan}}{\hat{\dot{m}}_s \hat{c}_p}$	Transitions from State 3 to State 2 ($S_{3\to2}$)
$r_{13} = T_{o,3\to2} - T_{m,3\to2}$	Transitions from State 3 to State 2 ($S_{3\to2}$)

(applicable in S_4) is impacted by Fault 1 and the residuals r_6 (in S_2), r_7 and r_8 (in S_3), r_9 (in S_4), r_{10} and r_{11} (in $S_{2\to3}$), and r_{12} and r_{13} (in $S_{3\to2}$) are impacted by Fault 2. Therefore, Fault 1 is detectable in S_4 and Fault 2 is detectable in S_2, S_3, S_4, $S_{2\to3}$, and $S_{3\to2}$ using the residual analysis (see also, Table 3.3).

In the rest of the manuscript, we will utilize Fault 1 in State 1 and Fault 2 in State 2 for the purpose of comparison. To understand the residual definitions better, consider the mathematical representation of the residual r_5, applicable in State 1 (S_1) given by Eq. (3.1):

$$r_5 = f_{design} - \frac{T_{m,1} - T_{r,1}}{T_{o,1} - T_{r,1}} \tag{3.1}$$

where $T_{m,1}$, $T_{r,1}$, and $T_{o,1}$ are mixed, return, and outdoor air temperatures in S_1 and f_{design} is the estimated (design) value of outdoor air fraction. In addition, the residual r_6, that can be used in State 2 (S_2), is computed using Eq. (3.2):

$$r_6 = T_{s,2} - T_{m,2} - \frac{\hat{\dot{W}}_{fan}}{\hat{\dot{m}}_s \hat{c}_p} \tag{3.2}$$

where $T_{s,2}$ and $T_{m,2}$ are supply and mixed air temperatures in S_2, respectively, $\hat{\dot{W}}_{fan}$ is the design value of supply fan power, $\hat{\dot{m}}_s$ is the design value of mass flow rate of supply air, and $\hat{c}_p$ is the design value of specific heat of the moist air mixture at constant pressure. The design values of the parameters in Eq. (3.1) and Eq. (3.2) were determined through numerical experiments and are listed in Table 3.2.

Table 3.2: Design values of parameters in the model-based residuals

Parameter	Value
f_{design}	0.3
$\hat{\dot{W}}_{fan}$	7.14 kW
$\hat{\dot{m}}_s$	10.53 kg/s
$\hat{c}_p$	1.02 kJ/(kg °C)

Remark 1. *Note that the testbed is simulated based on established component and system models (see Norford and Haves (1997), Haves et al. (1998), and DeSimone (1995)). Based on idealized flow relationships of a single-duct VAV AHU and the zones it serves, the models are implemented in HVACSIM+ (see Park et al. (1985)). Significant changes to the simulation testbed, described in detail by Norford and Haves (1997), are outlined in Seem and House (2009). All simulations are implemented for one year of operation under either normal or faulty conditions.*

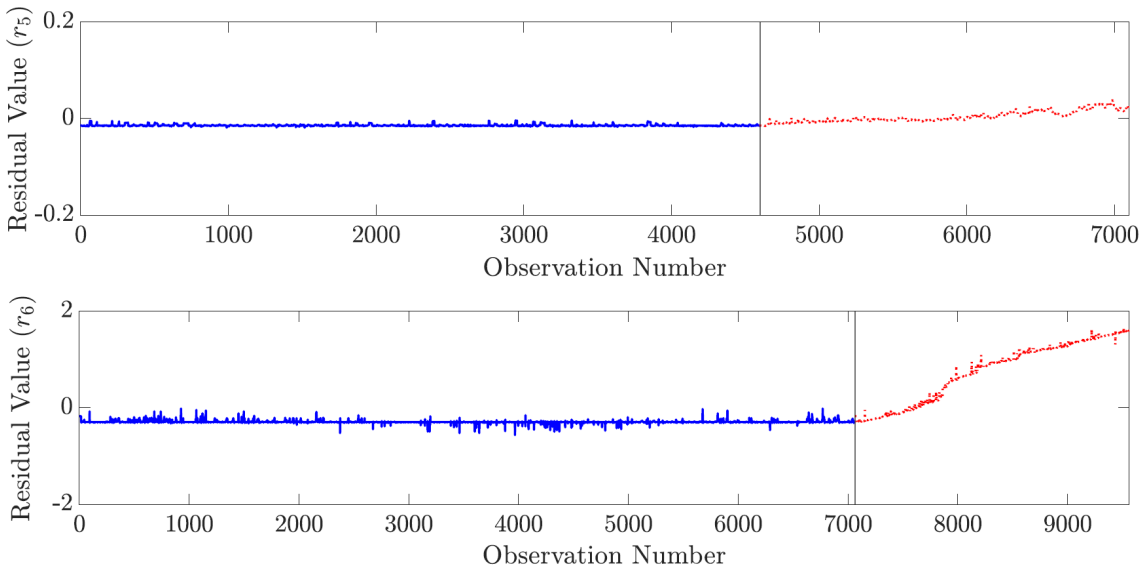


Figure 3.5: Residuals for the normal (solid blue lines) and faulty (dotted red lines) operating conditions: Residual r_5 when Fault 1 occurs (top) and Residual r_6 when Fault 2 occurs (bottom). The vertical gray line separates normal and faulty data.

Application of the model-based residuals

In order to assess the capability of the model-based residuals (r_5 and r_6) to detect Faults 1 and 2, respectively in S_1 and S_2, the values of these residuals are computed for normal and faulty conditions as shown in Fig. 3.5. It should be noted that the observations are not contiguous in time because the values are only plotted when the system is operating in the state where each residual is applicable. For this reason, the number of observations for the residual r_6 (in S_2) in the normal condition is greater than those for the residual r_5 (in S_1), due to the fact that the AHU was operating in S_2 more than S_1 (see, also Fig. 3.3). The expected values for the residuals r_5 and r_6 in the normal operations are calculated using Eq.(3.1) and Eq. (3.2), respectively, and shown in Fig. 3.5. Note that the residual r_5 is unitless, while the residual r_6 has units of temperature (°C in this case).

From Eq. (3.1), the residual r_5 is the difference between the design minimum fraction

of outdoor air, f_{design}, and the difference between the mixed air and return air temperatures, $T_{m,1} - T_{r,1}$, divided by the difference between the outdoor air and return air temperatures, $T_{o,1} - T_{r,1}$. Recall that in S_1 the minimum amount of outdoor air that is needed for ventilation is used (cold outdoor conditions). Given that $T_{o,1} < T_{r,1}$, from an energy balance it can be concluded that $T_{m,1} < T_{r,1}$. Thus both numerator and denominator values in the second term of this equation have negative values. In the presence of Fault 1 (return air temperature sensor offset of 2 °C), that results in the lower sensor reading, $T_{r,1} - \beta$, where β is the offset value, changing from 0 °C to 2 °C gradually over time, the value of second term in Eq. (3.1) becomes less than the nominal value considering that $T_{m,1} - T_{r,1} > T_{o,1} - T_{r,1}$ and $\beta < \mid T_{o,1} - T_{r,1} \mid$. Subtracting this value from f_{design} results in increasing the value of residual r_5 in the presence of Fault 1 (the magnitude of the increase of the residual depending on the offset value). Similarly, Eq. (3.2) reveals that the difference between the supply and mixed air temperatures should be equal to the temperature rise caused by heat from the fan in S_2 for the normal conditions. Recall that the mixing-box dampers are modulated in S_2 to control the supply air temperature. In the presence of Fault 2 (mixed air temperature sensor offset of 2 °C), the lower sensor reading, $T_{m,2} - \beta$, results in increasing the value of the residual r_6.

To evaluate the residual ability to detect these faults, the observed values of the residuals are calculated in the presence of faults and shown in Fig. 3.5. As can be seen, while there is some increase in the value of r_5, this increase is not significant compared to the normal conditions. In contrast, the difference between the normal and faulty values of the residual r_6 can be readily observed over time. Thus, from a qualitative standpoint, it can be concluded that Fault 1 is not detectable in S_1, but Fault 2 can be detected in S_2 using the residual analysis. A key recognition that must be emphasized here is that it is possible to change the definition of r_5 to make it more sensitive to Fault 1, and thus be able to detect it. The contribution of the present manuscript is to show how residuals defined on the basis of first-principles knowledge

can be supplemented with data-driven approaches to detect faults without having to re-calibrate the first-principles residuals when, for instance, they become less sensitive to certain faults due to changes in the operating conditions.

3.2.3 Fault detection with statistical approaches

In this section, the objective is to utilize Hotelling's T^2 and square prediction error (SPE) charts to detect and monitor abnormal behaviors of the process (see e.g., Yoon and MacGregor (2000), Yoon and MacGregor (2001), Kourti (2005), and MacGregor and Cinar (2012)). Principal component analysis (PCA) is first employed to not only reduce the dimensionality of the data space, but also address multi-collinearity among process variables. The PCA technique projects the data samples onto the new orthogonal variable space of principal components (PCs) or latent variables space, which are sorted such that the first few explain the highest possible amount of variation in the original space of real variables. The mathematical formulation of PCA model is defined as:

$$X = TP^T + E \tag{3.3}$$

where X is an $(N \times K)$ standardized matrix containing K process variables (columns) and N observed values (rows), T is an $(N \times A)$ matrix of latent variable scores (new mutually uncorrelated variables), and P is a $(K \times A)$ matrix of orthogonal loading vectors that determines the orientation of the latent space. Note that the loading vectors are the eigenvectors of the covariance matrix of X $(X^TX/(N-1))$ relevant to the A largest eigenvalues $(A < K)$. In Eq. (6.8), the first term (TP^T) explains the modeled variations of X and the second term (E) defines the residual or the unmodeled variations of X.

In order to develop a PCA model, the data matrix of N observations on K variables is first mean centered and scaled to obtain zero mean and unit variance in each column of preprocessed matrix X. Next, orthogonalization techniques such as singular value decomposition (SVD) or nonlinear iterative partial least squares (NIPALS) are applied to calculate the principal component loading vectors (SVD is employed in this work to calculate principal components). There also exist several criteria for choosing the number of PCs. To this end, cumulative percent variance (CPV) (see e.g., Li et al. (2000)) or cross validation techniques (see e.g., Wold (1978) and Eastment and Krzanowski (1982)) can be performed to determine the number of PCs. After computing the loading vectors, the score matrix that contains the values of scores corresponding to each observation is computed as follows:

$$T = XP \tag{3.4}$$

Having obtained the PCA model (loading vectors) describing the normal condition of the process, the Hotelling's T^2 and SPE statistics can be calculated for any new observation to detect an abnormal behavior that may occur throughout the process. The value of Hotelling's T^2 statistic for the i^{th} observation is calculated by Eq. (3.5):

$$T_i^2 = \sum_{a=1}^{A} \left(\frac{t_{i,a}}{s_a} \right)^2 \tag{3.5}$$

where A is the number of PCs, s_a^2 is the variance of the corresponding component (latent variable). Assuming that data samples follow a multivariate normal distribution (see e.g., Jackson and Mudholkar (1979), Kourti and MacGregor (1996), and Joe Qin (2003)), the Hotelling's T^2 statistic is computed using Eq. (3.6):

$$T^2_{UCL} = \frac{(N-1)(N+1)A}{N(N-A)} F_{\alpha(A,N-A)} \tag{3.6}$$

where $F_{\alpha(A,N-A)}$ denotes the upper $100\alpha\%$ critical point of the F-distribution with $(A, N-A)$ degrees of freedom. The SPE statistic for i^{th} observation determines the residual distance between the sample and its projection onto the model plane (PCs subspace) and is given by:

$$SPE_i = \tilde{x}_i^T \tilde{x}_i = x_i^T (I - PP^T) x_i \tag{3.7}$$

where $\tilde{x}$ is the residual vector $(x_i - \hat{x}_i)$, and $\hat{x}_i$ is the reconstructed vector of the observation. The upper control limit for the SPE statistic with a significance level α is calculated as follows (see Jackson and Mudholkar (1979)):

$$SPE_{UCL} = \theta_1 \left(\frac{c_\alpha \sqrt{2\theta_2 h_0^2}}{\theta_1} + \frac{\theta_2 h_0 (h_0 - 1)}{\theta_1^2} + 1 \right)^{1/h_0} \tag{3.8}$$

where c_α is the normal deviate relevant to the upper $(1-\alpha)$ percentile. θ_i (i = 1, 2, 3) and h_0 are given by the following equations:

$$\theta_i = \sum_{j=A+1}^{K} \lambda_j^i \tag{3.9}$$

$$h_0 = 1 - \frac{2\theta_1 \theta_3}{3\theta_2^2} \tag{3.10}$$

where λ_j is the j^{th} eigenvalue corresponding to the covariance matrix in descending order.

The Hotelling's T^2 represents the distance from the center of the model plane to the projection of a data point onto the model plane, while SPE denotes the perpendicular distance between the data point and its projection onto the model plane. The possible deviations from the normal correlation existing in the process variables can be captured by measuring these two variables. Hence, they are commonly used for fault detection and diagnosis.

Statistical model-based fault detection design

The statistical model-based fault detection strategy is designed in this section. The PCA models, developed using historical data for normal operating conditions, are used to detect future faults that may occur in the system. The procedure for fault detection can be summarized as follows:

1. Standardizing training data and building the PCA model:

   (a) The normal data samples (training data) for temperature measurements in different locations of the AHU, which are supply air temperature (T_{sa}), return air temperature (T_{ra}), outdoor air temperature (T_{oa}), and mixed air temperature (T_{ma}), are first segregated into 8 categories, corresponding to different operational states or state transitions for which the residual models are developed. These 8 operating conditions are S_1, S_2, S_3, S_4, $S_{1\to2}$, $S_{2\to1}$, $S_{2\to3}$, and $S_{3\to2}$ (see Table. 3.1).

   (b) Each data category is mean-centered and scaled to unit variance to develop the PCA model for that particular state.

   (c) Next, the upper control limits for Hotelling's T^2 and SPE statistics are determined with 99% control limits.

2. Monitoring new observations (fault detection):

(a) A new observation is then measured online and standardized (centered and scaled). Note that the mean and standard deviation values of the training data (used to construct PCA model for normal conditions) are used to standardize test sets.

(b) The mean centered and scaled data are projected onto the existing PCA model (projection of the new observation onto the principle component and residual subspaces).

(c) The Hotelling's T^2 and SPE statistics of the new observation are computed.

(d) Finally, the score plot, Hotelling's T^2, and SPE values of the new observation are analyzed to detect an abnormal condition. A fault is indicated if either the Hotelling's T^2 or SPE statistic (or both) exceeds its corresponding upper control limit.

A flow chart depicting the steps for the data-driven fault detection approach (without considering the steps required for the hybrid approach which are colored by blue) is shown in Fig. 3.6.

Remark 2. *Note that choosing the number of principal components (PCs) in a PCA model is an important issue to retain as much original variance as possible. In this work, cumulative percent variance (CPV), one of the popularly used methods, is used in order to determine the number of PCs. In CPV, the number of PCs is chosen such that the variance, explained by the PCA model, reaches a predetermined percentage (e.g., 75% in this paper). In addition, in order to determine the number of PCs, it is considered that the number of PCs (A) does not exceed half of the number of variables (K/2), but this condition is considered only if the previous condition is satisfied.*

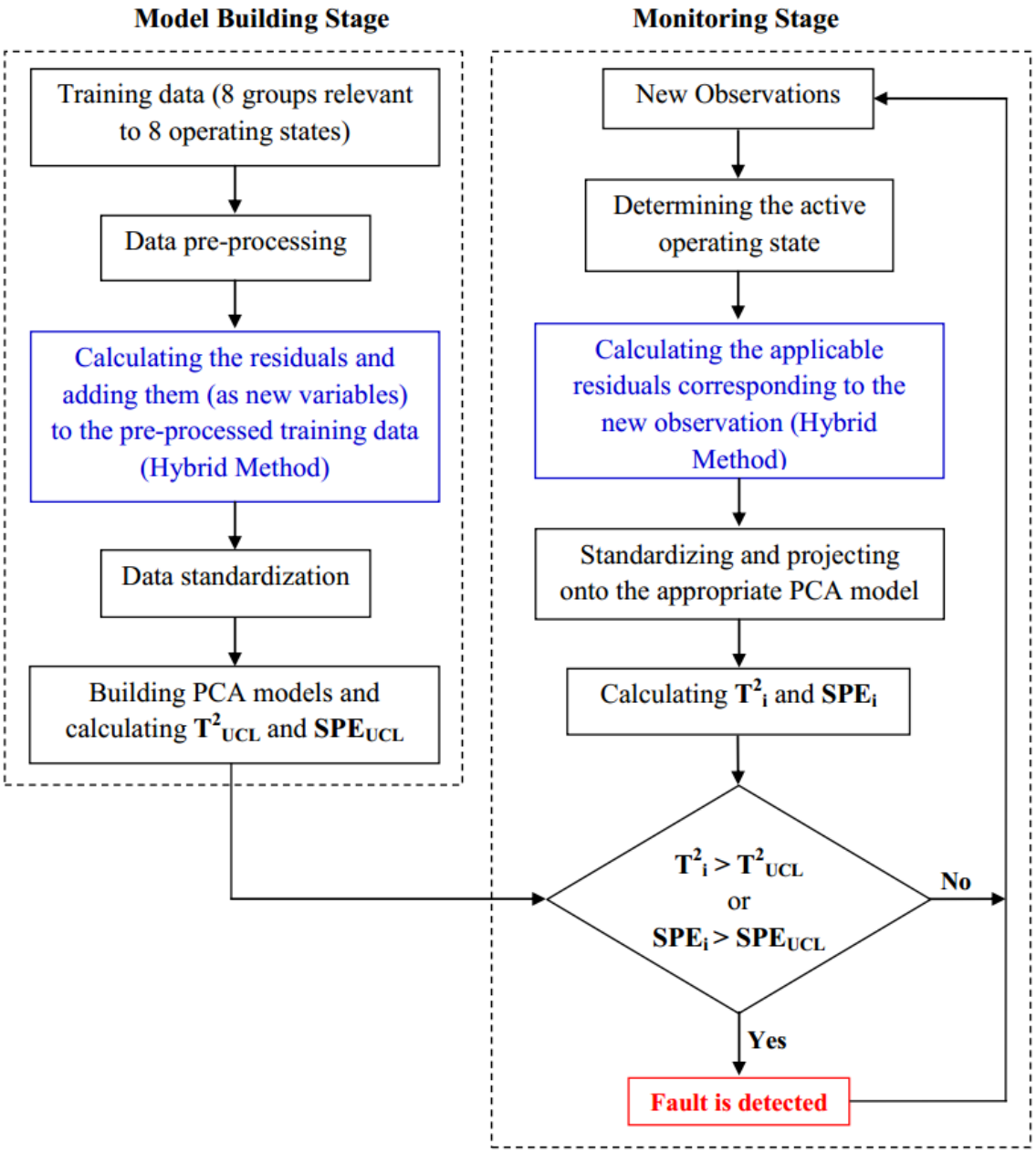


Figure 3.6: Outline of the data-driven and hybrid fault detection strategies.

Application of statistical model-based fault detection design

In this section, the statistical model-based approach described in Section 3.2.3 is applied to the benchmarks, mentioned earlier, to assess the capability of the data-driven

method to detect faults. The normal data samples corresponding to the temperature measurements in different locations of the AHU, as shown in Fig. 3.4, are divided into 8 categories depending on the operating state of the AHU. Then the data category corresponding to State 1 (S_1) is used to build the PCA model. As shown in Fig. 3.7 (left half), when the number of PCs is two, the CPV value exceeds the predetermined percentage (75%), and therefore, two principal components are considered to build the PCA model resulting in 82% variance explanation. Note that the data set related to the S_1 contains 4590 observations for one year of normal operation. Having obtained the PCA model for the normal conditions in S_1, the procedure explained above is applied to the new observations (test set) corresponding to Fault 1. The test set contains the first 2500 faulty observations in January, February, and March. Fig. 3.8 compares the score values relevant to normal conditions with those obtained when Fault 1 occurred. As can be seen, there are many overlaps between normal and faulty data samples, making it difficult to distinguish between normal and faulty observations.

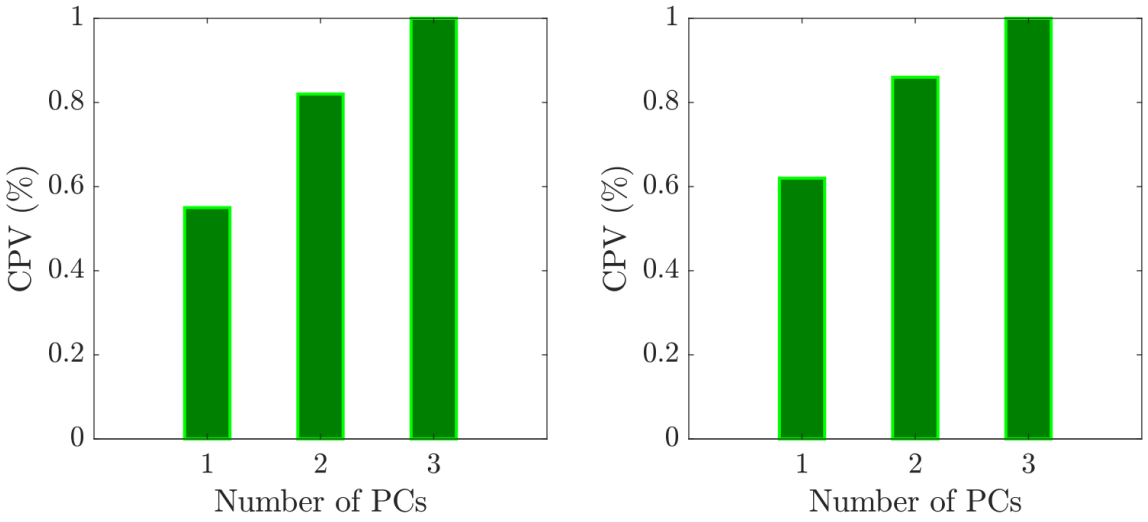


Figure 3.7: Plot of the CPV values versus the number of PCs to train the PCA model (in the data-driven approach) for State 1 (left half) and State 2 (right half) of the AHU.

In addition, the statistics Hotelling's T^2 and SPE are evaluated in Fig 3.9. Although the Hotelling's T^2 values for many observations go beyond the control limit (especially

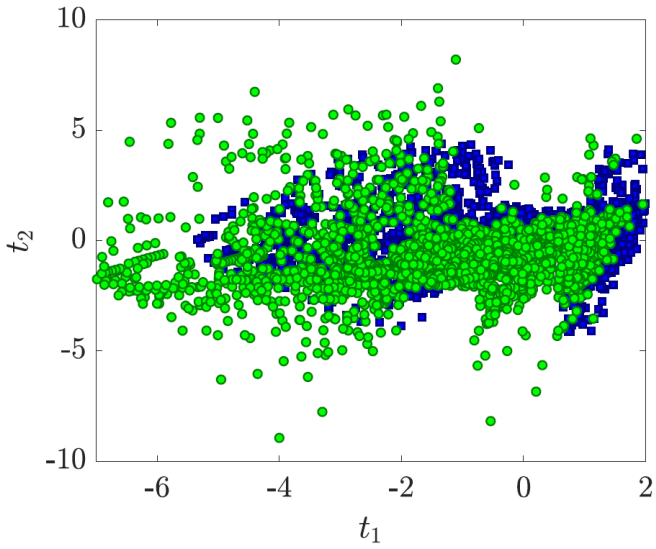


Figure 3.8: Comparison of score values for the normal (blue squares) and faulty (green circles) conditions in the presence of Fault 1 (return air temperature sensor offset) obtained using the data-driven method (PCA on temperatures).

those that are approximately between the 1000^{th} and 1500^{th} observations after the fault occurrence), most of them are below the control limit, which makes it difficult to identify faulty data samples. On the other hand, the SPE values exceed the control limit at approximately the 1600^{th} observation after the fault occurrence, resulting in detection of approximately 36.0% of faulty data samples. Recall that the bias faults are introduced gradually over a three-month period from 0 °C to 2 °C (final bias value), but the present analysis focuses on January, February, and March for all the approaches to evaluate the ability to detect incipient faults.

In order to detect Fault 2 in State 2 (S_2), the PCA model is first developed using the normal data samples related to the second operating state of the AHU. Fig. 3.7 (right half) shows that the CPV value exceeds the predetermined percentage when the number of PCs is two. Thus, two principal components, resulting in 86% variance explanation, are considered to construct the PCA model for S_2. The steps mentioned in the previous section are then applied for fault detection. It should be mentioned

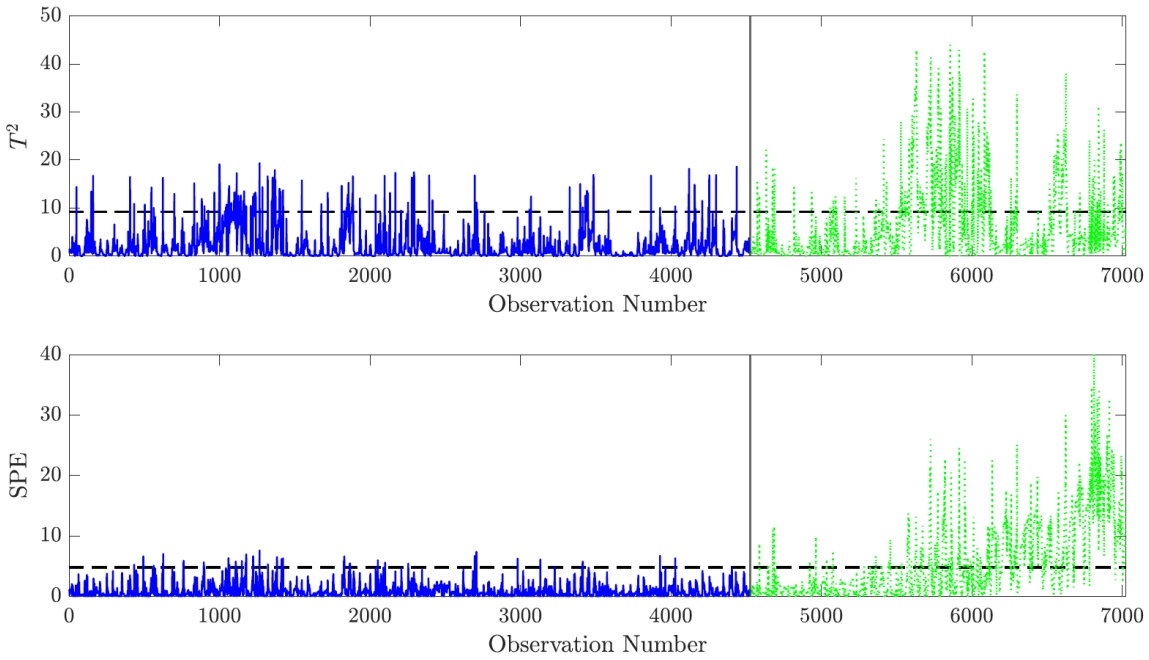


Figure 3.9: Comparison of Hotelling's T^2 (top) and SPE values (bottom) for the normal and faulty conditions (in the presence of Fault 1) obtained using the data-driven method: Training data (solid blue lines) and observations (dotted green lines) corresponding to Fault 1. The vertical gray line separates normal and faulty data.

that the normal training data used to build the PCA model for AHU in S_2 contains 7060 observations in one year of operation and the test set contains the first 2500 faulty observations for the months of January, February, and March. The score values for the normal and faulty conditions are shown in Fig. 3.10. It is not possible to distinguish faulty samples from normal ones due to the overlaps in the observations. Fig. 3.11 shows the Hotelling's T^2 and SPE values corresponding to the normal and faulty conditions. As shown in the Hotelling's T^2 plot, the variation of the projected values of the faulty observations onto the principal component subspace is masked by the variation of observations corresponding to the normal conditions (training data), and therefore, this fault is not detectable using Hotelling's T^2 statistic. A similar scenario can be observed in the variability of data samples in the residual subspace using the SPE statistic.

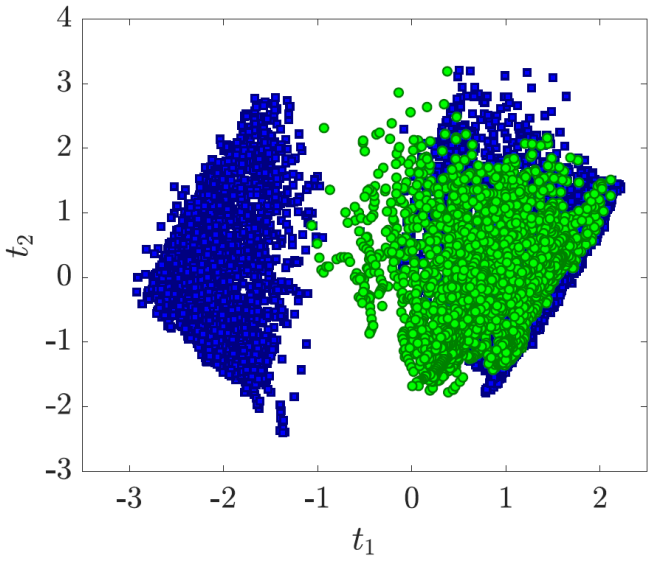


Figure 3.10: Comparison of score values for the normal (blue squares) and faulty (green circles) conditions in the presence of Fault 2 (mixed air temperature sensor offset) obtained using the data-driven method (PCA on temperatures).

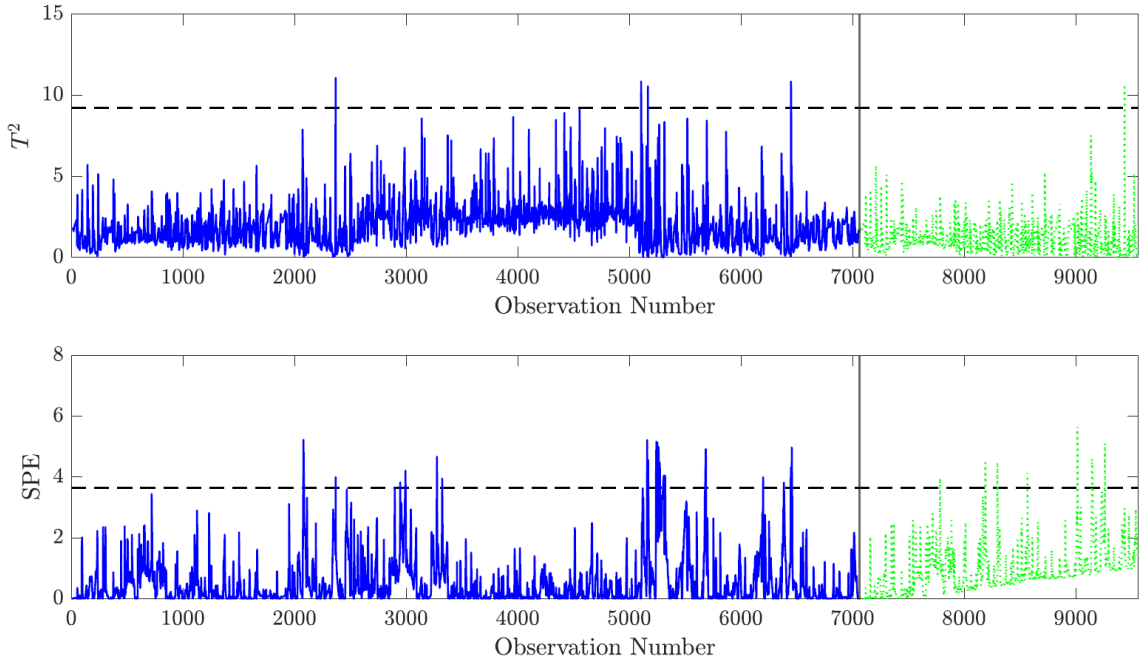


Figure 3.11: Comparison of Hotelling's T^2 (top) and SPE values (bottom) for the normal and faulty conditions (in the presence of Fault 2) obtained using the data-driven method: Training data (solid blue lines) and observations (dotted green lines) corresponding to Fault 2. The vertical gray line separates normal and faulty data.

3.3 Proposed hybrid approach for fault detection in HVAC systems

The performance of the model-based residuals and the data-driven method for detecting two faults was assessed in the previous sections. Although it was shown that the apparent impact of Fault 1 in S_1 on the residuals is small, further data analysis of the residuals may reveal some useful information for fault detection. On the other hand, it was observed that the purely data-driven method is not able to clearly distinguish faulty data from normal data. Therefore, in the hybrid approach for fault detection, the objective is to integrate the first-principles knowledge (model-based residuals) with a data-driven approach (PCA) to first model the process behavior for the normal conditions and then employ the hybrid model for fault detection of HVAC systems. As mentioned in Section 3.2, the first step to develop the PCA model is to segregate training data into the various categories according to the operating state of the AHU. In addition, the residual models were developed depending on the operating state in the AHU (Seem and House (2009)). The key idea in the proposed approach is to introduce the applicable residual(s) as a new variable(s) (new column or columns) to the training matrix data corresponding to each particular operating state. Then the hybrid PCA model can be developed using the temperature measurements and the new residual data for each state of the AHU. In summary, the hybrid-based fault detection strategy can be performed using similar steps as outlined in Section 3.2.3 for the data-driven approach except that the residuals relevant to each operating state are added to the temperature variables in the first stage to build the PCA model. The flow chart describing the proposed hybrid fault detection strategy is depicted in Fig. 3.6.

3.3.1 Application of hybrid-based fault detection design

The hybrid approach is applied to the benchmarks described above. The observations corresponding to the residual r_5, developed in State 1 and described by Eq. (3.1), are utilized together with the temperature measurements (T_{sa}, T_{ra}, T_{oa}, and T_{ma}) to build the hybrid PCA model for the normal conditions. Fig. 3.12 (left half) shows that 74% of the original data variance is explained using two principal components, which is deemed sufficiently close to the predetermined percentage of 75% to justify using two rather than three or more principal components. This model is then employed to detect Fault 1. Note that the residual r_5 is the only residual applicable in State 1. It should also be noted that the variance explanation using the hybrid PCA model is less than the variance explained by the PCA model obtained in previous approach because the same number of principal components (two components) are considered to explain the variance of the data with more features (the temperature measurements and the residual). That said, even though the PCA model explains less of the variance (due to a larger number of variables), it is able to capture the variable relationships better. In particular, as shown in Fig. 3.13, the differences between the score values of the normal and faulty conditions can be easily observed. It should be mentioned that the data samples in the upper right-hand corner of this figure correspond to the largest sensor offsets. Thus, the presence of the fault becomes more obvious as the severity of the fault increases. The Hotelling's T^2 and SPE statistics are illustrated in Fig. 3.14. As can be seen in this figure, the Hotelling's T^2 statistic exceeds the 99% control limit at approximately the 470^{th} observation after the fault occurrence and continues to increase, resulting in the detection of 81.2% of faulty data samples. The differences between the variation of the projected faulty samples and normal observations onto the principal component subspace can be readily observed using this method compared to the purely data-driven approach (see Fig. 3.9). In addition, the SPE values exceed the limit at approximately the 160^{th} observation after the fault

occurrence, leading to the detection of 93.6% of faulty data samples. Hence, applying the hybrid approach not only improves the capability of fault detection (the differences between normal and faulty behaviors can be readily observed using all indices), but also reduces the time needed to detect the fault.

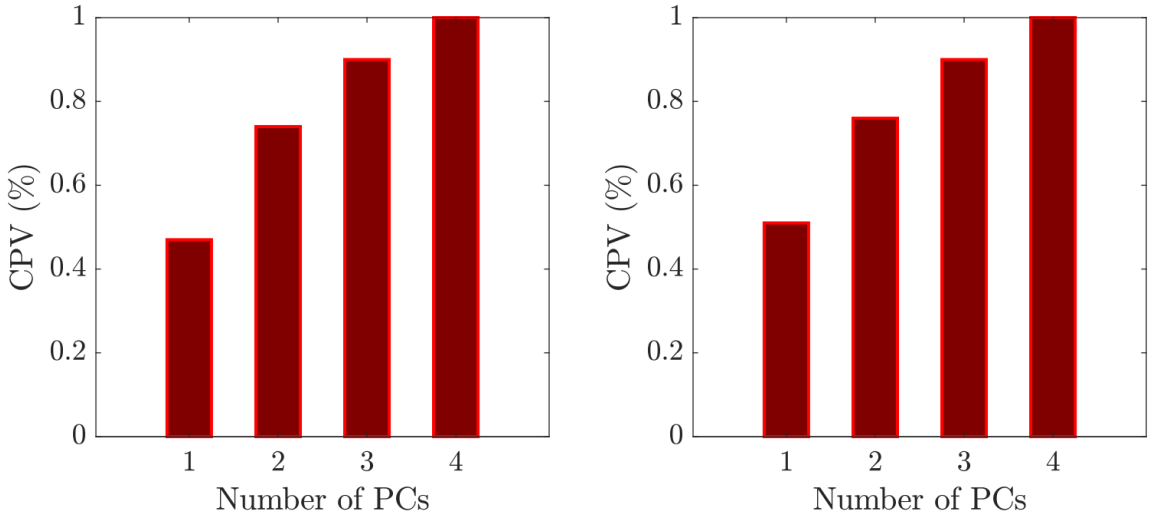


Figure 3.12: Plot of the CPV values versus the number of PCs to train the PCA model (in the hybrid approach) for State 1 (left half) and State 2 (right half) of the AHU.

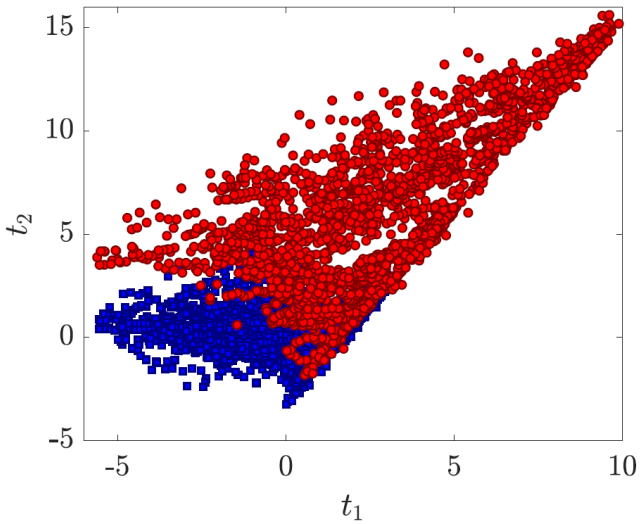


Figure 3.13: Comparison of score values for the normal (blue squares) and faulty (red circles) conditions in the presence of Fault 1 (return air temperature sensor offset) obtained using the hybrid method (PCA on temperatures and residuals).

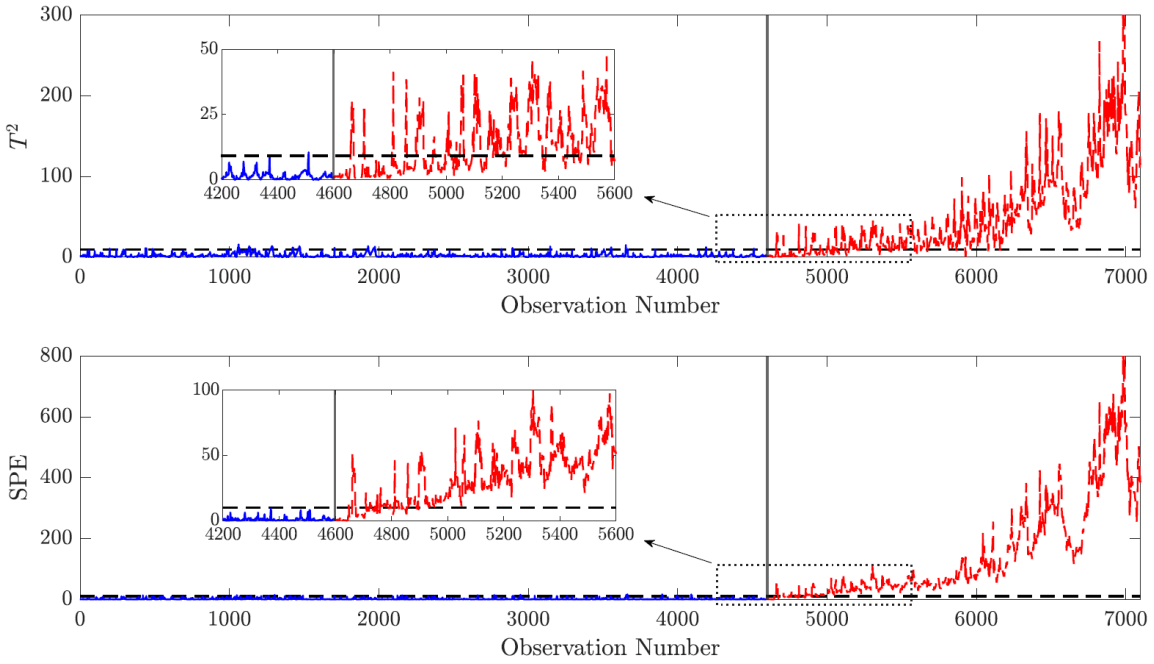


Figure 3.14: Comparison of Hotelling's T^2 (top) and SPE values (bottom) for the normal and faulty conditions (in the presence of Fault 1) obtained using the hybrid method: Training data (solid blue lines) and observations (dashed red lines) corresponding to Fault 1. The vertical gray line separates normal and faulty data.

The hybrid PCA model is developed for State 2 using the residual r_6, given by Eq. (3.2), and temperature variables. Note that the residual r_6 is the only residual applicable in State 2. The model uses two principal components which explain 76% of the original data variance (see Fig. 3.12 (right half)). The hybrid PCA model is then employed in the fault detection strategy, described above, to identify Fault 2. As illustrated in Fig 3.15, a significant difference between score values of normal and faulty observations can be readily observed. In addition, as the severity of the fault increases, the fault becomes more conspicuous. This can be understood by observing the data samples in the lower right-hand corner of this figure. Fig. 3.16 shows the values of Hotelling's T^2 and SPE statistics for normal and faulty conditions. As can be seen in this figure, the variation of the projected values of the faulty observations onto both the principal component and residual subspaces is readily distinguishable from the observations corresponding to the normal conditions. Both Hotelling's T^2

and SPE indices quickly exceed the 99% control limits at approximately the 200^{th} and 140^{th} observations after the fault occurrence, respectively, resulting in the detection of 92.0% and 94.4% of faulty data samples.

Remark 3. *It should be mentioned that it is possible to observe more than one cluster for each specific normal and faulty data set based on the operating conditions of the AHU. As shown in Figs. 3.10 and 3.15, there are two distinct clusters for data samples relevant to the normal conditions. The blue squares in these figures show the normal data samples corresponding to State 2 for one year of operation. Because there are two values for the supply air temperature set-point (12.78 °C for April 1 to October 31 and 15.56 °C for the remaining months) and the fact that State 2 can be active in both scenarios (see Fig. 3.3), each cluster contains the data samples related to each set-point value. Furthermore, the faulty data forms one cluster (green circles in Fig. 3.10) because the considered fault is analyzed in January, February, and March when the set-point value is fixed at 15.56 °C. On the other hand, one cluster is observed for normal data samples corresponding to State 1, as shown in Figs. 3.8 and 3.13 owing to the fact that when the system operates in this state the set-point value is also fixed at 15.56 °C.*

In addition, further analyses are performed that revealed Fault 1 and Fault 2 could be identified in other operating states and transitions. As before, the residual information relevant to each state or transition is used, together with the temperature measurements to first develop the hybrid PCA model and the fault detection strategy is then applied. Note that for operating states and transitions with more than one residual such as S_3, $S_{2\to3}$, and $S_{2\to3}$ (see Table 3.1), all residual variables can be integrated with the temperature measurements to build the hybrid PCA model. For instance, two residuals r_7 and r_8 are used in State 3 (S_3), in addition to the temperature measurements, to construct the hybrid PCA model for this operating state. A summary of the fault detection results for Fault 1 and Fault 2 and operation

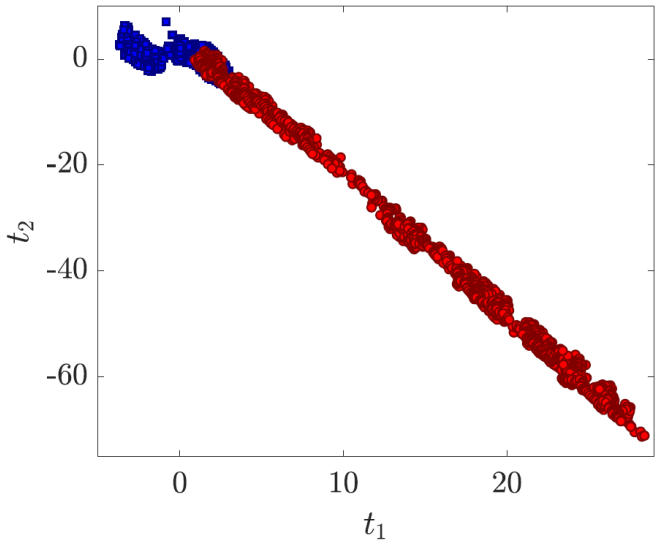


Figure 3.15: Comparison of score values for the normal (blue squares) and faulty (red circles) conditions in the presence of Fault 2 (mixed air temperature sensor offset) obtained using the hybrid method (PCA on temperatures and residuals).

corresponding to all states and transitions is presented in Table 3.3.

As summarized in Table 3.3, the hybrid method proposed here demonstrates the best performance of the methods and for the faults considered. As can be seen in Fig. 3.5, a visual inspection of the residual data for Fault 1 in State 1 (S_1) does not reveal an obvious deviation from operation under the normal conditions. However, the data-driven approach is able to identify this fault using SPE statistics. In addition, applying the hybrid method enables us to detect this fault, using all indices (score plot, Hotelling's T^2, and SPE statistics), earlier than the data-driven approach. In addition, the residual analysis is able to detect Fault 2 in State 2 (S_2), but this fault is not detectable using the data-driven approach, while the hybrid-based technique can detect this fault using all indices. Note in Table 3.3 that an asterisk superscript denotes a state in which the fault is detectable using one index (SPE, for example) over a longer time period (similar to Fault 1 in S_1). This table also shows the superior capability of the proposed hybrid approach to detect the considered faults in the other

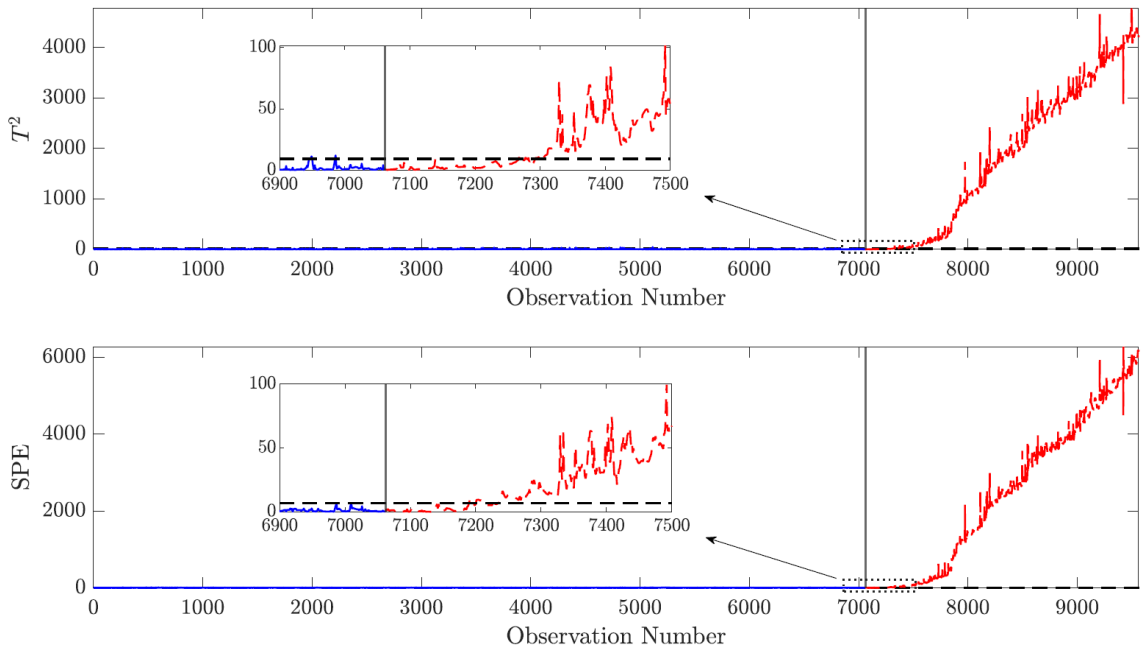


Figure 3.16: Comparison of Hotelling's T^2 (top) and SPE values (bottom) for the normal and faulty conditions (in the presence of Fault 2) obtained using the hybrid method: Training data (solid blue lines) and observations (dashed red lines) corresponding to Fault 2. The vertical gray line separates normal and faulty data.

operating states of the AHU compared to the first-principles (residual analysis) and the data-driven methods. It can be observed that Fault 1 is only detectable in S_4 based on the residual analysis. Although this fault is detectable in all operating states of the AHU using the data-driven and hybrid techniques, quicker detection in S_1 can be achieved using the hybrid method. The ability to detect Fault 2 in all states is achieved by using the hybrid approach compared to the residual analysis and the data-driven method, which can detect this fault in only some states as summarized in the Table 3.3.

Remark 4. *Note that the hybrid method uses the residual models developed considering that steady-state conditions prevail in the AHU. Thus, this method is applicable while the AHU is operating in steady-state conditions. However, in the cases where dynamic models of the residuals can be developed, they can be integrated with dynamic*

Table 3.3: Summary of states and state transitions where faults can be detected with different approaches

Fault	**Residual Analysis**	**Data-driven Method**	**Hybrid Method**
Fault 1	S_4	S_1^*, S_2, S_3, S_4, $S_{1\to2}$, $S_{2\to1}$, $S_{2\to3}$, $S_{3\to2}$	S_1, S_2, S_3, S_4, $S_{1\to2}$, $S_{2\to1}$, $S_{2\to3}$, $S_{3\to2}$
Fault 2	S_2, S_3 , S_4, $S_{2\to3}$, $S_{3\to2}$	S_4^*, $S_{1\to2}$, $S_{2\to1}$, $S_{2\to3}^*$, $S_{3\to2}^*$	S_1, S_2, S_3, S_4^*, $S_{1\to2}$, $S_{2\to1}$, $S_{2\to3}$, $S_{3\to2}$

* The fault is detectable using one index.

principal component analysis (DPCA) to develop a dynamic hybrid-based fault detection methodology.

Remark 5. *Note that the residual equations are both linear (Eq. 3.2) and nonlinear (Eq. 3.1) functions of the temperatures in different locations of the AHU. Although the principal components are constructed using linear combination of the temperature variables (in the case of purely data-driven approach), adding the residual r_6, which is the linear function of supply and mixed air temperatures enables the hybrid approach to improve the fault detection performance as shown for Fault 2 in State 2. This can be understood as follows: in this instance, use of r_6 enables a rescaling of variables in a fashion that uses the first-principles knowledge, thus making the resultant residuals more sensitive to faults, resulting in successful fault detection. On the other hand, improved performance of the hybrid approach in comparison to nonlinear residuals can be understood as follows: It is possible that a re-definition of the residuals (likely involving additional temperature measurements) would enable first-principles residuals to work just as effectively as the hybrid approach. The benefit of the hybrid approach is that it allows the use of appropriate information available in the data for the purpose of fault detection.*

3.4 Conclusion

In this study, a hybrid modeling technique, integrating first-principles knowledge with data-driven modeling approach, is proposed for fault detection in HVAC systems. The residuals (developed from first-principles models) and temperature variables are utilized to first develop a hybrid PCA model in each operating state of the AHU. Then the model is employed to detect faults in operational data corresponding to that state. The superior capability of the proposed method is demonstrated compared to two other approaches, the residual analysis and the data-driven method (PCA on temperatures), through the simulation results for two fault cases (return and mixed air temperature sensor faults).

Bibliography

Beghi, A., Brignoli, R., Cecchinato, L., Menegazzo, G., Rampazzo, M., and Simmini, F. (2016). Data-driven fault detection and diagnosis for HVAC water chillers. *Control Engineering Practice*, 53:79–91.

Chen, Y. and Lan, L. (2010). Fault detection, diagnosis and data recovery for a real building heating/cooling billing system. *Energy Conversion and Management*, 51(5):1015–1024.

DeSimone, M. (1995). A standard simulation testbed for the evaluation of control algorithms and strategies related to variable air volume HVAC systems. *Doctoral dissertation, Massachusetts Institute of Technology.*

Du, M. and Mhaskar, P. (2014). Isolation and handling of sensor faults in nonlinear systems. *Automatica*, 50(4):1066–1074.

Du, Y. and Du, D. (2018). Fault detection and diagnosis using empirical mode decomposition based principal component analysis. *Computers & Chemical Engineering*, 115:1–21.

Du, Z., Chen, L., and Jin, X. (2017). Data-driven based reliability evaluation for measurements of sensors in a vapor compression system. *Energy*, 122:237–248.

Du, Z., Jin, X., and Wu, L. (2007). Fault detection and diagnosis based on improved PCA with JAA method in VAV systems. *Building and Environment*, 42(9):3221–3232.

Eastment, H. T. and Krzanowski, W. J. (1982). Cross-validatory choice of the number of components from a principal component analysis. *Technometrics*, 24(1):73–77.

Fan, B., Du, Z., Jin, X., Yang, X., and Guo, Y. (2010). A hybrid FDD strategy

for local system of AHU based on artificial neural network and wavelet analysis. *Building and Environment*, 45(12):2698–2708.

Ghosh, D., Hermonat, E., Mhaskar, P., Snowling, S., and Goel, R. (2019). Hybrid modeling approach integrating first-principles models with subspace identification. *Industrial & Engineering Chemistry Research*, 58(30):13533–13543.

Haves, P., Norford, L. K., and DeSimone, M. (1998). A standard simulation test bed for the evaluation of control algorithms and strategies (rp-825). *ASHRAE Transactions*, 104(1):460.

Hosen, M. A., Hussain, M. A., and Mjalli, F. S. (2011). Control of polystyrene batch reactors using neural network based model predictive control (NNMPC): An experimental investigation. *Control Engineering Practice*, 19(5):454–467.

House, J. M., Vaezi-Nejad, H., and Whitcomb, J. M. (2001). An expert rule set for fault detection in air-handling units/discussion. *Ashrae Transactions*, 107:858.

Jackson, J. E. and Mudholkar, G. S. (1979). Control procedures for residuals associated with principal component analysis. *Technometrics*, 21(3):341–349.

Joe Qin, S. (2003). Statistical process monitoring: basics and beyond. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(8-9):480–502.

Kourti, T. (2005). Application of latent variable methods to process control and multivariate statistical process control in industry. *International Journal of adaptive control and signal processing*, 19(4):213–246.

Kourti, T. and MacGregor, J. F. (1996). Multivariate SPC methods for process and product monitoring. *Journal of quality technology*, 28(4):409–428.

Li, G. and Hu, Y., . (2019). An enhanced PCA-based chiller sensor fault detection method using ensemble empirical mode decomposition based denoising. *Energy and Buildings*, 183:311–324.

Li, S. and Wen, J. (2014). A model-based fault detection and diagnostic methodology based on PCA method and wavelet transform. *Energy and Buildings*, 68:63–71.

Li, W., Yue, H. H., Valle-Cervantes, S., and Qin, S. J. (2000). Recursive PCA for adaptive process monitoring. *Journal of process control*, 10(5):471–486.

Liang, J. and Du, R. (2007). Model-based fault detection and diagnosis of HVAC systems using support vector machine method. *International Journal of refrigeration*, 30(6):1104–1114.

MacGregor, J. and Cinar, A. (2012). Monitoring, fault diagnosis, fault-tolerant control and optimization: data driven methods. *Computers & Chemical Engineering*, 47:111–120.

Muller, T., Réhault, N., and Rist, T. (2013). A qualitive modeling approach for fault detection and diagnosis on HVAC systems. *Proceedings of the 13th International Conference for Enhanced Building Operations*, pages Montreal, Canada, October 8–11.

Norford, L. K. and Haves, P. (1997). A standard simulation testbed for the evaluation of control algorithms and strategies. final report of ashrae rp-825. *American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., Atlanta.*

O'Neill, Z., Pang, X., Shashanka, M., Haves, P., and Bailey, T. (2014). Model-based real-time whole building energy performance monitoring and diagnostics. *Journal of Building Performance Simulation*, 7(2):83–99.

Park, C., Clark, D. R., and Kelly, G. E. (1985). An overview of HVACSIM+, a dynamic building/HVAC/control systems simulation program. *In Proceedings of the 1st Annual Building Energy Simulation Conference, Seattle*, 104(1):175–185.

Provan, G. (2011). Generating reduced-order diagnosis models for HVAC systems. *In Itnl. Workshop on Principles of Diagnosis*, Murnau, Germany, October 4–7.

Qin, S. J. (2006). An overview of subspace identification. *Computers & Chemical Engineering*, 30(10-12):1502–1513.

Qin, S. J. and Chiang, L. H. (2019). Advances and opportunities in machine learning for process data analytics. *Computers & Chemical Engineering*, 126:465–473.

Ren, N., Liang, J., Gu, B., and Han, H. (2008). Fault diagnosis strategy for incompletely described samples and its application to refrigeration system. *Mechanical Systems and Signal Processing*, 22(2):436–450.

Schein, J., Bushby, S. T., Castro, N. S., and House, J. M. (2006). A rule-based fault detection method for air handling units. *Energy and buildings*, 38(12):1485–1492.

Seem, J. E. and House, J. M. (2009). Integrated control and fault detection of air-handling units. *HVAC&R Research*, 15(1):25–55.

Seem, J. E., Park, C., and House, J. M. (1999). A new sequencing control strategy for air-handling units. *HVAC&R Research*, 5(1):35–58.

Shahnazari, H., Mhaskar, P., House, J. M., and Salsbury, T. I. (2018). Heating, ventilation and air conditioning systems: fault detection and isolation and safe parking. *Computers & Chemical Engineering*, 108:139–151.

Shahnazari, H., Mhaskar, P., House, J. M., and Salsbury, T. I. (2019). Modeling and fault diagnosis design for HVAC systems using recurrent neural networks. *Computers & Chemical Engineering*, 126:189–203.

Song, Y. H., Akashi, Y., and Yee, J. J. (2008). A development of easy-to-use tool for fault detection and diagnosis in building air-conditioning systems. *Energy and Buildings*, 40(2):71–82.

Thumati, B., Feinstein, M. A., Fonda, J. W., Turnbull, A., Weaver, F. J., Calkins, M. E., and Jagannathan, S. (2011). An online model-based fault diagnosis scheme

for HVAC systems. *In 2011 IEEE International Conference on Control Applications (CCA)*, pages 70–75.

Venkatasubramanian, V., Rengaswamy, R., and Kavuri, S. N. (2003a). A review of process fault detection and diagnosis: Part ii: Qualitative models and search strategies. *Computers & Chemical Engineering*, 27(3):313–326.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N., and Yin, K. (2003b). A review of process fault detection and diagnosis: Part iii: Process history based methods. *Computers & Chemical Engineering*, 27(3):327–346.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., and Kavuri, S. N. (2003c). A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Computers & Chemical Engineering*, 27(3):293–311.

Wang, S. and Cui, J. (2006). A robust fault detection and diagnosis strategy for centrifugal chillers. *HVAC&R Research*, 12(3):407–428.

Wang, S. and Qin, J. (2005). Sensor fault detection and validation of VAV terminals in air conditioning systems. *Energy Conversion and Management*, 46(15-16):2482–2500.

Wang, S. and Xiao, F. (2004a). AHU sensor fault diagnosis using principal component analysis method. *Energy and Buildings*, 36(2):147–160.

Wang, S. and Xiao, F. (2004b). Detection and diagnosis of AHU sensor faults using principal component analysis method. *Energy Conversion and Management*, 45(17):2667–2686.

Wold, S. (1978). Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics*, 20(4):397–405.

Wu, S. and Sun, J. Q. (2011). Cross-level fault detection and diagnosis of building HVAC systems. *Building and Environment*, 46(8):1558–1566.

Yang, H., Cho, S., Tae, C. S., and Zaheeruddin, M. (2008). Sequential rule based algorithms for temperature sensor fault detection in air handling units. *Energy Conversion and Management*, 49(8):2291–2306.

Yoon, S. and MacGregor, J. F. (2000). Statistical and causal model-based approaches to fault detection and isolation. *AIChE Journal*, 46(9):1813–1824.

Yoon, S. and MacGregor, J. F. (2001). Fault diagnosis with multivariate statistical models part i: using steady state fault signatures. *Journal of process control*, 11(4):387–400.

Zhang, X., Polycarpou, M. M., and Parisini, T. (2010). Fault diagnosis of a class of nonlinear uncertain systems with lipschitz nonlinearities using adaptive estimation. *Automatica*, 46(2):290–299.

Chapter 4

Integrating dynamic neural network models with principal component analysis for adaptive model predictive control

The contents of this chapter have been published in the *Chemical Engineering Research and Design* Journal.

- Integrating dynamic neural network models with principal component analysis for adaptive model predictive control. Hesam Hassanpour, Brandon Corbett, and Prashant Mhaskar, *Chemical Engineering Research and Design*, 2020, 161, 26-37. DOI: 10.1016/j.cherd.2020.03.031

Abstract

This work addresses one aspect of the overparameterization problem in using artificial/recurrent neural networks (ANN/RNN) based dynamic models for model predictive control (MPC) implementations. The manuscript presents an approach to handle situations where the training data may not be sufficiently rich, and in particular, for handling historical data with correlated inputs. Two approaches are proposed. The key idea in the first method is perform principal component analysis (PCA) on input space and then utilize the scores to build a PCA-RNN model. Next, a PCA-RNN-based MPC is designed to compute the optimal values of scores and subsequently determined the manipulated inputs. An alternative solution is proposed in the second approach by proposing a new constraint on squared prediction error (SPE) statistic in the RNN-based MPC to make prescribed inputs follow the PCA model constructed for training input data. Finally, an approach is presented that allows to break the correlation in the MPC implementation while maintaining model validity. This is done by first generating richer closed-loop data by implementing the SPE based MPC with slightly relaxed constraints (thus compromising only slightly on the closed loop performance). Then the new data is utilized to re-identify the model, and for use in the MPC. The efficacy of the proposed approaches to handle the problem of set-point tracking is evaluated using a chemical reactor example. The results are compared with a nominal MPC design, and the superior performance under the proposed formulations demonstrated.

4.1 Introduction

Automatic control systems have been widely utilized in industrial processes to improve operating efficiencies. Existing challenges in process operations such as nonlinearity, multi-variable interactions, and process constraints make the design of an optimal

controller an essential task. In this direction, advanced process control strategies and especially model predictive control have been successfully developed to cope with process constraints and achieve economic objectives. Addressing the problem of system identification, to build and develop a reliable and accurate process model as a part of an MPC framework, plays an important role in the successful implementation of this approach.

Given the increased availability of process data, and the challenges associated with developing and maintaining a first principles model, there is significant interest in utilizing data driven approaches such as subspace identification, artificial neural network (ANN), principal component analysis (PCA), and projection to latent structures (PLS) to identify and develop appropriate dynamic models.

Data collection strategy and the operating conditions under which data is collected can strongly affect the type of models and their performance. Perturbation of inputs using generally uncorrelated pseudo random binary sequence (PRBS) or random Gaussian signals (RGS), to excite the process, is one of the approaches to generate rich data for model identification (see e.g., Shariff et al. (2013)). However, the standard uncorrelated input signals sometimes are not applicable due to the necessity for the satisfaction of other constraints such as output constraints or integral controllability during identification process. Thus, alternative approaches to uncorrelated inputs are proposed in the literature to design appropriate input signals for identification of dynamic models, especially for ill-conditioned systems (see e.g., Darby and Nikolaou (2009) and Darby and Nikolaou (2014)). In other approaches, considerations such as unstable open-loop characteristic, production, economic, and safety concerns have motivated the application of closed-loop identification techniques (see e.g., Forssell and Ljung (1999), Ljung (1999), Qin and Ljung (2003), and Huang et al. (2005)). To this end, there are several approaches that address the problem of closed-loop identification using data obtained without excitation of the reference signals (see e.g.,

Shardt and Huang (2011a) and Shardt and Huang (2011b)), applicable to processes with sufficiently large time delay or adequately fast sampling time. Other approaches consider the minimal required excitation conditions for a reference signal to identify the process model (see e.g., Shardt et al. (2015)). In essence, these techniques rely on the inadequacy of the closed-loop controller to tease out the model dynamics from closed-loop data.

Data driven and machine learning (ML) based techniques have recently received significant attention to model process dynamics for the purpose of control. In one direction, subspace identification approaches are utilized to identify process dynamics to be employed as a part of MPC framework (see e.g., Shi and MacGregor (2001), Hale and Qin (2002), Corbett and Mhaskar (2016), Kheradmandi and Mhaskar (2018a), and Kheradmandi and Mhaskar (2018b)). Due to the significant impact of the features on the performance of the trained model, feature selection strategies are considered as a common preprocessing step utilized for most practical machine learning-based applications. In this direction, PCA is one of the popular multivariate statistical techniques used for dimensionality reduction. A number of PCA-based feature selection strategies appropriate for building process models are proposed in the literature (see e.g., Lu et al. (2007), Song et al. (2010), Janakiraman et al. (2013), and Drgoňa et al. (2018)). In addition, several approaches are proposed to design latent variable model predictive control algorithms based on PCA models (see e.g., Flores-Cerrillo and MacGregor (2005), Golshan et al. (2010), and Godoy et al. (2016)). Recently, other machine learning-based techniques such as artificial neural networks are seeing increased interest in being deployed to model and design MPC for nonlinear systems (see e.g., Bhat and McAvoy (1990), Ramchandran and Rhinehart (1995), Nikravesh et al. (2000), Kittisupakorn et al. (2009), Sadeghassadi et al. (2018), Wu et al. (2019a), and Wu et al. (2019b)). However, because of the large number of parameters, a significant amount of data is required to achieve a fairly accurate model, and the problem of over-fitting remains a challenge, especially in the context of closed-loop implemen-

tations. The negative implications of the over-fitting problem would get exacerbated in situations where the data has been generated under closed-loop conditions that might be inducing co-linearity in certain inputs and outputs- and if data generated from that model is to be used for model identification, to be in turn utilized in a model predictive control implementation without the original co-linearity in place. Note that one solution to the problem is to first redesign the existing control structure to break the correlation, generate new data, and then identify a model. Where possible, such a solution is certainly viable. The present manuscript, however, focuses on the problem where practical limitations (often arising in industrial settings) do not permit changing the existing control structure, but instead, a model identification and control strategy is required that is able to use the existing data to build a reasonably good MPC implementation, and then subsequently improve upon it, if possible.

Motivated by above considerations, the present manuscript addresses the problem of model identification in the presence of co-linearity in the inputs in the training data. A base case scenario is used for comparison where the training data is used to build an input output RNN model, and used directly in the MPC implementation. As expected, under MPC implementation, breaking of the correlation between the inputs leads the process to a region where the model is no longer valid, leading to poor performance. To address this, two approaches are presented. In the first approach, a combination of PCA and RNN is utilized to model the process dynamics and employed in MPC implementation in a way that naturally respects the correlation observed in the training data. In the second approach, the base case RNN model is used in MPC, but a new constraint on squared prediction error (SPE) is included to ensure that the control moves remain in the same plane as the training data. Next, an approach is presented to fully exploit the performance enhancement capabilities of MPC, i.e., allow the MPC to break the input correlation seen in the training data, while not loosing model validity. To achieve this, the process is first run under MPC, albeit with slightly relaxed constraints on the SPE, thus generating

richer data (while only slightly compromising on the performance). A model identified using this new data is next used in the MPC implementation to achieve superior performance. The rest of the manuscript is organized as follows: In Section 6.2, a brief description of a continuous stirred tank reactor (CSTR) simulation test bed is presented. Then a review of the RNN-based model identification technique, principal component analysis (PCA), and the RNN-based MPC design is provided. In addition, the base case RNN-based MPC is implemented to the motivating example to illustrate the specific problem being addressed. In Section 5.3, the proposed approach for model identification, combining PCA and RNN techniques, and the PCA-RNN-based MPC design is presented. Furthermore, a remedy for the base case RNN-based MPC approach is proposed. Then the efficacy of the proposed approaches is evaluated by implementing to the CSTR example. Next, model re-identification methodology is proposed in Section 4.4 to generalize the model validity and then the effectiveness of this approach is illustrated through implementation to the CSTR example. Finally, concluding remarks are presented in Section 6.5.

4.2 Preliminaries

In this section, a Continuous Stirred Tank Reactor (CSTR) example is first described to provide the motivation for proposed approach. Next, a brief review of the application of artificial neural networks, especially RNN, to model process dynamics, followed by a brief description of principal component analysis is provided. Finally, a base case RNN model based MPC is implemented to illustrate the key problem being addressed.

4.2.1 Motivating Example: Continuous Stirred Tank Reactor (CSTR)

Consider the CSTR example (Du and Mhaskar (2014)), in which an irreversible elementary exothermic reaction (A $\rightarrow$ B) occurs, for which a mathematical model describing the dynamics takes the following form:

$$\begin{aligned}
\dot{C_A} =& \frac{F}{V}(C_{A0} - C_A) - k_0 e^{-E/RT_R} C_A \\
\dot{T_R} =& \frac{F}{V}(T_0 - T_R) + \frac{(-\Delta H)}{\rho c_p} k_0 e^{-E/RT_R} C_A - \frac{UA}{\rho c_p V}(T_R - T_c) \\
\dot{T_c} =& \frac{F_c}{V_c}(T_{cf} - T_c) + \frac{UA}{\rho_c c_{pc} V_c}(T_R - T_c)
\end{aligned} \tag{4.1}$$

where C_A, T_R, and T_c are concentration of reactant A, the temperature in the reactor, and the temperature in the cooling jacket, respectively. F represents the volumetric flow rate to the reactor with temperature T_0 and concentration C_{A0} and F_c is the cooling stream volumetric flow rate to the jacket with temperature T_{cf}. The values of process parameters and the steady-state values of the variables are listed in Table 6.1.

Table 4.1: Parameter and steady-state values for the CSTR example

$V = 100$ L	$c_{pc} = 4.20$ J/(g K)
$k_0 = 7.20 \times 10^{10}$ min^{-1}	$C_{A0} = 1$ mol/L
$E/R = 8750$ K	$T_0 = 350$ K
$\Delta H = -2 \times 10^4$ J/mol	$C_{A_s} = 0.83$ mol/L
$\rho = 1000$ g/L	$T_{R_s} = 307.14$ K
$c_p = 0.239$ J/(g K)	$T_{c_s} = 303.13$ K
$UA = 5 \times 10^4$ J/(min K)	$F_s = 14.60$ L/min
$V_c = 20$ L	$F_{c_s} = 4.70$ L/min
$\rho_c = 1000$ g/L	$T_{cf_s} = 293$ K

The CSTR example is used to represent one instance of a control structure where data may be available from process operation with more than one manipulated input being used to control the same output. Thus, in this example, the reactor and cooling jacket temperatures are controlled by manipulating feed flow rate to the reactor (F), the cooling stream flow rate to the jacket (F_c), and the cooling stream temperature (T_{cf}) by using PI controllers. The feed flow rate (F) is manipulated to control the reactor temperature (T_R) in the first control loop and the cooling stream flow rate and its inlet temperature (F_c and T_{cf}) are manipulated in the second control loop to maintain the jacket temperature (T_c), with the control actions being computed as follows:

$$U(t) = K\left(K_c E(t) + \frac{K_c}{T_I}\int_0^t E(t')dt'\right) \tag{4.2}$$

where U is the control signal, E is the error between set points s (R) and measured process variables (Y, thus $E = R - Y$), K_c is the proportional gain, and T_I is the integral time constant. Therefore, there are two pairings of controlled and manipulated variables: 1. (F) with (T_R) in the first loop and 2. (F_c and T_{cf}) with (T_c) in the second loop. The values of the tuning parameters for the two PI controllers are listed in Table 6.3. For the loop with two manipulated inputs and one output, the vector $K = [k_1\ k_2]^T$ in Eq. (6.2) is utilized to apply different degrees of the effectiveness on the manipulated variables. For the first loop, where one manipulated input is utilized to control one controlled variable, K is a scalar, and set to one. The values of k_1 and k_2 are used as -2 and 1, respectively for manipulated inputs (F_c) and (T_{cf}) in the second loop.

Instances of process data are generated by simulating the CSTR example in MATLAB. Specifically, data for training and validation is generated by applying set-point changes

Table 4.2: Tuning parameters for PI controllers

$K_{c_1} = 0.66$	$K_{c_2} = 0.37$
$T_{I_1} = 0.24$	$T_{I_2} = 0.51$

in T_R and T_c for the process under the PI controllers. The simulation input/output data relevant for training stage is shown in Fig. 4.1. Note that the over or under shoots, in the times when the set-points are constant, are due to the fact that a set-point change for the other output acts as a disturbance for the variable under consideration. It should also be noted that the output deviation variables, $\bar{T}_R = T_R - T_{R_s}$ and $\bar{T}_c = T_c - T_{c_s}$, and input deviation variables, $\bar{F} = F - F_s$, $\bar{F}_c = F_c - F_{c_s}$, and $\bar{T}_{cf} = T_{cf} - T_{cf_s}$, are shown here.

4.2.2 System identification using artificial neural networks

The application of artificial neural networks in the area of modeling and identification of dynamic systems has received significant attention in recent years. One approach to model dynamics within ANN is to use the recurrent neural networks (RNN). Unlike feedforward neural networks, the existence of delayed feedback loops in the structure of RNN provides the network with a dynamic memory. Different types of RNN such as nonlinear autoregressive network with exogenous inputs (NARX), layer recurrent network (LRN), long short-term memory (LSTM) and gated recurrent unit (GRU) have been developed and applied to process control (see e.g., Govindhasamy et al. (2005), Wang et al. (2017), and Wu and Christofides (2019)), optimization (see e.g., Govindhasamy et al. (2005), Kusiak and Xu (2012), and Zhang et al. (2019)), and fault diagnosis (see e.g.,Shahnazari et al. (2019)).

In this study, a NARX network is utilized to identify a dynamic model of the process for illustrative purposes. The key idea of the approach remains applicable with other kinds of dynamic ANN models. The NARX model can be mathematically expressed

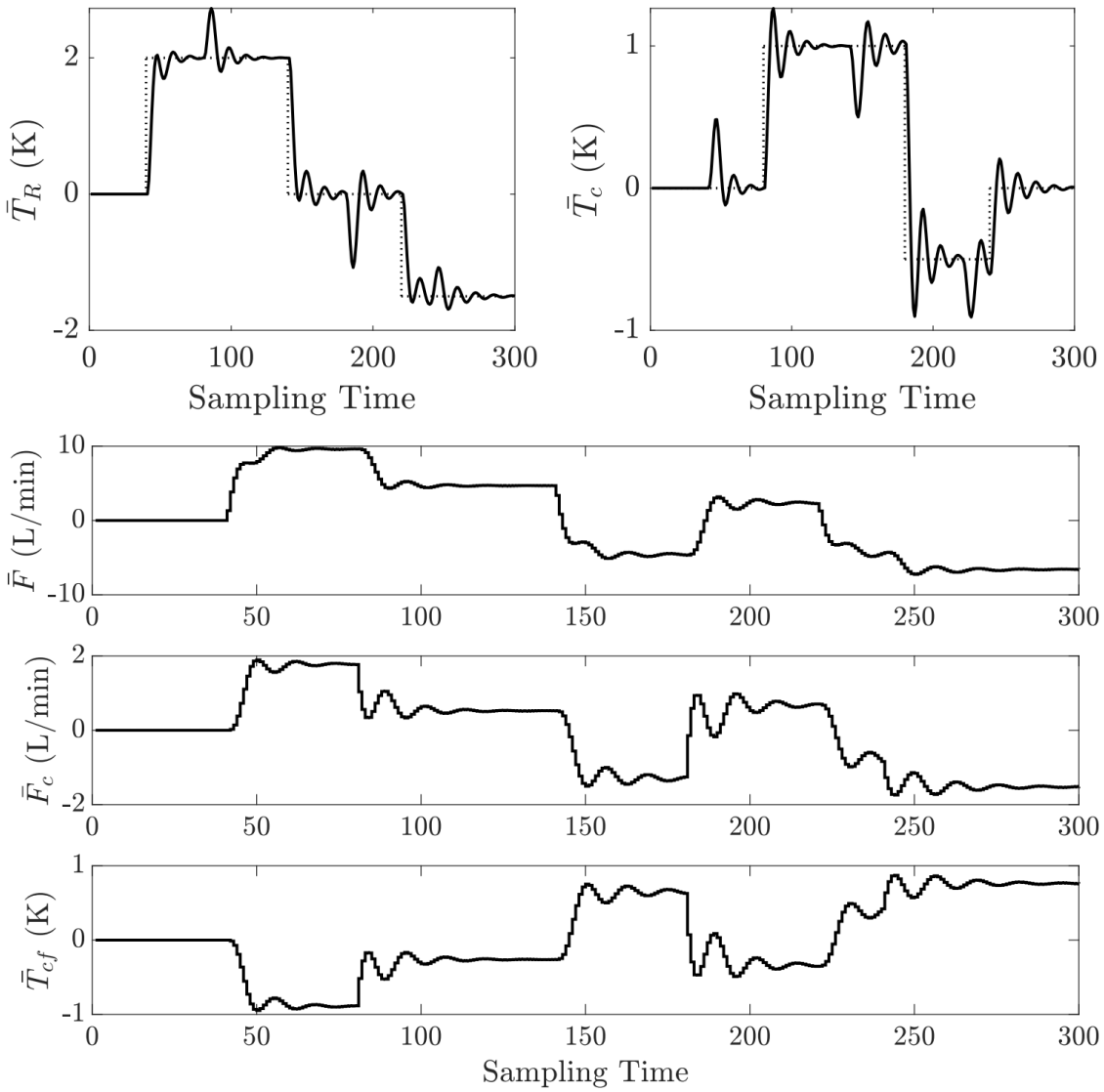


Figure 4.1: Model training data: measured outputs and manipulated variables under PI controller (solid lines) and set-points (dotted lines).

as follows:

$$y(\tau + T_s) = f[y(\tau), y(\tau - T_s), ..., y(\tau - L_y T_s), u(\tau), u(\tau - T_s), ..., u(\tau - L_u T_s)] \quad (4.3)$$

where y $\in \mathbb{R}^{n_y}$ and u $\in \mathbb{R}^{n_u}$ represent the vector of measured outputs and inputs of the system, respectively. τ, L_y, and L_u represent time, the number of lagged outputs, and the number of lagged inputs required for prediction. In other words, the next value of the dependent variables is regressed on past and current values of the dependent and independent (exogenous) variables by NARX models. A multilayer perceptron (MLP), a class of feedforward artificial neural networks, with the embedded memory can be utilized to approximate the nonlinear function 'f' in Eq. (5.3). The resulting model is referred to as a NARX recurrent dynamic neural network (see e.g., Narendra and Parthasarathy (1990)). A schematic of a NARX neural network is shown in Fig. 4.2 (left half). In the present manuscript, we use this RNN structure to illustrate the key idea of the proposed approach.

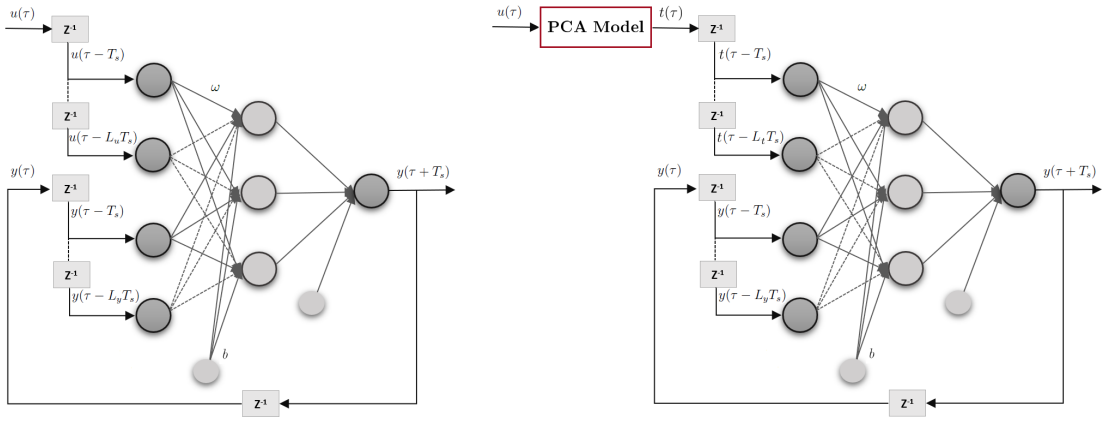


Figure 4.2: Schematic of a NARX neural network based on: Inputs (left) and Scores (right).

The data samples are first normalized and then fed to NARX network in MATLAB Machine Learning and Deep Learning toolbox. The structure of the network (the number of hidden layers and neurons) is chosen according to existing trial and error approaches in the literature (see e.g., Zhang et al. (1998)). One hidden layer including 5 neurons with hyperbolic tangent activation function ($tanh(x) = 2/(1 + e^{-2x}) - 1$) is utilized. Linear activation function is utilized for the output layers and Bayesian regularization backpropagation algorithm, updating the bias and weight values based

on Levenberg-Marquardt optimization, is chosen to determine the value of these parameters in the model.

The capability of the trained RNN model to capture process dynamics is examined in the validation stage. The validation data is generated by applying set-point changes for PI controllers. The data samples for two process outputs, T_R, and T_c, are shown in Fig. 4.3, based on deviation variables. The output and input values are fed to the trained NARX network to predict the outputs in the first sampling time. Subsequently, the predicted outputs in this step are utilized to predict the outputs in the next sampling time. This process is repeated to predict the process outputs in the next sampling times and the results are shown in Fig. 4.3. As demonstrated, the RNN model (dashed lines) is able to predict the values of the outputs fairly well.

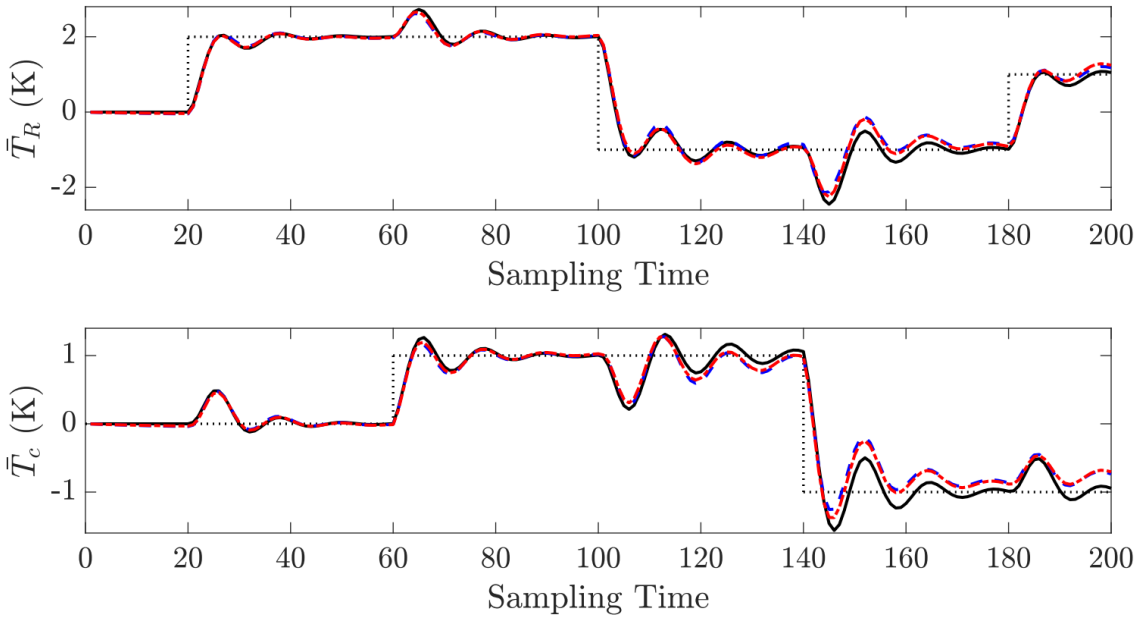


Figure 4.3: Model validation results: measured outputs (solid lines), RNN model prediction (dashed lines), PCA-RNN model prediction (dash-dotted lines), and set-points (dotted lines).

4.2.3 Review of principal component analysis

In this section, some of the fundamental concepts behind principal component analysis (PCA), one of the most popular latent variable methods, are reviewed. The objective of PCA is to transform a set of observations of possibly correlated variables into a latent space, where the new variables or principal components are uncorrelated. The mathematical representation of PCA method can be stated as follows in Eq. (4.4):

$$X = TP^T + E \tag{4.4}$$

where X is a matrix with columns and rows containing the process variables and their observations, respectively. T is a score matrix representing the new mutually uncorrelated variables (principal components) by columns and projected values of the observations by rows. P is a loading matrix, containing orthogonal loading vectors, determining the orientation of the latent space. The principal components are ordered so that the largest and smallest amount of variation in the data set is described by the first and the last components, respectively. Note that the first term of Eq. (4.4) (TP^T) explains the modeled variations of X and the second term (E) denotes the residual or the unmodeled variations of X.

In order to apply PCA algorithm, the data matrix is first mean centered and scaled to have zero mean and unity variance. Then, orthogonalization techniques such as singular value decomposition (SVD) or nonlinear iterative partial least squares (NIPALS) can be utilized to compute the principal components. Cross validation techniques (see e.g., Wold (1978) and Eastment and Krzanowski (1982)) can be applied to determine the number of proper principal components.

4.2.4 RNN-based model predictive control

Given an appropriate RNN model, the objective is to utilize the model in MPC implementation. A representative RNN-based MPC formulation can be described as follows:

$$\begin{aligned} \min_{u_k,\ldots,u_{k+P}} \quad & \sum_{j=1}^{P} \|\tilde{y}_{k+j} - y^{SP}_{k+j}\|^2_{Q_{y_r}} + \|u_{k+j} - u_{k+j-1}\|^2_{R_{du}} \\ s.t. \quad & RNN\ based\ Predictive\ Model, \\ & u_{min} \leq u \leq u_{max} \end{aligned} \tag{4.5}$$

where P represents the prediction horizon, y^{SP}_{k+j} is the desired set-point values for the outputs, $\tilde{y}_{k+j}$ is the prediction of output at time $(k+j)T_s$, $T_s = \Delta\tau$ is the sampling time, Q_{y_r} and R_{du} are penalty matrices corresponding to the output deviations from the set-points and the rate of change in the inputs, and $u_{min} \in \mathbb{R}^{n_u}$ and $u_{max} \in \mathbb{R}^{n_u}$ represent the lower and upper bounds of the manipulated inputs. It should be noted that the RNN-based predictive model computes multi-step ahead predictions. Thus, the initial values of inputs and outputs (lagged data) are fed to the model and then the values of the outputs are calculated one step ahead. Afterwards, the predicted outputs, along with candidate future input values are fed to the model to predict the outputs two steps ahead. Subsequently, this procedure is performed recursively to predict the outputs multi steps ahead up to the prediction horizon.

Next, the above representative RNN based MPC is implemented on the motivating example, using the RNN model identified in the earlier section. Specifically, the RNN-based identified model of the process, along with the constraints on the manipulated inputs, where $\bar{F} \in [0 - F_s \; 60 - F_s]$, $\bar{F}_c \in [0 - F_{c_s} \; 15 - F_{c_s}]$, and $\bar{T}_{cf} \in [290.5 - T_{cf_s} \; 299 -$

T_{cf_s}], are included in the RNN-based MPC represented by Eq. (4.5). The optimization problem for the RNN-based MPC is solved using *fmincon* solver in MATLAB and the parameters are chosen as follows: sampling time $T_s = 30$ s, prediction horizon $P = 4$, $Q_{y_r} = [1000\ 0; 0\ 1000]$, and $R_{du} = [1\ 0\ 0; 0\ 1\ 0; 0\ 0\ 1]$. The output and input profiles obtained by implementing the RNN-based MPC are shown in Fig. 4.4 (solid lines). As shown, this approach fails to track the set-point values (dotted lines) of the reactor and jacket temperature.

The reason why this approach fails can be explained by analyzing the capability of the RNN model embedded in the RNN-based MPC framework. In particular, the use of the RNN model in the above formulation leads to a plant-model mismatch due to the fact that the model was identified using data with the two of the three inputs being correlated, while in the MPC implementation, the inputs are allowed to take independent values.

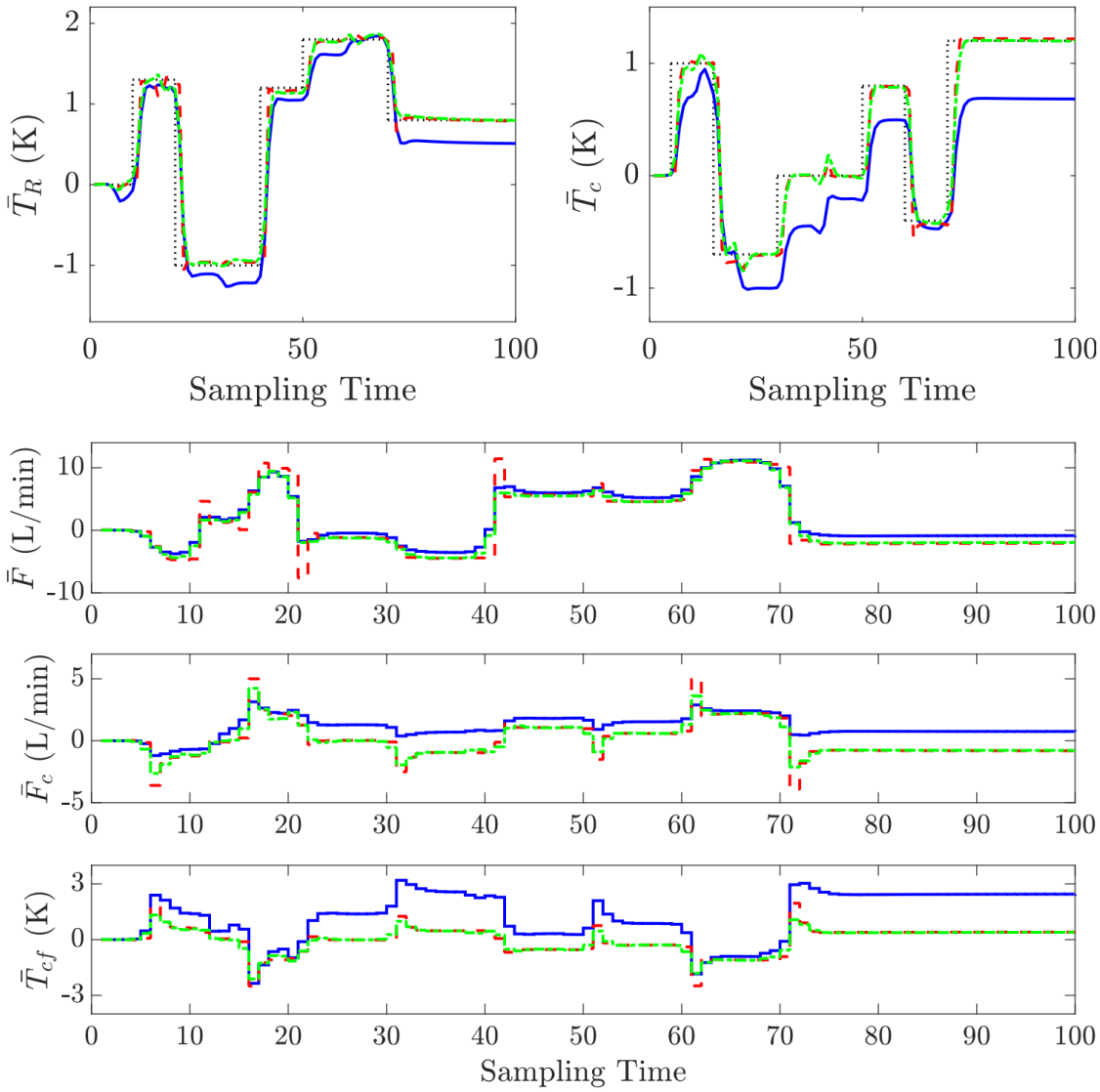


Figure 4.4: Comparison of the closed-loop trajectories for the measured variables and the profiles of the manipulated variables obtained from RNN-based MPC (blue solid lines), PCA-RNN-based MPC (red dashed lines), and constrained RNN-based MPC (green dash-dotted lines).

4.3 Proposed approaches for model identification and MPC design

4.3.1 PCA-RNN-based model identification

In the proposed approach for system identification, PCA is first utilized to eliminate the correlations existing in the input data samples. Then, the scores (which are by definition uncorrelated) are used to develop the model for the system using appropriate machine learning-based techniques. Recall that co-linearity in the input space means that it is impossible to identify the unique impact of each input variable. As a result, any data-driven model that uses the raw inputs as manipulated variables without considering the co-linearity would end up assigning arbitrary effects to the inputs. By contrast, the scores calculated in a PCA model are inherently independent and therefore represent directions of variation for which sufficient information is available to identify the unique impact of this reduced dimensional input space on output variables. The NARX utilizing the principle components takes the following form:

$$y(\tau + T_s) = f[y(\tau), y(\tau - T_s), ..., y(\tau - L_y T_s), t(\tau), t(\tau - T_s), ..., t(\tau - L_t T_s)] \quad (4.6)$$

where y $\in \mathbb{R}^{n_y}$ and and t $\in \mathbb{R}^{n_t}$ represent the vector of measured outputs and scores, respectively, and the current time is indicated by τ. The number of lagged outputs and scores required for prediction are represented by L_y and L_t, respectively. A schematic of a NARX neural network, utilizing PCA as a preprocessor, is shown in Fig. 4.2 (right half).

The NARX network is employed, using MATLAB Machine Learning and Deep Learn-

ing toolbox, to determine the parameters in the function 'f' in Eq. (4.6). To this end, the data samples are first normalized and then PCA is applied to the input training data samples shown in Fig. 4.1. Not surprisingly, the *R-squared* value suggests using two principal components in the identified model. Next, the two scores and outputs are utilized to train the RNN model. One hidden layer with 4 neurons is utilized in the NARX network. The hyperbolic tangent activation function and linear transfer function are employed in hidden and output layers, respectively. The Bayesian regularization backpropagation algorithm is chosen to update the values of weights and biases based on Levenberg-Marquardt optimization. The capability of the PCA-RNN model to predict outputs is shown in Fig. 4.3. It can be observed that the PCA-RNN model has nearly the same capability as the RNN model to capture the process dynamics. This is due to the fact that for the collected data (both for training and validation) the same correlation between the two inputs (F_c and T_{cf}) exists. It should be noted that the objective of this work is not to show the performance improvement of the PCA-RNN model over the RNN model for the same training and validation data, but to readily enable the use of the resultant model in MPC to ultimately compare the capability of the RNN-based MPC with the PCA-RNN-based MPC to handle the problem of set-point tracking.

4.3.2 PCA-RNN-based model predictive control

With an appropriate PCA-RNN model being available, this section presents a model predictive control algorithm utilizing this model, denoted as the PCA-RNN-based MPC. To this end, the proposed PCA-RNN-based identification technique is first applied to determine function 'f' in Eq. (4.6). Then the identified PCA-RNN model is embedded in MPC framework. Therefore, the scores, and subsequently control actions, at each sample time can be computed as follows:

$$\begin{aligned} \min_{t_k,\ldots,t_{k+P}} \quad & \sum_{j=1}^{P} \|\tilde{y}_{k+j} - y^{SP}_{k+j}\|^2_{Q_{y_p}} + \|t_{k+j} - t_{k+j-1}\|^2_{R_{dt}} \\ s.t. \quad & PCA - RNN\ based\ Predictive\ Model, \\ & \bar{u}_{min} \leq tP^T \leq \bar{u}_{max} \end{aligned} \tag{4.7}$$

where Q_{y_p} and R_{dt} are penalty matrices correspond to the output deviations from the set-points and the rate of change in the scores, respectively, $\bar{u}_{min} \in \mathbb{R}^{n_u}$ and $\bar{u}_{min} \in \mathbb{R}^{n_u}$ are corresponding mean-centered and scaled values of the lower and upper limits for the manipulated variables, and the rest of the parameters are as defined earlier. From Eq. (4.7), we see that the scores are the decision variables of this optimization problem. Therefore, applying PCA can be thought as a transformation that enables us to design the controller in the uncorrelated space of scores.

Remark 1. *The first objective and benefit of building and utilizing a PCA-RNN based model is that the MPC computes the scores instead of the inputs- with the inputs being computed by using the optimal score values. The inputs by design then retain the same correlation as in the training stage, thus helping maintain validity. The other benefit is computational. First off, there is dimensionality reduction in building the RNN model, thus alleviating the computational cost (which admittedly is not a big concern, given that the identification is off-line). The other compounded benefit though, is that the number of decision variables in the MPC reduces, and the predictive model in the MPC itself is slightly easier to compute- both resulting in a reduction in computation times compared to a classical RNN-MPC implementation (see the simulation results for a demonstration).*

4.3.3 Constrained RNN-based model predictive control

In this section, a remedy for standard RNN-based MPC approach is proposed. To this end, a new constraint forcing the manipulated inputs to move in a certain direction is proposed. In other words, this constraint is posed so as to maintain the correlations in the manipulated input similar to those existing in training data sample. This is achieved by first performing a PCA on the input space, and then, in the MPC implementation, still computing the original inputs, but enforcing the values of squared prediction error (SPE) to not breach a certain threshold.

Recall that the SPE-statistic determines the residual distance between an actual sample and its projection into the model plane. Therefore, the SPE of the ith vector of observed variables, $x_i \in \mathbb{R}^n$, can be expressed as follows:

$$SPE_i = \tilde{x}_i^T \tilde{x}_i = x_i^T (I - PP^T) x_i \tag{4.8}$$

where $\tilde{x}$ is the residual vector $(x_i - \hat{x}_i)$, and $\hat{x}_i \in \mathbb{R}^n$ is the reconstructed vector of the observation. $SPE_i = 0$ indicates that the observation is on the model plane.

Given the SPE-statistic, the MPC formulation of Eq. (4.5) can be modified using Eq. (4.9) as additional constraint on SPE values. The value of ε_i should be small enough to confine the movements of decision variables (manipulated inputs) to the PCA model plane, constructed from the input training data samples. This value can be determined by calculating the $95\% - 99\%$ confidence limit for SPE-statistic in the PCA model.

$$\begin{aligned} \min_{u_k,\ldots,u_{k+P}} \quad & \sum_{j=1}^{P} \|\tilde{y}_{k+j} - y^{SP}_{k+j}\|^2_{Q_{y_r}} + \|u_{k+j} - u_{k+j-1}\|^2_{R_{du}} \\ s.t. \quad & RNN\ based\ Predictive\ Model, \\ & SPE_i \leq \varepsilon_i, \\ & u_{min} \leq u \leq u_{max} \end{aligned} \tag{4.9}$$

Remark 2. *The constrained RNN-based MPC formulation, with the constraint set to zero, effectively reverts to the PCA-RNN-based MPC formulation. Such a formulation is invariably more computationally expensive compared to the PCA-RNN-based MPC- due to reasons outlined in Remark 1. It would also be more computationally expensive than say the RNN-based MPC (without the constraints), due to the presence of the additional constraint. The reasons for presenting this formulation are two-fold: 1) To recognize the existence of one alternative means of achieving what is achieved by the PCA-RNN-based MPC and 2) set it up so the re-identification based MPC of Section 4.4 can be easily presented.*

Remark 3. *Note that the present work considers first addressing the problem of model identification based on input data samples containing co-linearity, and then, using the identified model in the MPC framework. It should be noted that other kinds of correlations might exist in real-world data samples such as correlated outputs. In subsequent research, the method will be generalized to handle other kinds of correlations in the training and validation data sets.*

Remark 4. *The efficacy of the proposed PCA-RNN-based and constrained RNN-based MPCs in the presence of uncertainty and measurement noise can be evaluated for more complex systems, and remains outside the scope of the present work. For such implementations, a version of robust MPC or offset-free MPC technique, utilizing the embedded recurrent neural network-based models, can be utilized to handle plant-*

model mismatch. The key recognition is that regardless of the MPC formulation being utilized, the proposed approach is expected to lead to better outcomes due to inherently better handling the model identification problem.

4.3.4 Implementation of the PCA-RNN-based MPC to the motivating example

This section presents the application of the PCA-RNN-based MPC to the chemical reactor example described in Section 6.2. To this end, the PCA-RNN model is first identified using the proposed algorithm in Section 5.3. Then the model is employed in MPC formulation of Eq. (4.7) to compute the optimal values of scores. The values of parameters are considered as follows: sampling time $T_s = 30$ s, prediction horizon $P = 4$, $Q_{y_p} = [1000\ 0;\ 0\ 1000]$, and $R_{dt} = [1\ 0;\ 0\ 1]$. The output trajectories and manipulated inputs (computed by transforming the values of scores) are shown in Fig. 4.4.

In contrast to the RNN-based approach, the PCA-RNN-based MPC can handle the problem of set-point tracking for jacket temperature, where two correlated inputs (F_c) and (T_{cf}) are manipulated to reach desired set-points. The reason why the RNN-based MPC approach fails can be explained by analyzing the optimal values of inputs. Fig. 4.5 compares the SPE values obtained using the manipulated inputs in each specific approach with the SPE values corresponding to the PCA model constructed using training data. Note that the observations are not contiguous in time. As shown in Fig. 4.4, the MPC actions are calculated for 100 sampling instants (for each approach). As a result, the observations from 0 to 300 are the SPE values of the PCA model on training data and the observations from $300-400$, $400-500$, and $500-600$ are the SPE values related to the manipulated inputs obtained using the RNN-based MPC, the PCA-RNN-based MPC, and the constrained RNN-based MPC, respectively. As

can be seen, the SPE values of the manipulated inputs calculated using the RNN-based MPC are higher than zero (control limit in this case) for most observations, while these values are similar to the SPE values of the trained PCA model for the PCA-RNN-based MPC method. The manipulated inputs in the PCA-RNN-based MPC follow the trained PCA model plane resulting in fairly accurate prediction and subsequently good control.

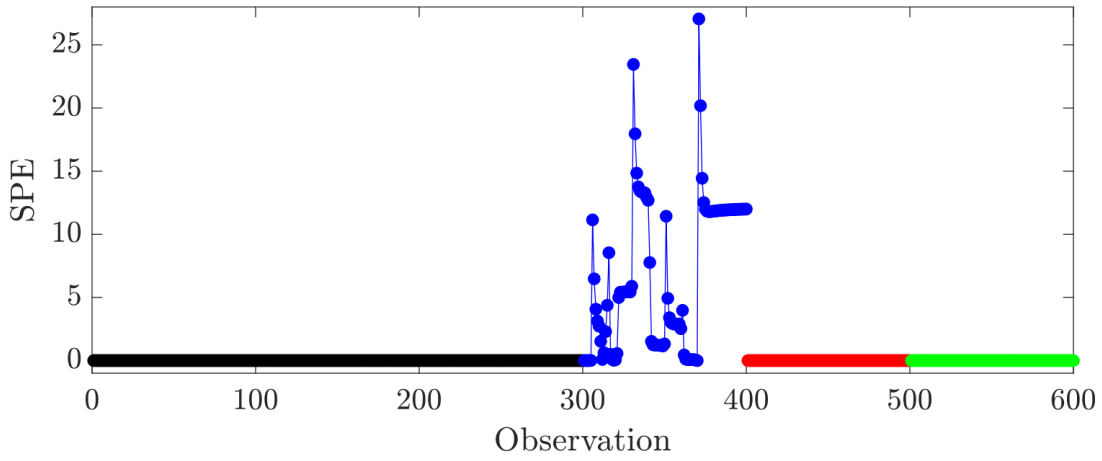


Figure 4.5: Comparison of the SPE values for training data (black circles), manipulated variables computed by RNN-based MPC (blue circles), manipulated variables computed by PCA-RNN-based MPC (red circles), and manipulated inputs computed by constrained RNN-based MPC (green circles).

4.3.5 Implementation of the constrained RNN-based MPC to the motivating example

The RNN model identified in Section 6.2 is employed in Eq. (4.9), along with the new constraints on SPE-statistic. The value of ε_i is chosen to be 10^{-5} with the other parameters same as before. The output trajectories and manipulated inputs are illustrated in Fig. 4.4. It can be observed that this method is successful in handling the problem of set-point tracking compared to the RNN-based MPC method. As shown in Fig. 4.5, the SPE values relevant to the manipulated inputs in this

method are close to zero (nearly similar to the SPE values corresponding to the trained PCA model). In addition, the performance of the predictive model utilized in this MPC implementation is evaluated and shown in Fig. 4.6. This figure compares the predicted values of the outputs (obtained by the models) with the actual process outputs by applying the manipulated inputs computed by all three MPC approaches (the manipulated inputs are shown in Fig. 4.4). It can be observed that the RNN model in the RNN-based MPC approach has poor predictions compared to the other methods, leading to poor control.

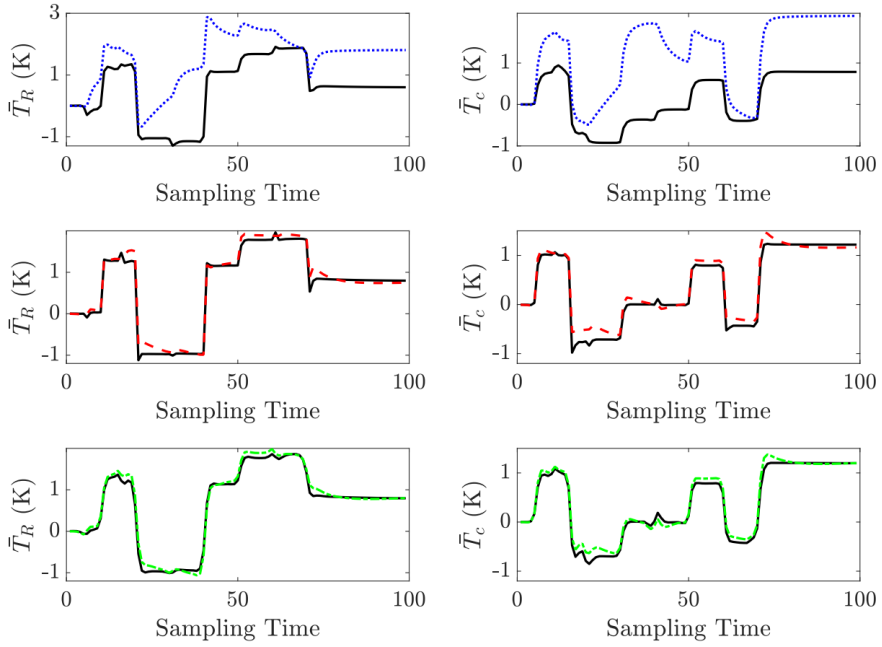


Figure 4.6: Model prediction results based on the MPC actions; measured variables (solid lines), RNN-based predictive model (blue dotted lines), PCA-RNN-based predictive model (red dashed lines), and constrained RNN-based predictive model (green dash-dotted lines).

For further analysis, the prediction capabilities of the RNN-model in the RNN-based MPC and the constrained RNN-based MPC approaches are evaluated, in each sampling instant, and shown in Fig. 4.7. In this figure, the process and predictive model outputs are compared at the 79^{th} sampling instant of the MPC optimization problem for the RNN-based MPC and constrained RNN-based MPC. In this direction, the

sequence of control actions over the prediction horizon ($P = 4$) are calculated and then applied to the process and the RNN model. Fig. 4.8 compares the manipulated inputs computed using the RNN-based MPC and the constrained RNN-based MPC.

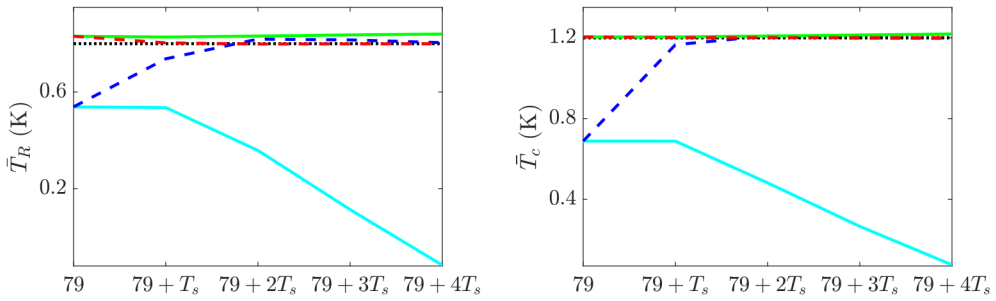


Figure 4.7: Comparison of the predicted values and the process output at 79^{th} sampling instant of the MPC optimization problem: process outputs (cyan solid lines) and the RNN model outputs (blue dashed lines) by applying the RNN-based MPC actions, process outputs (green solid lines) and the RNN model outputs (red dashed lines) by applying the constrained RNN-based MPC actions, and set-points (dotted lines).

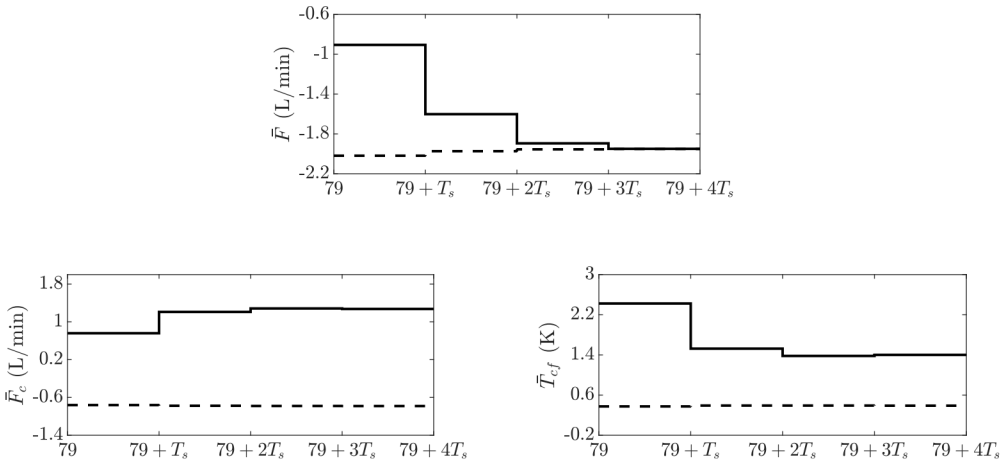


Figure 4.8: Comparison of the control actions at 79^{th} sampling instant of the MPC optimization problem: RNN-based MPC actions (solid lines) and constrained RNN-based MPC actions (dashed lines).

Recall that the RNN model utilized in both methods is the same. However, the manipulated inputs obtained using both methods are different due to the SPE constraint in the constrained RNN-based MPC approach which forced the inputs to move in a

certain direction. Consequently, as shown in Fig. 4.7, the RNN model in the constrained RNN-based MPC is able to predict (red dashed line) the process behavior (green solid line) with reasonable accuracy. However, the RNN model in the RNN-based MPC fails to predict (blue dashed line) the process behavior (cyan solid line) using the inputs calculated without considering the SPE constraint.

4.4 Constrained RNN-based model predictive control with re-identification

The previous section presented a couple of approaches that manage the plant-model mismatch issue by effectively making the inputs under the MPC follow the same correlation as the training data. While resulting in good control performance, such implementations might miss out on possible economic performance enhancements achievable by utilizing the full 'space' of movements of the manipulated inputs- which in turn requires a model that remains valid over the entire range of input moves. This in turn requires richer training data. Richer training data can be acquired in the following ways: 1) redesigning the original control structure to generate new data, or 2) to utilize the constrained RNN- based MPC to generate data that is richer, and yet does not result in significant performance degradation. The present manuscript addresses this problem by implementing the RNN-based MPC with relaxed SPE constraints, i.e., replacing ε with δ in Eq. (4.9), with $\delta > \varepsilon$. It is quite obvious that this causes offset because of poorer prediction of the RNN-based model. However, if the value of δ is determined appropriately, it not only does not degrade the controller performance significantly, but also help generate data samples with more variation that can be utilized in model re-identification stage.

4.4.1 Re-identification based MPC implementation results

The re-identification algorithm is considered in this section to re-build the new RNN model of CSTR example. As mentioned earlier, the variable ε is first replaced with the variable δ in Eq. (4.9). Then the value of δ should be chosen in a way that not only breaks the existing correlations, but also does not severely degrade the control performance. A value of $\delta = 2$ is chosen based on trial and error. It should be noted that one can introduce small changes with different magnitudes in this value, gradually over time, to generate richer data samples.

Given the new constraints on the value of δ, the constrained RNN-based MPC is implemented for CSTR example and the set-point tracking results and corresponding control actions are shown in Fig. 4.9. As expected, the MPC performance is slightly degraded by applying the new constraints. Recall that the objective is to break the existing co-linearity to provide richer data samples for model re-identification stage. Next, these new data samples were utilized to train the new RNN model and the model validation results are shown in Fig. 4.9. In the next step, the objective is to utilize the re-identified RNN model in the MPC framework and evaluate the efficacy of new approach compared to the previous one.

As mentioned earlier, the RNN model can predict fairly well when the manipulated inputs move in a certain direction, but sometimes they are required to move without considering a specific correlation to satisfy constraints and accomplish the control objective. One instance of this situation is considered here to illustrate the significance of re-identification in the MPC implementation. To this end, the constrained RNN-based MPC, as presented in Eq. (4.9), is utilized to make the process outputs track the set point changes as shown in Fig. 4.10. As can be seen, this controller can handle the problem of set-point tracking until the 30^{th} sampling time. The reason why this approach fails after the second change in the cooling jacket temperature set-point can

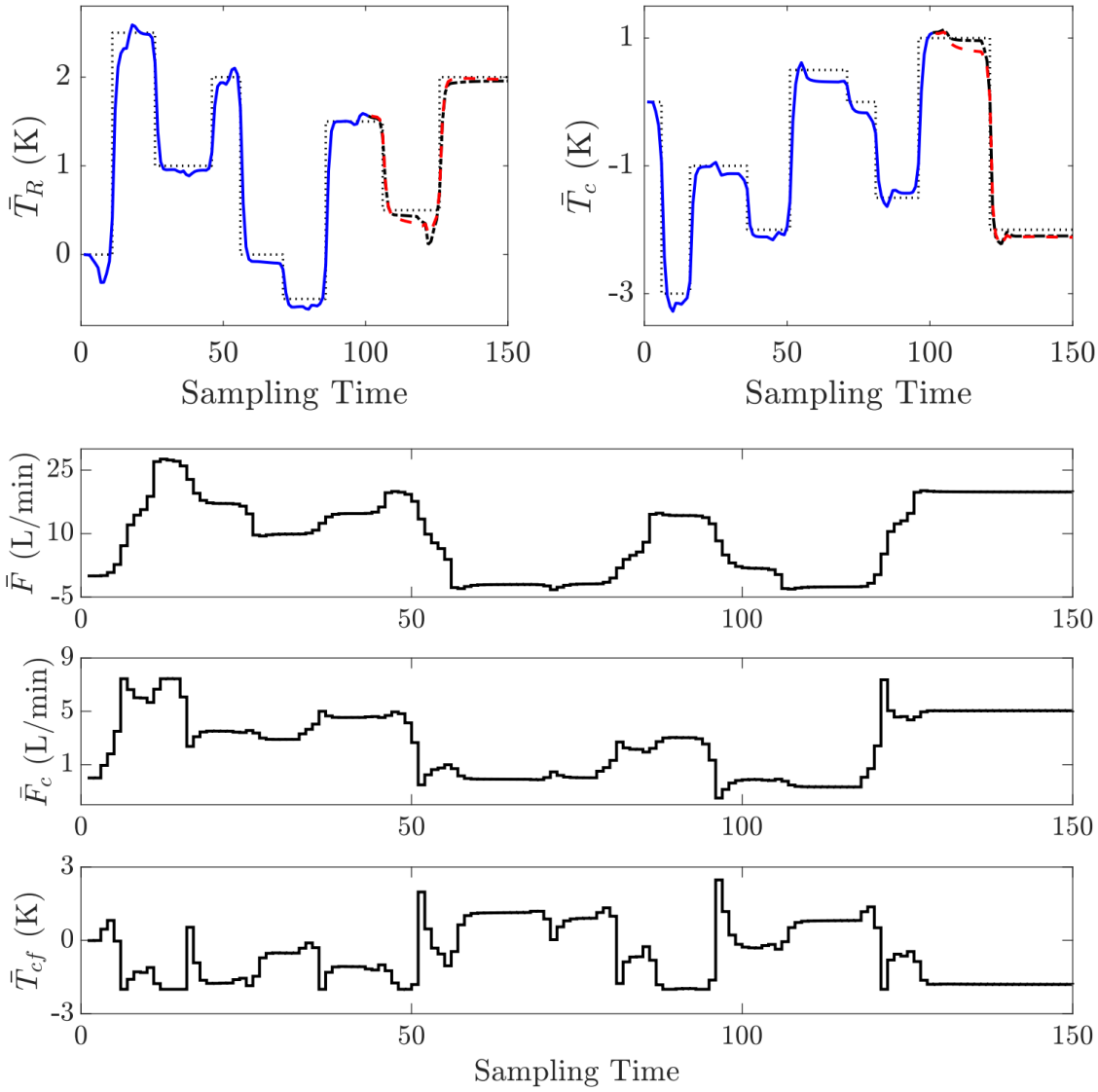


Figure 4.9: Model training data for re-identification and validation result: measured outputs and manipulated variables under the constrained RNN-based MPC with relaxed constraints (solid lines), validation data (black dash-dotted lines), RNN model prediction (red dashed lines), and set-points (dotted lines).

be explained by analyzing the values of manipulated variables in Fig. 4.10. It can be observed that the third manipulated input, T_{cf}, is saturated on the lower bound when the set-point changes from -1 to -2.5. This causes the second manipulated input, F_c, to be fixed at the certain value due to the SPE constraints. In this situation, the

controller is not able to drive the outputs to the set-points. Note that even if the second input is able to move so as to eliminate the offset, the model prediction will be degraded because the existing correlation is broken. Consequently, it is necessary to utilize more generalized model which can predict the system behavior in different range of inputs. As shown in Fig. 4.10, the MPC with the re-identified model can readily drive the outputs to the set-points.

Remark 5. *The dimensionality reduction achieved under the PCA-RNN based MPC helps not only the over-fitting issue, but, by virtue of reducing the resultant RNN, also has computational implications. For the simulation example, the total computation times required for different MPC implementations using Intel® Core(TM) i7-8700 at 3.20 GHZ and 8.0 GB of RAM, for 100 sampling times are 632 s, 392 s for the RNN-based and PCA-RNN-based MPCs, respectively (the RNN-based MPC being 61% more expensive than the PCA-RNN based MPC). The constrained RNN-Based MPC, on the other hand takes 1950 s, with the potential of improved closed-loop performance trading off against the increased computational time.*

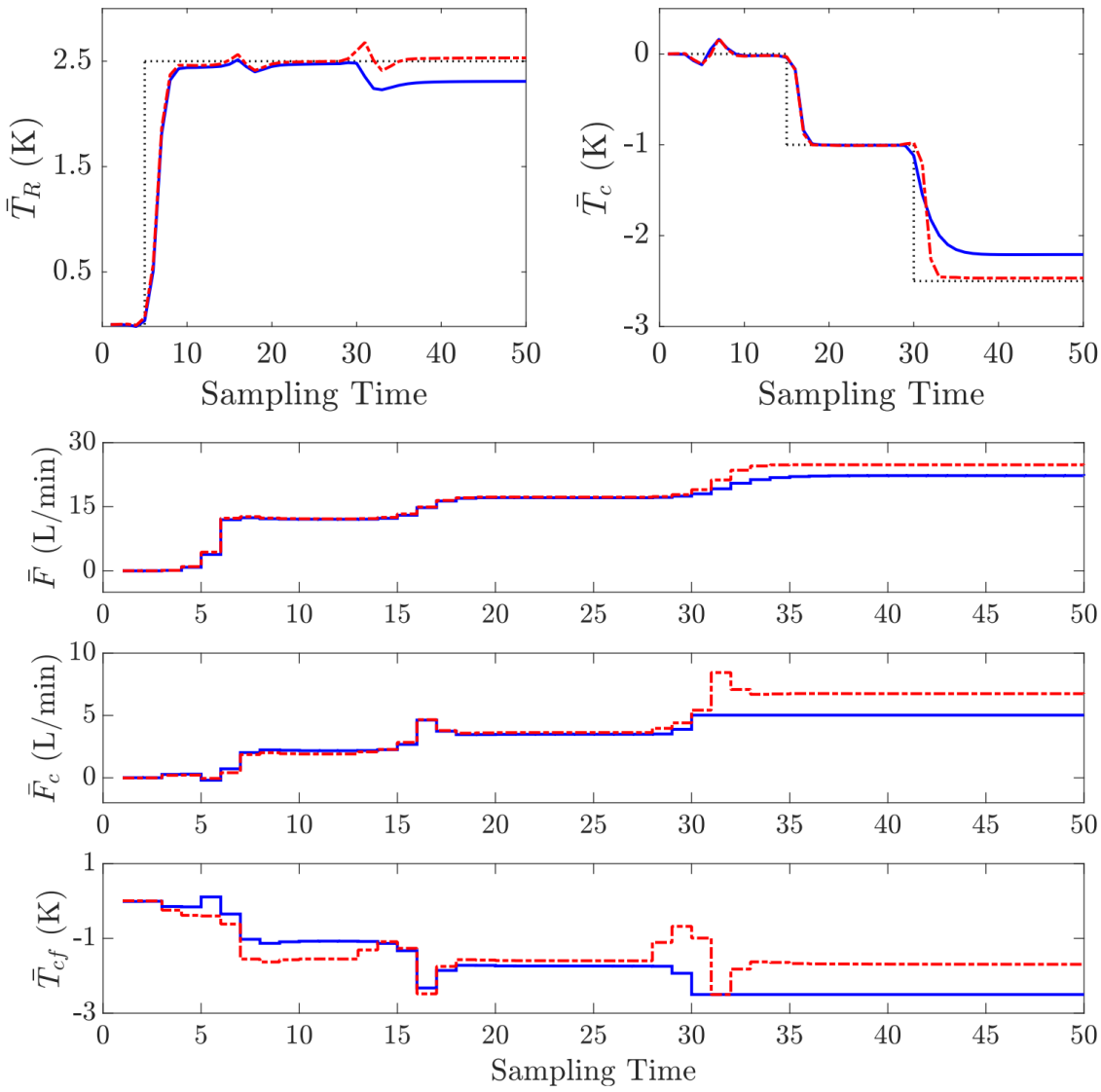


Figure 4.10: Comparison of the closed-loop trajectories for the measured variables and the profiles of the manipulated variables obtained from constrained RNN-based MPC (blue solid lines) and constrained RNN-based MPC using re-identified model (red dash-dotted lines).

4.5 Conclusion

This work considered the problem of developing the artificial neural network-based predictive models in the presence of correlated inputs to be employed in the MPC

implementations. A novel PCA-RNN-based MPC, along with the constrained RNN-based MPC approaches are developed to address the problem of model prediction and appropriately designed MPCs. PCA on input data samples is performed in the first approach and then scores are employed to build the PCA-RNN model. This model is then utilized in MPC implementation to calculate the optimal values of scores. In the second approach, new constraint on SPE-statistic is considered to ensure that input movements are confined to the PCA model plane to guarantee appropriate performance of the RNN predictive model. Finally, a new method is proposed to break the existing correlation in the computed MPC actions in order to generate richer data samples. This is performed by relaxing SPE constraint in the constrained RNN-based MPC approach. Next, the new data is utilized to re-identify the RNN model and then it is employed in MPC framework. The implementation of proposed approaches is illustrated using the chemical reactor example. The results show the effectiveness of the proposed approaches to handle the problem of set-point tracking compared with a nominal MPC design and the constrained RNN-based MPC approach.

Bibliography

Bhat, N. and McAvoy, T. J. (1990). Use of neural nets for dynamic modeling and control of chemical process systems. *Computers & Chemical Engineering*, 14(4-5):573–582.

Corbett, B. and Mhaskar, P. (2016). Subspace identification for data-driven modeling and quality control of batch processes. *AIChE Journal*, 62(5):1581–1601.

Darby, M. L. and Nikolaou, M. (2009). Multivariable system identification for integral controllability. *Automatica*, 45(10):2194–2204.

Darby, M. L. and Nikolaou, M. (2014). Identification test design for multivariable model-based control: an industrial perspective. *Control Engineering Practice*, 22:165–180.

Drgoňa, J., Picard, D., Kvasnica, M., and Helsen, L. (2018). Approximate model predictive building control via machine learning. *Applied Energy*, 218:199–216.

Du, M. and Mhaskar, P. (2014). Isolation and handling of sensor faults in nonlinear systems. *Automatica*, 50(4):1066–1074.

Eastment, H. T. and Krzanowski, W. J. (1982). Cross-validatory choice of the number of components from a principal component analysis. *Technometrics*, 24(1):73–77.

Flores-Cerrillo, J. and MacGregor, J. F. (2005). Latent variable mpc for trajectory tracking in batch processes. *Journal of process control*, 15(6):651–663.

Forssell, U. and Ljung, L. (1999). Closed-loop identification revisited. *Automatica*, 35(7):1215–1241.

Godoy, J. L., González, A. H., and Normey-Rico, J. E. (2016). Constrained latent

variable model predictive control for trajectory tracking and economic optimization in batch processes. *Journal of Process Control*, 45:1–11.

Golshan, M., MacGregor, J. F., Bruwer, M. J., and Mhaskar, P. (2010). Latent variable model predictive control (lv-mpc) for trajectory tracking in batch processes. *Journal of process control*, 20(4):538–550.

Govindhasamy, J. J., McLoone, S. F., Irwin, G. W., French, J. J., and Doyle, R. P. (2005). Neural modelling, control and optimisation of an industrial grinding process. *Control Engineering Practice*, 13(10):1243–1258.

Hale, E. T. and Qin, S. J. (2002). Subspace model predictive control and a case study. *In Proceedings of the American Control Conference*, 6:4758–4763.

Huang, B., Ding, S. X., and Qin, S. J. (2005). Closed-loop subspace identification: an orthogonal projection approach. *Journal of process control*, 15(1):53–66.

Janakiraman, V. M., Nguyen, X., and Assanis, D. (2013). Nonlinear identification of a gasoline hcci engine using neural networks coupled with principal component analysis. *Applied Soft Computing*, 13(5):2375–2389.

Kheradmandi, M. and Mhaskar, P. (2018a). Data driven economic model predictive control. *Mathematics*, 6(4):51.

Kheradmandi, M. and Mhaskar, P. (2018b). Model predictive control with closed-loop re-identification. *Computers & Chemical Engineering*, 109:249–260.

Kittisupakorn, P., Thitiyasook, P., Hussain, M. A., and Daosud, W. (2009). Neural network based model predictive control for a steel pickling process. *Journal of Process Control*, 19(4):579–590.

Kusiak, A. and Xu, G. (2012). Modeling and optimization of hvac systems using a dynamic neural network. *Energy*, 42(1):241–250.

Ljung, L. (1999). *System Identification: Theory for the User (2nd edition)*. Prentice Hall PTR, Upper Saddle River, New Jersey.

Lu, Y., Cohen, I., Zhou, X. S., and Tian, Q. (2007). Feature selection using principal feature analysis. *In Proceedings of the ACM international conference on Multimedia*, pages 301–304.

Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, 1(1):4–27.

Nikravesh, M., Farell, A. E., and Stanford, T. G. (2000). Control of nonisothermal cstr with time varying parameters via dynamic neural network control (dnnc). *Chemical Engineering Journal*, 76(1):1–16.

Qin, S. J. and Ljung, L. (2003). Closed-loop subspace identification with innovation estimation. *IFAC Proceedings Volumes*, 36(16):861–866.

Ramchandran, S. and Rhinehart, R. R. (1995). A very simple structure for neural network control of distillation. *Journal of Process Control*, 5(2):115–128.

Sadeghassadi, M., Macnab, C. J., Gopaluni, B., and Westwick, D. (2018). Application of neural networks for optimal-setpoint design and mpc control in biological wastewater treatment. *Computers & Chemical Engineering*, 115:150–160.

Shahnazari, H., Mhaskar, P., House, J. M., and Salsbury, T. I. (2019). Modeling and fault diagnosis design for hvac systems using recurrent neural networks. *Computers & Chemical Engineering*, 126:189–203.

Shardt, Y. A. and Huang, B. (2011a). Closed-loop identification condition for armax models using routine operating data. *Automatica*, 47(7):1534–1537.

Shardt, Y. A. and Huang, B. (2011b). Closed-loop identification with routine operating data: Effect of time delay and sampling time. *Journal of Process Control*, 21(7):997–1010.

Shardt, Y. A., Huang, B., and Ding, S. X. (2015). Minimal required excitation for closed-loop identification: Some implications for data-driven, system identification. *Journal of Process Control*, 27:22–35.

Shariff, H. M., Rahiman, M. H. F., and Tajjudin, M. (2013). Nonlinear system identification: Comparison between prbs and random gaussian perturbation on steam distillation pilot plant. *In 2013 IEEE 3rd International Conference on System Engineering and Technology*, pages 269–274.

Shi, R. and MacGregor, J. F. (2001). A framework for subspace identification methods. *In Proceedings of the American Control Conference*, 5:3678–3683.

Song, F., Guo, Z., and Mei, D. (2010). Feature selection using principal component analysis. *In international conference on system science, engineering design and manufacturing informatization*, 1:27–30.

Wang, Y., Velswamy, K., and Huang, B. (2017). A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems. *Processes*, 5(3):46.

Wold, S. (1978). Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics*, 20(4):397–405.

Wu, Z. and Christofides, P. D. (2019). Economic machine-learning-based predictive control of nonlinear systems. *Mathematics*, 7(6):494.

Wu, Z., Tran, A., Rincon, D., and Christofides, P. D. (2019a). Machine learning-based predictive control of nonlinear processes. part i: Theory. *AIChE Journal*, 65:e16729.

Wu, Z., Tran, A., Rincon, D., and Christofides, P. D. (2019b). Machine-learning-based predictive control of nonlinear processes. part ii: Computational implementation. *AIChE Journal*, 65:e16734.

Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International journal of forecasting*, 14(1):35–62.

Zhang, Z., Wu, Z., Rincon, D., and Christofides, P. D. (2019). Real-time optimization and control of nonlinear processes using machine learning. *Mathematics*, 7(10):890.

Chapter 5

Artificial neural network based model predictive control: Implementing achievable set-points

The contents of this chapter have been published in the *AIChE* Journal.

- Hesam Hassanpour, Brandon Corbett, and Prashant Mhaskar, Artificial neural network based model predictive control: Implementing achievable set-points. *AIChE*, 2022, 68(1), e17436. DOI: 10.1002/aic.17436

Abstract

This paper addresses the problem of determining achievable set-points for artificial neural network (ANN)-based model predictive control (MPC) designs. In particular, this work considers a case where a first-principles model may not be readily available for a nonlinear process, while sufficient closed-loop data containing possibly correlated outputs is available, such that an ANN-based model that captures the nonlinear dynamics reasonably well can be identified. The paper addresses implementation aspects with such an ANN-based MPC design- specifically that of ensuring that achievable set-points are prescribed to the MPC. The key idea is to perform principal component analysis (PCA) on the training data in order to recognize existing collinearity and determine the upper confidence limit of squared prediction error (SPE) statistic. An optimization problem subject to the SPE constraint is then defined to calculate the achievable set-points, that can in turn be provided to an MPC design. The efficacy of the proposed approach is illustrated via implementations on a chemical reactor example. The results reveal the superior tracking performance of MPC using the achievable set-points over the case where arbitrarily prescribed set-points are used in the MPC implementations.

5.1 Introduction

The problem of control system design for chemical processes faces several challenges such as nonlinearity, multi-variable interactions, process constraints, and uncertainties. Model predictive control (MPC) is a widely implemented control strategy that can cope with these challenges (Qin and Badgwell (2003); Rawlings and Mayne (2009)). For MPC design, developing a model that can predict the process behaviour with reasonable accuracy is an essential task. In this regard, various modeling techniques such as first-principles models and data-driven modeling approaches have been

developed and used for MPC implementations.

Due to the challenges that exist in developing and maintaining first-principles models and with the recent development of computing power and data storage, data-driven and machine learning techniques have received significant attention. A number of data-driven methods such as subspace identification (Moonen et al. (1989); Huang et al. (2005); Qin and Ljung (2003)), artificial neural networks (ANN) (Bhat and McAvoy (1990); Zhang et al. (1998); Prasad and Bequette (2003); Jalanko et al. (2021)), and latent variable methods such as principal component analysis (PCA) and projection to latent structures (PLS) (Kresta et al. (1994)) have been proposed and used for system identification. Data-driven models have also been utilized for MPC implementations. Subspace-based identification techniques are used to model process dynamics for being employed in MPC design (Hale and Qin (2002); Corbett and Mhaskar (2016)). Latent variable model predictive control (LV-MPC) algorithms are also developed and successfully used to handle trajectory tracking and disturbance rejection problems (Golshan et al. (2010); Godoy et al. (2016)).

Machine learning techniques such as ANN have received significant attention in the context of system identification and process control due to their abilities to capture nonlinearities (Kittisupakorn et al. (2009); Mohanty (2009); Ferreira et al. (2012); Wang et al. (2017)) Among several machine learning models, recurrent neural networks (RNN) have demonstrated excellent performances in modeling dynamic behavior of complex nonlinear systems (Janakiraman et al. (2013); Wong et al. (2018); Koschwitz et al. (2018); Wu et al. (2019)). In order to develop RNN models with reasonably good prediction performance, a significant amount of data with high quality is required. However, missing data, high dimensionality, redundancy, and collinearity in most historical process data lead to difficulties in extracting useful information. To address these issues, a number of data mining and pre-processing strategies have been proposed (Han et al. (2011)).

Data collection strategies and conditions under which data is collected can significantly impact the type of data (relationship between features), and therefore, the performance of the models to be developed. Unstable open-loop systems and economic and safety concerns have motivated the use of closed-loop data for system identification (Forssell and Ljung (1999)). Several methods have also been proposed based on minimal required excitation of reference signals (Shardt et al. (2015)). In addition, due to the presence of a large number of features in historical datasets, the choice of proper features plays a key role in developing a model with high accuracy. Due to the possible existence of irrelevant and redundant (highly correlated) features in the historical data, feature selection and extraction strategies have been commonly used as a pre-processing step for developing most machine learning models (Ayinde and Zurada (2017); Bellemans et al. (2018)). The use of feature extraction methods also helps reduce computational complexity, avoid over-fitting, and finally improve prediction performances of the models.

PCA is one of the widely used tools for extracting useful information and reducing the dimensionality of data. A number of machine learning models have been developed that use PCA as a pre-processing tool (Yan et al. (2016); Drgoňa et al. (2018)). In Zhan and Chong (2021), it is shown that applying PCA for feature extraction and using principal components for clustering analysis can enhance the performance to distinguish between the clusters that exist in the datasets. In Janakiraman et al. (2013), PCA is used for feature extraction to obtain a smaller set of uncorrelated inputs. These inputs are then used to train the neural network models. It is demonstrated that doing PCA enables reducing the computational time required to identify the optimal hyper-parameters. In Kristjanpoller and Minutolo (2018), PCA is used to calculate the principal components, and then, the components which are uncorrelated, are used to facilitate the process of training the neural network models.

Recently, there has been increasing interest in the use of RNN-based models in MPC

implementations (Temeng et al. (1995); Sadeghassadi et al. (2018); Wong et al. (2018); Hassanpour et al. (2020); Wu et al. (2020)). In order to train an RNN model with fairly good prediction accuracy, a significant amount of data with sufficient variation is required due to the existence of a large number of parameters needed to be identified. In addition, the problem of over-fitting remains a challenging problem in the context of neural network modeling. The negative effect of the over-fitting problem may get exacerbated when the training datasets are not informative, or they contain redundant information. This problem is likely to occur when the datasets are collected under closed-loop operations, where collinearities are induced between some input and output variables that must be appropriately accounted for in the modeling stage. Implementation of an MPC, where the inputs are moved independently, may necessitate predictions by the model in regions where it was not trained, leading to poor closed-loop performance. In Hassanpour et al. (2020), a method is proposed for the case where datasets contain correlated inputs. PCA on input space is performed and the principal components are then used to develop an RNN model. The MPC is then designed to calculate the optimal scores and associated inputs. Alternative solution is also proposed in this reference that considers applying a new constraint on squared prediction error (SPE) in order to force the input movements into following the same collinearity seen in the training data. The existing results, however, do not address the situation where the process outputs could be correlated either due to the control structure or process itself. In this situation, if the collinearity is not properly addressed, the prescribed set-points assuming the outputs to be independent may not be achieved by MPC, either due to fundamental process limitations, or the resultant mismatch when the MPC tries to use the model in regions where it was not identified. Thus, in the case where the dataset contains correlated outputs, it is essential to first recognize the collinearity in order to be able to calculate the achievable set-points. This in turn leads to maintaining model validity, which further results in the successful implementation of MPC.

Motivated by these considerations, this manuscript presents a method for extracting the achievable set-points when the datasets contain correlated outputs. The key idea is to first apply PCA on the output variables in order to recognize the collinearity. Next, SPE is calculated and the upper confidence limit for this statistic is determined. An optimization problem is then defined to minimize the sum of squared error between the desired and achievable set-points subject to the SPE constraint. This makes it possible to extract the achievable set-points from the arbitrarily chosen set-points. The calculated achievable set-points are then used in implementing the RNN-based MPC. To illustrate the robustness of the proposed approach against plant-model mismatch, especially in light of the fact that RNNs that give similar behavior during training/validation/testing could give quite different predictions in a new operating region, the following scenario is considered: different RNN models, with almost similar prediction performance, are used in the MPC implementations to show that the closed-loop responses can vary if the arbitrarily prescribed set-points are used. However, the use of the calculated achievable set-points makes all the RNN-based MPC stabilize the system around the calculated set-points. The rest of the manuscript is organized as follows: Section 6.2 presents a continuous stirred tank reactor (CSTR) under cascade control as the motivating example, followed by an overview of recurrent neural network and principal component analysis. A standard RNN-based MPC for the motivating example is then implemented to illustrate the specific problem being addressed. Section 5.3 presents the proposed approach for calculating the achievable set-points. The application of the proposed approach to the motivating example is then presented in Section 5.4. The use of the achievable set-points in the RNN-based MPC is seen to address possible oscillatory behaviors resulting from using the arbitrary desired set-points. Finally, concluding remarks are presented in Section 6.5.

5.2 Preliminaries

To provide the motivation for proposed approach, a Continuous Stirred Tank Reactor (CSTR) example, under cascade control, is presented. Then, a brief review of the application of recurrent neural networks (RNN) for system identification is followed by a brief description of principal component analysis (PCA).

5.2.1 Motivating Example: Continuous Stirred Tank Reactor (CSTR)

Consider a nonlinear CSTR example (Du and Mhaskar (2014)), where an irreversible elementary exothermic reaction (A $\rightarrow$ B) takes place, the mathematical model describing the dynamics takes the following form:

$$\begin{aligned}
\dot{C_A} =& \frac{F}{V}(C_{A0} - C_A) - k_0 e^{-E/RT_R} C_A \\
\dot{T_R} =& \frac{F}{V}(T_0 - T_R) + \frac{(-\Delta H)}{\rho c_p} k_0 e^{-E/RT_R} C_A - \frac{UA}{\rho c_p V}(T_R - T_c) \\
\dot{T_c} =& \frac{F_c}{V_c}(T_{cf} - T_c) + \frac{UA}{\rho_c c_{pc} V_c}(T_R - T_c)
\end{aligned} \tag{5.1}$$

where C_A is the concentration of reactant A, and T_R and T_c are the temperature in the reactor and the temperature in the cooling jacket, respectively. F is the volumetric flow rate to the reactor with temperature T_0 and concentration C_{A0} and F_c is the cooling stream flow rate to the jacket with temperature T_{cf}. The values of the process parameters and the steady-state values are listed in Table 6.1 and Table 6.2, respectively.

Table 5.1: Parameter values for the CSTR example

Parameter	Value	Unit
F	14.60	L/min
V	100	L
V_c	20	L
k_0	7.20×10^{10}	min^{-1}
ΔH	-5×10^4	J/mol
ρ	1000	g/L
ρ_c	1000	g/L
c_p	0.239	J/(g K)
c_{pc}	4.20	J/(g K)
UA	5×10^4	J/(min K)
E/R	8750	K
T_0	350	K
T_{cf}	293	K

Table 5.2: Steady-state values of the manipulated variables and the process outputs

Parameter	Value	Unit
C_{A0_s}	1.00	mol/L
F_{c_s}	4.70	L/min
C_{A_s}	0.50	mol/L
T_{R_s}	324.82	K
T_{c_s}	315.81	K

In the CSTR example, it is assumed that the data is collected from the operation of the CSTR where the concentration of species A (C_A) is controlled by manipulating the inlet concentration (C_{A0}) using a PI controller (single-loop). The temperature of the reactor (T_R) is also controlled by manipulating the cooling stream flow rate (F_c) using a cascade control structure, with two PI controllers in the inner and outer loops. The outer-loop controller adjusts the set-points of inner loop (T_c^{SP}). The inner-loop controller then manipulates the cooling stream flow rate to control reactor temperature. Note that the manipulated variables and the process outputs are represented in deviation variable form, i.e. $\bar{C}_A = C_A - C_{A_s}$, $\bar{T}_R = T_R - T_{R_s}$, and $\bar{T}_c = T_c - T_{c_s}$ for the output deviation variables and $\bar{C}_{A0} = C_{A0} - C_{A0_s}$, $\bar{F}_c = F_c - F_{c_s}$ for the input deviation variables. PI control actions are calculated as follows:

$$U(t) = K_c \left(E(t) + \frac{1}{T_I} \int_0^t E(t')dt' \right) \tag{5.2}$$

where U is the control signal, E is the error between the set-points and measured outputs, K_c is the proportional gain, and T_I is the integral time constant. The tuning parameters for the PI controllers are listed in Table 6.3.

Table 5.3: Tuning parameters for PI controllers

PI controller	Gain	Integral time constant
Single-loop	1.09	4.32
Outer-loop	0.11	2.31
Inner-loop	-1.95	1.02

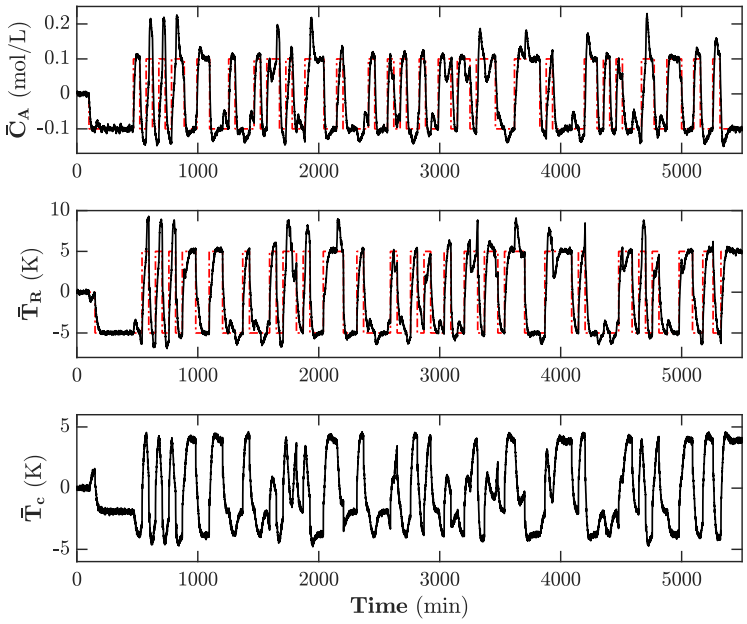
To reflect reality, the process outputs are corrupted with measurement noise. The noise is considered to be Gaussian white noise with mean μ and standard deviation σ as listed in Table 5.4. A moving average filter is also utilized to smooth the noisy measurement signal before it is used in the PI controller. Closed-loop data is generated by simulating the CSTR example. To generate training, validation, and test datasets, set-point changes in $\bar{C}_A$ and $\bar{T}_R$ are applied according to the pseudo random binary sequence (PRBS) signals, shown in Fig. 5.1(a). The generated data corresponding to the process outputs and the manipulated variables are shown in Fig. 6.1.

Table 5.4: Noise distribution parameters

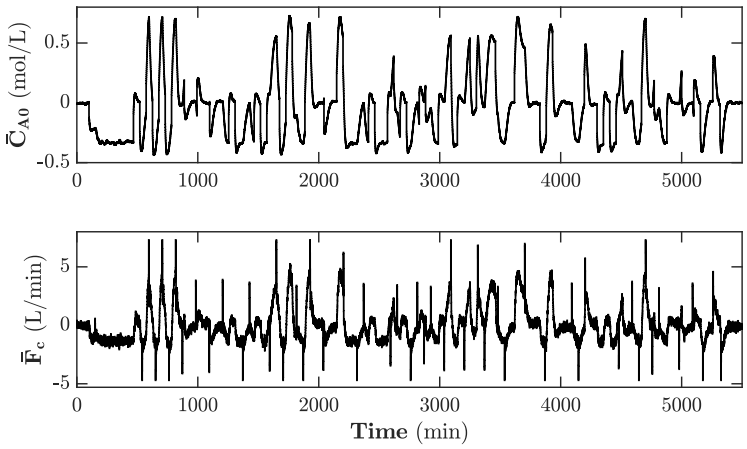
Output variable	μ	σ	Unit
C_A	0	0.003	mol/L
T_R	0	0.100	K
T_c	0	0.100	K

Having this data available for training, the next section reviews an RNN-based approach for model identification.

Remark 1. *Note that the closed-loop simulations are performed to generate data in*



(a)



(b)

Figure 5.1: Closed-loop data for system identification: (a) Process outputs (solid black lines) and set-points (dash-dotted red lines) (b) Manipulated inputs.

this work. However, in situations where there is the flexibility to run open-loop test where inputs are independently perturbed, the present approach would not be necessary. In general, there are a number of reasons for utilizing (generating) closed-loop

data in practice such as unstable open-loop systems, economic problems (open-loop datasets are more expensive to obtain in real plants), and safety concerns. For these reasons and in order to reflect practical considerations, it was decided to use closed-loop simulations under PI controllers, which are widely used in industries to maintain the process outputs at desired set-points, to generate data for system identification. Note that in this situation (closed-loop conditions), some collinearities between process inputs and outputs could be induced. Therefore, an appropriate pre-processing steps must be utilized to eliminate/recognize the collinearity before doing system identification and implementing any optimal control strategies such as MPC.

5.2.2 Recurrent neural networks for system identification

Recurrent neural networks (RNN) have been utilized in a number of applications to model nonlinear dynamic systems for being used in control designs (Hassanpour et al. (2020); Alhajeri et al. (2021); Wu et al. (2021)), optimization (Kusiak and Xu (2012)), and fault diagnosis (Han et al. (2020)). Unlike feedforward neural networks, the presence of delayed feedback loops in the RNN structure provides the network with a memory, enabling identifying dynamic behaviors of the systems. Different types of RNN such as nonlinear autoregressive network with exogenous inputs (NARX), long short-term memory (LSTM), bidirectional RNN, and gated recurrent unit (GRU) have been developed for different applications such as time series prediction and natural language processing.

In this work, NARX neural network modeling is used to model dynamic behavior of the system. The key idea of the proposed method remains applicable with other types of the RNN models. Mathematically, a NARX model is described as follows:

$$y(k) = f[y(k-1), ..., y(k-n_y), u(k-1), ..., u(k-n_u)] \tag{5.3}$$

where $y \in \mathbb{R}^l$ and $u \in \mathbb{R}^m$ represent the vector of measured outputs and manipulated inputs, respectively. n_y and n_u represent the number of lagged outputs and inputs required for prediction. As shown in Eq. (5.3), the next value of the dependent variables $y(k)$ is regressed on the previous values of the dependent and independent (exogenous) variables by NARX models. To approximate the nonlinear mapping function $f(\cdot)$, a multilayer perceptron (MLP), a class of feedforward neural networks, can be used, resulting in a NARX neural network model (Narendra and Parthasarathy (1990)). A schematic of a NARX neural network is shown in Fig. 5.2.

Note that the NARX neural network model is trained in order to perform one-step ahead prediction using past input and output measurements. After training, the NARX model is used to perform multi-step ahead predictions in the validation and test stages. To that end, assuming the initial values of the inputs and outputs are available (as the initial conditions), these values are fed to the trained NARX model to predict the outputs in the first sampling time. The predicted outputs in this step, together with the inputs, are then used to predict the outputs in the second sampling time. Subsequently, these predicted values are used to predict the outputs in the third sampling time. This process is repeated in order to do multi-step ahead predictions of the outputs.

RNN based modeling of the motivating example

In this section, NARX-based models are developed for the motivating example. To this end, the generated data is first divided into the training and test data. The first

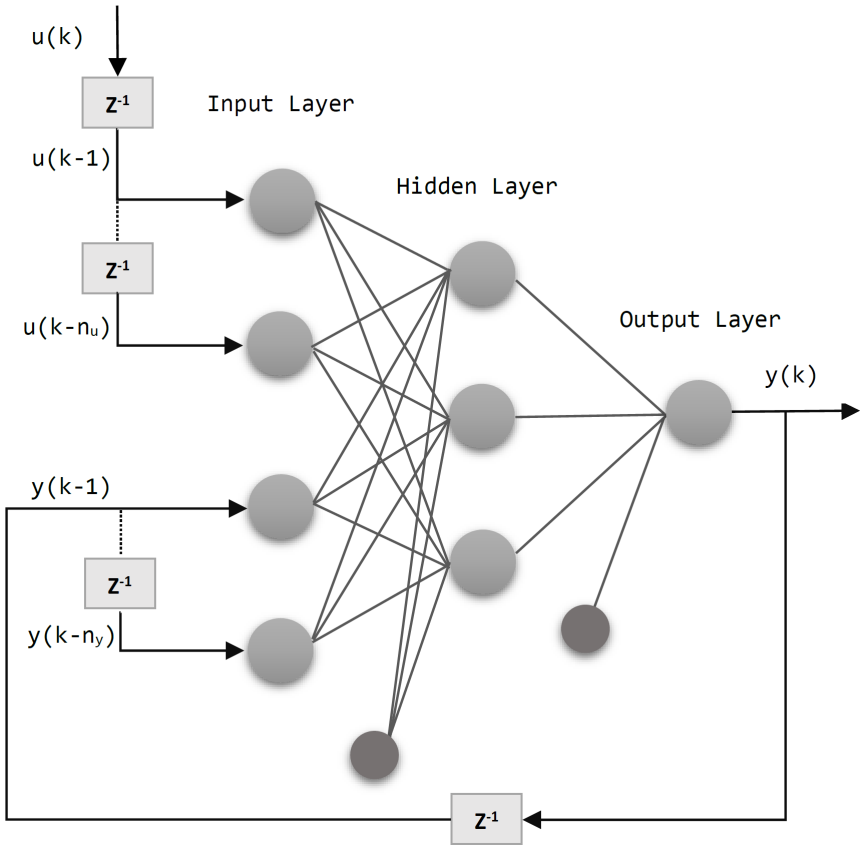


Figure 5.2: Schematic of a NARX neural network.

10000 data samples are considered as the training data and the last 1000 data samples are used as the test data to evaluate the performance of the trained models on the unseen data. In addition, a portion of the training data containing the first 8000 data samples is used to fit the model and the last 2000 data samples are considered as the validation data in order to provide an unbiased assessment of the model fit on the training data while tuning the model hyperparameters.

The input and output data samples are first normalized (mean-centered and scaled to unit variance), and then fed to a NARX network in MATLAB Machine Learning and Deep Learning toolbox. In the NARX network, different hyperparameters such as the number of hidden layers, the number of neurons in each layer, activation functions, and the number of lagged inputs and outputs need to be tuned. To this end, different

structures with 1 and 2 hidden layers, including different number of neurons, and two activation functions which are *sigmoid* and *tanh* are considered. Different number of lagged information (1, 2, and 3) is also used. Linear activation function is also used for the output layers and Bayesian regularization backpropagation algorithm that updates the model parameters (weights and biases) based on Levenberg-Marquardt optimization is selected. Depending on the different initialization of the weights and biases, the results can be different each time a neural network model is trained. Therefore, several runs for each case, mentioned above, are conducted to obtain some insights into the choice of the proper model hyperparameters. In order to quantify the prediction error, scaled root mean square error (SRMSE) metric is used, given by Eq. (6.5):

$$\text{SRMSE} = \sum_{i=1}^{l} \frac{\sqrt{\frac{\sum_{j=1}^{n} (y_{i,j} - \hat{y}_{i,j})^2}{n}}}{\sigma_i} \tag{5.4}$$

where $y_{i,j}$ and $\hat{y}_{i,j}$ are the j^{th} observation (real value) corresponding to the i^{th} output and its prediction, respectively, and σ_i represents the standard deviation of the i^{th} output. l and n also represent the number of outputs and the number of observations, respectively. This metric is used to evaluate the prediction performance of each trained model using the validation data. The results reveal that the model with 1 hidden layer including 5 neurons using *tanh* activation function has the best performance among all considered structures. In order to further evaluate the model performance based on this structure, the model is trained 100 times. The values of SRMSE for all these trained models are shown in Fig. 5.3. As mentioned earlier, different results can be obtained in each neural network training due to different initializations of the weights and biases and non-convex nature of the optimization problem that is trying to calculate the optimal values of these parameters. The asterisks, in Fig. 5.3(b), also

represent the models with the best prediction performance. The values of SRMSE for these models in the validation stage are shown in Table 5.5. The performance of these three models is also evaluated on the unseen data (test data). The SRMSE values corresponding to these models in the test stage are shown in Table 5.5. In addition, the predictions obtained using these models using the test data are shown in Fig. 5.4. As demonstrated, the NARX network models are able to predict the values of the outputs with reasonable accuracy. These three models are then used in the next step to implement the RNN-based model predictive control.

Table 5.5: SRMSE of the validation and test stages

Model	SRMSE (validation)	SRMSE (test)
RNN_1	0.8419	1.0670
RNN_2	0.8204	1.0091
RNN_3	0.7855	0.8021

Remark 2. *Note that the key idea in the present work is to use NARX network modeling technique in order to build a dynamic model of the process, and then, implementing MPC to evaluate its performance based on the arbitrarily prescribed and achievable set-points. However, other types of RNN modeling such as LSTM models can be developed and used in the MPC implementations, as long as they are able to accurately capture the process dynamics.*

Remark 3. *In practice, it is not easy to realize the trained model is over-fitted. Thus, it is essential to make sure that the model is working properly before employing that in production. This can be achieved by analyzing the model performance on unseen (new) data. Therefore, in the process of developing a neural network model (or any data-driven models), it is useful to divide the dataset into three categories including training data, validation (hold-out) data, and test (unseen) data. The model is trained only using the training data. The validation data is used in order to evaluate the progress and draw conclusions to improve the model ability. After analyzing the model performance on the validation data, the best (final) model can be selected for being*

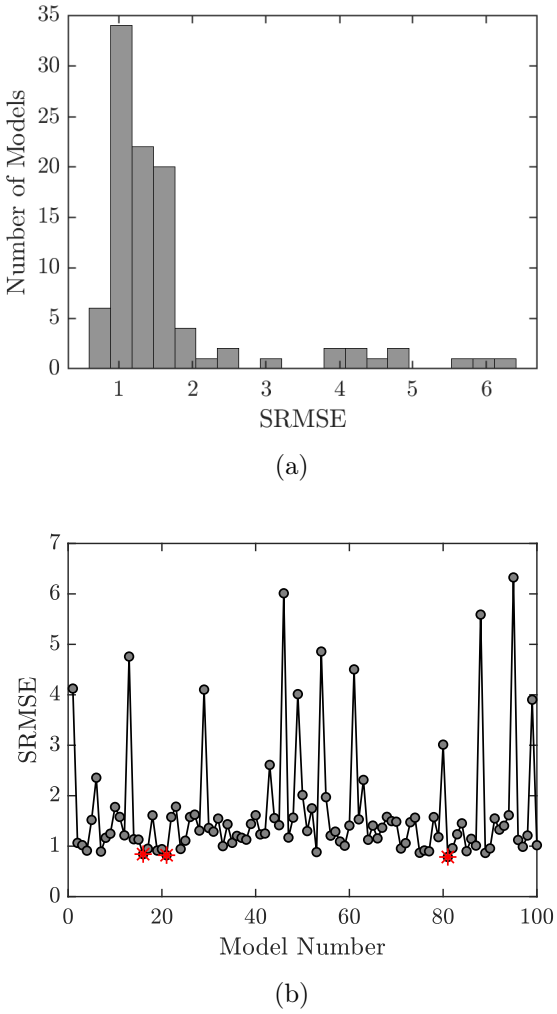


Figure 5.3: (a) Histogram of SRMSE values associated with the predictions obtained based on 100 trained models using validation data, (b) SRMSE plot (asterisks represent the three models with the smallest SMRSE).

tested on the unseen data. This allows us to assess how well the prediction performance of the model is on new data.

Remark 4. *As mentioned in the previous remark, after training the neural network models, the validation data is used to provide us with an unbiased opinion on how well the trained models work in order to choose the final model. Note that, the final model needs to be selected by analyzing the prediction performance of different trained models on the validation data (not test data). In this work, the three best models*

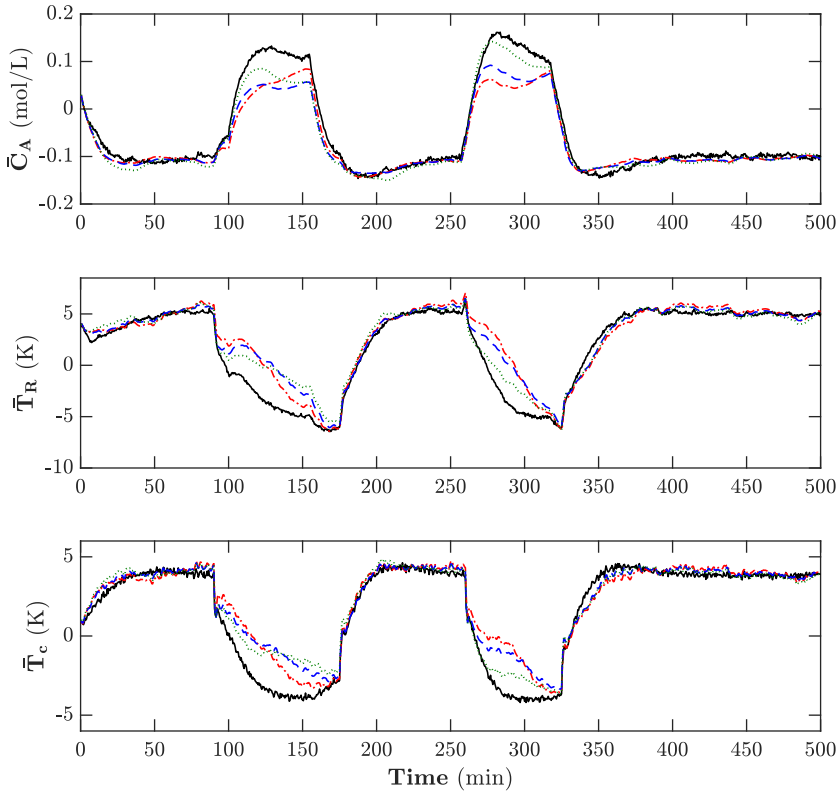


Figure 5.4: Model test results: measured outputs (solid black lines), RNN_1 prediction (dashed blue lines), RNN_2 prediction (dash-dotted red lines), and RNN_3 prediction (dotted green lines).

(among 100 trained models) are chosen as the representative models of the process. The three best models are used to implement different MPCs to illustrate a key issue with RNNs. As shown in Fig. 5.4, the RNN models have almost similar prediction performance, but as you will see in the next sections, the MPC performance can vary (stable, oscillatory, and unstable) when the arbitrarily prescribed set-points are used. However, implementing the achievable set-points makes all the MPCs show almost similar performance (stabilization around the calculated achievable set-points).

5.2.3 RNN-based model predictive control of the motivating example based on the given set-points

In this section, the three best models (RNN_1, RNN_2, and RNN_3), identified in the previous section, are employed, respectively, to implement the RNN-based MPCs (MPC_1, MPC_2, and MPC_3). A representative RNN-based MPC formulation can be defined as follows:

$$\begin{aligned} \min_{u_k,\ldots,u_{k+P}} \quad & \sum_{j=1}^{P} \|\tilde{y}_{k+j} - y_{k+j}^{SP}\|_{Q_y}^2 + \|u_{k+j} - u_{k+j-1}\|_{R_{du}}^2 \\ s.t. \quad & RNN - based\ predictive\ model, \\ & u_{min} \leq u \leq u_{max} \end{aligned} \tag{5.5}$$

where P denotes the prediction horizon, y_{k+j}^{SP} and $\tilde{y}_{k+j}$ are the desired set-point trajectory for the outputs and the prediction of output at time $(k+j)\Delta t$ (Δt is the sampling time), respectively. Q_y and R_{du} are penalty matrices corresponding to the output deviations from the set-points and the rate of change in the inputs, and $u_{min} \in \mathbb{R}^m$ and $u_{max} \in \mathbb{R}^m$ represent the lower and upper bounds of the manipulated variables. Note that the RNN-based predictive model calculates multi-step ahead predictions. Thus, the initial values of inputs and outputs are fed to the model, and then one-step ahead prediction of the outputs are calculated. Next, the predicted outputs, along with candidate future inputs are fed to the model to perform two-step ahead predictions of the outputs. This procedure is performed recursively to predict the outputs multi steps ahead up to the prediction horizon.

Using the best identified models in the earlier section, three MPCs are designed to handle the problem of set-point tracking for the motivating example. Each MPC

uses one of the models, along with the constraints on the manipulated inputs, where $\bar{C}_{A0} \in [0 - C_{A0_s},\ 2 - C_{A0_s}]$ and $\bar{F}_c \in [0 - F_{c_s},\ 15 - F_{c_s}]$. The optimization problem for the MPC is solved using *fmincon* solver in MATLAB and the parameters are chosen as follows: $\Delta t = 0.5$ min, prediction horizon $P = 4$, $Q_y = \text{diag}([10^4,\ 10^1,\ 10^1])$ and $R_{du} = \text{diag}([10^{-1},\ 10^{-1}])$. Several scenarios for the set-point tracking problem are considered, where arbitrarily prescribed set-points are first applied to evaluate the performance of the MPCs. For the sake of brevity, the results of two of these scenarios are discussed. The values of the given set-points (y_g^{SP}) in these scenarios are shown in the first rows of Tables 6.4 and 6.5.

Note that the measurement noise, similar to that used in the data generation step, is added to the process outputs (distribution parameters are listed in Table 5.4). To attenuate the effect of noise, a moving average filter is also employed before the output signals are used in the MPC controllers. It should also be mentioned that the same set of noise seeds are used in order to compare the results more readily. The input and output profiles obtained by implementing different MPCs are shown in Figs. 5.6-5.11 (dotted black lines). Figs. 5.6, 5.8, and 5.10 show the results obtained using MPC_1, MPC_2, and MPC_3, respectively, in Scenario 1, and Figs. 5.7, 5.9, and 5.11 show these results in Scenario 2. As can be seen, the given set-points, that are arbitrarily selected assuming the output moves are independent, are not achieved by different MPCs. In addition, applying the given set-points in some cases (e.g., MPC_2 in Scenario 1 and MPC_3 in Scenario 2) causes the occurrence of oscillations in the manipulated variables, which in turn leads to the oscillatory behavior of the outputs. This can also result in unstable closed-loop responses, as shown in Fig 5.9 for MPC_2 in Scenario 2. The reason for the oscillatory behavior is the MPC attempt to drive the process outputs to their set-points which are chosen without considering the inherent correlation between the outputs.

5.2.4 Principal component analysis (PCA)

A brief overview of principal component analysis (PCA) is presented in this section (see Wise et al. (1990); Nomikos and MacGregor (1994) for more details). PCA is one the commonly used latent variable methods designed for transforming a set of observations of possibly correlated variables into a latent space, where the new variables (principal components) are uncorrelated. Consider a $(n \times k)$ data matrix (X) of n observations on k variables that is mean-centered and scaled to unit variance, applying PCA results in a set of principal components calculated as follows:

$$T = XP \tag{5.6}$$

where T is the $(n \times a)$ principal component (score) matrix (a is the number of principal components and usually $a < k$ for dimensionality reduction) and P is the $(k \times a)$ loading matrix, containing orthogonal loading vectors, that determines the orientation of the latent space. The following relationship also holds:

$$X = TP^T + E \tag{5.7}$$

where TP^T explains the modeled variations of X and E is the residual or the unmodeled variations of X. PCA can be performed via an eigen-decomposition of the covariance matrix $S = X^T X/(N-1)$ or singular value decomposition (SVD) of the mean-centered and scaled data matrix (X). Alternatively, nonlinear iterative partial least squares (NIPALS) algorithm can be applied to calculate the loadings. To determine the number of principal components, several criteria such as cumulative percent variance (CPV) (Li et al. (2000)) or cross validation techniques (Wold (1978)) can be utilized.

Squared prediction error (SPE) statistic

Having obtained the PCA model, the SPE statistic which is defined as the residual distance between an actual observation x_i and its projection onto the model plane can be calculated as follows:

$$SPE_i = \tilde{x}_i^T \tilde{x}_i = x_i^T (I - PP^T) x_i \tag{5.8}$$

where $\tilde{x} \in \mathbb{R}^k$ is the residual vector $(x_i - \hat{x}_i)$, and $\hat{x} \in \mathbb{R}^k$ is the reconstructed vector of the observation x_i ($SPE_i = 0$ if the observation is on the model plane). The upper confidence limit of the SPE statistic at a significance level α can be estimated as follows (a weighted Chi-square distribution):

$$SPE_\alpha = \frac{\nu}{2m} \chi(\frac{2m^2}{\nu}) \tag{5.9}$$

where m and ν represent the sample mean and variance of the SPE sample (Yoon and MacGregor (2004)).

5.3 Proposed approach to calculate the achievable set-points

In this section, the proposed method for computing the achievable set-points from the given set-points is presented. The key idea is to use PCA for recognizing the collinearity in the process outputs. An optimization problem is then defined where a constraint on SPE statistic is considered to maintain the collinearity while calculating the achievable set-points.

5.3.1 Optimization problem to calculate the achievable set-points

The optimization problem used to calculate the achievable set-points from the arbitrarily prescribed (given) set-points can be formulated as follows:

$$\begin{aligned} \min_{y_a^{SP}} \quad & \|y_a^{SP} - y_g^{SP}\|^2_{Q_{sp}} \\ s.t. \quad & SPE \leq \varepsilon \end{aligned} \tag{5.10}$$

where $\|y_a^{SP} - y_g^{SP}\|^2_{Q_{sp}} = (y_a^{SP} - y_g^{SP})^T Q_{sp} (y_a^{SP} - y_g^{SP})$ is the cost function that must be minimized, $y_a \in \mathbb{R}^l$ and $y_g \in \mathbb{R}^l$ represent the achievable and given set-points, respectively, and Q_{sp} is a positive definite matrix used to penalize the achievable set-points deviation from the given set-points (note that different penalties (weights) can be used for each variable based on the importance of that variable). The SPE constraint is considered in order to ensure that the decision variables (achievable set-points) are calculated in a way that maintains a similar correlation to that seen in the training output data. In addition, the SPE value in Eq. (6.6) is calculated as follows:

$$SPE = (\bar{y}_a^{SP})^T (I - PP^T)(\bar{y}_a^{SP}) \tag{5.11}$$

where $\bar{y}_a^{SP}$ is the vector of decision variables which is mean-centered and scaled based on the mean and standard deviation corresponding to the training output data used to build the PCA model, and P is the loading matrix. The value of ε in Eq. (6.6) should be small enough to confine the choice of the decision variables to the PCA model, thus maintaining the same collinearity. To this end, this value can be determined by computing the upper confidence limit for the SPE statistic.

5.4 Application to the motivating example

The effectiveness of the proposed approach to calculate the achievable set-points from arbitrarily prescribed set-points, and then, the problem of set-point tracking by implementing the MPC controllers are shown in this section.

5.4.1 Achievable set-points

Using the proposed approach in Section 5.3.1, an optimization problem is defined in order to calculate the achievable set-points from the arbitrarily given set-points defined in the two scenarios in Section 6.2.3. Recall that the given set-points in these scenarios, assuming the outputs can be independently controlled, are not achieved by MPC due to the existing collinearity in the output space. Thus, the objective is to address this issue by first recognizing the correlation between the process outputs using PCA. To this end, the first 8000 data samples corresponding to three outputs in deviation forms ($\bar{C}_A$, $\bar{T}_R$, and $\bar{T}_c$), used to fit the models in Section 6.2.2, are utilized to build the PCA model, and thus, calculating the loading vectors. The result shows that approximately 98% of the original data variance is explained using two principal components, and therefore, these two principal components are considered to build the PCA model.

In the next step, SPE analysis is performed to determine the upper bound for SPE constraint in the optimization problem. Two cases are considered in order to choose this value. The first case considers using the value of 95% confidence limit (1.8×10^{-1}) as the upper bound, and the second case considers applying tighter constraint by selecting a smaller value (10^{-3}) to ensure that the correlation is maintained when calculating the achievable set-points. Finally, the optimization problem is solved using a similar penalty matrix to that used for Q_y in the MPC optimization problem

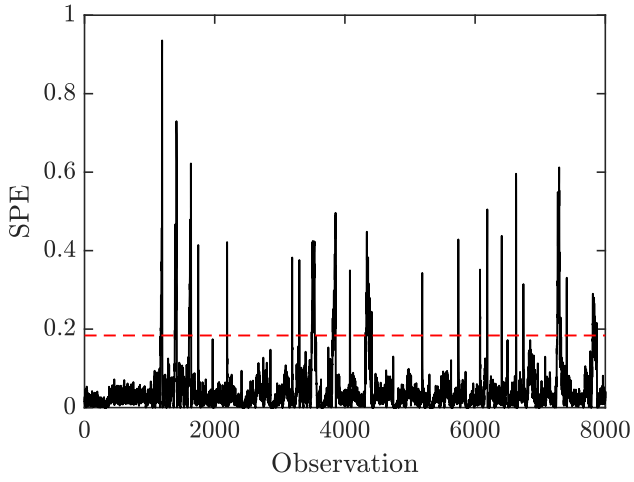


Figure 5.5: SPE values corresponding to the training data (solid lines) and 95% confidence limit (dashed line).

($Q_{sp} = \text{diag}([10^4, 10^1, 10^1])$). The set-points obtained by applying two different values, 1.8×10^{-1} and 10^{-3}, as the upper bound for the SPE constraint are listed in the second and third rows of Tables 6.4 and 6.5 as $y_{a_1}^{SP}$ and $y_{a_2}^{SP}$, respectively. Having obtained these set-points, the next step is to evaluate the RNN-based MPC performance for the problem of set-point tracking.

Table 5.6: Given set-points and achievable set-points in Scenario 1

Scenario 1	C_A^{SP}	T_R^{SP}	T_c^{SP}
y_g^{SP}	0.1000	-5.00	3.00
$y_{a_1}^{SP}$	0.0394	-3.14	-0.97
$y_{a_2}^{SP}$	0.0241	-2.67	-1.98

Table 5.7: Given set-points and achievable set-points in Scenario 2

Scenario 2	C_A^{SP}	T_R^{SP}	T_c^{SP}
y_g^{SP}	0.0800	2.00	3.00
$y_{a_1}^{SP}$	0.0570	2.70	1.50
$y_{a_2}^{SP}$	0.0417	3.17	0.49

5.4.2 RNN-based model predictive control based on the achievable set-points

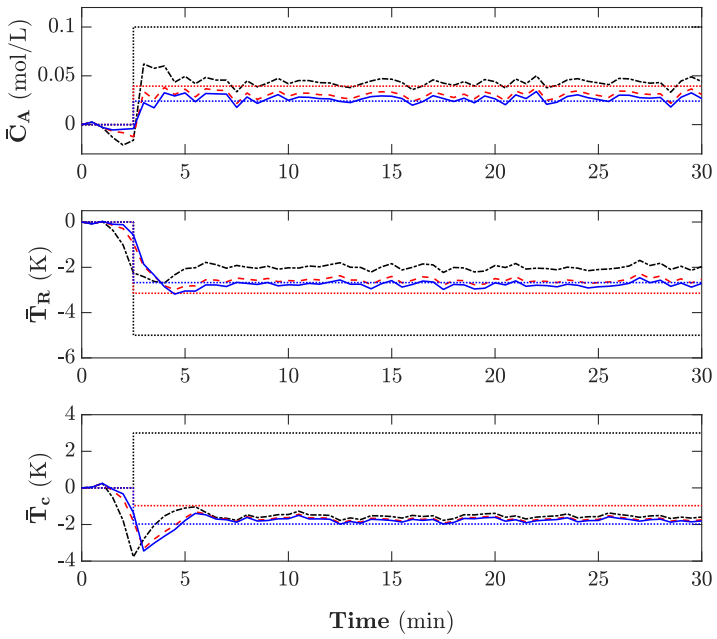
The performance of the three MPCs (MPC_1, MPC_2, and MPC_3) to stabilize the system around the given set-points was shown in Section 6.2.3. Various system responses were observed based on a specific set of the given set-points, although the RNN models, used in these implementations, have approximately similar prediction accuracy (see e.g., the performance of the three MPCs in Scenario 2 using the given set-points). This happened because the existing correlation between the process outputs was not accounted for when defining the set-points, resulting in oscillatory or unstable closed-loop responses. In this section, the performance of these three MPCs to track the set-points obtained using the optimization problem in the previous section is assessed. The input and output profiles achieved by implementing different MPCs are shown in Figs. 5.6-5.11 (red lines for $y_{a_1}^{SP}$ and blue lines for $y_{a_2}^{SP}$).

As expected, applying these set-points significantly reduces the output deviations from the set-points, showing that these set-point are more achievable than the given set-points used in Section 6.2.3. However, implementing $y_{a_2}^{SP}$ leads to a smaller offset due to the fact that this set-point is obtained using a tighter constraint on the SPE, thus maintaining a more accurate correlation between the variables in the output space. This behavior can be observed by implementing the three MPCs in Scenario 1, shown in Figs. 5.6, 5.8, and 5.10, using the achievable set-points. An interesting point here is that applying the achievable set-points eliminates the oscillatory behavior observed by implementing MPC_2 using the given set-point. Finally, as shown in Figs. 5.7, 5.9, and 5.11 for MPC implementations in Scenario 2, implementing the achievable set-point $y_{a_2}^{SP}$ enables all MPCs to drive the process outputs to their set-points accurately enough. However, applying the achievable set-point $y_{a_1}^{SP}$ causes the outputs to have larger offsets, compared to implementing the achievable set-point $y_{a_1}^{SP}$ in Scenario 1.

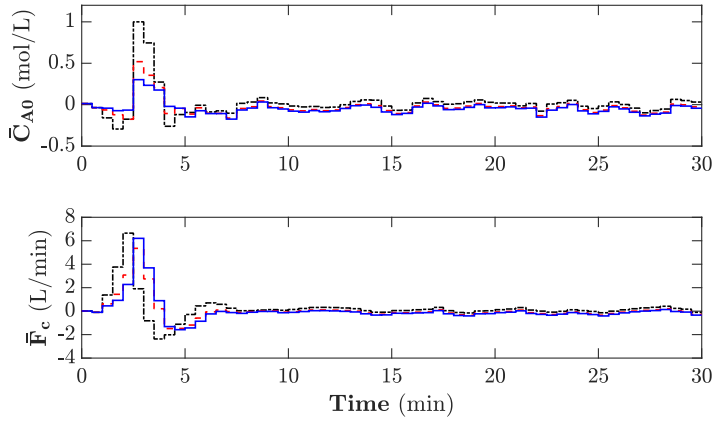
This can also result in oscillations, as illustrated in Fig. 5.9 for MPC_2 implementation. This result, therefore, reveals that using the value corresponding to 95% confidence limit as the upper bound for the SPE constraint in the optimization problem may not be sufficient. Thus, it is essential to consider a tighter constraint in order to make sure that the collinearity between calculated set-points, and subsequently the process outputs, is preserved.

Remark 5. *Note that the main objective of the present work is to show that the arbitrarily prescribed set-points, assuming the process outputs can move independently, may not be achieved while designing data-driven model-based control, when first-principles knowledge of the process is not available. Therefore, in this case it is not reasonable to expect the controller to drive the process outputs to the arbitrarily selected set-points. The key benefit of the PCA-based optimization problem is to calculate the achievable set-points so that they are as close as possible to the arbitrarily prescribed set-points by choosing an appropriate penalty matrix Q_{sp}.*

Remark 6. *Note that the small offsets observed between some of the achievable set-points and the process outputs, indicate that these set-points are not exactly achievable. This is due to the linear nature of the PCA-based approach, utilized to recognize the existing collinearity in the output space. As a result, choosing a method being able to identify the correlation between the process outputs, while handling the nonlinearity, such as autoencoder, can lead to more accurate selection of the achievable set-points. This remains the subject of future work*

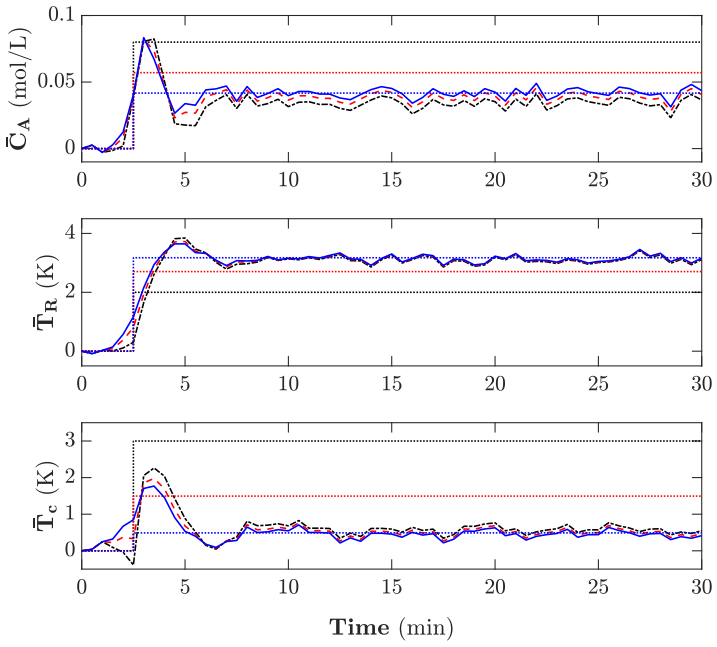


(a)

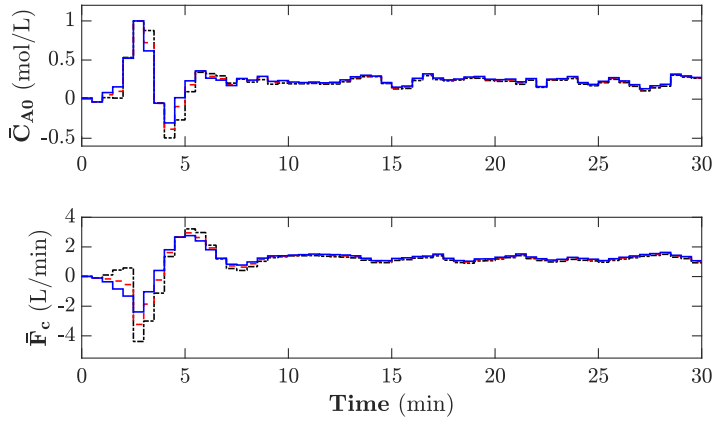


(b)

Figure 5.6: MPC$_1$ in Scenario 1:(a) y_g^{SP} (dotted black lines) and the output profiles based on that (dash-dotted lines), $y_{a_1}^{SP}$ (dotted red lines) and the output profiles based on that (dashed lines), and $y_{a_2}^{SP}$ (dotted blue lines) and the output profiles based on that (solid lines), (b) Manipulated input profiles obtained by applying y_g^{SP} (dash-dotted lines), $y_{a_1}^{SP}$ (dashed lines), and $y_{a_2}^{SP}$ (solid lines).

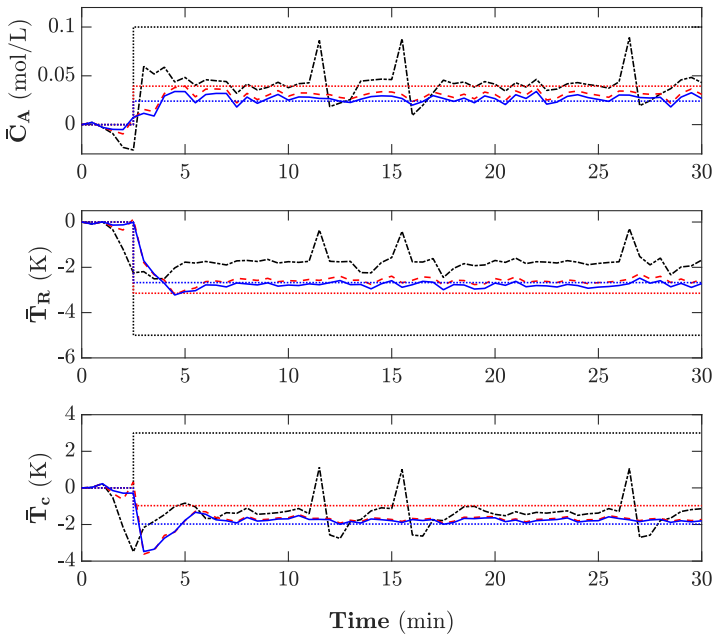


(a)

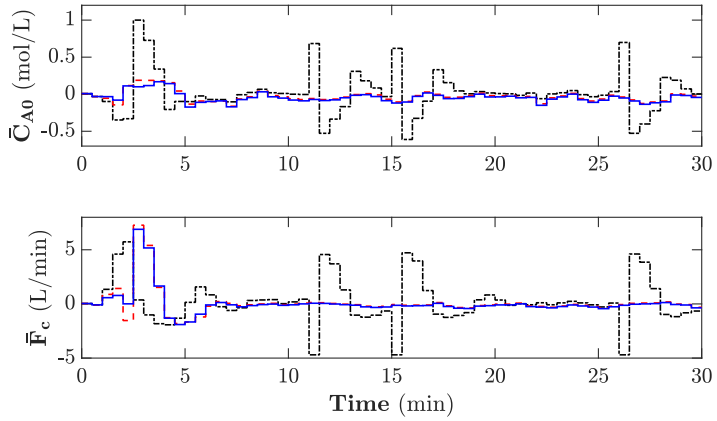


(b)

Figure 5.7: MPC$_1$ in Scenario 2:(a) y_g^{SP} (dotted black lines) and the output profiles based on that (dash-dotted lines), $y_{a_1}^{SP}$ (dotted red lines) and the output profiles based on that (dashed lines), and $y_{a_2}^{SP}$ (dotted blue lines) and the output profiles based on that (solid lines), (b) Manipulated input profiles obtained by applying y_g^{SP} (dash-dotted lines), $y_{a_1}^{SP}$ (dashed lines), and $y_{a_2}^{SP}$ (solid lines).

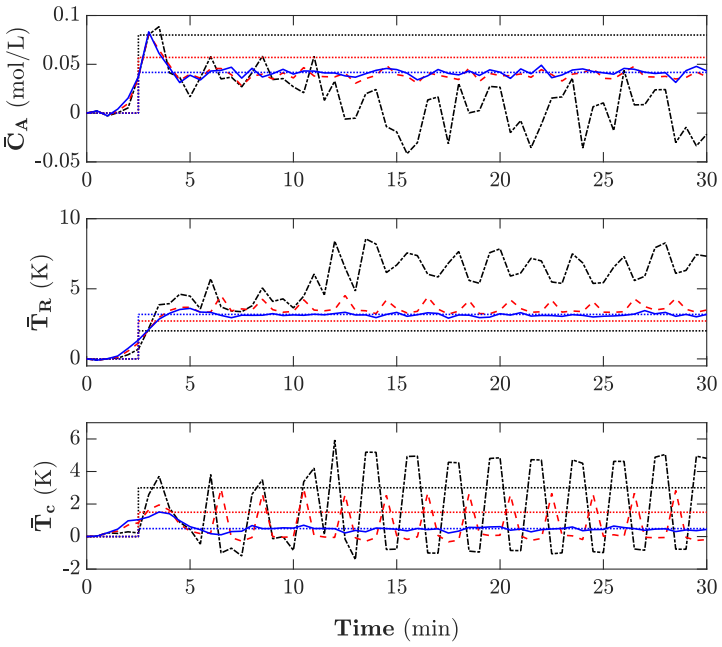


(a)

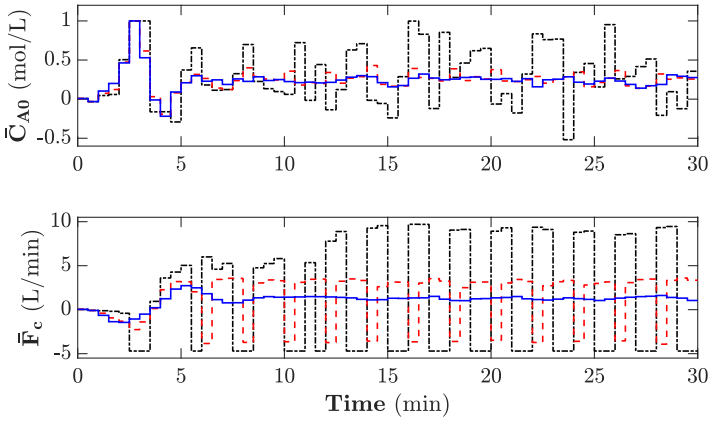


(b)

Figure 5.8: MPC$_2$ in Scenario 1:(a) y_g^{SP} (dotted black lines) and the output profiles based on that (dash-dotted lines), $y_{a_1}^{SP}$ (dotted red lines) and the output profiles based on that (dashed lines), and $y_{a_2}^{SP}$ (dotted blue lines) and the output profiles based on that (solid lines), (b) Manipulated input profiles obtained by applying y_g^{SP} (dash-dotted lines), $y_{a_1}^{SP}$ (dashed lines), and $y_{a_2}^{SP}$ (solid lines).

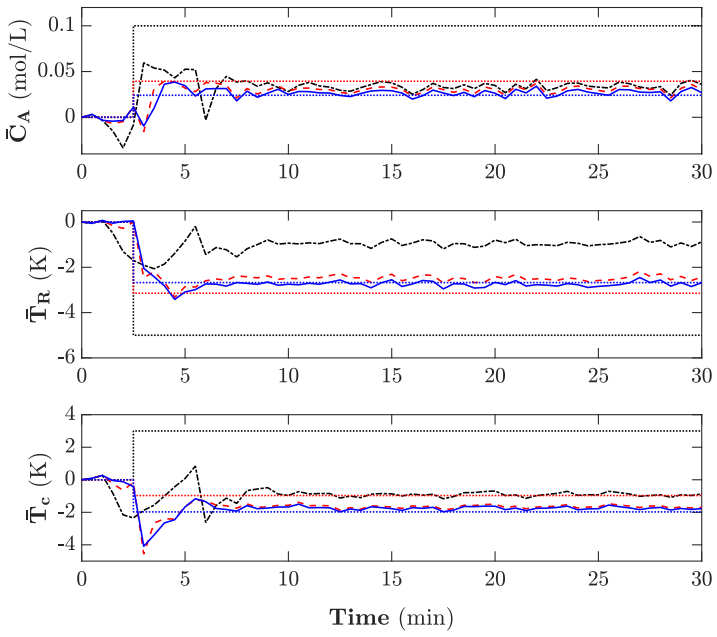


(a)

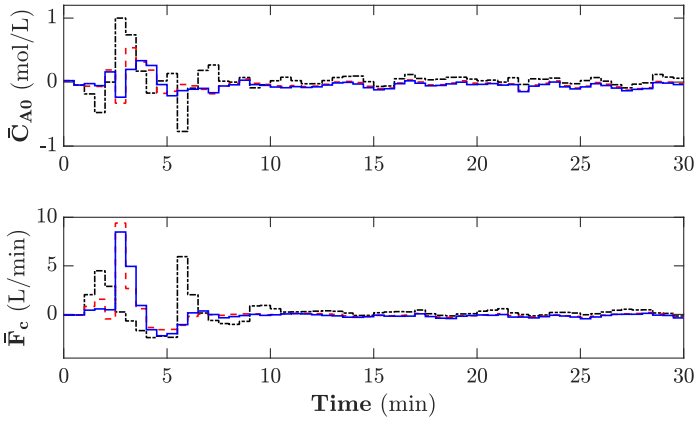


(b)

Figure 5.9: MPC$_2$ in Scenario 2:(a) y_g^{SP} (dotted black lines) and the output profiles based on that (dash-dotted lines), $y_{a_1}^{SP}$ (dotted red lines) and the output profiles based on that (dashed lines), and $y_{a_2}^{SP}$ (dotted blue lines) and the output profiles based on that (solid lines), (b) Manipulated input profiles obtained by applying y_g^{SP} (dash-dotted lines), $y_{a_1}^{SP}$ (dashed lines), and $y_{a_2}^{SP}$ (solid lines).

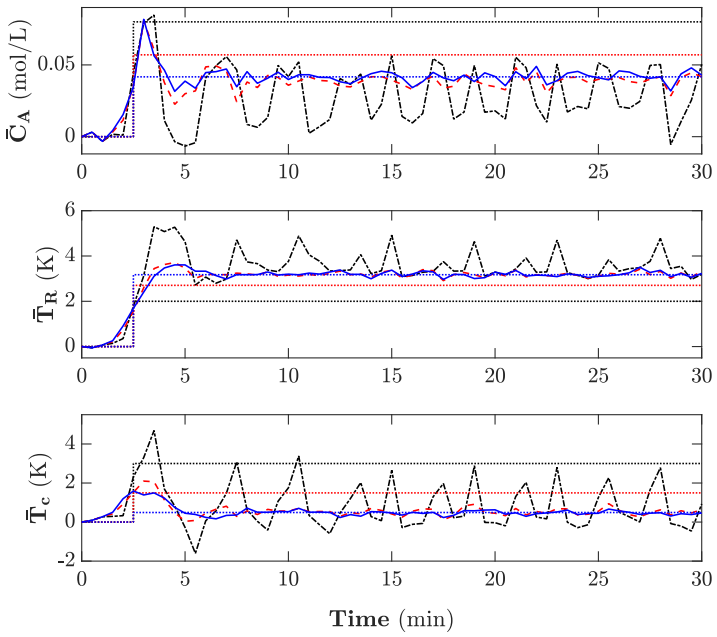


(a)

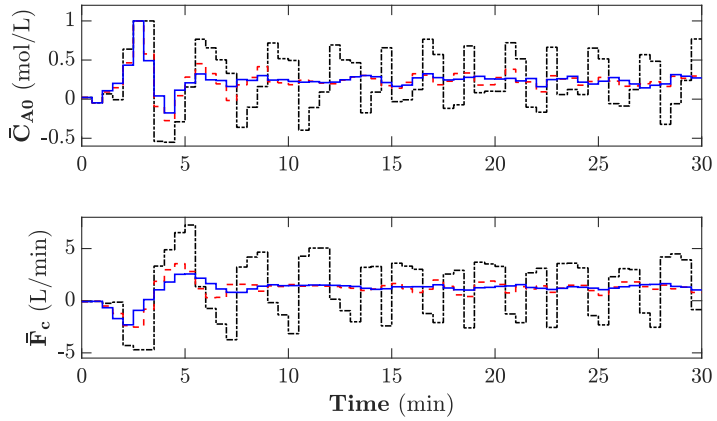


(b)

Figure 5.10: MPC$_3$ in Scenario 1:(a) y_g^{SP} (dotted black lines) and the output profiles based on that (dash-dotted lines), $y_{a_1}^{SP}$ (dotted red lines) and the output profiles based on that (dashed lines), and $y_{a_2}^{SP}$ (dotted blue lines) and the output profiles based on that (solid lines), (b) Manipulated input profiles obtained by applying y_g^{SP} (dash-dotted lines), $y_{a_1}^{SP}$ (dashed lines), and $y_{a_2}^{SP}$ (solid lines).



(a)



(b)

Figure 5.11: MPC$_3$ in Scenario 2:(a) y_g^{SP} (dotted black lines) and the output profiles based on that (dash-dotted lines), $y_{a_1}^{SP}$ (dotted red lines) and the output profiles based on that (dashed lines), and $y_{a_2}^{SP}$ (dotted blue lines) and the output profiles based on that (solid lines), (b) Manipulated input profiles obtained by applying y_g^{SP} (dash-dotted lines), $y_{a_1}^{SP}$ (dashed lines), and $y_{a_2}^{SP}$ (solid lines).

5.5 Conclusion

In this work, a novel PCA-based approach is proposed to determine the achievable set-points using the process data that contains possibly correlated outputs, in the situation where a first-principles model of the process is not easily available. To this end, an optimization problem subject to the SPE constraint is defined in order to ensure that the calculated achievable set-pints follow a similar correlation to that seen in the output data. Three MPCs that use different RNN models, with almost the same prediction performance, are then implemented to show the effectiveness of the proposed approach. The results reveal the superior performance of the MPC implementations using the calculated achievable set-point compared to the case where arbitrarily prescribed set-points are directly used in the MPC implementations.

Bibliography

Alhajeri, M. S., Wu, Z., Rincon, D., Albalawi, F., and Christofides, P. D. (2021). Machine-learning-based state estimation and predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 167:268–280.

Ayinde, B. O. and Zurada, J. M. (2017). Nonredundant sparse feature extraction using autoencoders with receptive fields clustering. *Neural Networks*, 93:99–109.

Bellemans, A., Aversano, G., Coussement, A., and Parente, A. (2018). Feature extraction and reduced-order modelling of nitrogen plasma models using principal component analysis. *Computers & Chemical engineering*, 115:504–514.

Bhat, N. and McAvoy, T. J. (1990). Use of neural nets for dynamic modeling and control of chemical process systems. *Computers & Chemical Engineering*, 14(4-5):573–582.

Corbett, B. and Mhaskar, P. (2016). Subspace identification for data-driven modeling and quality control of batch processes. *AIChE Journal*, 62(5):1581–1601.

Drgoňa, J., Picard, D., Kvasnica, M., and Helsen, L. (2018). Approximate model predictive building control via machine learning. *Applied Energy*, 218:199–216.

Du, M. and Mhaskar, P. (2014). Isolation and handling of sensor faults in nonlinear systems. *Automatica*, 50(4):1066–1074.

Ferreira, P. M., Ruano, A. E., Silva, S., and Conceicao, E. Z. E. (2012). Neural networks based predictive control for thermal comfort and energy savings in public buildings. *Energy and Buildings*, 55(6):238–251.

Forssell, U. and Ljung, L. (1999). Closed-loop identification revisited. *Automatica*, 35(7):1215–1241.

Godoy, J. L., González, A. H., and Normey-Rico, J. E. (2016). Constrained latent variable model predictive control for trajectory tracking and economic optimization in batch processes. *Journal of Process Control*, 45:1–11.

Golshan, M., MacGregor, J. F., Bruwer, M. J., and Mhaskar, P. (2010). Latent variable model predictive control (LV-MPC) for trajectory tracking in batch processes. *Journal of process control*, 20(4):538–550.

Hale, E. T. and Qin, S. J. (2002). Subspace model predictive control and a case study. *In Proceedings of the American Control Conference*, 6:4758–4763.

Han, J., Kamber, M., and Pei, J. (2011). *Data mining concepts and techniques third edition*. The Morgan Kaufmann Series in Data Management Systems.

Han, Y., Ding, N., Geng, Z., Wang, Z., and Chu, C. (2020). An optimized long short-term memory network based fault diagnosis model for chemical processes. *Journal of Process Control*, 92:161–168.

Hassanpour, H., Corbett, B., and Mhaskar, P. (2020). Integrating dynamic neural network models with principal component analysis for adaptive model predictive control. *Chemical Engineering Research and Design*, 161:26–37.

Huang, B., Ding, S. X., and Qin, S. J. (2005). Closed-loop subspace identification: an orthogonal projection approach. *Journal of process control*, 15(1):53–66.

Jalanko, M., Sanchez, Y., Mahalec, V., and Mhaskar, P. (2021). Adaptive system identification of industrial ethylene splitter: A comparison of subspace identification and artificial neural networks. *Computers & Chemical Engineering*, 147:107240.

Janakiraman, V. M., Nguyen, X., and Assanis, D. (2013). Nonlinear identification of a gasoline HCCI engine using neural networks coupled with principal component analysis. *Applied Soft Computing*, 13(5):2375–2389.

Kittisupakorn, P., Thitiyasook, P., Hussain, M. A., and Daosud, W. (2009). Neural network based model predictive control for a steel pickling process. *Journal of Process Control*, 19(4):579–590.

Koschwitz, D., Frisch, J., and Van Treeck, C. (2018). Data-driven heating and cooling load predictions for non-residential buildings based on support vector machine regression and NARX recurrent neural network: A comparative study on district scale. *Energy*, 165:134–142.

Kresta, J. V., Marlin, T. E., and MacGregor, J. F. (1994). Development of inferential process models using PLS. *Computers & Chemical Engineering*, 18(7):597–611.

Kristjanpoller, W. and Minutolo, M. C. (2018). A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. *Expert Systems with Applications*, 109:1–11.

Kusiak, A. and Xu, G. (2012). Modeling and optimization of HVAC systems using a dynamic neural network. *Energy*, 42(1):241–250.

Li, W., Yue, H. H., Valle-Cervantes, S., and Qin, S. J. (2000). Recursive PCA for adaptive process monitoring. *Journal of process control*, 10(5):471–486.

Mohanty, S. (2009). Artificial neural network based system identification and model predictive control of a flotation column. *Journal of Process Control*, 19(6):991–999.

Moonen, M., De Moor, B., Vandenberghe, L., and Vandewalle, J. (1989). On-and off-line identification of linear state-space models. *International Journal of Control*, 49(1):219–232.

Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, 1(1):4–27.

Nomikos, P. and MacGregor, J. F. (1994). Monitoring batch processes using multiway principal component analysis. *AIChE Journal*, 40(8):1361–1375.

Prasad, V. and Bequette, B. W. (2003). Nonlinear system identification and model reduction using artificial neural networks. *Computers & Chemical Engineering*, 27(12):1741–1754.

Qin, S. J. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764.

Qin, S. J. and Ljung, L. (2003). Closed-loop subspace identification with innovation estimation. *IFAC Proceedings Volumes*, 36(16):861–866.

Rawlings, J. B. and Mayne, D. Q. (2009). *Model Predictive Control Theory and Design*. Nob Hill Publishing, LLC.

Sadeghassadi, M., Macnab, C. J., Gopaluni, B., and Westwick, D. (2018). Application of neural networks for optimal-setpoint design and MPC control in biological wastewater treatment. *Computers & Chemical Engineering*, 115:150–160.

Shardt, Y. A., Huang, B., and Ding, S. X. (2015). Minimal required excitation for closed-loop identification: Some implications for data-driven, system identification. *Journal of Process Control*, 27:22–35.

Temeng, K. O., Schnelle, P. D., and McAvoy, T. J. (1995). Model predictive control of an industrial packed bed reactor using neural networks. *Journal of Process Control*, 5(1):19–27.

Wang, Y., Velswamy, K., and Huang, B. (2017). A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems. *Processes*, 5(3):46.

Wise, B. M., Ricker, N. L., Veltkamp, D. F., and Kowalski, B. R. (1990). A theoretical basis for the use of principal component models for monitoring multivariate processes. *Process Control and Quality*, 1(1):41–51.

Wold, S. (1978). Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics*, 20(4):397–405.

Wong, W. C., Chee, E., Li, J., and Wang, X. (2018). Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. *Mathematics*, 6(11):242.

Wu, Z., Luo, J., Rincon, D., and Christofides, P. (2021). Machine learning-based predictive control using noisy data: Evaluating performance and robustness via a large-scale process simulator. *Chemical Engineering Research and Design*, 168:275–287.

Wu, Z., Rincon, D., Luo, J., and Christofides, P. D. (2020). Machine learning modeling and predictive control of nonlinear processes using noisy data. *AIChE Journal*, page e17164.

Wu, Z., Tran, A., Rincon, D., and Christofides, P. D. (2019). Machine learning-based predictive control of nonlinear processes. part i: Theory. *AIChE Journal*, 65:e16729.

Yan, R., Ma, Z., Kokogiannakis, G., and Zhao, Y. (2016). A sensor fault detection strategy for air handling units using cluster analysis. *Automation in Construction*, 70:77–88.

Yoon, S. and MacGregor, J. F. (2004). Principal-component analysis of multiscale data for process monitoring and fault diagnosis. *AIChE Journal*, 50(11):2891–2903.

Zhan, S. and Chong, A. (2021). Building occupancy and energy consumption: Case studies across building types. *Energy and Built Environment*, 2(2):167–174.

Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International journal of forecasting*, 14(1):35–62.

Chapter 6

Artificial neural network-based model predictive control using correlated data

The contents of this chapter have been published in the *Industrial & Engineering Chemistry Research* Journal.

- Hesam Hassanpour, Brandon Corbett, and Prashant Mhaskar, Artificial neural network-based model predictive control using correlated data. *Industrial & Engineering Chemistry Research*, 2022, 61(8), 3075-3090. DOI: 10.1021/acs.iecr.1c04339

Abstract

This work addresses the problem of implementing model predictive control (MPC) in situations where the training data available for modeling contains possible correlations, and an artificial neural network (ANN)-based model is being used. In particular, we consider a problem where datasets are collected from a process that operates under the closed-loop condition in which correlations are induced between several input and output variables. In this situation, if the correlation problem is not addressed, manipulated inputs (calculated by MPC without considering the specific correlation in the input space), and independently prescribed set-points may require predictions in regions where the model is not trained, resulting in a poor closed-loop performance. To address this issue, a principal component analysis (PCA)-based strategy is applied to both the input and output spaces in a way that maintains model validity. To that end, a new constraint on the squared prediction error (SPE) is incorporated into the ANN-based MPC optimization problem to make control actions follow the PCA model built using the training input data. Next, a PCA model is developed using the training output data, and then an optimization problem subject to the SPE constraint is defined to calculate the set-points which are achievable. The effectiveness of the proposed ANN-based MPC to track these set-points is demonstrated using a chemical reactor example. Finally, a new autoencoder-based strategy is proposed to compute the achievable set-points. This is performed by replacing the PCA-based constraint with the autoencoder-based constraint in the optimization problem to calculate the set-points. The results indicate that the ANN-based MPC performance is improved when the autoencoder-based set-points are used.

6.1 Introduction

Operation of chemical processes to achieve economic and safety objectives has considerably relied on the use of advanced process control strategies. This fact, together with the need to cope with challenges such as multi-variable interactions and process input/output constraints has motivated the utilization of model predictive control (MPC) (Mayne et al. (2000); Qin and Badgwell (2003); Mhaskar et al. (2006); Rawlings and Mayne (2009)). The key step to successfully develop and implement of this control strategy is the use of a reliable model within the MPC framework. To that end, several modeling strategies have been utilized such as first-principles (Mayne et al. (2000); Mhaskar et al. (2006); Ganesh et al. (2021)) and data-driven modeling techniques (Moonen et al. (1989); Qin and Ljung (2003); Huang et al. (2005); Wu et al. (2019)).

While first-principles models have good extrapolation capabilities, they are generally difficult to develop and maintain due to reasons such as uncertainty in the physical parameters and unmeasured physical states. With the recent advance in computing and data storage technologies, there has been growing interest in the use of data-driven and machine learning techniques for system identification. In these methods, a model structure is first selected, and then, the available historical data is used to identify the parameters. In this direction, several statistical-based methods have been developed such as subspace identification (Moonen et al. (1989); Van Overschee and De Moor (1995); Qin and Ljung (2003); Huang et al. (2005)) and projection to latent structures (PLS) (Kresta et al. (1994); Hu et al. (2012)). Subspace-based (Hale and Qin (2002); Kheradmandi and Mhaskar (2018); Patel et al. (2021)) and latent variable-based models (Golshan et al. (2010); Godoy et al. (2016)) have also been incorporated in MPC designs for a variety of applications due to their predictive ability and simplicity (linear characteristic) of these models. In addition, several data-

driven approaches have been proposed to transform a nonlinear dynamic system into a higher dimensional space, in which its evolution is almost linear, using Koopman-based linear predictors (Korda and Mezić (2018); Narasingam and Kwon (2019); Narasingam et al. (2021)). The linear predictors are then utilized to design MPC for the nonlinear system.

Machine learning techniques such as artificial neural networks (ANN) have been widely used due to their abilities to capture process nonlinearity. Recurrent neural networks (RNN), a class of ANN, have shown excellent performance for dynamic modeling of complex nonlinear systems (Janakiraman et al. (2013); Wong et al. (2018); Koschwitz et al. (2018); Aliabadi et al. (2020); Sun et al. (2020); Wu et al. (2021a)). RNN models have also been utilized to design and implement RNN-based MPC in a wide range of applications (Temeng et al. (1995); Sadeghassadi et al. (2018); Wong et al. (2018); Hassanpour et al. (2020a); Wu et al. (2021b,a)). One of the challenges in RNN modeling is to fine-tune hyperparameters for a given dataset. Although these hyperparameters are usually determined heuristically, several tuning approaches have been used to objectively search different values of hyperparameters and select a subset that leads to a model with the best performance on the given dataset. More systematic approaches have also been proposed to optimally configure the network hyperparameters by analyzing the trade-off between the accuracy and over-fitting aspects using a multi-objective optimization methodology (Miriyala et al. (2016); Miriyala and Mitra (2020); Inapakurthi et al. (2020)). Notwithstanding these advances, a significant amount of high-quality data is usually required to develop an RNN model due to the presence of many parameters. However, some challenges that often exist in most historical datasets, such as high dimensionality and redundancy, present some difficulties for developing RNN models. In addition, the negative impact of the over-fitting issue can get amplified when datasets contain redundant information. This problem can happen when datasets are collected from a process that operates under the closed-loop condition. In this situation, the use of data-mining strategies is a promising way to

extract useful information for system identification purposes (Han et al. (2011)).

Feature selection and extraction algorithms have been widely used to reduce computational complexity and avoid over-fitting when developing machine learning-based models (Ayinde and Zurada (2017); Bellemans et al. (2018)). Principal component analysis (PCA) is one of the commonly used statistical approaches to reduce the dimensionality of data and remove redundant (correlated) variables (Yoon and MacGregor (2004); Song et al. (2010); Hassanpour et al. (2020b)). This approach has been incorporated in developing several machine learning models (Janakiraman et al. (2013); Yan et al. (2016); Bellemans et al. (2018); Drgoňa et al. (2018); Kristjanpoller and Minutolo (2018); Zhan and Chong (2021)). In some applications, it is shown that the use of PCA can facilitate the neural network model training process (reduce the computational time required to tune the hyperparameters), and increase the model accuracy (Janakiraman et al. (2013); Kristjanpoller and Minutolo (2018)). PCA is also used in the context of clustering analysis to improve the performance of the clustering techniques to identify the existing clusters in the datasets (Zhan and Chong (2021)).

Recently, autoencoder-based feature extraction strategies have received significant attention due to their ability to handle the nonlinear relationships that exist in most process datasets (Hinton and Salakhutdinov (2006)). Similar to PCA, an autoencoder neural network is an unsupervised learning algorithm that can be used for feature selection and extraction. Different variations of autoencoders have also been developed in many applications to perform anomaly detection tasks (Cheng et al. (2019); Li et al. (2019); Ma and Li (2020); Zheng and Zhao (2020); Nguyen et al. (2021); Belkadi et al. (2021)). More specifically, autoencoders in conjunction with neural networks have been used to identify nonlinear state-space models for being used in the model-based control designs such as MPC (Masti and Bemporad (2021)). These approaches, however, have not been integrated with the MPC implementations to address the

problem where correlated data samples are available for system identification. In this situation, the use of autoencoder neural networks could improve the recognition of the nonlinear correlation between the variables that must be maintained to successfully implement the MPC controller.

As mentioned earlier, applying an appropriate pre-processing step is an essential task for developing any machine learning-based models, such as RNN, when data samples are correlated. Thus, to design an RNN-based MPC, while using correlated data, the first step is to recognize the correlation between the variables. Then, this correlation must be properly considered, as a constraint, in the MPC optimization problem. Otherwise, if the manipulated inputs are moved independently, the MPC controller may require model predictions in regions where the model is not identified, which in turn leads to a poor control performance. To address this problem, two methods have recently been proposed (Hassanpour et al. (2020a)), when training input data samples are correlated. In the first approach, PCA is performed on the input data, and then, the calculated principal components (scores) are utilized to construct a model using RNN. In the next step, the MPC is implemented to compute the optimal scores (and associated manipulated inputs). An alternative method is also presented by incorporating a new squared prediction error (SPE) constraint into the standard MPC (MPC using the RNN model trained on the original input-output data) to confine the input movements to the PCA model plane developed using the training input data. In addition, a PCA-based strategy is presented to calculate the achievable set-points for the RNN-based MPC, Where data samples related to the process outputs are correlated (Hassanpour et al. (2021)). The objective is to maintain the correlation existing between the outputs, while calculating the set-points. Otherwise, the independently given set-points (prescribed without considering the correlation in the output space) cannot be achieved, either because of the plant–model mismatch (caused when the MPC controller uses the model for extrapolation), or the fundamental process limitations. As discussed, the existing approaches have focused on developing RNN-based

MPC where there is a correlation either only between process inputs or only between process outputs, but not addressed the scenario where both may be existing. These methods also consider applying PCA-based strategies, assuming the process behaves linearly, which can lead to poorer control performance when used for nonlinear systems. Finally, in the previous example (Hassanpour et al. (2021)), the correlation in the output space was dictated by the process dynamics, not by the fact that the region of operation resulted in correlation between the outputs.

Motivated by the above considerations, the first objective of this work is to address the problem of RNN-based MPC design, where the collected data from the process contains correlation in both the input and output spaces due to the closed-loop control structure (correlation is due to the closed-loop condition, and the outputs can in principle be controlled independently). To maintain the model validity, the MPC optimization problem must be defined in a way that the input and output movements follow the same correlations as those observed in the training data. Thus, we first conduct PCA on the training data so as to identify the existing correlations between the variables in the input and output spaces. The control moves are guaranteed to remain on the same plane as the PCA model plane (developed using the training input data) by including a constraint on the SPE statistic. However, implementing the MPC just by considering the SPE constraint on input moves does not necessarily guarantee that the process outputs are driven to follow the arbitrarily prescribed (given) set-points due to the existing correlation between the output data samples. To address this problem, an optimization problem is formulated so as to minimize the sum of squared error between the given and achievable set-points by applying a SPE constraint, determined based on the PCA model built using the training output data. Due to the ability of autoencoders to capture nonlinear correlations, an autoencoder-based optimization problem is also proposed to find the set-points which are achievable. The results show the RNN-based MPC performance is enhanced when using the autoencoder-based set-points (compared to the PCA-based set-points).

The rest of the article is organized as follows: The motivating example, a continuous stirred-tank reactor (CSTR) under closed-loop control, is presented in Section 6.2. A brief description of recurrent neural networks, principal component analysis, and autoencoder neural networks is then provided. Next, the CSTR example is used to implement a standard RNN-based MPC in order to illustrate the particular problem being addressed. In Section 6.3, the PCA-based strategies for computing the achievable set-points and maintaining model validity within the RNN-based MPC are presented. Applications of these approaches to the motivating example are also demonstrated. The proposed autoencoder-based technique to calculate the achievable set-points, together with its application is presented in Section 6.4. Finally, concluding remarks are made in Section 6.5.

6.2 Preliminaries

6.2.1 Motivating Example

A continuous stirred-tank reactor, where an irreversible first-order reaction (A $\rightarrow$ B) occurs, is considered (Du and Mhaskar (2014)). Mass and energy balances lead to the following first-principles model, describing the evolution of the concentration of reactant A, C_A, the reactor temperature, T_R, and the cooling jacket temperature, T_c:

$$\begin{aligned}
\dot{C_A} =& \frac{F}{V}(C_{A0} - C_A) - k_0 e^{-E/RT_R} C_A \\
\dot{T_R} =& \frac{F}{V}(T_0 - T_R) + \frac{(-\Delta H)}{\rho c_p} k_0 e^{-E/RT_R} C_A - \frac{UA}{\rho c_p V}(T_R - T_c) \\
\dot{T_c} =& \frac{F_c}{V_c}(T_{cf} - T_c) + \frac{UA}{\rho_c c_{pc} V_c}(T_R - T_c)
\end{aligned} \tag{6.1}$$

where F is the feed flow rate to the reactor with temperature T_0 and concentration C_{A0} and F_c is the cooling liquid flow rate with temperature T_{cf}. V and V_c are the volume of the reactor and the volume of cooling jacket, respectively. k_0 is the pre-exponential constant, E is the activation energy, and R is the ideal gas constant. ΔH is the enthalpy of the reaction, ρ is the density, and c_p is the heat capacity of the fluid in the reactor. U and A are the overall heat transfer coefficient and the heat transfer area, respectively. ρ_c is the density and c_{pc} is the heat capacity of the cooling liquid in the jacket. The model parameters in nominal conditions and the steady-state values of the inputs and outputs are reported in Table 6.1 and Table 6.2, respectively.

Table 6.1: Parameters of the CSTR example

Parameter	Value	Unit
V	100	L
V_c	20	L
k_0	7.20×10^{10}	min^{-1}
ΔH	-5×10^4	J/mol
ρ	1000	g/L
ρ_c	1000	g/L
c_p	0.239	J/(g K)
c_{pc}	4.20	J/(g K)
UA	5×10^4	J/(min K)
E/R	8750	K
T_0	350	K
T_{cf}	293	K

Table 6.2: Steady-state values of the process inputs and outputs

Parameter	Value	Unit
C_{A0_s}	1.00	mol/L
F_s	14.60	L/min
F_{c_s}	4.70	L/min
C_{A_s}	0.50	mol/L
T_{R_s}	324.82	K
T_{c_s}	315.81	K

Closed-loop simulations are carried out to produce data samples for system identification. Note that there are several reasons due to which only closed-loop data may

be available in practice, such as unstable open-loop systems, production/economic reasons, and safety concerns. Thus, to reflect practical considerations, the data samples are generated from the CSTR operating under the closed-loop condition where the concentration (C_A) is controlled by manipulating the inlet concentration (C_{A0}) using a PI controller (PI_1). In addition, the reactor temperature (T_R) is controlled by manipulating two manipulated variables which are the feed flow rate (F) and the cooling stream flow rate (F_c) using two PI controllers (PI_2 and PI_3, respectively). It should be mentioned that this implementation is used to illustrate an example of a control structure where more than one manipulated input is utilized to control one process output. Note that the deviation forms are used to represent the variables as follows: $\bar{C}_{A0} = C_{A0} - C_{A0_s}$, $\bar{F} = F - F_s$, and $\bar{F}_c = F_c - F_{c_s}$ (for the manipulated inputs), and $\bar{C}_A = C_A - C_{A_s}$, $\bar{T}_R = T_R - T_{R_s}$, and $\bar{T}_c = T_c - T_{c_s}$ (for the outputs). PI control actions are computed as follows:

$$U(t) = K_c \left(E(t) + \frac{1}{T_I} \int_0^t E(t')dt' \right) \tag{6.2}$$

where U is the control action and E is the error between the process output and the set-point. K_c and T_I also represent the proportional gain and the integral time constant, respectively. The tuning parameters of the controllers are reported in Table 6.3.

Table 6.3: Tuning parameters of the PI controllers

PI controller	K_c	T_I
PI_1	4.00	8.00
PI_2	1.00	10.00
PI_3	-0.50	16.67

To generate data samples, pseudo random binary sequence (PRBS) signals are used to change the set-point for $\bar{C}_A$ and $\bar{T}_R$, as shown in Fig. 6.1(a). The process is

simulated for 10000 minutes, with a sampling rate of 0.5 minutes. The manipulated inputs, calculated by the PI controllers, together with the process outputs are shown in Fig. 6.1. Having obtained the data samples (20000 data samples), an RNN model is trained in Section 6.2.2.

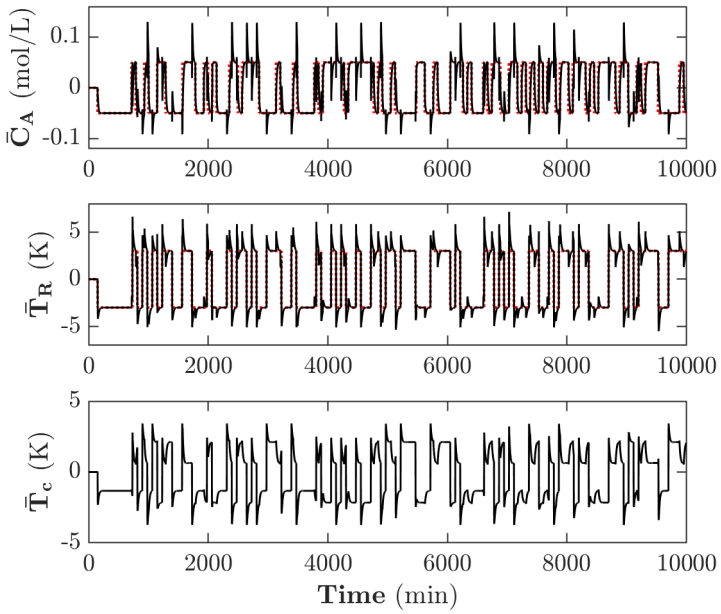
Remark 1. *Note that the present manuscript does not consider the problem of generating or using sufficiently rich data. Thus, if, say open-loop data were available where the inputs were moved independently, and as such the data was information rich, one could readily utilize that data to build a good model to in turn use for control. The present manuscript addresses the problem where closed-loop, imperfect data is available, and the above simulation example is simply used to replicate such practical scenarios.*

6.2.2 Recurrent neural networks

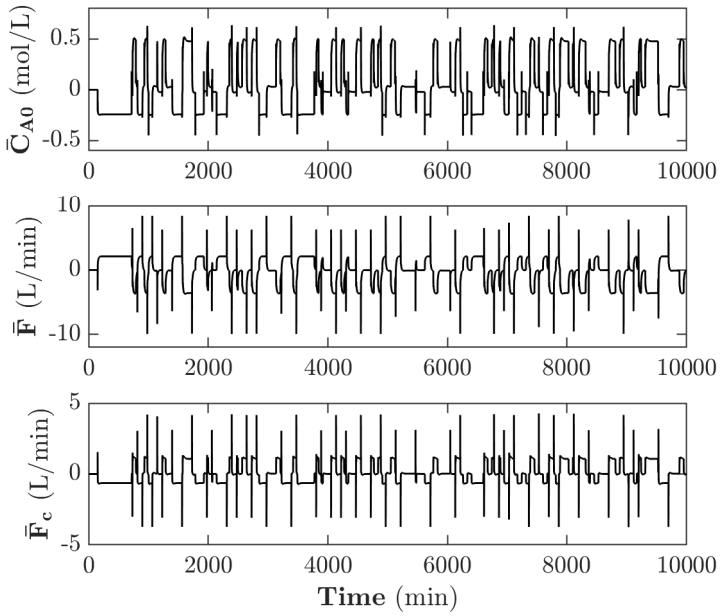
Recurrent neural networks (RNN) have been used for an extensive range of system identification and time-series modeling problems (Cadenas et al. (2016); Boussaada et al. (2018); Hassanpour et al. (2020a); Han et al. (2020); Wu et al. (2021a)). In this section, a brief review of a particular class of RNN, nonlinear autoregressive network with exogenous inputs (NARX), is presented. This neural network architecture is utilized to model the dynamic behaviour of the CSTR example. The open-loop and closed-loop architectures of the NARX neural network, shown in Fig. 6.2, are mathematically described via Eqs. (6.3) and (6.4) as follows:

$$\hat{y}(k+1) = f[y(k), ..., y(k-n_y), u(k), ..., u(k-n_u)] \tag{6.3}$$

$$\hat{y}(k+1) = f[\hat{y}(k), ..., \hat{y}(k-n_y), u(k), ..., u(k-n_u)] \tag{6.4}$$



(a)



(b)

Figure 6.1: Closed-loop data: (a) Measured outputs (solid black lines) and set-points (dotted red lines) (b) Manipulated inputs.

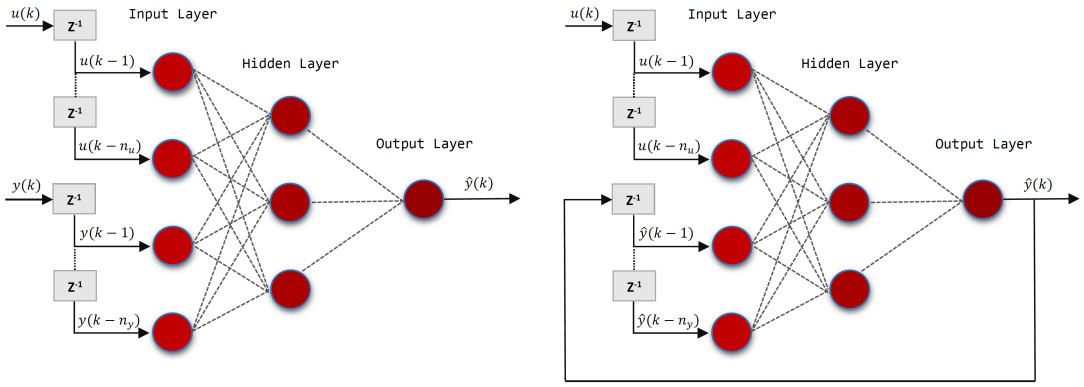


Figure 6.2: Schematic of open-loop (left) and closed-loop (right) NARX neural network architecture.

where $f(\cdot)$ is the nonlinear mapping function of the neural network. $y \in \mathbb{R}^l$ and $\hat{y} \in \mathbb{R}^l$ represent the vector of the measured outputs and the predicted outputs, respectively, and $u \in \mathbb{R}^m$ represents the vector of manipulated inputs. n_y and n_u are also the number of output and input delays, respectively. As can be seen in Eq. (6.3) and Fig. 6.2 (left half), the future value of the outputs is predicted using the present and past values of the inputs, together with the measured values of the present and past outputs in the open-loop architecture. However, the predicted values of the present and past outputs, instead of the measured values, are used in the closed-loop architecture to predict the future value of the outputs, as shown in Eq. (6.4) and Fig. 6.2 (right half).

Due to the availability of the output measurements during the training process, the open-loop architecture is utilized to develop a NARX model. The resulting model is a feedforward neural network that enables us to use the traditional training algorithm (static backpropagation) during training. This model is able to perform one-step ahead prediction. After the training process, the NARX neural network is converted into the closed-loop architecture in order to perform multi-step ahead predictions.

Model identification using RNN

The data generated as described in Section 6.2.1 is used to develop a NARX model for the CSTR example. The data samples are first divided into three categories: (1) Training data (14000 data samples), (2) Validation data (4000 data samples), and (3) Testing data (2000 data samples). Note that this data segmentation is necessary to develop a model and evaluate its prediction performance before use in the MPC implementation. The training data is used to train the NARX networks, and then the validation data is used to fine-tune the hyperparameters (in order to achieve the final/best model). Finally, the testing data is used in order to evaluate the model ability to predict the unseen data.

The optimization problem to calculate the optimal values of the parameters in the NARX neural network is non-convex due to the use of nonlinear activation functions. This leads to two challenging problems: (1) Difficulty in finding a global solution and (2) Difficulty in ensuring that the same solution is found every time a NARX network is trained with the same architecture. In recognition of these challenges, the NARX network is trained multiple times with random re-initializations of the parameters in order to obtain a good local optimum. This can also provide insights into the selection of the hyperparameters by comparing the results obtained based on multiple runs for each architecture.

To build a model, the training data samples are first normalized (mean-centered and scaled to unit variance). The normalized data is then fed to a NARX neural network (Note that the NARX model is trained using Matlab Machine Learning and Deep Learning toolbox). The NARX neural network has several hyperparameters that must be tuned, such as the number of hidden layers and neurons, type of activation functions, the number of input and output delays, and the number of epochs. As discussed in Section 6.1, several methods have recently been proposed to more

systematically determine the network hyperparameters using a multi-objective optimization framework (Miriyala et al. (2016); Miriyala and Mitra (2020); Inapakurthi et al. (2020)). However, in this work, the hyperparameters are tuned by searching in a subset resulting in a model with higher prediction performance. To that end, different NARX architectures including 1 and 2 hidden layers with different number of neurons are used. In addition, *tanh* and linear activation functions are used for the neurons in the hidden and output layers, respectively. We also use different number of input and output delays. The Levenberg-Marquardt backpropagation technique is used to train the models. The training process stops when the number of epochs reaches a certain number for each specific architecture. To that end, the number of epochs is initially set to 100, then modified as needed. To evaluate the prediction performance of the trained model, scaled root mean square error (RMSE), given by Eq.(6.5), is utilized:

$$\text{Scaled RMSE} = \sum_{i=1}^{l} \frac{\sqrt{\frac{\sum_{j=1}^{n} (y_{i,j} - \hat{y}_{i,j})^2}{n}}}{\sigma_i} \tag{6.5}$$

where $y_{i,j}$ and $\hat{y}_{i,j}$ are the j^{th} true and predicted values of the i^{th} output, respectively. σ_i is the standard deviation of the i^{th} output, l is the number of outputs, and n is the number of data samples.

To account for the non-uniqueness problem mentioned earlier, the NARX neural network is trained 50 times for each specific architecture. Fig. 6.3 summarizes the results obtained based on different NARX architectures. Note that ($NARX_1$, $NARX_2$, $NARX_3$, and $NARX_4$ use 1 hidden layer with 6, 8, 10, and 12 neurons, respectively, and $NARX_5$, $NARX_6$, $NARX_7$, and $NARX_8$ use 2 hidden layers with (6, 4), (8, 4), (10, 6), and (12, 6) neurons, respectively). These results, together with the scaled RMSE mean and variance results obtained based on each architecture, indicate the NARX model using 1 hidden layer with 10 neurons ($NARX_3$) has the best perfor-

mance among all considered architectures. Thus, the model with this architecture and the smallest scaled RMSE is selected as the final model. Having trained the NARX model, the closed-loop NARX architecture (see Fig. 6.2) is utilized for multi-steps ahead predictions (in the validation and testing phases). To do this, the initial input and output data samples are first fed to the trained NARX model to predict the outputs in the first sampling instant. The predicted values are then used recursively to predict the outputs in the next sampling instants. The prediction performance of the model on the testing data is shown in Fig. 6.4. As can be seen, this model can predict the process outputs with reasonable accuracy. Thus, this model is used in the MPC implementation.

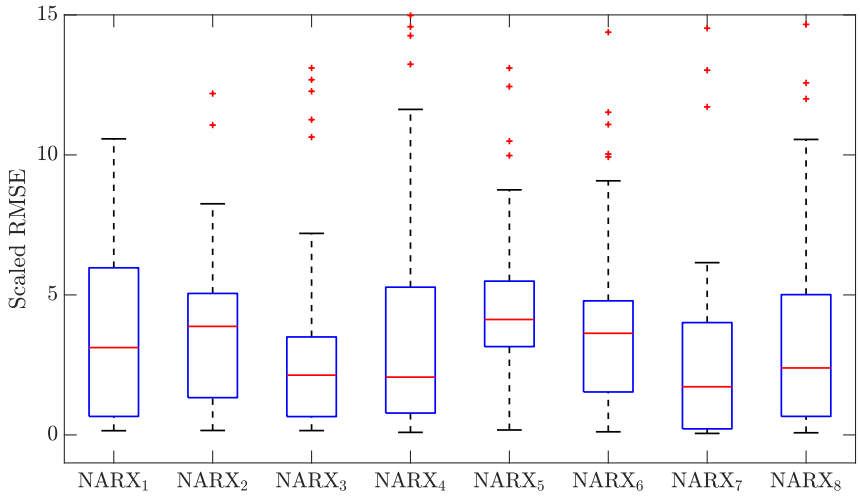


Figure 6.3: Scaled RMSE of the NARX models with different architectures in the validation phase.

6.2.3 Standard RNN-based model predictive control

The optimization problem to implement the (standard) RNN-based MPC is formulated as follows:

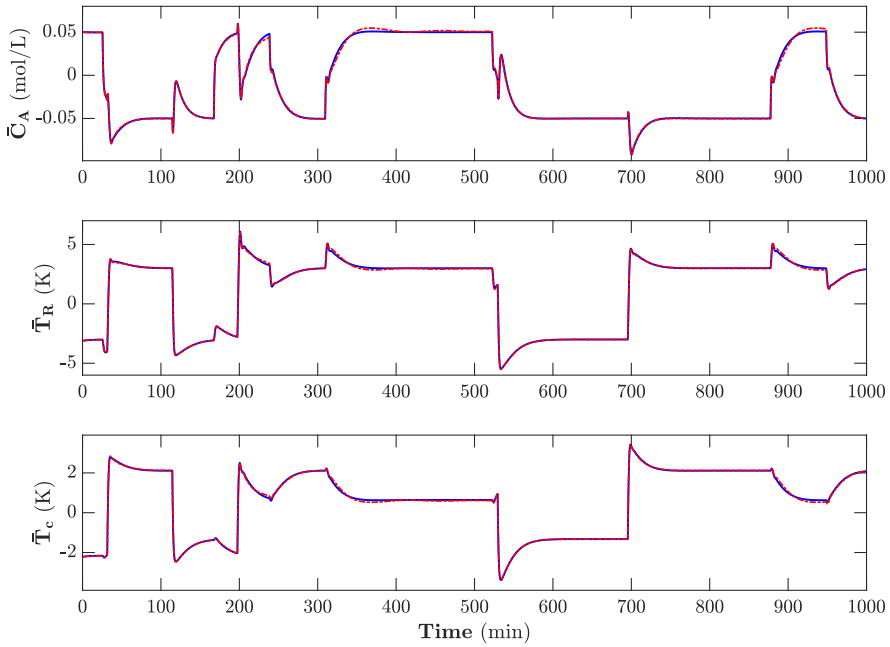


Figure 6.4: Model testing: Measured outputs (solid blue lines) and predictions (dash-dotted red lines).

$$\begin{aligned} \min_{u_k,\ldots,u_{k+P}} \quad & \sum_{j=1}^{P} \|\tilde{y}_{k+j} - y^{SP}_{k+j}\|^2_{Q_y} + \|u_{k+j} - u_{k+j-1}\|^2_{R_{du}} \\ s.t. \quad & RNN\ model, \\ & u_{min} \leq u \leq u_{max} \end{aligned} \tag{6.6}$$

where y^{SP}_{k+j} is the set-point, $\tilde{y}_{k+j}$ is the predicted output trajectory at the time $(k+j)\Delta t$, Δt is the sampling time, and P is the prediction horizon. $Q_y \in \mathbb{R}^{l\times l}$ is a positive definite matrix used to penalize the output deviation from the set-point, $R_{du} \in \mathbb{R}^{m\times m}$ is a positive semi-definite matrix used to penalize the rate of change in the manipulated inputs. $u_{min} \in \mathbb{R}^m$ and $u_{max} \in \mathbb{R}^m$ are also the lower and upper bounds of the inputs. It should be noted that the NARX model, with closed-loop

architecture, is used to predict the outputs multi-steps ahead. Thus, the initial input and output data samples are fed to the NARX model to perform one-step ahead prediction. The predicted outputs, together with the candidate future inputs are then used for two-step ahead prediction. This procedure is repeated recursively to calculate the predicted outputs multi-steps ahead to the end of the prediction horizon. The RNN-based MPC calculates the optimal input sequence over the prediction horizon $(u_k, ..., u_{k+P})$, and then the first control action is sent to the process, and the MPC implemented in a standard receding horizon fashion.

Standard RNN-based MPC implementation

This section demonstrates the implementation of a representative RNN-based MPC. Thus, the identified model from Section 6.2.2 is utilized to control the CSTR example. Two scenarios are also defined to evaluate the RNN-based MPC performance using the arbitrarily given set-points. The values of the given set-points are listed in the first row of Tables 6.4 and 6.5. The parameters for the RNN-based MPC are selected as follows: $\Delta t = 0.5$ min, prediction horizon $P = 4$, $Q_y = \text{diag}([10^4,\ 10^1,\ 10^1])$ and $R_{du} = \text{diag}([10^{-1},\ 10^{-1},\ 10^{-1}])$ (diag$(\cdot)$ refers to a diagonal matrix with the argument on the diagonal). The manipulated inputs are subject to the following constraints: $\bar{C}_{A0_s} \in [0 - C_{A0_s},\ 3 - C_{A0_s}]$, $\bar{F} \in [0 - F_s,\ 60 - F_s]$, and $\bar{F}_c \in [0 - F_{c_s},\ 15 - F_{c_s}]$. The MPC optimization problem is solved using *fmincon* solver in the Matlab Optimization toolbox, and the results are shown in Figs. 6.5 and 6.6 (solid blue lines), for Scenarios 1 and 2, respectively.

As shown, the given set-points (prescribed assuming the outputs are independent) are not reached by the MPC. The reasons why the RNN-based MPC fails to drive the outputs to the set-points are as follows: (1) The use of the given set-points can result in plant–model mismatch. Recall that the model is developed using the data with correlated outputs. Thus, correlation being maintained is one indication of the process

Table 6.4: Scenario 1: Given set-points (y_g^{SP}) and achievable set-points, obtained based on the PCA ($y_{a_1}^{SP}$) and autoencoder ($y_{a_2}^{SP}$) approaches.

Scenario 1	$\bar{C}_A^{SP}$	$\bar{T}_R^{SP}$	$\bar{T}_c^{SP}$
y_g^{SP}	0.0400	3.00	2.00
$y_{a_1}^{SP}$	0.0321	3.38	1.27
$y_{a_2}^{SP}$	0.0382	3.29	1.07

Table 6.5: Scenario 2: Given set-points (y_g^{SP}) and achievable set-points, obtained based on the PCA ($y_{a_1}^{SP}$) and autoencoder ($y_{a_2}^{SP}$) approaches.

Scenario 2	$\bar{C}_A^{SP}$	$\bar{T}_R^{SP}$	$\bar{T}_c^{SP}$
y_g^{SP}	-0.0500	2.00	3.00
$y_{a_1}^{SP}$	-0.0621	2.58	1.88
$y_{a_2}^{SP}$	-0.0476	3.17	2.02

continuing to evolve in the same region as the training data, and in the absence of that the RNN model may not remain accurate, leading to MPC performance degradation. (2) In addition, the model is trained using the training data including the correlated inputs, while the manipulated inputs are allowed to take independent values in the MPC implementation. Thus, the use of the model within the MPC framework can lead to the plant-model mismatch due to not maintaining the correlation between the manipulated inputs.

To address these problems, it is necessary to recognize the existing correlations between the variables in the training input and output data. The next two sections review existing techniques that enable capturing the correlation between variables. Subsequently, this information is utilized to design and implement MPCs that maintain the same correlation as observed in the training input and output data, to in turn help maintain model validity.

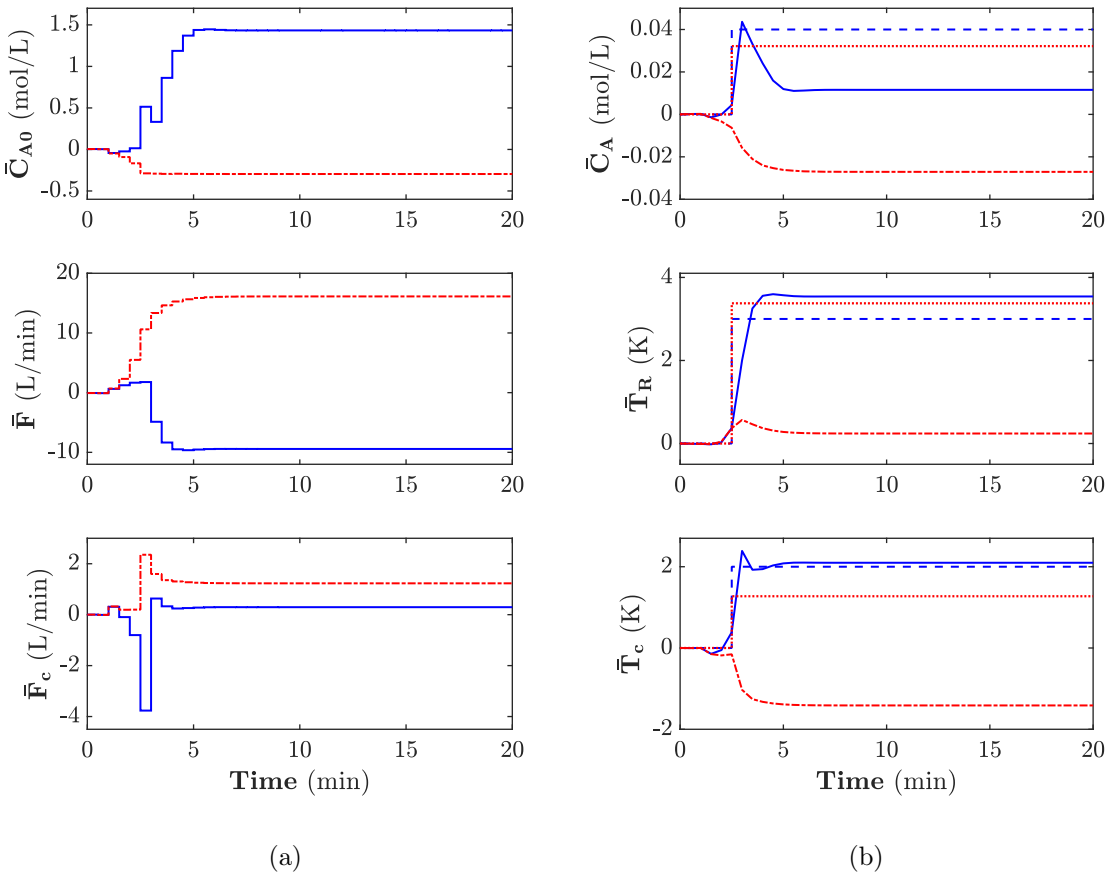


Figure 6.5: Standard MPC performance in Scenario 1: (a) the manipulated input profiles obtained using y_g^{SP} (solid blue lines) and $y_{a_1}^{SP}$ (dash-dotted red lines). (b) y_g^{SP} (dashed blue lines), the output profiles based on y_g^{SP} (solid blue lines), $y_{a_1}^{SP}$ (dotted red lines), and the output profiles based on $y_{a_1}^{SP}$ (dash-dotted red lines).

6.2.4 Principal component analysis

Principal component analysis (PCA) is a statistical approach allowing for reducing the dimensionality of the data, while retaining most of the variation in the original data. This method is designed to extract uncorrelated components from a set of correlated variables (Wise et al. (1990); Nomikos and MacGregor (1994)). Consider a mean-centered and scaled (to unit variance) data matrix X consisting of n observations

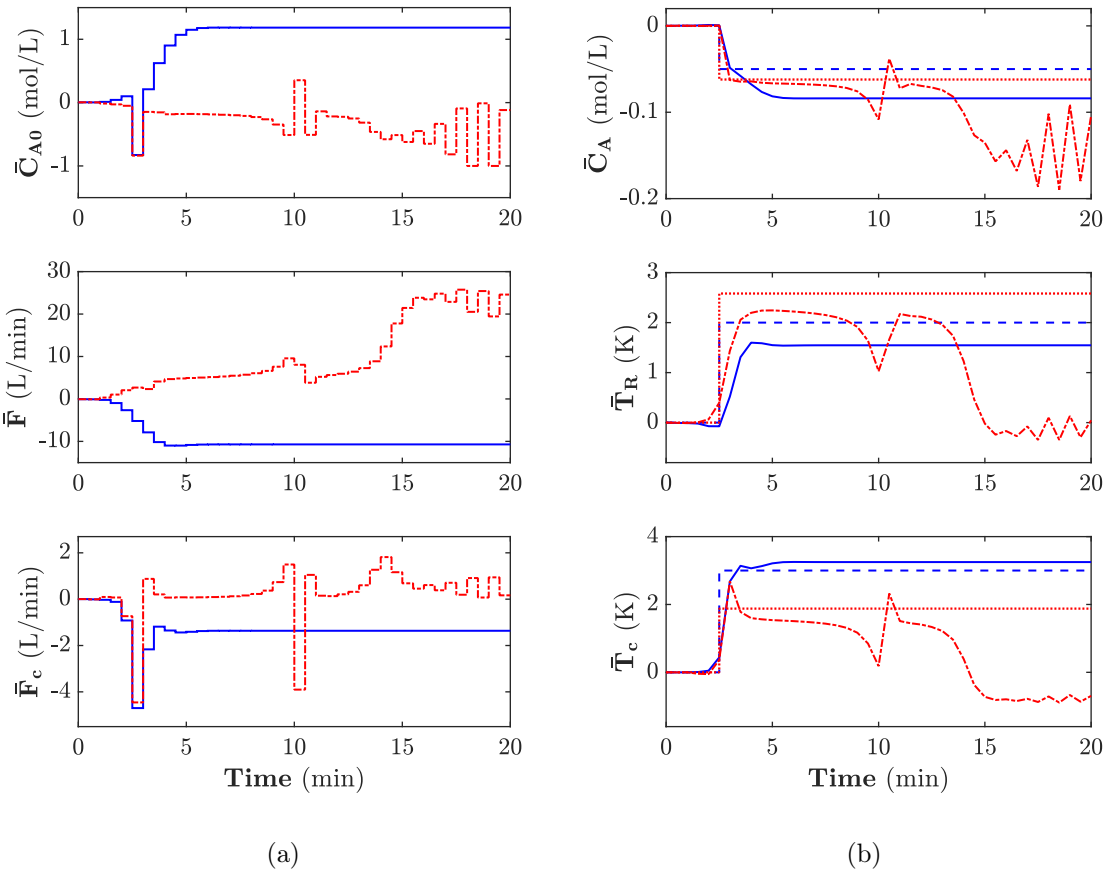


Figure 6.6: Standard MPC performance in Scenario 2: (a) the manipulated input profiles obtained using y_g^{SP} (solid blue lines) and $y_{a_1}^{SP}$ (dash-dotted red lines). (b) y_g^{SP} (dashed blue lines), the output profiles based on y_g^{SP} (solid blue lines), $y_{a_1}^{SP}$ (dotted red lines), and the output profiles based on $y_{a_1}^{SP}$ (dash-dotted red lines).

on k variables, performing PCA on X leads to a set of principal component scores, computed as follows:

$$T = XP \tag{6.7}$$

where T is the $(n \times a)$ score matrix and P is the $(k \times a)$ loading matrix that contains orthogonal loading vectors, determining the orientation of the latent space (Note that

a is the number of principal components, and $a < k$ for dimensionality reduction). Having obtained the principal components, the data matrix X is decomposed as follows:

$$X = TP^T + E \tag{6.8}$$

where TP^T and E represent the modeled and unmodeled variations of X, repectively. PCA can be performed through an eigen-decomposition of the sample covariance matrix $S = X^T X/(N-1)$ or a singular value decomposition (SVD) on the data matrix X. The nonlinear iterative partial least squares (NIPALS) technique can be applied alternatively to extract the principal components.

After obtaining the PCA model, the SPE statistic, which is defined as the residual distance between an observation x_i and its projection onto the PCA model plane, is computed as follows:

$$SPE_i^{pca} = \tilde{x}_i^T \tilde{x}_i = x_i^T (I - PP^T) x_i \tag{6.9}$$

where $\tilde{x}_i \in \mathbb{R}^k$ is the residual vector for each observation ($\tilde{x}_i = x_i - \hat{x}_i$), and $\hat{x}_i \in \mathbb{R}^k$ is the reconstructed vector of the observation x_i.

6.2.5 Autoencoder neural networks

Autoencoder is another unsupervised learning technique that leverages ANN capability to achieve an alternate representation of the original data. It generally consists of three layers, as shown in Fig. 6.7: (1) Input layer, (2) Hidden layer, and (3) Output layer. The objective of training an autoencoder is to copy the inputs to the outputs in a way that the hidden (latent) representation captures the useful features present in

the data. This can be achieved by constraining the hidden layer to have a smaller dimension (neurons) than the input layer (undercomplete autoencoder). Training such an undercomplete representation makes the autoencoder to learn the most salient features of the training data. On the other hand, the autoencoder could easily learn to memorize the training data if it is permitted to have too much capacity. It is the reduced dimension of the hidden layer that is served to perform noise elimination and dimensionality reduction (Hinton and Salakhutdinov (2006); Ramamurthy et al. (2020); Masti and Bemporad (2021)), and thus safeguard against an overfitted model.

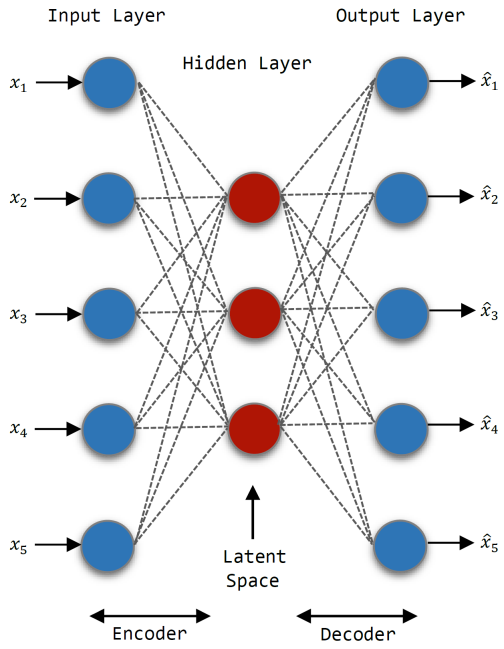


Figure 6.7: Schematic of a three-layer autoencoder neural network.

To train the autoencoder, the data samples $x \in \mathbb{R}^k$ are compressed into the low-dimensional latent space $\mathbb{R}^a$ by the encoder using a nonlinear function $g(\cdot)$ ($z = g(x) \in \mathbb{R}^a$), where z represents the a-dimensional latent space, and $a < k$. The encoded data samples are then decompressed into the output layer by the decoder using a function $h(\cdot)$ ($\hat{x} = h(z) \in \mathbb{R}^k$). Finally, the encoded-decoded data samples, from the output layer, are compared with the initial data samples, fed to the input layer, and then, the error is backpropagated through the network in order to update

the parameters (weight and bias values). The autoencoder is trained by minimizing the reconstruction error, defined by the following loss function:

$$L = \frac{1}{2n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|_2^2 \tag{6.10}$$

where n is the number of input data samples, and $\hat{x}_i$ is the reconstruction of the original input x_i. Recall that the key objective of the autoencoder is not to simply copy the input data samples but to learn latent representation with the aim of extracting the hidden structure that may exist in the data (i.e. correlations between the input variables). Indeed, the autoencoder aims to reduce the dimensionality of the data, while retaining the most information of data structure.

After developing the autoencoder model, the SPE value corresponding to each observation is calculated as follows:

$$SPE_i^{ae} = \tilde{x}_i^T \tilde{x}_i \tag{6.11}$$

where $\tilde{x}_i \in \mathbb{R}^k$ is the residual corresponding to the observation x_i ($\tilde{x}_i = x_i - \hat{x}_i$), $\hat{x}_i = h(g(x_i))$ is the reconstructed vector of this observation, and $h(g(\cdot))$ is the function describing the trained autoencoder.

6.3 Proposed PCA-based strategy for RNN-based MPC implementation

To address the problems discussed in Section 6.2.3, a formulation that integrates the ideas on handling output correlation (Hassanpour et al. (2021)) and input correlation (Hassanpour et al. (2020a)) is integrated.

6.3.1 Constrained RNN-based MPC: Implementing achievable set-points

As shown in Section 6.2.3, the prescribed set-pints are not achieved by MPC due to the use of model in domains where it is not trained. To maintain model validity, it is necessary to first recognize the correlation in the input space. Then, the constrained RNN-based MPC formulation is utilized to force the manipulated inputs to move in a specific direction so as to maintain the same correlation as that of the training input data. This is accomplished by adding a constraint on the SPE statistic in the standard RNN-based MPC formulation. Thus, the control action is computed as follows:

$$\begin{aligned} \min_{u_k,\ldots,u_{k+P}} \quad & \sum_{j=1}^{P} \|\tilde{y}_{k+j} - y^{SP}_{a_{k+j}}\|^2_{Q_y} + \|u_{k+j} - u_{k+j-1}\|^2_{R_{du}} \\ s.t. \quad & RNN\ model, \\ & SPE^{pca}_u \leq \varepsilon_u, \\ & u_{min} \leq u \leq u_{max} \end{aligned} \tag{6.12}$$

where $y^{SP}_{a_{k+j}}$ is the achievable set-point and ε_u is an upper bound for the SPE constraint,

and the rest of the parameters and variables are as defined earlier in Eq. (6.6). Note that ε_u must be small enough to restrict the manipulated input moves to the PCA model plane, constructed using the training input data. The value of this variable is determined by computing the upper confidence limit of the SPE statistic. The SPE value in Eq. (6.12) is also calculated as follows:

$$SPE_u^{pca} = \bar{u}^T(I - P_u {P_u}^T)\bar{u} \tag{6.13}$$

where $\bar{u}$ is the vector of the manipulated inputs, mean-centered and scaled using the mean and standard deviation of the training input data, and P_u is the loading matrix of the PCA model, built using the training input data.

The achievable set-point is computed using the following optimization problem:

$$\begin{aligned} \min_{y_a^{SP}} \quad & \|y_a^{SP} - y_g^{SP}\|^2_{Q_{sp}} \\ s.t. \quad & SPE_y^{pca} \leq \varepsilon_y \end{aligned} \tag{6.14}$$

where $y_a^{SP} \in \mathbb{R}^l$ is the achievable set-point (decision variables) to be calculated, $y_g^{SP} \in \mathbb{R}^l$ represents the given set-point, and Q_{sp} is a positive definite matrix used to penalize the deviation of the achievable set-point from the given set-point. The SPE constraint is used to make sure that a similar correlation to that of the training output data is obeyed when computing the achievable set-points. To this end, the upper bound of the SPE constraint, ε_y, must be sufficiently small so that the selection of the achievable set-points is limited to the PCA model plane. The SPE value is also computed as follows:

$$SPE_y^{pca} = (\bar{y}_a^{SP})^T (I - P_y P_y{}^T)(\bar{y}_a^{SP}) \tag{6.15}$$

where $\bar{y}_a^{SP}$ is the vector of the achievable set-points, mean-centered and scaled using the mean and standard deviation of the training output data, and P_y is the loading matrix of the PCA model developed based on the training output data. A schematic flowchart illustrating the steps required to implement the constrained MPC is shown in Fig. 6.8.

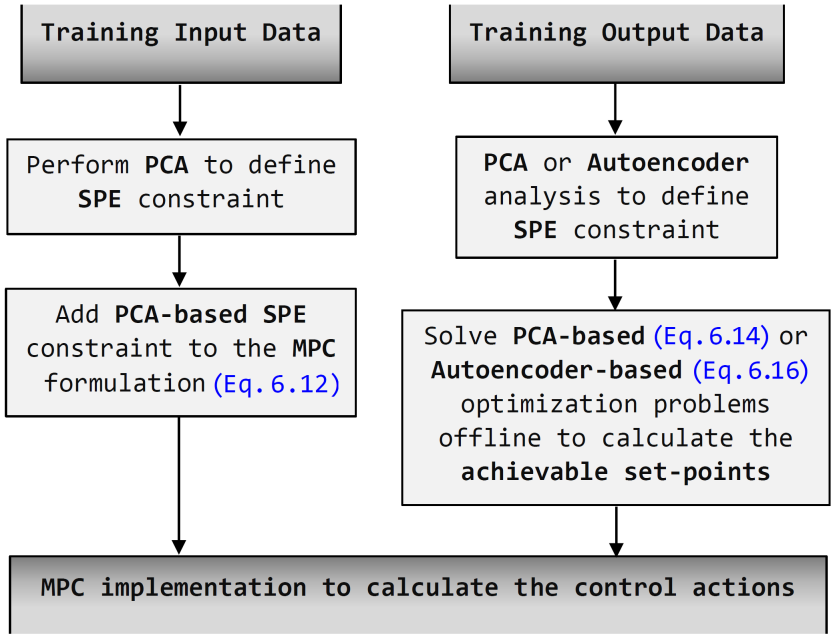


Figure 6.8: Schematic flowchart of the proposed constrained MPC.

6.3.2 Application to the motivating example

Consider the given set-points in Scenarios 1 and 2 introduced in Section 6.2.3, the achievable set-points are first calculated using the optimization problem given by Eq. (6.14). To define the SPE constraint, a PCA model is developed based on the training output data, described in Section 6.2.2. The PCA model is able to explain 99.7% of

the variance in the data using two principal components. The SPE constraint is then formulated using a sufficiently small value ($\varepsilon_y = 10^{-3}$) as the upper bound to make sure that the correlation between the calculated achievable set-points is maintained. Next, the optimization problem is solved by applying a penalty matrix, similar to Q_y, used in the RNN-based MPC implementation, thus $Q_{sp} = \text{diag}([10^4,\ 10^1,\ 10^1])$. It should be noted that different penalty weights can be used based on the importance of each variable. The achievable set-points, $y_{a_1}^{SP}$, obtained by solving the optimization problem subject to the PCA-based SPE constraint are reported in the second row of Tables 6.4 and 6.5 for Scenarios 1 and 2, respectively.

In addition, the SPE values of the training output data, the given set-point (y_g^{SP}), and the achievable set-point ($y_{a_1}^{SP}$), in Scenario 1, are shown in Fig. 6.9 (note that the observations are not contiguous in time; thus, observations 1 to 14000 are related to the training output data, and observations 14001 to 14040 and 14041 to 14080 correspond to the given and achievable set-points, respectively). As can be seen, the SPE values corresponding to y_g^{SP} exceed the 99% confidence limit, resulting in breaking the correlation that exists between the outputs in the training data, while applying $y_{a_1}^{SP}$ leads to maintaining the existing correlation.

The achievable set-points are next used to implement the constrained RNN-based MPC using Eq. (6.12). To define the SPE constraint on the manipulated inputs in the MPC optimization problem, a PCA model is developed using the training input data. Using two principal components, 99.3% of the data variance can be explained. The loading vectors of the PCA model are then utilized to implement the SPE constraint. In addition, the value corresponding to 99% confidence limit ($\varepsilon_u = 0.19$) is considered as the upper bound of the SPE constraint to guarantee that the calculated manipulated inputs follow a similar correlation to that observed in the training input data. Finally, the constrained RNN-based MPC is implemented, and the results are shown in Figs. 6.10 and 6.11 for Scenarios 1 and 2, respectively

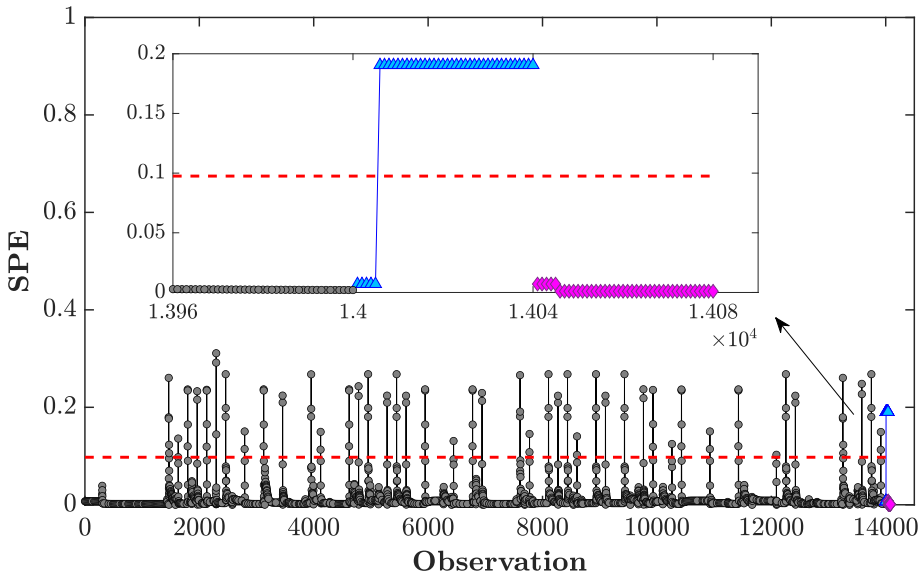


Figure 6.9: Comparison of the SPE values obtained based on the training output data (gray circles), the given set-points, y_g^{SP}, (blue triangles), the achievable set-points obtained based on PCA-based optimization problem, $y_{a_1}^{SP}$, (magenta diamonds), and 99% confidence limit (dashed line), in Scenario 1.

(dash-dotted red lines). As can be seen, the constrained RNN-based MPC is able to stabilize the outputs around the achievable set-points.

To compare the approach proposed in the present manuscript with the approaches in Hassanpour et al. (2020a) and Hassanpour et al. (2021), two implementations are presented. The first is the constrained RNN-based MPC, albeit with arbitrarily given set-points (Hassanpour et al. (2020a)) and the next is the standard RNN-based MPC with the achievable set-points (Hassanpour et al. (2021)). As shown by the solid blue lines in Figs. 6.10 and 6.11, the given set-points ($y_{a_1}^{SP}$) are not achieved by the constrained RNN-based MPC due to the use of the set-points which are selected independently, although the correlation is maintained in the input space. In addition, as shown by the dash-dotted red lines in Figs. 6.5 and 6.6, the standard RNN-based MPC is not able to drive the outputs to their set-points, although the achievable set-points are implemented. In summary, the simulations show the necessity of accounting for

correlations in both the input and output spaces, and ensuring that the MPC is given the achievable set-points. The present section, however, utilizes PCA to compute the set-points. Possibly improved performance using autoencoders is demonstrated in the next section.

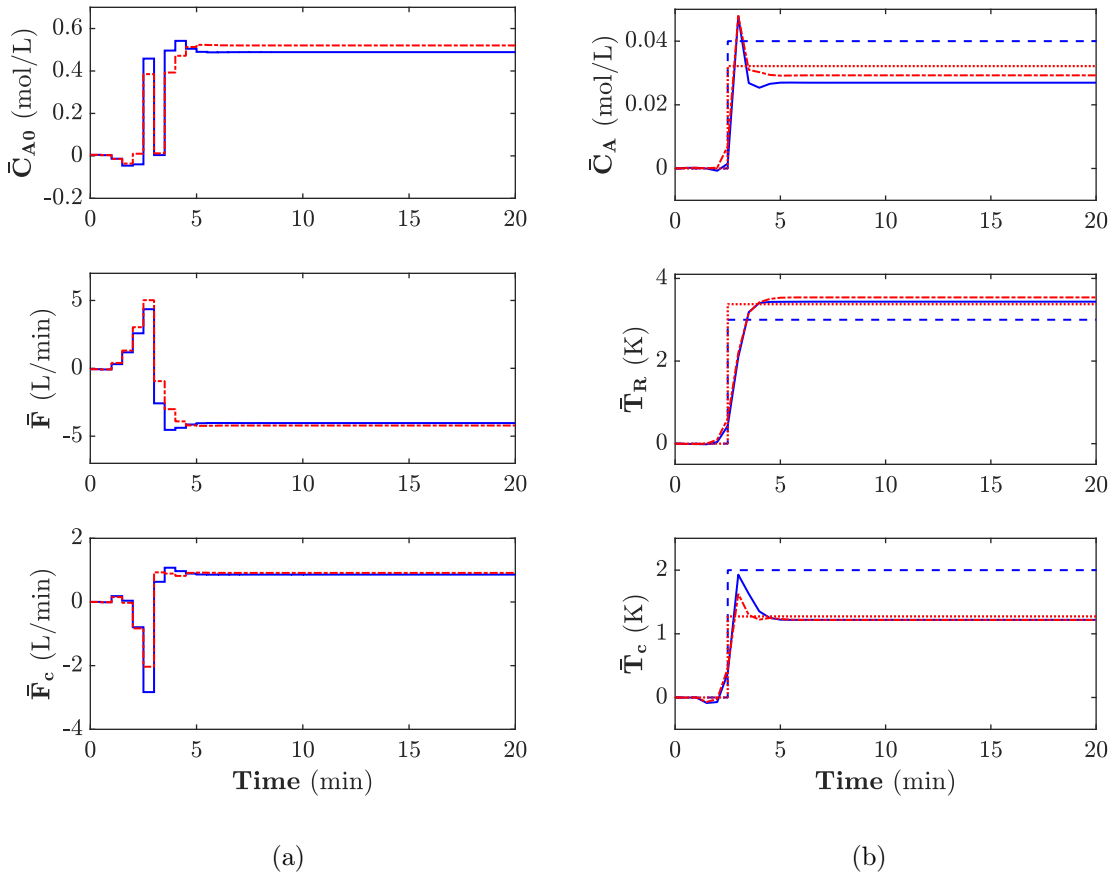


Figure 6.10: Constrained MPC performance in Scenario 1: (a) the manipulated input profiles obtained using y_g^{SP} (solid blue lines) and $y_{a_1}^{SP}$ (dash-dotted red lines). (b) y_g^{SP} (dashed blue lines), the output profiles based on y_g^{SP} (solid blue lines), $y_{a_1}^{SP}$ (dotted red lines), and the output profiles based on $y_{a_1}^{SP}$ (dash-dotted red lines).

Fig. 6.12 summarizes the SPE values associated with the training input data and the control actions obtained based on the standard and constrained RNN-based MPC implementations using y_g^{SP} and $y_{a_1}^{SP}$ in Scenario 1. It should be noted that the observations are not contiguous in time. Therefore, observations 1 to 14000 are related to

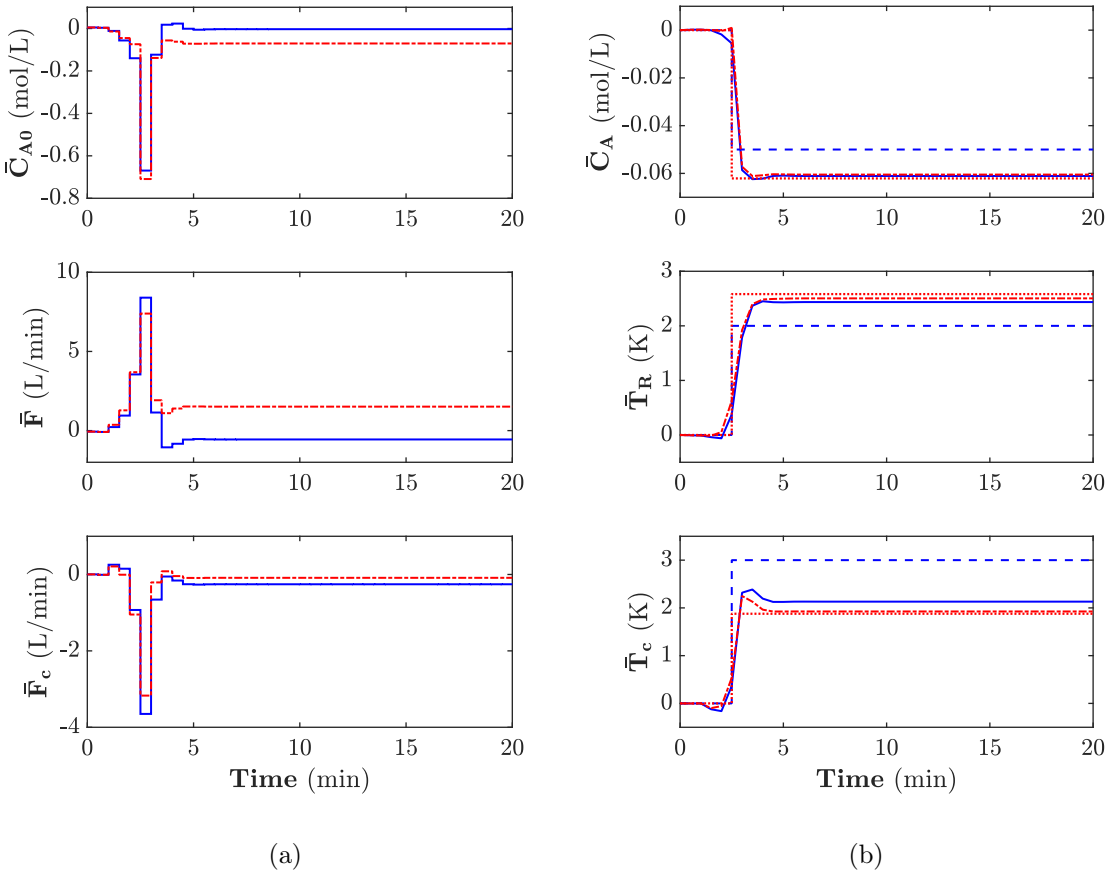


Figure 6.11: Constrained MPC performance in Scenario 2: (a) the manipulated input profiles obtained using y_g^{SP} (solid blue lines) and $y_{a_1}^{SP}$ (dash-dotted red lines). (b) y_g^{SP} (dashed blue lines), the output profiles based on y_g^{SP} (solid blue lines), $y_{a_1}^{SP}$ (dotted red lines), and the output profiles based on $y_{a_1}^{SP}$ (dash-dotted red lines).

the training input data, used to build the PCA model. Observations 14001 to 14040 and 14041 to 14080 are related to control actions calculated by the standard RNN-based MPC using y_g^{SP} and $y_{a_1}^{SP}$, respectively. As shown, these SPE values exceed the 99% confidence limit, indicating that the control actions do not follow the existing correlation between inputs in the training data. In addition, observations 14081 to 14120 and 14121 to 14160 are related to control actions calculated by the constrained RNN-based MPC using y_g^{SP} and $y_{a_1}^{SP}$, respectively. These results show the existing correlation can be maintained by applying the constrained RNN-based MPC. In sum-

mary, the results show that when the RNN model, developed using the training data containing correlated inputs and outputs, is utilized to implement MPC, it is necessary to maintain the existing correlations in both the input and output spaces, while calculating manipulated inputs and prescribing set-points, respectively.

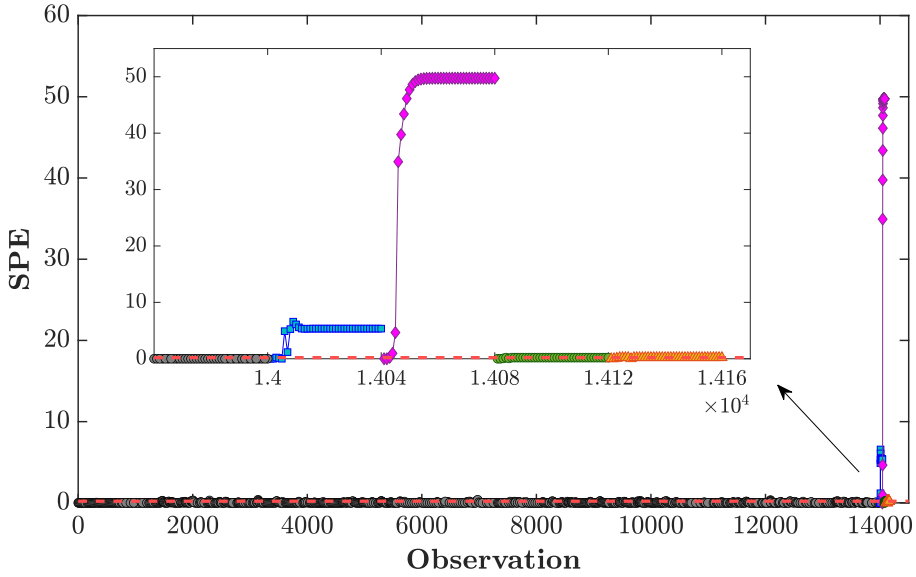


Figure 6.12: Comparison of the SPE values obtained based on the training input data (gray circles), standard RNN-based MPC control actions using y_g^{SP} (blue squares), standard RNN-based MPC control actions using $y_{a_1}^{SP}$ (magenta diamonds), constrained RNN-based MPC control actions using y_g^{SP} (green circles), constrained RNN-based MPC control actions using $y_{a_1}^{SP}$ (orange triangles), and 99% confidence limit (dashed line), in Scenario 1.

Remark 2. *As shown, applying the prescribed set-points, selected without considering the correlation in the output space, may result in a poor performance of the RNN-based MPC (or any data-driven model-based control strategy). Thus, we use the PCA-based optimization problem to compute the achievable set-points, as close as possible to the arbitrarily prescribed ones, while following the same correlation as that of the outputs in the training data. This can be done by selecting appropriate weights, when defining the penalty matrix Q_{sp}, according to the importance of each controlled variable.*

6.4 Proposed autoencoder-based approach to calculate the achievable set-points

6.4.1 Autoencoder-based optimization problem to calculate the achievable set-points

The constrained RNN-based MPC performance to track the achievable set-points, computed using the PCA-based approach, is shown in the previous section. As seen in Figs. 6.10 and 6.11, although the MPC controller is able to track these set-points with reasonable accuracy, small offsets are observed. This indicates that the achievable set-points are not truly achievable, with the error resulting from the linear nature of the PCA-based technique. Therefore, the use of techniques that allow for handling nonlinear correlation between the output variables can result in more accurate values of the set-points. Due to the ability of autoencoder neural networks to capture nonlinear relationships between the variables, an autoencoder-based technique is presented to compute the achievable set-points. In this regard, an optimization problem is defined with the same objective function as in Eq. (6.14), albeit with different SPE constraint, computed as follows:

$$\begin{aligned} \min_{y_a^{SP}} \quad & \|y_a^{SP} - y_g^{SP}\|^2_{Q_{sp}} \\ s.t. \quad & SPE_y^{ae} \leq \varepsilon_y^{ae} \end{aligned} \tag{6.16}$$

where SPE_y^{ae} is the autoencoder-based SPE constraint calculated using Eq. (6.11), and ε_y^{ae} is a sufficiently small value, considered as the upper bound for the SPE constraint.

6.4.2 Constrained RNN-based MPC using autoencoder-based achievable set-points

The autoencoder-based optimization problem, given by Eq. (6.16), is used to calculate the set-points for the motivating example, and then the constrained RNN-based MPC is implemented using Eqs. (6.12) and (6.13).

To compute the achievable set-points, the training output data is first normalized, and then used to develop an autoencoder model. Several architectures including 1 and 2 hidden layers with different activation functions are evaluated to tune the autoencoder hyperparameters. Different autoencoder networks are trained multiple times, and then their scaled RMSE values are assessed in the validation phase. The results indicate that the autoencoder with 1 hidden layer containing 2 neurons with *sigmoid* activation function, for both the encoder and decoder, has superior performance over other architectures. This autoencoder network is further trained multiple times, and the final model is selected based on the lowest value of the scaled RMSE. In the next step, an optimization problem subject to the autoencoder-based SPE constraint is formulated with the same penalty matrix (Q_{sp}), used in Section 6.3.1. A similar small value, $\varepsilon_y = 10^{-3}$, is also selected so as to maintain the correlation between the calculated set-points. The achievable set-points, $y_{a_2}^{SP}$, obtained by solving the optimization problem are listed in the third row of Tables 6.4 and 6.5 for Scenarios 1 and 2, respectively.

Having obtained the set-point $y_{a_2}^{SP}$, the constrained RNN-based MPC is implemented. As can be seen in Figs. 6.13 and 6.14 (for Scenarios 1 and 2), applying these set-points results in reducing the offsets that are obtained using the PCA-based achievable set-points ($y_{a_1}^{SP}$). In addition, in order to quantitatively evaluate the RNN-based MPC performance based on $y_{a_1}^{SP}$ and $y_{a_2}^{SP}$, weighted integral absolute error (IAE) is used and computed as follows:

$$\text{Weighted IAE} = \sum_{i=1}^{l} \sum_{j=1}^{n} W_i |E_{i,j}| \Delta t \tag{6.17}$$

where l is the number of outputs, n is the number of sampled data points for each output, W_i is the weight associated with each output, $E_{i,j}$ is the error between the i^{th} output and its set-point at j^{th} sampling point ($y_{i,j}^{SP} - y_{i,j}$), and Δt is the sampling time. Consider $W_1 = 50$, $W_2 = 1$, and $W_3 = 1$ for the first, second, and third outputs (C_A, T_R, T_c), respectively, and $\Delta t = 0.5$ min, the value of the weighted IAE is calculated based on $y_{a_1}^{SP}$ and $y_{a_2}^{SP}$, and listed in Table 6.6. The results show the performance of the RNN-based MPC is improved when the achievable set-points obtained by the autoencoder-based optimization problem are used.

Table 6.6: Comparison of the weighted IAE values obtained by applying $y_{a_1}^{SP}$ and $y_{a_2}^{SP}$.

Scenario	IAE ($y_{a_1}^{SP}$)	IAE ($y_{a_2}^{SP}$)
1	9.75	7.72
2	7.68	6.15

Remark 3. *Note that while the autoencoder is used in computing the achievable set-points, only the PCA-based SPE constraint is used in the RNN-based MPC to maintain the correlation. In principle, one can use an autoencoder-based SPE constraint in the RNN-based MPC to improve its performance. The autoencoder-based SPE constraint can lead to more computationally expensive optimization problem for the RNN-based MPC due to the use of the nonlinear autoencoder model (compared with the PCA-based SPE constraint that uses the linear PCA model). The use of the autoencoder-based constraint in the MPC can be used to improve the MPC performance when the existing correlation between the inputs in the training data is considerably nonlinear. On the other hand, because the optimization problem to find the achievable set-points is solved offline, the autoencoder-based constraint can be used with less concern about*

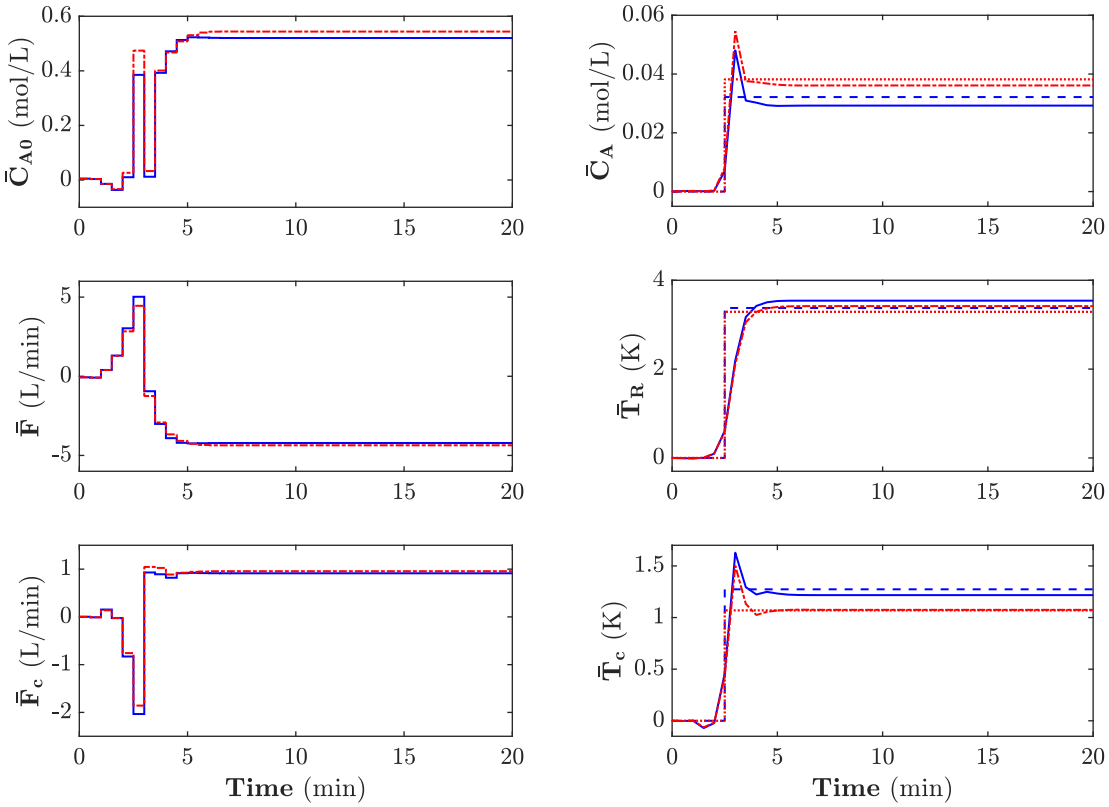
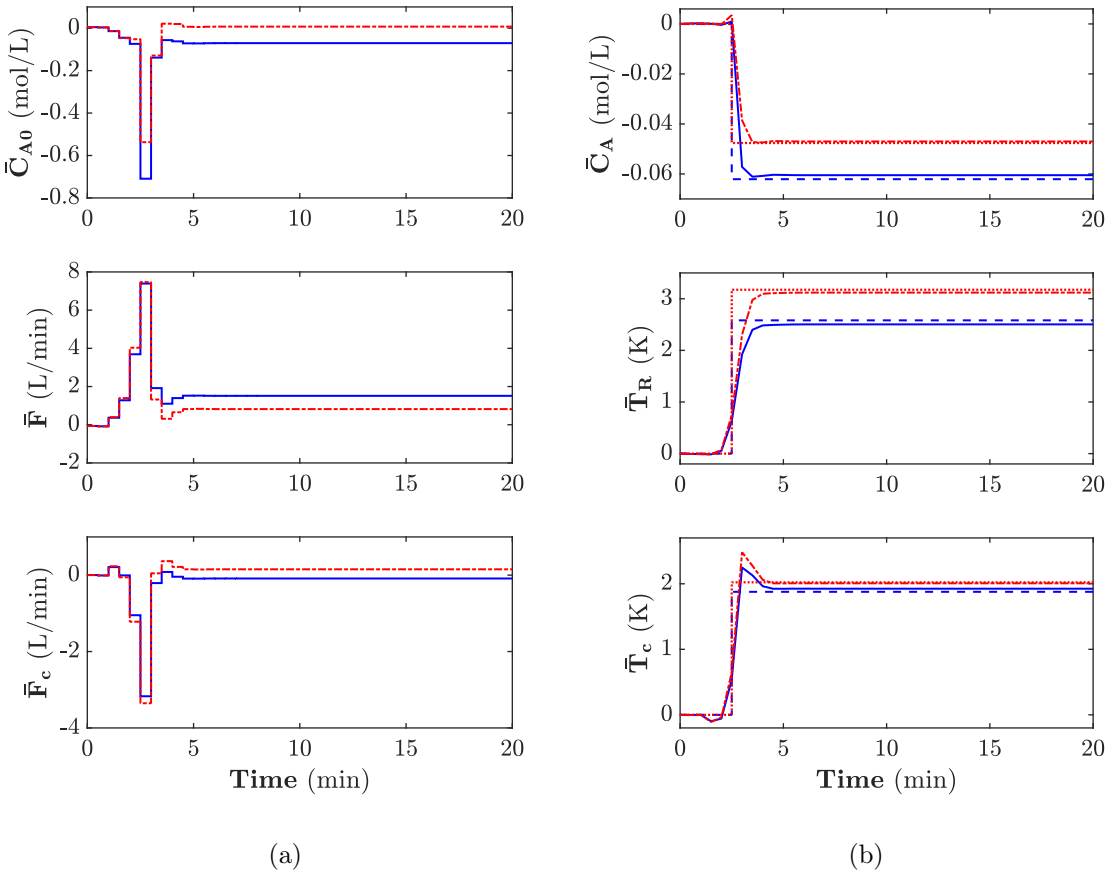


Figure 6.13: Constrained MPC performance in Scenario 1: (a) the manipulated input profiles obtained using $y_{a_1}^{SP}$ (solid blue lines) and $y_{a_2}^{SP}$ (dash-dotted red lines). (b) $y_{a_1}^{SP}$ (dashed blue lines), the output profiles based on $y_{a_1}^{SP}$ (solid blue lines), $y_{a_2}^{SP}$ (dotted red lines), and the output profiles based on $y_{a_2}^{SP}$ (dash-dotted red lines).

the computational complexity.

Remark 4. *In this work, a PCA-based SPE constraint on the input space is added to the MPC optimization problem to effectively handle the plant-model mismatch by restricting the input moves to the PCA model plane, and thus following the same correlation as that of the training input data. Although applying this technique results in an improved control performance, such an implementation may disregard possible improvements in economic performance that can be obtained by making use of the entire space of movements of the manipulated inputs. This, however, necessitates*



(a) (b)

Figure 6.14: Constrained MPC performance in Scenario 2: (a) the manipulated input profiles obtained using $y_{a_1}^{SP}$ (solid blue lines) and $y_{a_2}^{SP}$ (dash-dotted red lines). (b) $y_{a_1}^{SP}$ (dashed blue lines), the output profiles based on $y_{a_1}^{SP}$ (solid blue lines), $y_{a_2}^{SP}$ (dotted red lines), and the output profiles based on $y_{a_2}^{SP}$ (dash-dotted red lines).

a model capable of predicting the system behavior throughout the whole (or broader) range of input moves (developing such a model in turn requires richer training data). To address this issue, we proposed a method in order to produce richer data using a relaxed SPE constraint in the constrained RNN-based MPC, while not significantly degrading the controller performance (Hassanpour et al. (2020a)). In addition, several approaches are developed by integrating the idea of offset-free MPC with existing data-driven based strategies (Kheradmandi and Mhaskar (2018); Son et al. (2020)).

6.5 Conclusions

The problem of handling collinearity in inputs and outputs, and possibly nonlinear correlation in outputs, in RNN-based MPC implementation is addressed in the present manuscript. To maintain the model validity within the MPC framework, PCA-based strategies are first utilized to recognize the existing correlations in the input and output spaces. The constrained RNN-based MPC, using the PCA-based SPE constraint, is then designed to compute the control actions by maintaining a similar correlation to that of the training input data. An optimization problem subject to the PCA-based SPE constraint is also formulated to compute the achievable set-points by ensuring that they follow the same correlation as that of the training output data. Next, a new autoencoder-based optimization problem is proposed to calculate the achievable set-points (due to its capability to recognize the nonlinear correlations between the process outputs). The efficacy of the proposed approach is shown via simulation of a chemical reactor example. The results show the superior performance of the constrained RNN-based MPC over the standard RNN-based MPC to drive the process outputs to the achievable set-points. Finally, further improvements in closed loop performance are demonstrated with the use of an autoencoder-based strategy to calculate the achievable set-points.

Bibliography

Aliabadi, M. M., Emami, H., Dong, M., and Huang, Y. (2020). Attention-based recurrent neural network for multistep-ahead prediction of process performance. *Computers & Chemical Engineering*, 140:106931.

Ayinde, B. O. and Zurada, J. M. (2017). Nonredundant sparse feature extraction using autoencoders with receptive fields clustering. *Neural Networks*, 93:99–109.

Belkadi, M. A., Daamouche, A., and Melgani, F. (2021). A deep neural network approach to QRS detection using autoencoders. *Expert Systems with Applications*, 184:115528.

Bellemans, A., Aversano, G., Coussement, A., and Parente, A. (2018). Feature extraction and reduced-order modelling of nitrogen plasma models using principal component analysis. *Computers & Chemical engineering*, 115:504–514.

Boussaada, Z., Curea, O., Remaci, A., Camblong, H., and Mrabet Bellaaj, N. (2018). A nonlinear autoregressive exogenous (NARX) neural network model for the prediction of the daily direct solar radiation. *Energies*, 11(3):620.

Cadenas, E., Rivera, W., Campos-Amezcua, R., and Heard, C. (2016). Wind speed prediction using a univariate ARIMA model and a multivariate NARX model. *Energies*, 9(2):109.

Cheng, F., He, Q. P., and Zhao, J. (2019). A novel process monitoring approach based on variational recurrent autoencoder. *Computers & Chemical Engineering*, 129:106515.

Drgoňa, J., Picard, D., Kvasnica, M., and Helsen, L. (2018). Approximate model predictive building control via machine learning. *Applied Energy*, 218:199–216.

Du, M. and Mhaskar, P. (2014). Isolation and handling of sensor faults in nonlinear systems. *Automatica*, 50(4):1066–1074.

Ganesh, H. S., Seo, K., Fritz, H. E., Edgar, T. F., Novoselac, A., and Baldea, M. (2021). Indoor air quality and energy management in buildings using combined moving horizon estimation and model predictive control. *Journal of Building Engineering*, 33:101552.

Godoy, J. L., González, A. H., and Normey-Rico, J. E. (2016). Constrained latent variable model predictive control for trajectory tracking and economic optimization in batch processes. *Journal of Process Control*, 45:1–11.

Golshan, M., MacGregor, J. F., Bruwer, M. J., and Mhaskar, P. (2010). Latent variable model predictive control (LV-MPC) for trajectory tracking in batch processes. *Journal of process control*, 20(4):538–550.

Hale, E. T. and Qin, S. J. (2002). Subspace model predictive control and a case study. *In Proceedings of the American Control Conference*, 6:4758–4763.

Han, J., Kamber, M., and Pei, J. (2011). *Data mining concepts and techniques third edition*. The Morgan Kaufmann Series in Data Management Systems.

Han, Y., Ding, N., Geng, Z., Wang, Z., and Chu, C. (2020). An optimized long short-term memory network based fault diagnosis model for chemical processes. *Journal of Process Control*, 92:161–168.

Hassanpour, H., Corbett, B., and Mhaskar, P. (2020a). Integrating dynamic neural network models with principal component analysis for adaptive model predictive control. *Chemical Engineering Research and Design*, 161:26–37.

Hassanpour, H., Corbett, B., and Mhaskar, P. (2021). Artificial neural network based model predictive control: Implementing achievable set-points. *AIChE Journal*, page e17436.

Hassanpour, H., Mhaskar, P., House, J. M., and Salsbury, T. I. (2020b). A hybrid modeling approach integrating first-principles knowledge with statistical methods for fault detection in HVAC systems. *Computers & Chemical Engineering*, 142:107022.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Hu, B., Zhao, Z., and Liang, J. (2012). Multi-loop nonlinear internal model controller design under nonlinear dynamic PLS framework using ARX-neural network model. *Journal of Process Control*, 22(1):207–217.

Huang, B., Ding, S. X., and Qin, S. J. (2005). Closed-loop subspace identification: an orthogonal projection approach. *Journal of process control*, 15(1):53–66.

Inapakurthi, R. K., Miriyala, S. S., and Mitra, K. (2020). Recurrent neural networks based modelling of industrial grinding operation. *Chemical Engineering Science*, 219:115585.

Janakiraman, V. M., Nguyen, X., and Assanis, D. (2013). Nonlinear identification of a gasoline HCCI engine using neural networks coupled with principal component analysis. *Applied Soft Computing*, 13(5):2375–2389.

Kheradmandi, M. and Mhaskar, P. (2018). Model predictive control with closed-loop re-identification. *Computers & Chemical Engineering*, 109:249–260.

Korda, M. and Mezić, I. (2018). Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160.

Koschwitz, D., Frisch, J., and Van Treeck, C. (2018). Data-driven heating and cooling load predictions for non-residential buildings based on support vector machine regression and NARX recurrent neural network: A comparative study on district scale. *Energy*, 165:134–142.

Kresta, J. V., Marlin, T. E., and MacGregor, J. F. (1994). Development of inferential process models using PLS. *Computers & Chemical Engineering*, 18(7):597–611.

Kristjanpoller, W. and Minutolo, M. C. (2018). A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. *Expert Systems with Applications*, 109:1–11.

Li, C., Zhao, D., Mu, S., Zhang, W., Shi, N., and Li, L. (2019). Fault diagnosis for distillation process based on CNN–DAE. *Chinese Journal of Chemical Engineering*, 27(3):598–604.

Ma, Y. and Li, H. (2020). Gru-auto-encoder neural network based methods for diagnosing abnormal operating conditions of steam drums in coal gasification plants. *Computers & Chemical Engineering*, 143:107097.

Masti, D. and Bemporad, A. (2021). Learning nonlinear state–space models using autoencoders. *Automatica*, 129:109666.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814.

Mhaskar, P., El-Farra, N. H., and Christofides, P. D. (2006). Stabilization of nonlinear systems with state and control constraints using lyapunov-based predictive control. *Systems & Control Letters*, 55(8):650–659.

Miriyala, S. S. and Mitra, K. (2020). Deep learning based system identification of industrial integrated grinding circuits. *Powder Technology*, 360:921–936.

Miriyala, S. S., Mittal, P., Majumdar, S., and Mitra, K. (2016). Comparative study of surrogate approaches while optimizing computationally expensive reaction networks. *Chemical Engineering Science*, 140:44–61.

Moonen, M., De Moor, B., Vandenberghe, L., and Vandewalle, J. (1989). On-and off-line identification of linear state-space models. *International Journal of Control*, 49(1):219–232.

Narasingam, A. and Kwon, J. S. I. (2019). Koopman lyapunov-based model predictive control of nonlinear chemical process systems. *AIChE Journal*, 65(11):e16743.

Narasingam, A., Son, S. H., and Kwon, J. S. I. (2021). Data-driven feedback stabilisation of nonlinear systems: Koopman-based model predictive control. *International Journal of Control*, page 1.

Nguyen, H. D., Tran, K. P., Thomassey, S., and Hamad, M. (2021). Forecasting and anomaly detection approaches using LSTM and LSTM autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*, 57:102282.

Nomikos, P. and MacGregor, J. F. (1994). Monitoring batch processes using multiway principal component analysis. *AIChE Journal*, 40(8):1361–1375.

Patel, N., Corbett, B., and Mhaskar, P. (2021). Model predictive control using subspace model identification. *Computers & Chemical Engineering*, 149:107276.

Qin, S. J. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764.

Qin, S. J. and Ljung, L. (2003). Closed-loop subspace identification with innovation estimation. *IFAC Proceedings Volumes*, 36(16):861–866.

Ramamurthy, M., Robinson, Y. H., Vimal, S., and Suresh, A. (2020). Auto encoder based dimensionality reduction and classification using convolutional neural networks for hyperspectral images. *Microprocessors and Microsystems*, 79:103280.

Rawlings, J. B. and Mayne, D. Q. (2009). *Model Predictive Control Theory and Design*. Nob Hill Publishing, LLC.

Sadeghassadi, M., Macnab, C. J., Gopaluni, B., and Westwick, D. (2018). Application of neural networks for optimal-setpoint design and MPC control in biological wastewater treatment. *Computers & Chemical Engineering*, 115:150–160.

Son, S. H., Narasingam, A., and Kwon, J. S. I. (2020). Handling plant-model mismatch in koopman lyapunov-based model predictive control via offset-free control framework. *arXiv preprint arXiv:2010.07239*.

Song, F., Guo, Z., and Mei, D. (2010). Feature selection using principal component analysis. *In international conference on system science, engineering design and manufacturing informatization*, 1:27–30.

Sun, W., Paiva, A. R., Xu, P., Sundaram, A., and Braatz, R. (2020). Fault detection and identification using bayesian recurrent neural networks. *Computers & Chemical Engineering*, 141:106991.

Temeng, K. O., Schnelle, P. D., and McAvoy, T. J. (1995). Model predictive control of an industrial packed bed reactor using neural networks. *Journal of Process Control*, 5(1):19–27.

Van Overschee, P. and De Moor, B. (1995). A unifying theorem for three subspace system identification algorithms. *Automatica*, 31(12):1853–1864.

Wise, B. M., Ricker, N. L., Veltkamp, D. F., and Kowalski, B. R. (1990). A theoretical basis for the use of principal component models for monitoring multivariate processes. *Process Control and Quality*, 1(1):41–51.

Wong, W. C., Chee, E., Li, J., and Wang, X. (2018). Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. *Mathematics*, 6(11):242.

Wu, Z., Luo, J., Rincon, D., and Christofides, P. (2021a). Machine learning-based predictive control using noisy data: Evaluating performance and robustness via a

large-scale process simulator. *Chemical Engineering Research and Design*, 168:275–287.

Wu, Z., Rincon, D., Luo, J., and Christofides, P. D. (2021b). Machine learning modeling and predictive control of nonlinear processes using noisy data. *AIChE Journal*, 67(4):e17164.

Wu, Z., Tran, A., Rincon, D., and Christofides, P. D. (2019). Machine learning-based predictive control of nonlinear processes. part i: Theory. *AIChE Journal*, 65:e16729.

Yan, R., Ma, Z., Kokogiannakis, G., and Zhao, Y. (2016). A sensor fault detection strategy for air handling units using cluster analysis. *Automation in Construction*, 70:77–88.

Yoon, S. and MacGregor, J. F. (2004). Principal-component analysis of multiscale data for process monitoring and fault diagnosis. *AIChE Journal*, 50(11):2891–2903.

Zhan, S. and Chong, A. (2021). Building occupancy and energy consumption: Case studies across building types. *Energy and Built Environment*, 2(2):167–174.

Zheng, S. and Zhao, J. (2020). A new unsupervised data mining method based on the stacked autoencoder for chemical process fault diagnosis. *Computers & Chemical Engineering*, 135:106755.

Chapter 7

Conclusions and future work

The main contributions of the thesis, together with future work opportunities are summarized in this chapter.

7.1 Conclusions

This work considers addressing the problem of fault diagnosis and MPC implementations in the presence of limited and uninformative data. In Chapter 2, a hybrid machine learning approach is proposed by integrating a pre-trained RNN model (built using a large amount of data for a representative unit) with SubID models (developed to model residuals for the units in question) to handle unavailability of sufficient data. The key point is to take advantages of both the RNN and SubID modeling techniques (to handle over-fitting and achieve better performance while using different amount of data). The application of the proposed approach to predict the behaviour of different zones in a multi-zone fitness centre is evaluated. The results show the superior performance of the hybrid RNN-SubID approach in handling data shortage problem compared with the individual RNN and SubID models.

In Chapter 3, a hybrid PCA technique is proposed by integrating first-principles knowledge with data in order to detect faults in HVAC systems. The key point is to utilize the residuals obtained based on the first-principles knowledge to enhance the performance of purely data-driven based PCA method (when data samples are not sufficiently informative). The performance of the proposed hybrid approach is assessed using different examples. The results reveal the superior performance of the hybrid approach over the purely data-driven (PCA using data) and first-principles (residuals) based approaches.

In Chapter 4, the problem of handling correlated input data for the RNN-based MPC is addressed. Two approaches (PCA-RNN-based MPC and Constrained RNN-based

MPC) are developed to address this issue. In the first approach, PCA is performed on the input data, and then the scores are used to develop an RNN model. The PCA-RNN model is then employed within the MPC framework to compute the optimal scores. In the second approach, an RNN model (developed using the input and output data) is used in the MPC, but a new PCA-based SPE constraint is added to ensure that input movements are confined to the PCA model plane. The effectiveness of the proposed methods (over the standard RNN-based MPC implementation) is illustrated using different set-point tracking scenarios for a chemical reactor example. Finally, a new method is proposed to break the existing correlation in the constrained RNN-based MPC actions in order to generate richer data samples for re-identification to possibly enhance the economic performance of the RNN-based MPC.

In Chapter 5, the problem of implementing achievable set-points for the RNN-based MPC implementations, where output data are correlated is addressed. A PCA-based optimization problem is defined to compute the achievable set-points from arbitrarily prescribed ones. The results show the superior performance of the MPC performance using the achievable set-point compared to the case where the arbitrarily prescribed set-points are directly used.

In Chapter 6, the proposed methods in Chapters 5 and 6 are integrated to address the problem of handling correlated input and output data for the RNN-based MPC implementations. In addition, to address the problem of existing nonlinear correlation in the output space, a new AE-based optimization problem is formulated to compute achievable set-points. The efficacy of the proposed approach is shown via simulation of a chemical reactor example. The results reveal the superior performance of the constrained RNN-based MPC over the standard RNN-based MPC. The results also show that the closed-loop performance can be improved with the use of the achievable set-points obtained using the AE-based approach.

7.2 Future work

The results and contributions of this thesis suggest the following topics for future work:

1. Investigating transfer learning methods for process modeling in the presence of insufficient data
2. Implementing MPC using the proposed hybrid machine learning (RNN-SubID) modeling technique
3. Evaluating the effect of time delay on the prediction performance of the PCA-RNN models
4. Handling correlated data (with nonlinear relationship in the input space) for the ANN-based MPC implementations
5. Developing an implementable model-free reinforcement learning (RL) technique for process control design

In the transfer learning-based approaches, a pre-trained model (which is usually a DNN), developed for the source problem, is selected to be re-tuned using data from the target problem (modifying all or some of parameters in different layer). In Chapter 2, we propose an alternative approach to address data shortage problem, where the existing pre-trained model of the source problem is a shallow neural network. Further research could be considered on how to extend the idea of transfer learning to cases where a shallow neural network model is available for the source problem (how to re-tune weights and biases in this network to capture the dynamics of the target problem). In addition, MPC implementations using the hybrid (RNN-SubID) approach, proposed in this chapter, can be considered as future work.

In Chapter 4, a PCA-RNN modeling approach is proposed for the case where input data are correlated. This technique is applied assuming two conditions: (1) the inputs are perfectly correlated and (2) the inputs have the same time delay with the outputs. Therefore, how time delay is considered when the inputs are compressed by PCA (for situations where the inputs have different time delays with the outputs) can be investigated as future research. In addition, in Chapters 4, 5, and 6, we propose different methods to address the problem of handling correlated input and output data for MPC implementations. Chapters 4 and 5 consider using PCA to recognize the correlations that exist between the variables in the input and output spaces. An AE-based approach is also proposed in Chapter 6 to handle the nonlinear correlation between the output variables. Another potential future work can investigate the use of AE to handle the nonlinear correlation between the variables in the input space. To this end, a new AE-based SPE constraint (obtained using the input data) can be developed and added to the MPC formulation.

RL-based techniques have shown promising potentials in many process control applications. However, the training process of an RL controller requires a significant amount of online data. Safety and economic problems are obstacles to apply online training in many practical situations. Model-based control techniques such as model-based RL and nonlinear MPC are alternatives to the model-free RL control designs. However, these techniques require a model to be maintained (this can result in additional costs). Another future work can consider developing an implementable model-free RL controller by leveraging offline MPC calculations (this information can be obtained using a pre-implemented industrial MPC that uses a simple step-response model). The RL agent is trained using offline MPC calculations, and then further retuned online by interacting with the actual process to continuously learn and improve its performance.