Identification of Finite Dimensional Spatiotemporal Systems

Hidayat, Z.

DOI
10.4233/uuid:c9f8cf50-e1b5-43e1-9746-8f13b72ce59a

Publication date
2025

Document Version
Final published version

Citation (APA)
Hidayat, Z. (2025). *Identification of Finite Dimensional Spatiotemporal Systems*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:c9f8cf50-e1b5-43e1-9746-8f13b72ce59a

IDENTIFICATION OF FINITE DIMENSIONAL SPATIOTEMPORAL SYSTEMS

IDENTIFICATION OF FINITE DIMENSIONAL SPATIOTEMPORAL SYSTEMS

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op **mandaag 28 april 2025 om 15.00 uur**

door

Zulkifli HIDAYAT

elektrotechniek ingenieur,
Universiteit Twente, Nederland
geboren te Jakarta, Indonesië.

This dissertation has been approved by the promotors.

Prof.dr. R. Babuska
Prof.dr.ir. B.H.K.. De Schutter
Dr. A.A. Nunez Vicencio

Composition of the doctoral committee:

Rector Magnificus	Chairperson
Prof. dr. R. Babuska	Delft University of Technology, promotor
Prof. dr. B.H.K. De Schutter	Delft University of Technology, promotor
Dr. A.A. Nunez Vicencio	Delft University of Technology, promotor
Independent Members:	
Prof.dr.ir. J.W. van Wingerden	Delft University of Technology
Prof.Dr.ing. D. Sáez Hueichapan	University of Chile, Chile
Prof.dr. I. Škrjanc	University of Ljubljana, Slovenia
Dr. B. Shyrokau	Delft University of Technology
Prof.dr.ir. J. Hellendoorn	Delft University of Technology, reserve member

Keywords: System identification, spatiotemporal systems, model selection, nonlinear state estimation
Printed by: Gildeprint

ISBN (print) 978-94-6384-754-4
ISBN (pdf) 978-94-6518-032-8

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

For my parents
and all my teachers in life

SUMMARY

IDENTIFICATION OF FINITE DIMENSIONAL SPATIOTEMPORAL SYSTEMS

Spatiotemporal systems are systems whose dynamics depend on time and space and are commonly found in real life. These systems are mathematically modeled using partial differential equations and are also known as distributed-parameter systems. Due to their structure and the high number of variables involved, control and estimation for this class of systems are very challenging. This thesis addresses two problems related to spatiotemporal systems: state estimation and system identification.

Monitoring the states of a control system is important to ensure the behavior of the system is achieving the control objectives. This can be achieved, among others, by using state observers that estimate the states of the systems regularly. First, we present a literature review of observer design methods for distributed parameter systems. In general, the design requires a dimension-reduction approach to implement the observer. From the dimension reduction, the design approaches can be classified into late and early lumping. In the late lumping perspective, model reduction is performed at the end of the observer design. In the early lumping perspective, dimension reduction is applied to the model of the system. We incorporate both approaches in our literature review.

We also compare distributed Kalman filter methods for distributed-parameter systems to the centralized Kalman filter. These methods are the parallel information filter, the distributed information filter, the distributed Kalman filter with consensus filter, and the distributed Kalman filter with weighted averaging. The comparison is performed by simulating a set of sensors that can communicate with each other to form a sensor network. We evaluate the characteristics of the methods, including whether the technique:

- Requires centralized estimate computation,
- Requires full connectivity of the sensors,
- Requires intensive communication among sensors,
- Computes global estimates.

We find that the resulting estimates depend on the connectivity level of the sensors. When the sensors are fully connected, the estimates resulting from the distributed Kalman filter are comparable to those of the centralized Kalman filter.

State observer design requires the model of the systems. This thesis also presents a system identification method for distributed-parameter systems. The identification of such systems typically requires spatially dense and regular measurements, followed by selecting sensors that provide significant measurements to the model to reduce the model complexity. However, these requirements may be challenging to fulfill. In case

the sensor locations are irregular and sparse in space, we propose the use of lumped-parameter system identification. In the proposed method, the identification is applied for measurements of each sensor in the system using the following regressors: (1) the measurements at the current sensor, (2) the measurements and system inputs from the neighboring current sensors.

Applying the proposed method results in coupled linear models representing the system at the sensor locations. By incorporating measurements from the neighboring sensors, the obtained models may be unnecessarily complex due to the large number of regressors. For practical application, the models need simplification. In this case, the number of regressors is reduced using lasso.

For models with a large number of regressors, we propose a method for reducing the number of regressors using a tree representation. The tree is a way to list models with different numbers of regressors. From all possible regressors for the model, the proposed method builds the tree from the simplest models, i.e., models with one regressor. The number of regressors in the models is incrementally increased to one or more models with the best performance. The addition is repeated until the tree contains models with the desired maximum number of regressors. Using the proposed method, it is also possible to implement greedy search or exhaustive search. Compared to exhaustive search, the proposed method delivers a model with comparable performance, but it requires significantly less computation time.

System identification is typically performed using a complete data set, i.e., for each input sample, there is an associated output sample available. However, there are cases in which some output samples are not recorded in the data set, making the identification data incomplete. This thesis also considers the problem of incomplete data for Takagi-Sugeno (TS) fuzzy system identification using the product space clustering method. This method comprises two steps: fuzzy clustering and rules construction. The first proposed method enables the use of incomplete system identification data to fuzzy c-means clustering algorithm developed for incomplete classification, which yields different estimates for a missing sample. This can be achieved by fusing those different values into a single value. The second proposed method treats missing samples as optimization variables during the identification process. The optimization is repeated until the change of all optimization variables is small.

In the main part of this thesis, state estimation problems of linear systems are analyzed. In the appendix, we examine the nonlinear state estimation problem in traffic applications. More specifically, we design a TS fuzzy observer for the METANET traffic flow model. It is already known that TS fuzzy models can accurately approximate many nonlinear systems. The state observer development starts from developing an affine TS fuzzy representation of the METANET model using the sector nonlinearity method. The observer gains are computed using linear matrix inequalities based on the stability conditions for TS fuzzy systems.

SAMENVATTING

IDENTIFICATIE VAN EINDIG-DIMENSIONALE SPATIO-TEMPORELE SYSTEMEN

Spatio-temporele systemen zijn systemen waarvan de evolutie afhankelijk is van tijd en ruimte en die vaak in de praktijk voor komen. Deze systemen kunnen wiskundig gemodelleerd worden met behulp van partiële differentiaalvergelijkingen en ze staan ook bekend als systemen met gedistribueerde parameters. Vanwege hun structuur en het grote aantal betrokken variabelen zijn de besturing en schatting voor deze klasse van systemen zeer uitdagend. Dit proefschrift behandelt twee problemen gerelateerd aan spatio-temporele systemen: toestandsschatting en systeemidentificatie.

Het monitoren van de toestanden van een regelsysteem is belangrijk om ervoor te zorgen dat het gedrag van het systeem voldoet aan de regeldoelen. Dit kan onder andere worden bereikt door toestandswaarnemers te gebruiken die regelmatig de toestand van het systeem schatten. Eerst presenteren we een literatuuronderzoek naar ontwerpmethoden voor waarnemers voor systemen met gedistribueerde parameters. Over het algemeen vereist het ontwerp een dimensiereductie-aanpak om de waarnemer te implementeren. Vanuit de dimensiereductie kunnen de ontwerpmethoden worden geclassificeerd in late en vroege *lumping*. In het late *lumping*-perspectief wordt modelreductie aan het einde van het waarnemerontwerp uitgevoerd. In het vroege *lumping*-perspectief wordt dimensiereductie toegepast op het model van het systeem. Beide benaderingen nemen we op in ons literatuuronderzoek.

We vergelijken ook gedistribueerde Kalman-filtermethoden voor systemen met gedistribueerde parameters met het gecentraliseerde Kalman-filter. Deze methoden zijn het parallelle informatiefilter, het gedistribueerde informatiefilter, het gedistribueerde Kalman-filter met consensusfilter en het gedistribueerde Kalman-filter met gewogen middelen. De vergelijking wordt uitgevoerd door een reeks sensoren te simuleren die met elkaar kunnen communiceren en zo een sensornetwerk vormen. We evalueren de kenmerken van de methoden, waaronder of de techniek:

- gecentraliseerde schattingberekening vereist,
- volledige connectiviteit van de sensoren vereist,
- intensieve communicatie tussen sensoren vereist,
- globale schattingen berekent.

We constateren dat de resulterende schattingen afhangen van het connectiviteitsniveau van de sensoren. Wanneer de sensoren volledig zijn verbonden, zijn de schattingen die voortvloeien uit het gedistribueerde Kalman-filter vergelijkbaar met die van het gecentraliseerde Kalman-filter.

Het ontwerp van toestandswaarnemers vereist een model van de systemen. Dit proefschrift presenteert ook een methode voor systeemidentificatie voor systemen met gedistribueerde parameters. De identificatie van dergelijke systemen vereist doorgaans ruimtelijk dichte en regelmatige metingen, gevolgd door het selecteren van sensoren die significante metingen leveren aan het model om zo de modelcomplexiteit te verminderen. Deze vereisten kunnen echter moeilijk te vervullen zijn. In het geval dat de sensorlocaties onregelmatig en schaars in de ruimte zijn, stellen we het gebruik van systeemidentificatie met *lumped* parameters voor. In de voorgestelde methode wordt de identificatie toegepast op de metingen van elke sensor in het systeem met behulp van de volgende regressoren: (1) de metingen bij de huidige sensor, (2) de metingen en systeeminvoer van de naburige huidige sensoren.

Het toepassen van de voorgestelde methode resulteert in gekoppelde lineaire modellen die het systeem bij de sensorlocaties representeren. Door metingen van de naburige sensoren op te nemen, kunnen de verkregen modellen onnodig complex zijn vanwege het grote aantal regressoren. Voor praktische toepassing moeten de modellen worden vereenvoudigd. In dit geval wordt het aantal regressoren verminderd met behulp van lasso.

Voor modellen met een groot aantal regressoren stellen we een methode voor om het aantal regressoren te verminderen met behulp van een boomrepresentatie. De boom is een manier om modellen met verschillende aantallen regressoren op te sommen. Uit alle mogelijke regressoren voor het model bouwt de voorgestelde methode de boom van de eenvoudigste modellen, d.w.z. modellen met één regressor. Het aantal regressoren in de modellen wordt stapsgewijs verhoogd tot één of meer modellen met de beste prestaties. De toevoeging wordt herhaald totdat de boom modellen bevat met het gewenste maximale aantal regressoren. Met de voorgestelde methode is het ook mogelijk om een *greedy* zoekmethode of een uitputtende zoekmethode te implementeren. Vergeleken met uitputtende zoekmethoden levert de voorgestelde methode een model met vergelijkbare prestaties, maar vereist het aanzienlijk minder rekentijd.

Systeemidentificatie wordt doorgaans uitgevoerd met behulp van een volledige dataset, d.w.z. voor elke invoermonster is er een bijbehorend uitvoermonster beschikbaar. Er zijn echter gevallen waarin sommige uitvoermonsters niet zijn opgenomen in de dataset, waardoor de identificatiegegevens onvolledig zijn. Dit proefschrift behandelt ook het probleem van onvolledige gegevens voor Takagi-Sugeno (TS) *fuzzy* systeemidentificatie met behulp van de productruimte-clustermethode. Deze methode bestaat uit twee stappen: *fuzzy clustering* en constructie van regels. De eerste voorgestelde methode maakt het mogelijk om onvolledige systeemidentificatiegegevens te gebruiken voor een *fuzzy-means clustering*algoritme dat ontwikkeld is voor onvolledige classificatie, wat verschillende schattingen oplevert voor een ontbrekend monster. Dit kan worden bereikt door die verschillende waarden te combineren tot één enkele waarde. De tweede voorgestelde methode behandelt ontbrekende monsters als optimalisatievariabelen tijdens het identificatieproces. De optimalisatie wordt herhaald totdat de verandering van alle optimalisatievariabelen klein genoeg is.

In het hoofdgedeelte van dit proefschrift worden toestandschattingsproblemen van lineaire systemen geanalyseerd. In de appendix onderzoeken we het niet-lineaire toestandschattingsprobleem in verkeersapplicaties. Meer specifiek ontwerpen we een TS

fuzzy waarnemer voor het verkeersstroommodel. Het is al bekend dat TS *fuzzy* modellen veel niet-lineaire systemen nauwkeurig kunnen benaderen. De ontwikkeling van de toestandswaarnemer begint met het ontwikkelen van een affiene TS *fuzzy* representatie van het model met behulp van de sector-niet-lineariteitsmethode. De versterkingsfactoren van de waarnemers worden berekend met behulp van lineaire matrixongelijkheden op basis van de stabiliteitsvoorwaarden voor TS *fuzzy* systemen.

CONTENTS

1

INTRODUCTION

This thesis addresses estimation problems related to distributed-parameter systems, a class of systems whose states depend on both time and space. These kinds of systems are called spatiotemporal systems or infinite-dimensional systems. Different from lumped-parameter systems that are represented by ordinary differential equations, distributed-parameter systems are modeled by partial differential equations.

Two problems are considered in this thesis: the system identification problem and the state estimation problem. For the identification problem, the approximation of the model in space and time is addressed. Model reduction and identification with incomplete data are also discussed. For the state estimation problem, we survey methods for observer design and compare distributed Kalman filtering methods.

1.1. BACKGROUND AND MOTIVATION

There are real physical processes that are described by partial differential equations, as opposed to ordinary differential equations for lumped parameter systems. Common examples include:

- Computational electromagnetic fields [213];
- Elastic deformations of bodies under forces [12];
- Process control [192], e.g., heating and cooling problems, chemical reactors, polymer processing operations.
- Mechanical systems [192], e.g., mechanical structure problems, vehicle structure design, control of flexible mechanical platforms;
- Resource recovery [192], e.g., identification of underground oil, reservoirs, and fishery management;
- Environment [192], e.g., environment quality modeling, monitoring, and control;
- Physiological systems [192], e.g., modeling the distribution and effect of drugs and other chemicals in humans and animals.

1

There are two main approaches for simulation, estimation, and control of partial differential equations [191, 225]. These approaches involve approximating the models by finite-dimensional representations, i.e., reducing the number of states of the models. The first approach is called early lumping. In this approach, the model is first approximated in space and then methods for control and estimation for lumped-parameter systems are applied. The second approach is called late lumping. In contrast to the early lumping, in this approach, the design process directly uses partial differential equations.

In the early-lumping approach, the spatial discretization methods that are commonly used are finite-difference and finite-element methods. The finite-difference method approximates the derivatives with discrete values and then solves the resulting system of algebraic equations, while the finite-element method uses a piecewise polynomial over a mesh of the spatial domain to approximate the solution. A disadvantage of this approach is that the approximation error is present from the beginning, and the error is propagated to the end of the design step. In the case of small spatial grid sizes, the dimension of the discretized models is commonly large, so that techniques for large-scale systems are required. These techniques incur high computation costs due to the high model complexity.

In the late-lumping approach, the discretization error only arises in the last step of the design process. A disadvantage of late lumping is that there is no universal design strategy that encompasses different types of partial differential equations. In addition, the computation cost should be carefully considered because of the increasing complexity of the solution after discretization in the case of real-time implementation.

A problem presented in this thesis is the state estimation for distributed-parameter systems. Here, we survey the literature on deterministic state estimation methods using observers. In addition, different distributed Kalman filter algorithms are compared to assess their performance for estimating distributed-parameter systems. The comparison is performed by computer simulation experiments.

The advances in the field of computing devices and in the algorithms for large-scale systems, see, e.g., [92] compensate better the disadvantages of the early-lumping approach than those of the late-lumping approach. Therefore, the early-lumping approach is used in this thesis for system identification by using the finite-difference method for spatial discretization because of the ease of implementation, even for complex spatial geometry. As the aim of system identification is to have a mathematical representation of a system, the early-lumping approach pertains to the parametrization of coupled ordinary differential equations. Depending on the spatial grid size, the number of equations might be very large, where each model equation is interconnected to other equations based on the corresponding location in the grid. Writing those equations into a state-space form model results in a large but sparse state matrix [20]. This leads to computation cost reduction.

The cases considered in this thesis differ from the ones typically found in the literature. Commonly, a discretization of a partial differential equation with the finite difference method makes use of a relatively small and uniform grid, e.g., as in [44, 49, 182]. In [44, 49, 182], the model is known so that the discretization can be easily computed. While this assumption might hold in some cases, there are cases in which no partial differential equation model is available to approximate the process. The only information known is

that the system's behavior is spatially dependent. What one can then do is install sensors to measure the variables of interest and develop a model from those measurements. This is the case we consider in the thesis. For each measurement location, a black-box model is identified. To keep the spatial connections, the model takes measurements from neighboring measurement locations as inputs. These neighbors are not necessarily the nearest adjacent neighbors. Furthermore, the measurement locations might not be arranged regularly in the grid.

The use of measurements from neighboring locations as inputs for a black-box model with different time lags creates a regression model with a large number of parameters. This might lead to an overfitted model, i.e., a complex model with low prediction performance. To avoid overfitting, methods to reduce the regression model complexity have been presented in the literature. In general, they can be classified into two classes. In the first class of approaches, a regressor is added or removed based on statistical tests or performance gains. This includes stepwise regression, backward elimination, and exhaustive search [72]. The second class of approaches includes methods that are based on least-squares optimization with regularization, which increasingly penalizes models with a larger number of parameters. Methods that fall into this class of approaches are ridge regression [113], least absolute selection and shrinkage [217], and the elastic net method [252]. These methods have been commonly used, especially in the machine learning community to simplify regression models [98]. In this Ph.D. thesis, we propose a method that extends the stepwise regression by a tree representation to perform a search for the best model with respect to a specified model structure.

Another problem considered in this thesis involves the identification of nonlinear systems with a Takagi-Sugeno (TS) fuzzy system structure [214]. In this problem, we consider the identification of TS fuzzy systems with incomplete data, namely, with samples that are not available at random time instants in the data acquisition experiment. These missing samples might be caused by, e.g., faults in the sensing devices. In general, the identification of black-box models involves solving a least-squares problem in matrix form. Missing samples create missing elements in the matrix, which cannot be addressed by ordinary identification algorithms. As the TS fuzzy system identification considered here consists of two steps: antecedent parameter estimation by fuzzy clustering and consequent parameter estimation by least squares, the missing samples affect both steps.

Incomplete data problems are hard to deal with for fuzzy clustering because distance computation between the cluster centers and an incomplete data vector is not possible, while the distance is critical information required in the clustering process. In this thesis, we proposed a method to tackle the problem by extending the algorithm of [5]. The fuzzy c-means clustering for the regression problem in [5] is enhanced to allow clustering of system identification data containing missing samples.

For the consequent parameter estimation, methods for parameter estimation of linear system identification can be applied for incomplete data, e.g., the maximum likelihood method [97]. Solving the problem typically involves estimating the missing values in the algorithms. The other proposed method involves an iterative approach between the identification and the estimation of missing values.

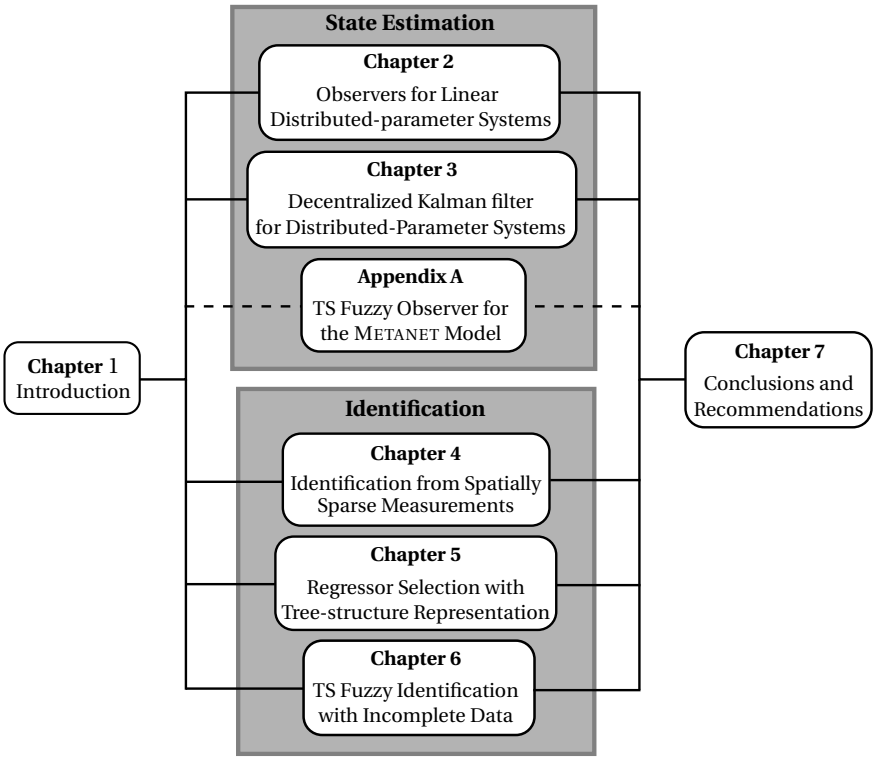


Figure 1.1: Structure of the thesis.

1.2. OVERVIEW OF THE THESIS

1.2.1. STRUCTURE OF THE THESIS

Two topics are discussed in this thesis: state estimation and system identification. The state estimation topic is in Chapter 2, Chapter 3, and Appendix A, and the system identification topic is treated in Chapter 4 to Chapter 6. However, each chapter can be read independently. The structure of this thesis is shown in Figure 1.1.

1.2.2. OUTLINE AND CONTRIBUTIONS

The work presented in this thesis makes the following contributions to the state of the art:

Chapter 2: A survey on observer design for distributed-parameter systems is presented. The survey is focused on the design of several classes of linear systems from the 1970s up to around 2010, including methods for distributed estimation. Some open problems for observer design are also presented. This chapter is based on [112].

Chapter 3: Different methods of noncentralized Kalman filtering for distributed-parameter system applications are considered. The performance of the algorithms is compared for their ability to estimate the states of the process. In addition, we

assess their consistency in terms of the estimate variations as the process reaches a steady state. This chapter is based on [111].

Chapter 4: A system identification method for distributed-parameter systems is proposed. The assumptions are that the underlying partial-differential-equations model is unknown and that the locations of actuators and sensors are given and fixed. The resulting models are black-box models that involve inputs and measurements from other locations. This chapter is based on [108].

Chapter 5: The models obtained in Chapter 4 are complex and may have a large number of parameters. This is because the inputs and measurements from other locations are included in the model. In this chapter, a model reduction method is proposed that searches for the best black-box model based on a specified performance measure by growing a tree representation of the regressors used in the model. This chapter is based on [109].

Chapter 6: Two methods for TS fuzzy system identification in case of incomplete identification data are presented in this chapter. The methods are based on product space clustering, which is a two-step method. The first step performs fuzzy clustering to estimate antecedent parameters, and the second step solves a least-squares problem to estimate consequent parameters. The first identification method is based on completing missing samples by fuzzy clustering so that the data are complete for the antecedent parameter estimation. The second identification method is an iterative method that alternates the identification step and the missing sample estimation step until the performance change is below a prespecified value. This chapter is based on [110].

Appendix A: A TS fuzzy representation of the METANET traffic flow model is derived, and the corresponding observer design is presented. This chapter is based on [107].

2

OBSERVERS FOR LINEAR DISTRIBUTED-PARAMETER SYSTEMS

2.1. INTRODUCTION

In many real-world problems, the states, inputs, and outputs of a mathematical model of a system depend on a spatial variable, which is usually a position in a one-dimensional or multi-dimensional space. These kinds of systems are called distributed-parameter systems, in contrast to lumped-parameter systems, the variables of which do not depend on spatial parameters. Examples of distributed-parameter systems can be found in process control [191], e.g., robotics [151], bioreactors [59], glass feeders [103], biomedical engineering [40], flexible structures [17], and vibrations [175]. An overview of distributed-parameter system applications can be found in [192].

For the operation of a control system, the knowledge of the states of the system is important. In most cases, it is not possible to have full information on the system's states due to the fact that not all variables can be measured. Installing all the necessary sensors may not be physically possible, or the costs may become prohibitive. In such a case, the states can be estimated using state estimators (observers). One of the basic state estimators for linear lumped-parameter systems is the Luenberger observer [162]. For this class of observers, the state estimates are computed by using the model of the system in combination with the difference between the estimated states and the corresponding measurements scaled by a gain. This type of observer has attracted much attention and various design methods have been proposed for both the linear and the nonlinear cases. For surveys on observer design methods for lumped-parameter systems, see, e.g., [171] for nonlinear observers and [209] for sliding-mode observers.

Control and state estimator design for distributed-parameter systems is more complex than in the lumped-parameter case [13, 172, 194, 225]. The presence of spatial variables imposes limitations to the design, e.g., observation and/or actuation may occur at the

boundaries only. Research on observer design for distributed-parameter systems has not been so extensive as in the case of lumped-parameter systems. Furthermore, papers on distributed-parameter observers are scattered in the literature and, to the author's best knowledge, there are no surveys on this topic. This chapter aims at filling this gap. We survey the techniques that are currently available for the design of observers for first-order and second-order linear distributed-parameter systems. In addition, we identify challenges for future research in this field.

2.2. LUMPED-PARAMETER OBSERVER DESIGN

In this section, we briefly recall the observer design problem for lumped-parameter systems. Consider the following linear time-invariant system:

$$\begin{aligned} \dot{x}(t) &= \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \\ y(t) &= \boldsymbol{C}x(t) \end{aligned} \tag{2.1}$$

where $x(t) \in \mathbb{R}^n$, $u \in \mathbb{R}^r$, and $y(t) \in \mathbb{R}^q$ are the input, state, and output vector, respectively, and $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ are state, input, and output matrices that have appropriate dimensions. The dot over a variable indicates the time derivative of the variable. The estimate of the state, $\hat{x}(t)$, is obtained as the output of the following observer [83]:

$$\dot{\hat{x}}(t) = \boldsymbol{A}\hat{x}(t) + \boldsymbol{B}u(t) + \mathscr{L}\big(y(t) - \boldsymbol{C}\hat{x}(t)\big) \tag{2.2}$$

where $\mathscr{L} \in \mathbb{R}^{n\times q}$ is the observer gain. The corresponding observer design problem is to find $\mathscr{L}$ such that the estimation error $e(t) = \hat{x}(t) - x(t)$ asymptotically converges to zero: $\lim_{t\to\infty} e(t) = 0$. This can be achieved if the real parts of all the eigenvalues of $\boldsymbol{A} - \mathscr{L}\boldsymbol{C}$ are negative.

2.3. DISTRIBUTED-PARAMETER SYSTEMS

Distributed-parameter systems are modeled by partial differential equations. The evolution of the states of a partial differential equation is described in an infinite-dimensional space. Distributed-parameter systems are therefore called infinite-dimensional systems [219]. Take the following normalized 1D heat equation[1] as an example:

$$\frac{\partial x(z,t)}{\partial t} = \frac{\partial x(z,t)}{\partial z} + u(z,t), \qquad z \in (0,1) \tag{2.3a}$$

$$\frac{\partial x(0,t)}{\partial z} = \frac{\partial x(1,t)}{\partial z} = 0, \qquad t \geq 0 \tag{2.3b}$$

$$x(z,0) = x_0(z), \qquad z \in [0,1] \tag{2.3c}$$

where x and u are, respectively, the temperature and the input heat flow, and z is the spatial variable. The first equation describes the dynamics, while the second and third equations are the boundary condition and the initial condition, respectively.

[1] The normalized 1D heat equation describes the temperature distribution over a rod with unit length as a function of time under the effect of a heat source.

A partial differential equation like (2.3), when expanded with an output equation, can also be expressed as an abstract state space equation:

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \tag{2.4a}$$

$$y(t) = \boldsymbol{C}x(t),\ x(0) = x_0 \tag{2.4b}$$

It should be noted that $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ are operators defined on a normed linear space [56] and that they have different meanings from those in (2.1). For (2.3) the operator $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ can be defined by taking the trajectory segment $x(\cdot, t) = \{x(z, t) \mid 0 \le z \le 1\}$ as the state and by identifying the state space $\mathscr{X}$ with the normed space $L_2(0,1)$ of functions $x(\cdot, t)$ that are square-integrable on the spatial interval $[0,1]$ with $\|x(\cdot, t)\| = \left(\int_0^1 |x(z,t)|^2 \,\mathrm{d}z\right)^{\frac{1}{2}}$ [56]. Define the operators $\boldsymbol{A}$ and $\boldsymbol{B}$ on $\mathscr{X}$ to be

$$\begin{aligned}
\boldsymbol{A} &= \frac{\mathrm{d}^2}{\mathrm{d}z^2} \quad \text{with the domain} \\
\mathscr{D}(\boldsymbol{A}) &= \left\{ h \in L_2(0,1) \,\middle|\, h, \frac{\mathrm{d}h}{\mathrm{d}z} \text{ are absolutely continuous,} \right. \\
&\qquad \left. \frac{\mathrm{d}^2 h}{\mathrm{d}z^2} \in L_2(0,1), \quad \text{and} \quad \frac{\mathrm{d}h(0)}{\mathrm{d}z} = \frac{\mathrm{d}h(1)}{\mathrm{d}z} = 0 \right\} \\
\boldsymbol{B} &= I
\end{aligned}$$

where $\mathscr{D}(A)$ is the domain of $\boldsymbol{A}$, I is the identity, and the function $x_0(\cdot, 0) \in L_2(0,1)$ is the initial state. The output operator $\boldsymbol{C}$ is determined once the sensor locations have been selected.

The input trajectory $u(\cdot, t)$ and output trajectory $y(\cdot, t)$ are defined similarly to the state. The abstract model simplifies the representation of the partial differential equation model by incorporating the boundary condition into the definition of the domain of $\boldsymbol{A}$, $\mathscr{D}(\boldsymbol{A})$.

For observer and controller design, the dimension of the system must be reduced. This process is called lumping, and there are two kinds of lumping [191, 225]: early lumping and late lumping. In early lumping, the first step of the design process is to reduce the system dimensionality using spatial approximation methods, e.g., finite-difference or finite-element methods. The dimension reduction step results in a finite-dimensional system of ordinary differential equations that serve as the basis for observer design. Next, a temporal discretization can be applied to obtain discrete-time system models. In late lumping, the infinite-dimensional model is used during analysis and design. The resulting observer has an infinite number of dimensions, which are then lumped for implementation.

Several order reduction methods can be used for both early and late lumping. They are the Galerkin method [181], proper orthogonal decomposition [76], and eigenfunction expansion [210].

The Galerkin method works on the weak form of partial differential equations. The weak form involves the approximation of the equation by using a weighted function, which is called a test function, and expressing the equation into the integral form. The purpose is to reduce the smoothness requirement of the solution, which is not easily fulfilled in

many real-world applications and is used in finite element software [250]. The solution, which is in an infinite-dimensional space, is approximated as a linear combination of weighted basis functions, not necessarily orthogonal, in finite-dimensional subspaces. The approximation is called the trial function. The Galerkin method uses the same space for the test and the trial functions. They are typically in different spaces [30].

Proper orthogonal decomposition is applied to obtain reduced-order models to linear or nonlinear partial differential equations. The method is also known as principal component analysis or Karhunen-Loeve expansion [14]. The method starts by collecting a considerable number of measurements at a finite number of spatial locations (states) into a matrix called the snapshot matrix. It is assumed that each state evolution vector can be represented by a linear combination of proper orthogonal basis vectors, i.e., a set of orthonormal basis vectors. Using the singular value decomposition, the decreasing singular values of the snapshot matrix specify the importance of the vectors concerning the information present in the data. The importance of the singular values is connected to the energy in the matrix. The basis vectors containing the most energy are selected in the reduced-models [11].

Eigenfunction expansion is known as a method to solve a linear homogenous partial differential equation by separation of variables [93]. The infinite-dimensional state can be represented into an infinite series of eigenfunctions that corresponds to an infinite set of ordinary differential equations defining the time evolution of the Fourier coefficients of the system. For lumping purposes, the series is truncated at a finite number of them to make it computationally realizable [199].

From all three order reduction methods above, the Galerkin method and the proper orthogonal decomposition have wider applications as both are also applicable to nonlinear systems; see, e.g., [36, 144]. However, the proper orthogonal decomposition relies on the snapshot matrix that is built from measurements from a large number of spatial points. This may not be practical for physically large systems because a large number of measurements should be obtained before applying model reduction. For the eigenfunction expansion method, the application is limited to linear homogenous systems.

2.4. DISTRIBUTED-PARAMETER OBSERVERS

In the literature, there are several approaches to observer design for distributed-parameter systems. The first approach uses the partial differential equation model (2.3) and makes use of the available analysis methods for partial differential equations. The second approach uses the abstract model (2.4) and applies existing analysis tools of functional analysis.

For the partial differential equation model (2.3) the observers have the following form [225]:

$$\frac{\partial \hat{x}(z,t)}{\partial t} = \frac{\partial^2 \hat{x}(z,t)}{\partial z^2} + u(z,t)\ell(z)\big(\hat{x}(z,t) - x(z,t)\big),\ z \in (0,1) \tag{2.5a}$$

$$\frac{\partial \hat{x}(z,t)}{\partial z} = \ell_{\mathrm{b}}(z)\big(\hat{x}(z,t) - x(z,t)\big),\ \ t \geq 0,\ z \in \{0,1\} \tag{2.5b}$$

$$\hat{x}(z,0) = \hat{x}_0(z), \quad z \in [0,1] \tag{2.5c}$$

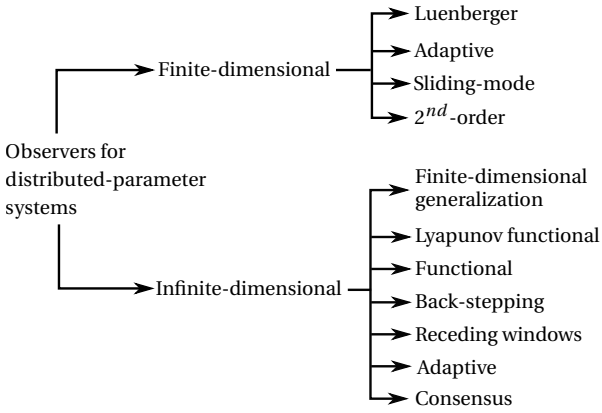


Figure 2.1: Overview of observer design approaches for linear distributed-parameter systems.

where $\ell(z)$ and $\ell_{\mathrm{b}}(z)$ are the observer gains. Inserting (2.3) and (2.5) into the error equation:

$$e(z,t) = \hat{x}(z,t) - x(z,t) \tag{2.6}$$

results in

$$\frac{\partial e(z,t)}{\partial t} = \frac{\partial^2 e(z,t)}{\partial z^2} + \ell(z)e(z,t), \qquad z \in (0,1) \tag{2.7a}$$

$$\frac{\partial \hat{e}(z,t)}{\partial z} = \ell_{\mathrm{b}}(z)e(z,t), \quad t \geq 0 \qquad z \in \{0,1\} \tag{2.7b}$$

$$e(z,0) = e_0(z) = \hat{x}_0(z) - x_0(z), \qquad z \in [0,1] \tag{2.7c}$$

The observer design problem is to determine the observer gains such that the estimation error goes to zero:

$$\lim_{t\to\infty} e(z,t) = 0, \qquad z \in [0,1]$$

For model (2.4), the observer has the following form:

$$\dot{\hat{x}}(t) = \boldsymbol{A}\hat{x}(t) + \boldsymbol{B}u(t) + \mathscr{L}\big(y(t) - \boldsymbol{C}\hat{x}(t)\big) \tag{2.8a}$$

$$\hat{x}(0) = \hat{x}_0 \tag{2.8b}$$

The observer design problem involves finding an observer gain operator $\mathscr{L}$ that makes the estimation error e, the evolution of which is described by

$$\begin{aligned} \dot{e}(t) &= (\boldsymbol{A} - \mathscr{L}\boldsymbol{C})e(t) \\ e(0) &= \hat{x}_0 - x_0\,, \end{aligned} \tag{2.9}$$

asymptotically tend to zero.

We can now proceed to an overview of observer design methods for distributed-parameter systems. A taxonomy of observers for linear distributed-parameter systems is shown in Figure 2.1.

2.4.1. FINITE-DIMENSIONAL OBSERVERS

Finite-dimensional observers are designed using the early-lumping approach. The lumping results in a system of ODEs, making the observer design methods similar to those of lumped-parameter systems. However, they often include additional elements, such as the influence of sensor locations.

LUENBERGER

Orner and Foster [181] have designed an optimal controller for a system modeled by a partial differential equation. The order of the model is reduced using the Galerkin approximation. The observer is designed to include the influence of the sensor location by optimizing the term $(\boldsymbol{A}-\mathscr{L}\boldsymbol{C})$ in (2.2).

Stavroulakis and Sarachik [210] have developed an observer for an optimal control system in which the eigenfunction expansion is used to reduce the order of the model. The observer design follows the method proposed in [242], in which it has been shown that the eigenvalues of observers used for optimal control systems cannot be assigned arbitrarily, and a systematic design approach has also been given.

An observer that estimates the states of the system and unknown input functions using output measurements has been introduced by Kobayashi and Hitotsuya [134]. In [134], it is assumed that the system in Hilbert space $\mathscr{H}$ is decomposable into an N-dimensional part:

$$\dot{x}_{\tilde{\mathrm{f}}}(t) = \boldsymbol{A}_{\tilde{\mathrm{f}}} x_{\tilde{\mathrm{f}}}(t) + \mathscr{P}\mathscr{U}(t), \quad x_{\tilde{\mathrm{f}}}(0) = x_{\tilde{\mathrm{f}},0}$$

and an infinite-dimensional part:

$$\dot{x}_{\tilde{\imath}}(t) = \boldsymbol{A}_{\tilde{\imath}} x_{\tilde{\imath}}(t) + \mathbb{Q}\mathscr{U}(t), \quad x_{\tilde{\imath}}(0) = x_{\tilde{\imath},0}$$

where $x_{\tilde{\mathrm{f}}}$ and $x_{\tilde{\imath}}$ are the state variables of the finite-dimensional part and the infinite-dimensional part, $\mathscr{P}$ and $\mathbb{Q}$ are orthogonal projections such that $\mathscr{H} = \mathscr{P}\boldsymbol{H} + \mathbb{Q}\boldsymbol{H}$ and $\mathbb{Q} = I - \mathscr{P}$, $\boldsymbol{A}_{\tilde{\mathrm{f}}}$ and $\boldsymbol{A}_{\tilde{\imath}}$ are $\boldsymbol{A}$ that restricted to the projection, i.e., to $\mathscr{P}\mathscr{H}$ and $\mathbb{Q}\mathscr{H}$ respectively, and $\mathscr{U}(t)$ is the unknown input function. An observer for the N-dimensional part has the following form:

$$\dot{\hat{x}}_{\tilde{\mathrm{f}}}(t) = \mathscr{R}\hat{x}_{\tilde{\mathrm{f}}}(t) + \mathscr{S}y(t), \qquad \hat{x}_{\tilde{\mathrm{f}}}(0) = x_{\tilde{\mathrm{f}},0}$$

for continuous linear operators $\mathscr{R}$ and $\mathscr{S}$ where $y(t) = \boldsymbol{H}x_{\tilde{\mathrm{f}}}(t) + \boldsymbol{H}x_{\tilde{\imath}}(t)$ is the measurement equation and where $\mathscr{R}$ fulfills the observer condition [161]: $\mathscr{R} = \boldsymbol{A}_{\tilde{\mathrm{f}}} - \mathscr{S}\boldsymbol{H}$. Using the solution of the dynamic error equation and assuming that the projected system is observable, the eigenvalues of $\mathscr{R}$ can be assigned arbitrarily. The estimated states are then used to approximate the input function $\mathscr{U}(t)$. In [133], Kobayashi has investigated the same problem but for a discrete-time system and unknown initial states.

ADAPTIVE

For systems with unknown parameters, adaptive observers are used to simultaneously reconstruct the states and asymptotically identify the parameters of the system. Results in this topic for lumped-parameter systems can be traced back to the beginning of the 1970s [37].

For the application of adaptive observer for distributed-parameter systems, in [152], Lilly has extended the result of [139] and shown that a reduced-order model can be used

to identify parameters and states of an infinite-dimensional system. It is shown that if the residual energy from the unmodeled dynamics is bounded over a finite interval and the input is persistently exciting, then the estimation will be bounded. This assumes no input boundedness or plant stability.

Demetriou and Ito [60] have studied adaptive observers of the following form to overcome additive perturbations from output feedback:

$$\begin{aligned} \dot{x}(t) &= (\boldsymbol{A}+\Delta)x(t)+\boldsymbol{B}u(t) \\ y(t) &= \boldsymbol{C}x(t) \end{aligned} \tag{2.10}$$

where the matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ have a finite rank, and Δ is the additive perturbation from output feedback. Several cases of Δ, assuming the same input-output locations (collocation), are presented and for each case, and an adaptation law obtained using the Lyapunov redesign method [127] is given.

SLIDING-MODE

Sliding-mode observers are a class of observers that apply the sliding-mode principle to compute the observer gain to get the estimates of the states [71]. Efe [76] has designed a sliding-mode observer for a reduced-order model obtained by proper orthogonal decomposition. The stability and convergence of the design are proved using Lyapunov stability conditions.

SECOND-ORDER

Flexible structures and vibration are examples of distributed-parameter systems that are modeled as second-order time derivative systems [164] that can be written as [63]:

$$\ddot{x}(t)+\mathcal{D}\dot{x}(t)+\mathcal{F}x(t)=\boldsymbol{B}u(t) \tag{2.11a}$$

$$y(t)=\boldsymbol{C}\begin{bmatrix} x^\top(t) & \dot{x}^\top(t)\end{bmatrix}^\top \tag{2.11b}$$

$$x(0)=x_0 \qquad \dot{x}(0)=v_0 \tag{2.11c}$$

where x and $\dot{x}$ are the position and velocity states respectively, $\mathcal{D}$ is the damping operator, $\mathcal{F}$ is the stiffness operator, and $\boldsymbol{C}$ is the output operator.

While (2.11) can be transformed into a standard state equation $\dot{x}(t)=\boldsymbol{A}x(t)+\boldsymbol{B}u(t)$ and uses a Luenberger observer, to keep the second-order structure, Demetriou [62] has designed the observer directly in the second-order form instead of designing it from the first-order representation. The method also modified the output equation by

$$y(t)=\begin{bmatrix} y_{\mathrm{p}}^\top(t) & y_{\mathrm{v}}^\top(t)\end{bmatrix}^\top=\begin{bmatrix} x^\top(t)\boldsymbol{C}_{\mathrm{p}}^\top & \dot{x}^\top(t)\boldsymbol{C}_{\mathrm{v}}^\top\end{bmatrix}^\top$$

where $\boldsymbol{C}_{\mathrm{p}}$ and $\boldsymbol{C}_{\mathrm{v}}$ are matrices with suitable dimensions. Then, the observer is designed using the following form:

$$\ddot{\hat{x}}(t)+\mathcal{D}\dot{\hat{x}}(t)+\mathcal{F}\hat{x}(t)=\boldsymbol{B}u(t)+\mathcal{L}_{\mathrm{p}}\big(y_{\mathrm{p}}(t)-\boldsymbol{C}_{\mathrm{p}}\hat{x}(t)\big)+\mathcal{L}_{\mathrm{v}}\big(y_{\mathrm{v}}^\top(t)-\boldsymbol{C}_{\mathrm{v}}\dot{\hat{x}}(t)\big) \tag{2.12}$$

where $\mathcal{L}_{\mathrm{p}}$ and $\mathcal{L}_{\mathrm{v}}$ are the observer gains. This type of observer is called natural observer [62]. In [62], the abstract representation, along with the stability and convergence analysis,

are presented. It can be seen that the observer is basically a Luenberger-type observer with a separate state gain for the velocity and position states. Another second-order observer method that allows observing systems with a positive semi-definite damping was proposed in [175].

The solution to (2.11a) can be calculated using eigenfunction expansion:

$$x(t) = \sum_{i=1}^{M} x_i(t)\phi_i$$

where ϕ_i is the eigenvector corresponding to the ith eigenvalue which corresponds to the ith mode of the system. Theoretically, $M = \infty$, but M is usually set to a finite number, say J, since, in practice, actuators and sensors are not able to deal with very high frequencies. The high-frequency modes $\sum_{i=J+1}^{\infty} x_i(t)\phi_i$ are called the residual or spillover [17].

When the frequencies of the modeled system and its residual are sufficiently separated, the measurement signals can be filtered using low-pass filters to remove high-frequency components. However, the separation principle does not apply anymore in this case. Instead of filtering the sensor output, Chait and Radcliffe [39] filtered the estimation error, and the filter is included in the observer design, which results in an augmented observer. In [41], an observer has been designed as a part of a vibration control method that is robust to the spillover.

2.4.2. INFINITE-DIMENSIONAL OBSERVERS

When the observer design uses the late-lumping approach, the resulting observer has an infinite number of dimensions and it has the form of (2.5) or (2.8).

GENERALIZATION OF THE THEORY FOR OBSERVERS OF FINITE-DIMENSIONAL SYSTEMS

One of the fundamental problems associated with the abstract model (2.4) is the generalization of observer theory for finite-dimensional systems to the infinite-dimensional case. We can see that the observer equations (2.8) are the same as those of finite-dimensional systems. However, we cannot directly apply the finite-dimensional observer equations to infinite-dimensional systems because of the different underlying spaces on which the operators are defined in the infinite-dimensional case [13].

To generalize observer theory to infinite-dimensional systems, Kitamura *et al.* [130] introduced conditions for the existence and realizability of the observer (2.8) with no input. The full-order and reduced-order cases are also considered with two types of measurements: spatially continuous and spatially discrete.

Gressang and Lamont [89] have investigated another generalization of (2.4), including a class of linear functional differential equations. In that paper, observability has been proven as a sufficient condition to design full-order and reduced-order observers. It was also shown that the eigenvalues of an observer-based state feedback system are the union of the eigenvalues of the closed-loop system and of the observer. However, the generalization does not allow the observer to have arbitrary eigenvalues.

Sakawa and Matsushita [196] have investigated feedback stabilization of parabolic partial differential equations and proved that controllability and observability are necessary and sufficient conditions to stabilize the system. This result is applied to the observation problem and it can be shown that observability guarantees the observers to converge.

An extension to the Luenberger observer in relation to sensor locations in distributed-parameter systems has been studied by El Jai and Amouroux [78]. This study has shown that sensor configurations have a relation with the detectability of the system and the existence of the observer. This means that there are certain sensor configurations that allow the observers to exist and, at the same time, there are also certain sensor configurations for which the observer does not exist.

Köhne [135] designed an observer for a heated-slab set-up by late lumping. The parameters of the observer are calculated via eigenfunction expansion of the dynamic estimation error equation.

LYAPUNOV FUNCTIONAL

Liu and Lapidus [157] have introduced a Lyapunov-based observer design for system (2.4) under the assumption that there are no inputs. The gain operator $\mathscr{L}$ can be obtained by first defining a Lyapunov functional based on a norm of the estimation error and next applying the Lyapunov stability condition to this Lyapunov functional.

FUNCTIONAL

A prime application of observers in control systems involves feedback stabilization. A functional observer design has been proposed by Fujii [84] for feedback stabilization for boundary control. The resulting observer involves convolution of the input, which results in an infinite-dimensional observer.

BACK-STEPPING

Another observer design method has been introduced by Smyshlyaev and Krstic [206] for boundary observation and control[2] of a class of parabolic partial differential equations (2.3). The observer is called a backstepping observer because a backstepping-like transformation[3] is applied to the estimation error $e(z,t)$. It has been shown that the observer gain computation in the new coordinates ensures the stability of the observer in the original coordinates. Miranda *et al.* [170] have added a sliding-mode term to the observer to obtain a sliding-mode observer.

The backstepping observer in [206] is also applied to feedback stabilization using boundary control, where the situations of collocated and anti-collocated[4] actuator and sensor are considered. A similar approach is taken in [45] for the sliding-mode control problem in a non-collocated actuator and sensor case.

RECEDING WINDOWS

Yaz *et al.* [238] have proposed a type of observer called receding-window observers for discrete-time infinite-dimensional systems. The observer is in the form (2.8) with the following observer gain calculation

$$\mathscr{L}\big(\boldsymbol{P}(N-1)\big) = \boldsymbol{A}\boldsymbol{P}(N-1)\boldsymbol{C}^{\top}(\boldsymbol{C}\boldsymbol{P}(N-1)\boldsymbol{C}^{\top}+\boldsymbol{R})^{-1} \tag{2.13}$$

[2] Boundary control and/or observation systems are a class of distributed-parameter systems in which the actuation and/or observation is performed at the spatial boundary of the system.

[3] The backstepping-like transformation converts a system into another stable system using an invertible kernel-based transformation.

[4] For a 1D heated bar, "anti-collocated" that means the heater is at one end and the sensor is at the other end.

where N is the receding window length of the observer and where $\boldsymbol{P}_N$ is computed by iterating the following Riccati and gain equations:

$$\begin{aligned}\mathscr{L}\big(\boldsymbol{P}(k)\big) &= \boldsymbol{A}\boldsymbol{P}(k)\boldsymbol{C}^\top\big(\boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{C}^\top+\boldsymbol{R}\big)^{-1}\\ \boldsymbol{P}(k+1) &= \Big(\boldsymbol{A}-\mathscr{L}\big(\boldsymbol{P}(k)\big)\boldsymbol{C}\Big)\boldsymbol{P}(k)\big(\boldsymbol{A}-\mathscr{L}\big(\boldsymbol{P}(k)\big)\boldsymbol{C}\big)^\top+\mathscr{L}\big(\boldsymbol{P}(k)\big)\boldsymbol{R}\mathscr{L}\big(\boldsymbol{P}(k)\big)^\top+\boldsymbol{Q}\end{aligned} \tag{2.14}$$

for discrete-time instant $k=0,\ldots,N-1$, with $\boldsymbol{P}_0$, $\boldsymbol{R}$, and $\boldsymbol{Q}$ positive-definite operators. In [238], the convergence of the observer and its bounded noise rejection and robustness to parameter perturbation properties are proved. We can see from (2.13) and (2.14) that the observer is similar to the Kalman filter (with $\boldsymbol{R}$ and $\boldsymbol{Q}$ the covariance matrices of respectively the measurement noise and the process noise).

ADAPTIVE

Curtain *et al.* [57] have extended the results of [60] for the non-collocated case. Reference [58] addresses the case of time-varying input parameters. The adaptation laws are obtained by using the Lyapunov redesign method [127].

Demetriou *et al.* [61] have proposed an adaptive fault detection observer to monitor and accommodate actuator faults. When the actuator fault occurs, the observer generates a non-zero residual signal. After the fault is detected, the residual signal is also used to automatically reconfigure the system.

CONSENSUS

A recent development in state estimation for distributed-parameter systems is distributed estimation using sensor networks as the measurement system. Sensor networks consist of several sensor nodes, where each node has embedded computation, communication, and power modules. A node acts as a local observer that computes estimates using its own model and measurements. The communication module allows the sensor nodes to share information with other nodes in the network within a specified communication topology.

Demetriou [64] has proposed a distributed Luenberger observer using a sensor network with the consensus method [179]. Each node I in the network has a model similar to (2.4) but with a modified measurement equation to reflect the sensor node index:

$$y_i(t)=\boldsymbol{C}_i x_i(t) \tag{2.15}$$

for $I\in\{1,\ldots,N\}$ where N is the number of nodes in the network. For each node I, the following local observer is applied:

$$\dot{\hat{x}}_i(t)=\boldsymbol{A}_i\hat{x}(t)+\boldsymbol{B}u(t)+\boldsymbol{G}_i y_i(t) \tag{2.16}$$

where $\boldsymbol{G}_i$ denotes the observer gain and it is such that $\boldsymbol{A}_i=\boldsymbol{A}-\boldsymbol{G}_i C_i$ generates a stable semigroup. The distributed observer with consensus for node i can be written as

$$\dot{\hat{x}}_i(t)=\boldsymbol{A}_i\hat{x}(t)+\boldsymbol{B}u(t)+\boldsymbol{G}_i y_i(t)+\alpha_i\Phi_i\sum_{j=1,\,i\neq j}^{N}(\hat{x}_j(t)-\hat{x}_i(t)) \tag{2.17}$$

where Φ_i is the consensus operator gain and α_i is an additional weighting term. In [64], an adaptive consensus observer has also been introduced in which the consensus term $\alpha_i\Phi_i$ is made adaptive.

Consensus filters are also proposed in [63] for second-order distributed-parameter systems based on Luenberger-type observers, resulting in a filter called natural Luenberger consensus filter.

2.5. SUMMARY

We have presented a review of state observer design methods for first-order and second-order linear distributed-parameter systems. We have addressed linear observers for infinite-dimensional and finite-dimensional systems. In addition, adaptive methods and distributed state estimation methods for sensor networks have been presented.

The observer design problem in distributed-parameter systems is more complex than for lumped-parameter systems due to its infinite dimensionality. At the same time, this field is less explored compared to that of lumped-parameter systems.

3

NON-CENTRALIZED KALMAN FILTER COMPARISON FOR DISTRIBUTED-PARAMETER SYSTEMS

3.1. INTRODUCTION

Current advances in sensor technology enable the design of small-scale, low-cost sensing devices (sensor nodes) endowed with embedded computing and communication capabilities. A number of these sensor nodes can be connected to each other following a particular topology called a sensor network. There are numerous applications of sensor networks, for instance, in the military, health care, or agriculture [3, 156]. An example of the use of sensor networks in the control field is networked control [105]. The use of wireless sensors for feedback control has also been reported [18].

An attractive feature of sensor networks is their ability to perform sensing and state estimation in environments with spatially distributed parameters. In this case, sensor nodes are placed at specified locations surrounding an environment to collect measurements that serve as inputs to the filter. To this end, the noncentralized variants of the Kalman filter can be used. Kalman filters are a class of estimators that minimize the variance of the estimation error.

One approach of centralized estimation for using the measurements is by collecting them into a specialized subsystem called a central processor, in which the measurements are used to produce global estimates. In this case, the sensor nodes do not have an active role in computing the estimates and act purely as a measurement device. Consequently, the system model is known only by the central processor. This approach is called the centralized Kalman filter.

As opposed to the centralized approach, the estimate computations can be performed

by the nodes, where local estimates are computed. The local estimates can subsequently be transmitted to a central processor that computes global estimates. Alternatively, local estimates can be communicated between two or more nodes using an algorithm to get the estimates of the global state. In this case, the sensor nodes have an active role in estimating the states by decentralization of estimate computations. This results in a decentralized approach to the Kalman filter. In this chapter, the decentralized approach encompasses methods that do not use a central processor to fuse the estimates and the corresponding estimate error covariance matrix from the connected sensors. The decentralization may employ a distributed algorithm[1] so that the method is called the distributed Kalman filter. For instance, the consensus Kalman filter [177] applies the consensus algorithm in computer science [53].

In the literature, several decentralized and distributed Kalman filter methods have been proposed. The most representative methods are the parallel information filter [208], the distributed information filter [190], the distributed Kalman filter with consensus filter [177], and the distributed Kalman filter with weighted averaging [7]. This chapter compares the mentioned methods to estimate the states of a linear distributed parameter system that has been discretized spatially and temporally, resulting in a linear discrete-time large-scale system. Note that the above list of methods is incomplete and does not include some recent methods like distributed Kalman filter with diffusion strategies [38]. However, that method closely resembles the distributed Kalman filter with consensus filter [177] and may thus have a similar performance.

A comparison of the non-centralized Kalman filters for wireless sensor networks has been made in [201], which focused on communication-related performance. In that paper, the comparison involved the performance of decentralized Kalman filters with and without data loss. The paper [201] also discussed the required communication for each method and concluded that the performance of the distributed Kalman filter with weighted averaging is the highest with the lowest communication requirements, but that on the other hand, that method also suffers the most from data loss. In the current chapter, our interest is the performance of the filters related to large-scale linear systems derived from the discretization of distributed parameter systems. We evaluate the performance of a system with a high number of states, not only in terms of the evolution of the states over time and space but also in terms of errors in the steady-state estimation.

3.2. KALMAN FILTER AND ITS DECENTRALIZED VARIANTS

The Kalman filter is an optimal stochastic discrete-time state estimator developed by Kalman [124]. Since then, the theory and applications of Kalman filters have been treated in different journal papers and books. There are several ways to derive the Kalman filter that can be found in standard textbooks on optimal estimation, see, e.g., [8, 32, 202]. This section briefly describes the Kalman filter and its decentralized variants based on [7, 177, 190, 202, 208].

Consider the following process modeled as a discrete-time linear system with a mea-

[1]The distributed algorithm communicates estimates among nodes with specified rules, e.g., based on the topology of the network. Estimates from the neighbors are then fused with the local estimates.

surement equation:

$$x(k) = \boldsymbol{F}x(k-1) + \boldsymbol{G}u(k-1) + w(k-1) \tag{3.1a}$$
$$z(k) = \boldsymbol{H}x(k) + v(k) \tag{3.1b}$$

where $x(k) \in \mathbb{R}^n$ is the state vector, $\boldsymbol{F} \in \mathbb{R}^{n\times n}$ the state matrix, $\boldsymbol{G} \in \mathbb{R}^{n\times m}$ the input matrix, $u(k) \in \mathbb{R}^m$ the input vector, $z(k) \in \mathbb{R}^p$ the measurement vector, $\boldsymbol{H} \in \mathbb{R}^{p\times n}$ the measurement matrix, $w(k)$ the process noise vector, and $v(k)$ the measurement noise vector. The process and measurement noise are assumed to be zero-mean Gaussian white noise signals that have the following properties:

$$\mathbb{E}\{w(k)\} = 0 \qquad \mathbb{E}\left\{w(k)w^\top(k)\right\} = \boldsymbol{Q}$$
$$\mathbb{E}\{v(k)\} = 0 \qquad \mathbb{E}\left\{v(k)v^\top(k)\right\} = \boldsymbol{R}$$

where $\mathbb{E}\{\cdot\}$ is the expectation operator, the superscript $\top$ denotes matrix transpose operation, and $\boldsymbol{Q}$ and $\boldsymbol{R}$ are, respectively, the covariance matrix of the process and measurement noise.

3.2.1. CENTRALIZED KALMAN FILTER

The Kalman filter equations consist of two parts of equations: time update and measurement update. The following time update equations compute the estimates at time step k based on the process model and the previous estimates to get *a priori* estimates:

$$\begin{aligned} \hat{x}^-(k) &= \boldsymbol{F}\hat{x}^+(k-1) + \boldsymbol{G}u(k-1) \\ \boldsymbol{P}^-(k) &= \boldsymbol{F}\boldsymbol{P}^+(k-1)\boldsymbol{F}^\top + \boldsymbol{Q} \end{aligned} \tag{3.2}$$

where $\boldsymbol{P}(k)$ is the estimation error covariance matrix, and the superscripts "$-$" and "$+$" respectively indicate the *a priori* and *a posteriori* estimates and the error covariance matrix. This step is also referred to as the prediction step. Once the measurements $z(k)$ at time step k are available, the measurement update corrects *a priori* estimates to get *a posteriori* estimates:

$$\begin{aligned} \boldsymbol{K}(k) &= \boldsymbol{P}^-(k)\boldsymbol{H}^\top[\boldsymbol{H}\boldsymbol{P}^-(k)\boldsymbol{H}^\top + \boldsymbol{R}]^{-1} \\ \hat{x}^+(k) &= \hat{x}^-(k) + \boldsymbol{K}(k)[z(k) - \boldsymbol{H}\hat{x}^-(k)] \\ \boldsymbol{P}^+(k) &= \left[(\boldsymbol{P}^-(k))^{-1} + \boldsymbol{H}^\top\boldsymbol{R}^{-1}\boldsymbol{H}\right]^{-1} \end{aligned} \tag{3.3}$$

where $\boldsymbol{K}(k)$ is the Kalman gain matrix. The initial conditions are $\hat{x}^+(0) = x_0$ and $\boldsymbol{P}^+(0) = \boldsymbol{P}_0$, where x_0 and $\boldsymbol{P}_0$ are, respectively, the initial guesses of the estimate and estimation error covariance matrix.

The Kalman filter in (3.2) and (3.3) is called centralized Kalman filter because the measurement vectors are treated in one measurement matrix. The estimates from the centralized Kalman filter are called global estimates.

Besides the form in (3.2)–(3.3), another form of the Kalman filter uses the inverse of the estimation error covariance matrix, called information matrix, denoted by $\mathcal{J}$ and defined as $\mathcal{J} = \boldsymbol{P}^{-1}$. The filter that uses this information matrix is called the information

filter. The *a priori* estimate equations of the information filter are:

$$\begin{aligned} \hat{x}^-(k) &= \boldsymbol{F}\hat{x}^+(k-1) + \boldsymbol{G}u(k-1) \\ \mathcal{J}^-(k) &= \left(\boldsymbol{F}\left(\mathcal{J}^+(k-1)\right)^{-1}\boldsymbol{F}^\top + \boldsymbol{Q}\right)^{-1} \end{aligned} \tag{3.4}$$

The measurement update computations are the following:

$$\begin{aligned} \Delta s(k) &= \boldsymbol{H}^\top \boldsymbol{R}^{-1} z(k) \\ \hat{x}^+(k) &= \hat{x}^-(k) + \Delta s(k) \\ \mathcal{J}^+(k) &= \mathcal{J}^-(k) + \Delta\mathcal{J} \end{aligned} \tag{3.5}$$

where Δs is the information vector and the information matrix update $\Delta\mathcal{J}$ is defined as:

$$\Delta\mathcal{J} = \boldsymbol{H}^\top \boldsymbol{R}^{-1}\boldsymbol{H} \tag{3.6}$$

The information filter avoids the need for matrix inverse computation, which is preferably avoided from a numerical point of view.

In applying the decentralized Kalman filter for distributed parameter systems with sensor networks, each sensor node i can compute its own estimate $\hat{x}_i(k)$ and the corresponding estimation error covariance matrix $\boldsymbol{P}_i(k)$. The estimate and/or the error covariance matrix are communicated to other nodes based on the network topology. In our case, consider a sensor network consisting of N sensor nodes. The nodes are connected to each other, following a specified network topology. In the network, nodes i and j are neighbors if there is a direct link between them. Let $\mathcal{N}_i$ be the set of neighbors of node i, including node i itself. We assume that each node has an identical process model (3.1a) and the corresponding process noise $\boldsymbol{Q}$, but a different measurement matrix. Because each node measures one or more state components of the system, and no state component is measured by two or more nodes, we can assume that each local measurement matrix $\boldsymbol{H}_i$ is one block row of the global measurement matrix $\boldsymbol{H}$. In other words, the global measurement matrix $\boldsymbol{H}$ is the stacked version of all local measurement matrices $\boldsymbol{H}_i$:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_i^\top & \cdots & \boldsymbol{H}_N^\top \end{bmatrix}^\top \quad .$$

Thus, the local measurement in node i is expressed as

$$z_i(k) = \boldsymbol{H}_i x_i(k) + v_i(k) \quad .$$

It is assumed that the measurement noise between nodes i and j is uncorrelated, or $R_{ij} = \mathbb{E}\left\{v_i(k)v_j^\top(k)\right\} = 0$ for $i \neq j$.

In this chapter, the measurement updates and the resulting estimates in each node are called local updates and local estimates, respectively.

3.2.2. PARALLEL INFORMATION FILTER (PIF)

The parallel information filter computes local an *a posteriori* estimate $\hat{x}_i^+(k) \in \mathbb{R}^n$ and the corresponding estimation error covariance matrix $\boldsymbol{P}_i^+$ in parallel in each node.

Then $\hat{x}_i^+(k)$ and $\boldsymbol{P}_i^+(k)$ are sent to a central processor in which the estimates are combined to get the global estimate $\hat{x}(k)$ [208]. The time and measurement update equations for node i are

- Local prediction:

$$\begin{aligned}\hat{x}_i^-(k) &= \boldsymbol{F}\hat{x}_i^+(k-1) + \boldsymbol{G}u(k-1)\\ \boldsymbol{P}_i^-(k) &= \boldsymbol{F}\boldsymbol{P}_i^+(k-1)\boldsymbol{F}^\top + \boldsymbol{Q}\end{aligned} \tag{3.7}$$

- Local measurement update:

$$\begin{aligned}\boldsymbol{K}_i(k) &= \boldsymbol{P}_i^-(k)\boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1}\\ \hat{x}_i^+(k) &= \hat{x}_i^-(k) + \boldsymbol{K}_i(k)[z_i(k) - \boldsymbol{H}_i\hat{x}_i^-(k)]\\ \left(\boldsymbol{P}_i^+(k)\right)^{-1} &= \left(\boldsymbol{P}_i^-(k)\right)^{-1} + \boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1}\boldsymbol{H}_i\end{aligned} \tag{3.8}$$

In the central processor, the estimates from all nodes are combined into one estimate. It is desired that the estimate is as certain as possible, or in other words, an estimate with a low uncertainty is preferable. In the case of estimates and uncertainties from N measurements, where the measurement of node i is independent of that of node j for $i \neq j$, estimates with lower uncertainty should be given larger weights. With such consideration, the weight for each measurement can be calculated as [208]

$$\omega_i(k) = \frac{\left(\mathrm{tr}\left\{\boldsymbol{P}_i^+(k)\right\}\right)^{-1}}{\sum_{i=1}^{N}\left(\mathrm{tr}\left\{\boldsymbol{P}_i^+(k)\right\}\right)^{-1}}$$

Once the weights have been determined, the global *a posteriori* estimate and its estimate error covariance matrix can be expressed as follows

$$\boldsymbol{P}^{-1}(k) = \sum_{i=1}^{N}\omega_i(k)\left(\boldsymbol{P}_i^+(k)\right)^{-1} \tag{3.9a}$$

$$\hat{x}(k) = \sum_{i=1}^{N}\omega_i(k)\boldsymbol{P}(k)\left(\boldsymbol{P}_i^+(k)\right)^{-1}\hat{x}_i^+(k) \tag{3.9b}$$

This method relies on the central processor to get the global estimates. Hence, it is necessary that all sensor nodes are neighbors of the central processor to ensure that all local measurements can be combined into global ones.

3.2.3. DISTRIBUTED INFORMATION FILTER (DIF)

The distributed information filter was proposed by Rao and Durrant-Whyte [190] to eliminate the need for a central processor in the decentralized Kalman filter. Using a central processor creates a hierarchy in the network. Furthermore, the network is highly dependent on the central processor. Eliminating the central processor ensures that all nodes are at the same level, and moreover, it removes the dependency on a single component.

The key idea of this method is expressed in the relation between information vectors and matrix updates, respectively, for the global estimates of the centralized method and local estimates in each node i. This is formulated as the sum of individual updates from

neighboring nodes:

$$\begin{aligned} \Delta s(k) &= \boldsymbol{H}^\top \boldsymbol{R}^{-1} z(k) = \sum_{i=1}^{N} \boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1} z_i(k) \\ \Delta \mathcal{I} &= \boldsymbol{H}^\top \boldsymbol{R}^{-1} \boldsymbol{H} = \sum_{i=1}^{N} \boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1} \boldsymbol{H}_i \end{aligned} \tag{3.10}$$

Local updates are computed in each node and sent to the neighboring nodes. Node i adds all information updates from its neighbors to its own updates and then computes the updated estimates and the estimation error covariance matrix. The estimates obtained after the communication of the nodes are called communication update estimates.

The prediction and measurement update equations for node i are

- Local prediction:

$$\begin{aligned} \hat{x}_i^-(k) &= \boldsymbol{F}\hat{x}_i^+(k-1) + \boldsymbol{G}u(k-1) \\ \boldsymbol{P}_i^-(k) &= \boldsymbol{F}\boldsymbol{P}_i^+(k-1)\boldsymbol{F}^\top + \boldsymbol{Q} \end{aligned}$$

- Information vector update:

$$\Delta s_i(k) = \boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1} z_i(k)$$

- Communication update:

$$\begin{aligned} \hat{x}_i^+(k) &= \boldsymbol{P}_i^+(k)\left[(\boldsymbol{P}_i^-(k))^{-1}\hat{x}_i^-(k) + \sum_{j\in\mathcal{N}_i} \Delta s_j(k)\right] \\ (\boldsymbol{P}_i^+(k))^{-1} &= (\boldsymbol{P}_i^-(k))^{-1} + \sum_{j\in\mathcal{N}_i} \Delta\mathcal{I}_i \end{aligned}$$

  for

$$\Delta\mathcal{I}_i = \boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1} \boldsymbol{H}_i \tag{3.11}$$

  the information matrix update for node i.

This method decentralizes the computation of global estimates to every node without the need for a central processor. If all nodes are fully connected, then the neighbors of the node i are all nodes across the network. This shows that the performance of the DIF method is equal to that of the centralized Kalman filter for the fully connected node, as shown in (3.10). Note that we include inputs in (3.2.3), which is not considered in [190].

3.2.4. DISTRIBUTED KALMAN FILTER WITH CONSENSUS FILTER (DKFCF)

The distributed Kalman filter with consensus filter has been proposed by Olfati-Saber [177]. The main feature of this method is the use of a consensus algorithm to obtain the communication update estimates. The consensus algorithm at node i is performed iteratively as follows: node i receives estimates from its neighbors for each consensus step. Node i subtracts its estimate from the estimate of each of its neighbors, weights the result with a factor γ, and adds the obtained value to its estimate.

Another feature of this method is the availability of stability analysis. Olfati-Saber *et al.* in [179] presented a stability analysis of the consensus algorithm using algebraic graph theory.

The prediction and measurement update equations for node i are:

- Local prediction:

$$\hat{x}_i^-(k) = \boldsymbol{F}\hat{x}_i(k-1) + \boldsymbol{G}u(k-1)$$
$$\boldsymbol{P}_i^-(k) = \boldsymbol{F}\boldsymbol{P}_i^+(k-1)\boldsymbol{F}^\top + \boldsymbol{Q}$$

- Information vector update:

$$\Delta s_i(k) = \boldsymbol{H}_i^\top \boldsymbol{R}_i^{-1} z_i(k)$$

- Measurement update:

$$(\boldsymbol{P}_i^+(k))^{-1} = (\boldsymbol{P}_i^-(k))^{-1} + \sum_{j \in \mathcal{N}_i} \Delta \mathcal{I}_j$$
$$\hat{x}_{\mathrm{c},i}^+(k) = \boldsymbol{P}_i^+(k)\left[(\boldsymbol{P}_i^-(k))^{-1}\hat{x}_i^-(k) + \sum_{j \in \mathcal{N}_i} \Delta s_j(k)\right]$$

  where $\hat{x}_{i,\mathrm{c}}^+(k)$ is the *a posteriori* estimate for node i. This estimate will be later sent to its neighbors at the consensus step.

- Consensus step:

$$\hat{x}_i^+(k) = \hat{x}_{\mathrm{c},i}^+(k) + \gamma \sum_{j \in \mathcal{N}_i} \left(\hat{x}_{\mathrm{c},j}^+(k) - \hat{x}_{\mathrm{c},i}^+(k)\right) \tag{3.12}$$

Up to the consensus step, this method is identical to the distributed information filter. In other words, DKFCF extends DIF by iterating local estimate exchange to reach the consensus of estimate; meaning that all nodes have the same value of estimates.

3.2.5. DISTRIBUTED KALMAN FILTER WITH WEIGHTED AVERAGING (DKFWA)

The distributed Kalman filter with weighted averaging has been proposed in [6, 7]. A feature of this method is the reduction of computation and communication load compared to DKFCF. The reduction is because the nodes only compute and send the estimates without the error estimation covariance matrix.

Unlike the previous methods, this method consists of two parts: an online and an offline part. The online part computes and communicates estimates. In the offline part, Kalman gains and weights are computed for each node. In this method, the Kalman gain and the weight $\boldsymbol{W}$ are computed once and used during the entire operation.

This method works as follows: node i receives estimates from its neighbors and weights them with a weight matrix W. Then, the weighted estimates are added to the estimate of node i.

The online steps of the distributed Kalman filter with weighted averaging at node i are the following:

- Prediction:

$$\hat{x}_i^{\mathrm{l}-}(k) = \boldsymbol{F}\hat{x}_i^{\mathrm{l}+}(k-1) + \boldsymbol{G}u(k-1)$$

  where $\hat{x}_i^{\mathrm{l}}(k)$ denotes the local estimates at node i

- Measurement update:

$$\hat{x}_i^{\mathrm{l}+}(k) = \hat{x}_i^{\mathrm{l}-}(k) + \boldsymbol{K}_i[z(k) - \boldsymbol{H}_i\hat{x}_i^{\mathrm{l}-}(k)] \tag{3.13}$$

- Information exchange:

$$\hat{x}_i^+(k) = \sum_{j \in \mathcal{N}_i} \boldsymbol{W}_{ij} \hat{x}_j^{l+}(k) \tag{3.14}$$

where W_{ij} is the weight of the estimate of node j that is used to compute the global estimates in node i. The value of W_{ij} is zero if node i is not connected to node j.

The off-line computations are performed to minimize the trace of the estimation error covariance matrix $\boldsymbol{P}^+(k)$, which is defined as:

$$\boldsymbol{P}^+(k) = \mathbb{E}\left\{(x(k) - \hat{x}^+(k))(x(k) - \hat{x}^+(k))^\top\right\}$$

for

$$x(k) = \begin{bmatrix} x_1^\top(k) & \cdots & x_N^\top(k) \end{bmatrix}^\top$$
$$\hat{x}^+(k) = \begin{bmatrix} \left(\hat{x}_1^+(k)\right)^\top & \cdots & \left(\hat{x}_N^+(k)\right)^\top \end{bmatrix}^\top$$

Using (3.14) and

$$\sum_{j \in N_i} W_{ij} = I \tag{3.15}$$

To get unbiased estimates, we obtain the following relation

$$\begin{aligned} \boldsymbol{P}^+(k) &= \boldsymbol{W}(x(k) - \hat{x}^{l+}(k))(x(k) - \hat{x}^{l+}(k))^\top \boldsymbol{W}^\top \\ &= \boldsymbol{W}\boldsymbol{P}^{l+}(k)\boldsymbol{W}^\top \end{aligned} \tag{3.16}$$

The covariance $\boldsymbol{P}^{l+}(k)$ in the last equation can be written as

$$\boldsymbol{P}^{l+}(k) = \begin{pmatrix} \boldsymbol{I} & \tilde{\boldsymbol{K}} \end{pmatrix} \Phi(k) \begin{pmatrix} I & \tilde{\boldsymbol{K}} \end{pmatrix}^\top \tag{3.17}$$

with

$$\Phi(k) = \begin{pmatrix} \boldsymbol{I} \\ -\tilde{\boldsymbol{H}} \end{pmatrix} \boldsymbol{P}^-(k) \begin{pmatrix} \boldsymbol{I} \\ -\tilde{\boldsymbol{H}} \end{pmatrix}^T + \begin{pmatrix} 0 & 0 \\ 0 & \boldsymbol{R} \end{pmatrix}$$

and

$$\begin{aligned} \tilde{\boldsymbol{K}} &= \text{blockdiag}(K_1, \ldots, K_N) \\ \tilde{\boldsymbol{H}} &= \text{blockdiag}(H_1, \ldots, H_N) \end{aligned} \quad .$$

It should be noted that for this method, it is possible to have a non-diagonal measurement covariance matrix $\boldsymbol{R}$ in contrast to DIF and DKFCF that require R to be a diagonal matrix.

To get an optimal filter, finding the values of $\boldsymbol{W}$ and $\tilde{\boldsymbol{K}}$ that minimize (3.17) is required. Instead of direct minimization of (3.17), the Kalman gain $\boldsymbol{K}$ and weight $\boldsymbol{W}$ are computed off-line by solving the following optimization problem:

$$\min_{\tilde{\boldsymbol{K}}, \boldsymbol{W}} \text{tr}\left\{\boldsymbol{W}\begin{pmatrix} \boldsymbol{I} & \tilde{\boldsymbol{K}} \end{pmatrix} \Phi \begin{pmatrix} \boldsymbol{I} & \tilde{\boldsymbol{K}} \end{pmatrix}^\top \boldsymbol{W}^\top\right\} \tag{3.18}$$
$$\text{s.t. } W_{ij} = 0 \text{ if node } i \text{ and } j \text{ are not connected and (3.15)}$$

The obtained gain $\tilde{\boldsymbol{K}}$ and weight $\boldsymbol{W}$ are employed in the on-line computation of the states. Details on how to solve the optimization problem (3.18) are given in [7].

Table 3.1: Comparison of the characteristics of the different Kalman filters

	CKF	PIF	DIF	DKFCF	DKFWA
Central processing	yes	yes	no	no	no
Connectivity	full	full	full	partial	partial
Communication	single	single	single	multi	single
Global estimates	yes	yes	no	no	no

In short, this method is a consensus filter but with only one information exchange. The motivation for the communication limitation is that communication in sensor networks draws more power than computation.

A comparison of the characteristics of the Kalman filter methods presented in this section is shown in Table 3.1. Concerning communication requirements, the DKFCF is the only one that needs iterations of estimate exchange to achieve estimate consensus among the nodes. This is denoted by the term 'multi' in the table. The other methods perform a single communication of estimates for each estimation step.

3.3. BENCHMARK: 1D CONDUCTION PROCESS MODEL

The Kalman filters presented in the preceding section are now simulated to estimate the states of a heat conduction process of a rod without input, i,e., the cooling process from an initial temperature. This process is an example of a distributed parameter system that is modeled as a first-order time-derivative and second-order spatial derivative PDE with specified boundary conditions. Therefore, the conduction process has the elements needed to compare different decentralized Kalman filter methods for distributed parameter systems.

Consider a rod with length L and cross-section radius r. The material's density, heat capacity, and thermal conductivity are denoted by ρ, C_p, and κ, respectively. Using energy balance equations [121], we can get the following partial differential equation

$$\frac{\partial T(z,t)}{\partial t} = \frac{\kappa}{\rho C_\mathrm{p}} \left[\frac{\partial^2 T(z,t)}{\partial z_1^2} \right], \forall z \in \mathscr{Z} \setminus \mathscr{Z}_\mathrm{b}, \forall t \tag{3.19a}$$

$$T(z,t) = T_\mathrm{b}(t), \quad \forall z \in \mathscr{Z}_\mathrm{b}, \forall t \tag{3.19b}$$

$$T(z,0) = T_0, \quad \forall z \in \mathscr{Z} \tag{3.19c}$$

where T is the temperature of the rod, T_e is the temperature of the environment, h is the heat transfer coefficient of the surface of the rod, x is the spatial coordinate of length, $P_\mathrm{e} = 2\pi r$ is the perimeter of the rod, and $A_\mathrm{T} = \pi r^2$ the area of the longitudinal section. Equations (3.19b) and (3.19c) are the boundary condition and initial condition, respectively. The rod's parameters are listed in Table 3.2.

The simulation of the Kalman filter requires discretization of (3.19a) in space and time. For the spatial discretization, the central approximation of the second-order derivative is employed:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta_x)^2} \tag{3.20}$$

where Δ_x is the spatial discretization interval. The spatial discretization results in a grid as shown in Figure 3.1. In the figure, the grid index increases from left to right. The distance between consecutive grid points is Δ_x. Applying (3.20) to (3.19a) and simplifying the results with respect to the grid point index, (3.19a) becomes an ordinary partial differential equation as follows:

$$\frac{dT_i(t)}{dt} = C_{\mathrm{x}} T_{i-1}(t) - (2C_{\mathrm{x}} + C_{\mathrm{Peh}}) T_i(t) + C_{\mathrm{x}} T_{i+1}(t) + C_{\mathrm{Peh}} T_{\mathrm{e}} \tag{3.21}$$

with i the node index that corresponds to the grid point index and

$$C_{\mathrm{x}} = \frac{\kappa}{\rho C_{\mathrm{p}} \Delta_x^2}$$
$$C_{\mathrm{Peh}} = \frac{h P_{\mathrm{e}}}{\rho C_{\mathrm{p}} A_{\mathrm{T}}}$$

For a grid with n grid points on the rod, we have n ODEs from (3.21), each of which corresponds to a grid point. These ODEs can be expressed as a state space equation

$$\dot{x} = \boldsymbol{A}x + \boldsymbol{B}u$$

with

$$x = \begin{bmatrix} T_1 & \cdots & T_n \end{bmatrix}^\top$$
$$u = \begin{bmatrix} T_{\mathrm{e}} & T_{\mathrm{b}} \end{bmatrix}^\top$$

The environment temperature T_{e} is 25 °C = 298.15 K. The state equation is discretized temporally by using the forward Euler approximation to get a discrete-time linear equation.

A number of N sensor nodes are located on the specified grid points and numbered sequentially from left to right. Each node j measures the specific temperature T_i so that the dimension of the measurement matrix H is $N \times n$. Each row of H corresponds to a measurement of node i, i.e., the i-th entry is 1 and the other entries are zero.

We will perform two simulations to compare the decentralized Kalman filter methods. The first simulation observes the estimation error performance of the decentralized methods. The goal of the second simulation is to observe the consistency of the decentralized methods. These two simulations are sufficient to assess the performance of the decentralized Kalman filters.

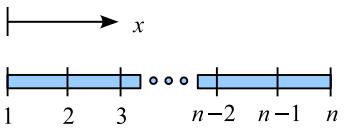


Figure 3.1: Numbering of the grid points on the rod

Table 3.2: Rod parameters

Parameters	Values	Units
ρ	8700	$\mathrm{kg\,m^{-3}}$
κ	400	$\mathrm{W\,m^{-1}\,K^{-1}}$
C_p	385	$\mathrm{J\,kg^{-1}\,K^{-1}}$
h	10	$\mathrm{W\,m^{-1}\,K^{-1}}$
T_e	298.15	K

Table 3.3: Simulation parameters

Parameters	Values	Units
L	4	m
n	201	–
T_0	50	K
x_0	80	–
T_b	298.15	K
σ_v^2	0.04	–
N	3	–

3.3.1. SIMULATION 1: ESTIMATION ERROR PERFORMANCE

In this simulation, the estimation error performance of the decentralized Kalman filters is compared with that of the centralized one. The simulation parameters are shown in Table 3.3. The simulation final time is 5000 s, and the sensor node locations are at grid points 2, 104, and 200. In addition, the consensus parameter for the DKFCF is selected small $\gamma = 0.05$ [178]. All sensor nodes are fully connected.

The simulation results are shown in Figure 3.2, where "pt." in the legend stands for "point". The first plot shows the estimates from the CKF in which the true states are plotted in thick grey lines. The other plots, (b)–(e), display the difference between the CKF and the decentralized methods for $\Delta_\mathrm{est} = \hat{x}_\mathrm{ckf} - \hat{x}_\mathrm{d}$. The subscript d in $\hat{x}_\mathrm{d}$ stands for "decentralized". The plotted estimates of the CKF show that the estimates' convergence rate depends on the sensor's spatial distance. For instance, the estimates of grid point 2 converge faster than those of grid points 30 and 75. However, the estimates of grid point 2 are noisy due to the measurement noise.

When comparing the decentralized methods to the CKF, Figure 3.2 shows that the performance of the DIF and DKFCF are identical. Furthermore, both methods have the slightest difference to the CKF. If the sensor nodes are fully connected, as in this simulation, the DKFCF is identical to the DIF but with an additional consensus step. The consensus step in the DKFCF costs more communication but does not deliver better estimates due to the full connectivity of the nodes.

The central processor in the PIF essentially performs weighted summations on the local estimates based on the estimates' uncertainties to obtain the global estimates. The larger the uncertainty of an estimate, the lower the weight assigned to compute the global estimate. This summing process is not, however, equivalent to the centralized treatment of the measurements in the CKF, where the central processor computes the

global estimates based on the measurements obtained from the nodes.

The lower performance of the DKFWA compared to the DIF or DKFCF is expected. The adaptive Kalman gain adapts better to the measurement noise than the fixed gain in the DKFWA. Merging with weighted estimates from the neighbors is not enough to improve the estimate accuracy.

3.3.2. SIMULATION 2: STEADY-STATE ESTIMATION PERFORMANCE

In this simulation, we investigate the steady-state estimates of the filters to assess the consistency of the filters. The simulations are performed by simulating from the initial temperature at 0 0 s until final time 15 000 s. The means and variances of the steady-state estimates are computed and plotted in Figure 3.3. The sensor node locations are at grid points 62, 104, and 146. The other simulation parameters are shown also in Table 3.3.

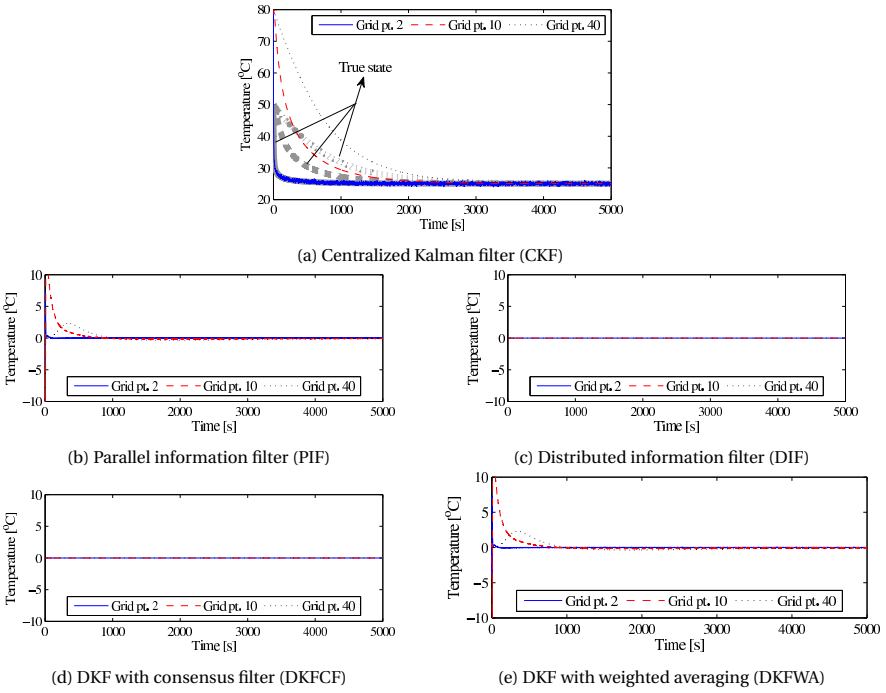
Figure 3.3 shows the mean of the estimates in blue circles, the variance of the estimates in blue bars, and the variance of the averaged measurement in grey. The figure shows that the estimates of states between two sensors are better than those on the outer side of the sensors, i.e., before the left-most and after the right-most sensors. The variance of the estimates for states located on the outer side increases as the distance to the sensor increases. The results show that the spatial distance between the grid point for which the temperature is estimated and the sensor location influences the accuracy of the steady-state estimates.

For the decentralized methods, the DIF and DKFCF give the closest estimates to those of the CKF, while the DKFWA gives the farthest estimates. As already mentioned in Simulation 1, the full connectivity of the nodes results in the equivalence of the CKF and DIF. The result of the PIF is better than that of the DKFWA. The increase of the estimates' variance of the PIF and the DKFWA is also higher compared to that of the CKF and the other two distributed methods. For the PIF, this can be explained as follows: the PIF sums up the estimates and, consequently, their uncertainty. As the uncertainty of the estimates increases, the total uncertainty also increases.

3.4. SUMMARY

This chapter has compared the centralized Kalman filter and some of its variants on a spatially and temporally discretized linear distributed parameter system. In general, the performance of the decentralized Kalman filters cannot be better than the centralized method. The performance of the decentralized methods can be equal to the centralized method provided there is full connectivity of the sensor nodes. This applies to the DIF and DKFCF. This situation has been considered in the simulation for the distributed information filter and distributed Kalman filter with consensus filter.

The PIF and the DKFWA suffer from the limitations of their approach. The parallel information filter requires a guarantee on the connection to the central processor. Otherwise, no global estimate is yielded, and every node has its own estimates that might be different from one to the other. Another problem can occur in case one of the nodes feeds low-quality estimates. These estimates will reduce the quality of the global estimates instead of improving them. The DKFWA was developed to reduce the large communication requirement of the DKFCF by communicating the estimates only once. In addition,



(a) Centralized Kalman filter (CKF)

(b) Parallel information filter (PIF)

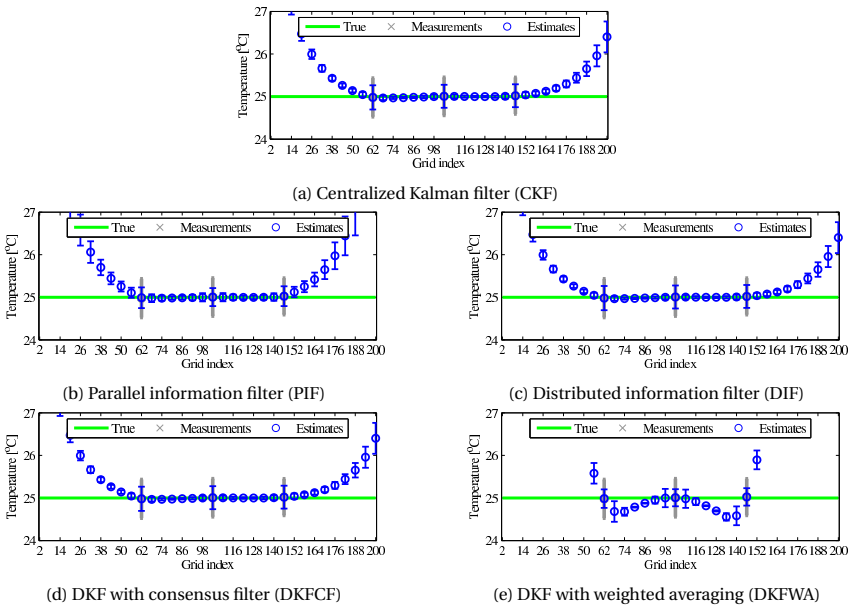
(c) Distributed information filter (DIF)

(d) DKF with consensus filter (DKFCF)

(e) DKF with weighted averaging (DKFWA)

Figure 3.2: Estimation of the decentralized Kalman filters compared to the centralized one

the Kalman gain is computed offline and used during the estimation. This reduces the computation load in the nodes. However, the reduction in computation load comes at the cost of a lower estimate accuracy compared to the other methods. This is mainly due to the static gain of the Kalman filter.



(a) Centralized Kalman filter (CKF)

(b) Parallel information filter (PIF)

(c) Distributed information filter (DIF)

(d) DKF with consensus filter (DKFCF)

(e) DKF with weighted averaging (DKFWA)

Figure 3.3: Steady-state estimation of the centralized and decentralized Kalman filters

4

IDENTIFICATION OF DISTRIBUTED-PARAMETER SYSTEMS FROM SPARSE MEASUREMENTS

4.1. INTRODUCTION

Many real-life processes are distributed-parameter systems. The states of this class of systems depend on one or more dimensional spatial variables. Examples include chip manufacturing plants [70]; process control systems such as packed-bed reactors [47], reverse flow reactors [75], and wastewater treatment plants [226]; flexible structures in atomic force microscopes [141]; ultraviolet disinfection installations in the food industry [228]; electrochemical processes [131]; or drying installations [74].

Distributed-parameter systems are typically modeled using partial differential equations. However, developing such models from first principles is a tedious and time-consuming process [150]. If input-output measurements are available, a model can be constructed by using system identification methods. However, due to the large number of spatially interdependent state variables, the identification of distributed-parameter systems is considerably more complex than the identification of lumped-parameter systems, and it is known to be an ill-posed inverse problem [138] because the solution is not unique [154]. There are three main approaches to the identification of distributed-parameter systems [142]: (i) direct identification, (ii) reduction to a lumped-parameter system, and (iii) reduction to an algebraic equation. The direct identification approach uses the infinite-dimensional system model to find the parameters of the systems. This case includes the identification of a certain parameter of interest related to an application, e.g., heat conduction [42, 43, 230]. The reduction-based approaches involve spatial discretization to create a set of ordinary differential equations in time to which identification methods

for lumped-parameter systems can be applied. This approach, also called time-space separation [150], is the subject of this chapter.

There are two recent books related to this chapter. The first book by Cressie and Wikle [54] extensively treats statistical modeling and analysis of spatial, temporal, and spatiotemporal data. The second book by Billings [25] addresses spatiotemporal discretized partial differential equations by using polynomial basis functions and model reduction by using orthogonal forward regression.

In this chapter, a method for the identification of finite-dimensional models for distributed-parameter systems with a *small number of fixed sensors* is proposed. Compared to other finite-difference identification methods in the literature [19, 25, 49, 81, 163, 182, 184, 246, 249], this method does not assume a dense set of measurement locations in space, and, in addition, the method also uses an input selection method to reduce the complexity of the model. The method also allows the use of external inputs in the model, a problem not addressed by Cressie and Wikle [54]. In addition, an application that, to our knowledge, has not yet been described in the literature is presented, namely the identification of a model for the temperature dynamics in a greenhouse.

4.2. PROBLEM FORMULATION

Consider a distributed-parameter system described by a partial differential equation with the associated boundary and initial conditions. For ease of notation and without loss of generality, a system that is first-order in time and second-order in a two-dimensional space is considered:

$$\frac{\partial g(z,t)}{\partial t} = f\Big(z, t, g(z,t), \frac{\partial g(z,t)}{\partial z_1}, \frac{\partial g(z,t)}{\partial z_2}, \frac{\partial g(z,t)}{\partial z_1 z_2}, \frac{\partial^2 g(z,t)}{\partial z_1^2}, \frac{\partial^2 g(z,t)}{\partial z_2^2}, u(z,t), w(z,t)\Big), \forall z \in \mathscr{Z} \setminus \mathscr{Z}_\mathrm{b}, \forall t \tag{4.1a}$$

$$0 = h\Big(z, t, g(z,t), \frac{\partial g(z,t)}{\partial z_1}, \frac{\partial g(z,t)}{\partial z_2}, u(z,t), w(z,t)\Big), \forall z \in \mathscr{Z}_\mathrm{b}, \forall t \tag{4.1b}$$

$$g(z,t_0) = g_0(z), \forall z \in \mathscr{Z} \tag{4.1c}$$

Here $g(\cdot,\cdot)$ is the variable of interest, $f(\cdot)$ is the system function, $h(\cdot)$ is the boundary value function, $z = (z_1, z_2) \in \mathscr{Z} \subset \mathbb{R}^2$ is the spatial coordinate, $t \in \mathbb{R}^+ \cup \{0\}$ is the continuous-time variable, $\mathbb{R}^+$ is the set of positive real numbers, $u(\cdot,\cdot)$ is the input function, $w(\cdot,\cdot)$ is the process noise, and $\mathscr{Z}_\mathrm{b}$ is the set of spatial boundaries of the system. Higher-order and multi-variable systems can be defined analogously.

Assume that a set of input-output measurements are available from the distributed-parameter system (4.1) with unknown functions $f(\cdot)$ and $h(\cdot)$. The sensors are located at specified points to measure $g(\cdot,\cdot)$, and there are also actuators that generate inputs $u(\cdot,\cdot)$ to the system. Since the actuators and the sensors are placed at known and fixed locations, the space is discretized with a set of grid points $\mathscr{M}_\mathrm{g}$ such that the actuator locations $\mathscr{M}_\mathrm{u}$ and the sensor locations $\mathscr{M}_\mathrm{s}$ are in $\mathscr{M}_\mathrm{g}$, i.e., $\mathscr{M}_\mathrm{u} \subset \mathscr{M}_\mathrm{g}$ and $\mathscr{M}_\mathrm{s} \subset \mathscr{M}_\mathrm{g}$. Assume that the measurements, concatenated in a vector $y(\cdot)$, are affected by additive Gaussian

noise $v(z_i,t) \sim \mathcal{N}(0,\sigma^2_{\mathrm{v}_i})$. The input and measurement vectors are defined as follows:

$$u(t) = \begin{bmatrix} u(z_{\mathrm{u},1},t) & \dots & u(z_{\mathrm{u},N_\mathrm{u}},t) \end{bmatrix}^\top \tag{4.2a}$$

$$y(t) = \begin{bmatrix} g(z_{\mathrm{g},1},t) + v(z_{\mathrm{g},1},t) & \dots & g(z_{\mathrm{g},N_\mathrm{s}},t) + v(z_{\mathrm{g},N_\mathrm{s}},t) \end{bmatrix}^\top \tag{4.2b}$$

where N_u is the number of actuators, N_s the number of sensors, the coordinates of the inputs are denoted by $z_{\mathrm{u},j} \in \mathcal{M}_\mathrm{u}$, the measurement coordinates by $z_{\mathrm{g},i} \in \mathcal{M}_\mathrm{g}$, and the superscript $\top$ denotes the transpose of a matrix or vector. Note that not every grid point is associated with a sensor or actuator.

The measurements are collected at discrete time steps $t_k = k \cdot T_\mathrm{s}$ with $k \in \mathbb{N} \cup \{0\}$, where T_s is the sampling period. To simplify the notation, the discrete-time instant t_k is subsequently written as k. The notation is further simplified by using an integer subscript assigned to the given sensor or actuator location:

$$u_j(k) = u(z_{\mathrm{u},j},t)\big|_{t=k\cdot T_\mathrm{s}}, \quad j = 1,\dots,N_\mathrm{u} \tag{4.3}$$

for the inputs and

$$y_i(k) = \Big(g(z_{\mathrm{g},i},t) + v(z_{\mathrm{g},i},t)\Big)\Big|_{t=k\cdot T_\mathrm{s}}, \quad i = 1,\dots,N_\mathrm{s} \tag{4.4}$$

for the outputs. The input and output data (4.3) and (4.4) are the only available information to construct a distributed finite-order model of (4.1).

4.3. IDENTIFICATION METHOD FOR DISTRIBUTED-PARAMETER SYSTEMS

The main idea of the method proposed in this chapter is to identify at each sensor location a lumped-parameter system described by a dynamic model. To take into account the spatial dynamics of the system, measurements from the neighboring locations are included as inputs. This is justified by the derivation of coupled discrete-time dynamic models obtained from spatial discretization of a partial-differential equation presented at the beginning of this section. Parameter estimation of the coupled models by solving a least-squares problem is then shown, subsequently followed by model reduction to simplify the models. As measurements and actuation are performed only at some spatial locations, sensors and actuators locations that related problems are briefly discussed. The summary of the method is given at the end of the section to give a big picture of the identification method.

4.3.1. CONSTRUCTION OF COUPLED DISCRETE-TIME DYNAMIC MODELS

The discretization of a partial differential equation in space by using the finite-difference method results in a set of coupled ordinary differential equations. At time instant t, the coupling spatially relates the value of the variable of interest at node I, $g_i(t)$, to values of the same variable at the neighboring nodes. The influence of more distant neighbors may be delayed due to the finite speed of spatial propagation of the quantity of interest. As an example, consider the following simplification of (4.1a) to an autonomous

one-dimensional case:

$$\frac{\partial g(z,t)}{\partial t} = m\left(\frac{\partial^2 g(z,t)}{\partial z^2}\right) \tag{4.5}$$

where $g(z,t) \in \mathbb{R}$ is the variable of interest, $z \in \mathbb{R}$ is the spatial coordinate, and $m(\cdot)$ is a nonlinear function. The system is spatially discretized using the finite-difference method by creating grid points, which, for the sake of simplicity, are uniformly spaced at distance Δ_z. Denote $g_i(t)$ for $g(z,t)$ at grid point $z = i \cdot \Delta_z$, called node I for short. The central approximation [147] of the second-order derivative in space is:

$$\left.\frac{\partial^2 g(z,t)}{\partial z^{2^2}}\right|_{z=i} \approx \frac{g_{i+1}(t) - 2g_i(t) + g_{i-1}(t)}{(\Delta_z)^2} \tag{4.6}$$

which results in:

$$\frac{\mathrm{d}g_i(t)}{\mathrm{d}t} = m\left(\frac{g_{i+1}(t) - 2g_i(t) + g_{i-1}(t)}{(\Delta_z)^2}\right) \tag{4.7}$$

Then, using the forward-difference approximation of the time derivative:

$$\left.\frac{\mathrm{d}g_i(t)}{\mathrm{d}t}\right|_{t=k} \approx \frac{g_i(k+1) - g_i(k)}{T_\mathrm{s}}$$

to discretize the left-hand side of (4.7), yields:

$$g_i(k+1) = g_i(k) + T_\mathrm{s} \cdot m\left(\frac{g_{i+1}(k) - 2g_i(k) + g_{i-1}(k)}{(\Delta_z)^2}\right) \tag{4.8}$$

or in a slightly more general form:

$$g_i(k+1) = q\Big(g_i(k), g_{i-1}(k), g_{i+1}(k), T_\mathrm{s}, \Delta_z\Big) \tag{4.9}$$

Note that in this example, $g_i(k)$ is influenced only by its immediate neighbors. For systems with a higher spatial order and with exogenous inputs (4.9) can be written as:

$$g_i(k+1) = q(g_{\mathcal{N}_{\mathrm{s},i}}(k), u_{\mathcal{N}_{\mathrm{u},i}}(k), T_\mathrm{s}, \Delta_z) \tag{4.10}$$

where $g_{\mathcal{N}_{\mathrm{s},i}}(k) = \{g_j(k) | j \in \mathcal{N}_{\mathrm{s},i}\}$ is the set of neighboring variables of interest, including $g_i(k)$ itself and $u_{\mathcal{N}_{\mathrm{u},i}}(k) = \{u_l(k) | l \in \mathcal{N}_{\mathrm{u},i}\}$ is the set of neighboring inputs including $u_i(k)$ itself.

In the system identification setting, Δ_z and T_s are known and fixed, and instead of $g_i(k)$, the measurement $y_i(k)$ is used (which includes the effect of measurement noise $v_i(k)$). Thus the following model is obtained:

$$y_i(k+1) = F\big(y_{\mathcal{N}_{\mathrm{s},i}}(k), u_{\mathcal{N}_{\mathrm{u},i}}(k), v_{\mathcal{N}_{\mathrm{s},i}}(k)\big) \tag{4.11}$$

where $y_{\mathcal{N}_{\mathrm{s},i}}(k)$ is the set of neighboring measurements at node i, including $y_i(k)$. The neighbors of node i can be simply the nodes that are within a specified distance ϱ, i.e., $y_{\mathcal{N}_{\mathrm{s},i}}(k) = \big\{y(z,k) | \|z - z_i\| \leq \varrho, z \in \mathcal{M}_\mathrm{u} \cup \mathcal{M}_\mathrm{s}\big\}$ for measurements and $u_{\mathcal{N}_{\mathrm{s},i}}(k) = \big\{u(z,k) | \|z - z_i\| \leq \varrho, z \in \mathcal{M}_\mathrm{u} \cup \mathcal{M}_\mathrm{s}\big\}$ for inputs, see Figure 4.1. A priori knowledge can be used to obtain a suitable value of ϱ.

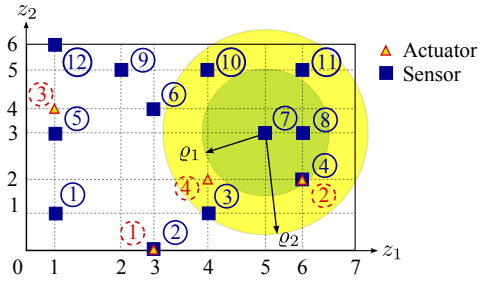


Figure 4.1: An illustration of the neighboring measurements and the inputs set with two possible neighborhoods of sensor 7 using a Euclidean distance criterion. The first set of neighbors is defined using distance ϱ_1 from sensor 7, namely $y_{\mathcal{N}_{s,7}} = \{4,7,8\}$ and $u_{\mathcal{N}_{s,7}} = \{2\}$. The second set using distance ϱ_2, $y_{\mathcal{N}_{s,7}} = \{3,4,7,8,10,11\}$ and $u_{\mathcal{N}_{s,7}} = \{2,4\}$.

An inappropriate choice of ϱ may, however, yield a large number of neighbors that are included in the model. In order to reduce the model complexity, an input or regressor selection method is applied. This topic is discussed later on in Section 4.3.3.

When $F(\cdot)$ in (4.11) is not known, an approximation can be designed using the available input-output data and linear or nonlinear system identification. Assuming that the system can be approximated by a linear model, linear system identification methods can be applied to (4.11), as described in the following section.

4.3.2. SYSTEM IDENTIFICATION AND PARAMETER ESTIMATION

Identification methods for linear systems (including linear-in-parameters nonlinear systems) use the following model representation:

$$\hat{y}_i(k+1) = \phi_i^\top(k)\theta_i \tag{4.12}$$

where $\hat{y}_i(k)$ is the predicted value of $y_i(k)$, $\phi_i(k)$ is the regressor vector at time step k, and θ_i is the vector of parameters. Note that the subscript index I corresponds to sensor I as in the previous section. The regressor vector contains lagged input-output measurements, including those of neighboring sensors and inputs. The parameter vector $\hat{\theta}_i$ can be estimated by using least-squares methods [27], so that the following prediction error is minimized:

$$\hat{\theta}_i = \arg\min_{\theta_i} \sum_{k=1}^{N} \left\| y_i(k+1) - \phi_i^\top(k)\theta_i \right\|_2^2 \tag{4.13}$$

$$= \arg\min_{\theta_i} \sum_{k=1}^{N} \|\epsilon_i(k)\|_2^2 \tag{4.14}$$

with $\epsilon_i(k) = y_i(k) - \hat{y}_i(k)$ the prediction error.

The use of neighboring measurements as inputs to the model may lead to a situation where the regressors are corrupted by noise. This requires an error-in-variables identification approach, solved, e.g., by using total least squares [207]. For a thorough discussion of the total least-squares method, the interested reader is referred to [224]. When noiseless

input variables to the actuators are among the regressors, a mixed ordinary-total least-squares method must be used [224]. In this chapter, however, the ordinary least squares is used to allow model simplification with methods that extend ordinary least squares, e.g., Lasso. In other words, it is assumed that the prediction error is Gaussian.

In nonlinear system identification, the problem is more difficult as there is no unique way to represent the nonlinear relation between the regressors and the output, and different methods are available to represent the nonlinearity. For instance, Wiener systems [248] and Hammerstein systems [68] use nonlinear functions cascaded with a linear system, Takagi-Sugeno fuzzy models combine local linear models by weighting them via membership functions [214], while neural networks use global nonlinear basis functions [174].

4

4.3.3. MODEL REDUCTION BY USING REGRESSOR SELECTION

Including neighboring measurements as inputs will result in a highly complex model and increase the size of the regressor vector $\phi_i(k)$. This size is determined by the number of neighboring inputs and the number of components of each neighboring regressor vector. A highly complex model may have low generalization performance, i.e., it may not correctly predict previously unseen data. So, the reduction-based approach is used to limit the size of the regressor vector and, therefore, only the most relevant components are kept in the model.

The large number of regressors might also cause another problem in the case of a limited number of input-output samples, i.e., the regressor matrix has more columns than rows, causing a non-unique least-squares solution. This shows that the least-squares model obtained is ill-posed. Furthermore, simple models are preferred for control applications that use sensors embedded with limited-performance computing devices, i.e., smart sensors. Using simple models reduces the prediction computation load inside the sensors and increases the ease of implementation of the method in real applications. For these reasons, among other reasons, it is desired to have a simpler model by removing inputs that do not contribute to the output to reduce the computational load, especially when the models are used in online control design.

Three methods are commonly used to reduce the number of regressors in standard linear regression [72]: stepwise regression, backward elimination, and exhaustive search. With these methods, the inclusion or exclusion of a regressor is decided based on statistical tests, such as the F-test.

One of the more recent methods is Lasso [217], which stands for the *least absolute shrinkage and selection operator*. Lasso is a least squares optimization approach with L_1 regularization through a penalty function with an ℓ_1-norm. In Lasso, the following regression model is assumed:

$$\hat{y} = \theta_0 + \phi^\top \theta \tag{4.15}$$

with $\theta = \begin{bmatrix}\theta_1 & \dots & \theta_{n_r}\end{bmatrix}^\top$ and θ_0 the parameters of the model and ϕ the vector of regressors. Lasso computes the parameters so that the parameters of the regressors that have the least importance are made zero by using a regularization parameter. More specifically,

Lasso solves the following optimization problem [217]:

$$\begin{bmatrix}\hat{\theta}_0 & \hat{\theta}^\top\end{bmatrix}^\top = \arg\min_{\theta_0,\theta} \sum_{i=1}^{N} \left(y_i - \theta_0 - \phi_i^\top \theta\right)^2, \quad \text{s.t.} \sum_{j=1}^{n_\mathrm{r}} |\theta_j| \leq \tau \tag{4.16}$$

where τ is a tuning parameter, and for the sake of simplicity, the scalar case of y is considered (extension to the vector case is straightforward). This problem can also be written as:

$$\begin{bmatrix}\hat{\theta}_0 & \hat{\theta}^\top\end{bmatrix}^\top = \arg\min_{\theta_0,\theta} \left(\frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \theta_0 - \phi_i^\top \theta\right)^2 + \lambda \sum_{j=1}^{n_\mathrm{r}} \left|\theta_j\right| \right) \tag{4.17}$$

where λ is a nonnegative regularization parameter. Note that formulation (4.16) and (4.17) are equivalent in the sense that for any $\tau \geq 0$ in (4.16), there exists a $\lambda \in [0,\infty)$ in (4.17) such that both formulations have the same solution, and vice versa.

As for nonlinear systems there is no unique representation, regressor selection becomes more complex. The simplest method, albeit computationally inefficient, is to directly search the most optimal set of regressors using exhaustive search. Regarding model-specific methods, forward regression has been used for polynomial models by using orthogonalization and selecting regressors that significantly reduce the estimation error [44, 165]; neural network models can be simplified by pruning, link weights that are sufficiently small [100]; for adaptive network fuzzy inference systems [80] by analyzing the correlation between input and output variables. For an example of the model-independent regressor selection method, one may refer to, e.g., [229], which uses fuzzy clustering to asses the significance of regressors to the measured output.

4.3.4. SENSOR AND ACTUATOR LOCATIONS AND INTERPOLATION

Measurements and actuations in distributed-parameter systems are commonly performed at spatially sampled locations. This practice raises two related problems in the control and estimation of partial differential equations:

1. The number and the locations of sensors and/or actuators. A short introduction to this topic is presented in a survey by Kubrusly and Malebranche [143]. Given the underlying partial differential equations model, the locations of the sensors will influence the identifiability of the distributed-parameter system [220].
   In this chapter, the partial differential equations model is assumed unknown and the sensor and actuator locations are assumed to be fixed and given. This resembles the case study of the greenhouse temperature model discussed in Section 4.4.2. The sensor and actuator location problem is considered a topic of further research.
   Similarly to the identification of lumped-parameter systems, it is necessary to generate sufficiently persistent excitation in data acquisition experiments for distributed-parameter systems. The notion of persistence of excitation for distributed-parameter systems is more complicated than that for the lumped-parameter systems; see, e.g., [180]. However, this topic is still an open problem, and as such, it is out of the scope of the chapter.
2. How to interpolate outputs at locations that are not measured? This problem naturally arises because the sensors provide information only at their locations [54].

For the interpolation problem, kriging and splines are commonly used methods [54]. However, ordinary kriging is used and briefly presented in this chapter following [54]. Kriging was initially developed to solve estimation-related problems in geology and it is able to interpolate in time and space. Because temporal interpolation is not required in our setting, only spatial kriging is considered in this section.

Given a spatial random process, also called a random field:

$$Y(z) = L(z) + V(z), \quad z \in \mathscr{Z} \tag{4.18}$$

where $Y(\cdot)$, $L(\cdot)$, and $V(\cdot)$ are, respectively the measured random field, the true but unknown random field, and the random measurement noise, z is the spatial coordinate, and $\mathscr{Z}$ is the spatial domain. As the spatial domain $\mathscr{Z}$ has been discretized using the finite-difference method, the measurements of the random field realizations can be written as $y_i = g_i + v_i$, where the subscript i is defined similarly to that of (4.7), from which the measurement vector y_z is defined as the stacked measurements from N_{y_z} sensor locations.

Remarks:

1. Depending on the purpose, a spatiotemporal random process

   $$Y(z,t) = L(z,t) + V(z,t), \quad \forall z \in \mathscr{Z}, \forall t \tag{4.19}$$

   for a certain fixed time t can be viewed as a random field $Y(z)$ or as a dynamic random process $Y(t)$ [73, 236]:

   $$Y(z) = L(z) + V(z), \quad \forall z \in \mathscr{Z} \tag{4.20a}$$
   $$Y(t) = L(t) + V(t), \quad \forall t \tag{4.20b}$$

   Stochastic analysis related to spatial variables can be performed to (4.20a), for instance, spatial-dependent correlation analysis. This is commonly used in geoscience, e.g., [200].
2. In the case of the proposed method, (4.2b) is the discrete-time realization of (4.20b) at sensor location $z_i \in \mathscr{M}_s$. This kind of random process is found in signals and systems analysis.

Kriging [54] is a linear estimation method to obtain the optimal spatial estimate of the second-order stationary process $G(z)$ at a coordinate location that is not measured $z_o \notin \mathscr{M}_s$, such that the mean square estimation error (MSE):

$$\text{MSE} = \mathbb{E}\left\{\left(g_{z_o} - \hat{L}(y_z)\right)^2\right\} \tag{4.21}$$

is minimized, where g_{z_o} is the true but unknown value of the process $L(z)$, $\hat{L}(y_z)$ is the estimator, and $\mathbb{E}\{\cdot\}$ is the expectation operator.

In ordinary kriging, the mean of $G(z)$ is assumed constant, i.e., $\mathbb{E}\{L(z)\} = \mu_G, z \in \mathscr{Z}$, and the covariance function $\text{Cov}(g_i, g_j)$ and the zero mean measurement error variance σ_V^2 are assumed to be known. The estimator has the following form:

$$\hat{L}_O(y_z) = \gamma^\top y_z \tag{4.22}$$

with the column vector $\gamma \in \mathbb{R}^{N_{yz}}$ the estimator parameter.

The problem of kriging is to find γ to minimize (4.21). To impose unbiasedness, $\gamma^\top \mathbf{1} = 1$ has to be fulfilled, where $\mathbf{1}$ is a column vector with 1 as the elements. By using the Lagrange multiplier ζ, the parameter vector γ is computed by solving the following optimization problem:

$$\arg\min_{\gamma} \left(\mathbb{E}\left\{ \left(g_{z_o} - \gamma^\top y_z \right)^2 \right\} - 2\zeta \cdot (\gamma^\top \mathbf{1} - 1) \right) \tag{4.23}$$

The solution of the above optimization problem is:

$$\gamma^* = \boldsymbol{C}_{y_z}^{-1} \left(\mathrm{Cov}\left(g_{z_o} y_z \right) + \zeta^* \mathbf{1} \right) \tag{4.24}$$

$$\zeta^* = \frac{1 - \mathbf{1}^\top \boldsymbol{C}_{y_z}^{-1} \mathrm{Cov}\left(g_{z_o} y_z \right)}{\mathbf{1}^\top \boldsymbol{C}_{y_z}^{-1} \mathbf{1}} \tag{4.25}$$

where γ^* and ζ^* are respectively the optimal parameter vector and Lagrange multiplier, and $\boldsymbol{C}_{y_z}$ is the covariance matrix of measurement vector y_z defined as:

$$\boldsymbol{C}_{y_z} = \begin{cases} \mathrm{Var}(y_i) + \sigma_V^2 & i = j \\ \mathrm{Cov}(y_i, y_j) & i \neq j \end{cases} \tag{4.26}$$

where $\mathrm{Var}(\cdot)$ is the variance. Substituting ζ^* in (4.24) and γ^* into (4.22) gives:

$$\hat{G}_O(y_z) = \left(\mathrm{Cov}(g_{z_o} y_z) + \mathbf{1} \frac{1 - \mathbf{1}^\top \boldsymbol{C}_{y_z}^{-1} \mathrm{Cov}(g_{z_o} y_z)}{\mathbf{1}^\top \boldsymbol{C}_{y_z}^{-1} \mathbf{1}} \right)^\top \boldsymbol{C}_{y_z}^{-1} y_z \tag{4.27}$$

with the corresponding mean square error:

$$\mathrm{MSE} = \mathrm{Var}(g_{z_o}) - \mathrm{Cov}(g_{z_o} y_z)^\top \boldsymbol{C}_{y_z}^{-1} \mathrm{Cov}(g_{z_o} y_z) + \frac{1 - \mathbf{1}^\top \boldsymbol{C}_{y_z}^{-1} \mathrm{Cov}(g_{z_o} y_z)}{\mathbf{1}^\top \boldsymbol{C}_{y_z}^{-1} \mathbf{1}} \tag{4.28}$$

Equation (4.27) can be rewritten as:

$$\hat{G}_O(y_z) = \hat{\mu}_G + \mathrm{Var}(g_{z_o})^\top \boldsymbol{C}_{y_z}^{-1} \cdot (y_z - \hat{\mu}_G \mathbf{1}) \tag{4.29}$$

with $\hat{\mu}_G$ the generalized least-squares estimator of μ_G [86]:

$$\hat{\mu}_G = \frac{\mathbf{1}^\top \boldsymbol{C}_{y_z}^{-1} y_z}{\mathbf{1}^\top \boldsymbol{C}_{y_z}^{-1} \mathbf{1}} \tag{4.30}$$

Another variant of kriging is universal kriging, which assumes $\mu_G(z)$ to be a linear model instead of a constant as in ordinary kriging. An interesting application of this kriging variant is the Kalman filter method for distributed motion coordination strategy of mobile robot positioning at critical locations [50].

4.3.5. MODEL VALIDATION

Once a model has been built, it needs to be validated [159]. In validation, the model performance is assessed by evaluating its performance to predict data that are not used in the identification, i.e., to predict validation data. To be used for control and estimation purposes, it is desired that the obtained model has a sufficiently good prediction performance based on a specified measure.

A model is typically presented as a one-step prediction model as shown in (4.12). In this case, the model performance is analyzed by evaluating the errors between the data and its one-step ahead predictions. Larger prediction steps might be required in some applications, e.g., in model predictive control. In that case, the n-th step prediction is obtained from

$$\hat{y}_i(k+n) = \hat{\phi}_i(k+n-1)\theta_i^T \tag{4.31}$$

where $\hat{y}_i(k+n)$ is the predicted value of $y_i(k)$ at discrete time step $k+n$, $\hat{\phi}_i(k+n-1)$ is the regressor vector containing lagged measurements $y_i(k+n-1)$ and/or their predictions $\hat{y}_i(k+n-1)$, and θ_i is the vector of parameters.

In general, a model with accurate predictions for a long horizon indicates that the behavior of the model is closer to the behavior of the real system. Setting the prediction horizon to infinity represents a very strict test of the model. This is also called the free-run simulation. In free-run simulation, the prediction is computed by using inputs and previous predictions and involving no output measurement.

4.3.6. IDENTIFICATION PROCEDURE

The identification procedure is presented in this section. Given the set of input-output measurements from an unknown distributed-parameter system, the proposed identification procedure proceeds as follows:

1. Create a spatial grid for the system so that each sensor and each actuator is associated with a grid point. The grid may have a uniform or a nonuniform spacing, depending on the actuator and sensor locations. Recall that not all grid points are occupied by sensors or actuators. The sensors and actuators are numbered consecutively: $i = 1,\ldots,N_s$ for the sensors and $j = 1,\ldots,N_u$ for the actuators. An illustration of a 2D system, with spatial grid points and labels for the sensors and actuators, is shown in Figure 4.2.

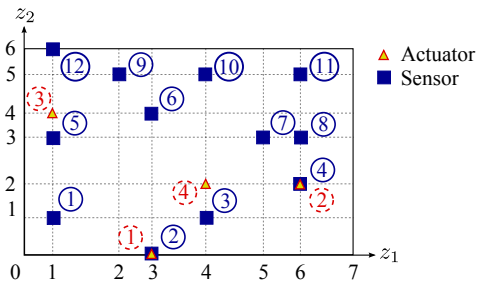


Figure 4.2: An illustration of a 2D system with a nonuniform spatial grid. Sensors and actuators are indicated by solid and dashed circles, respectively.

2. For each sensor i in the grid:
   (a) Determine the dynamic model structure using one of the available structures for lumped-parameter systems, such as auto-regressive with exogenous input (ARX), output error (OE), Box-Jenkins (BJ), etc.
   (b) Define the set of neighboring sensors and actuators, i.e., those that are located in a defined neighborhood. The neighboring measurements and inputs from neighboring actuators become inputs to the dynamic model of sensor i. Determine the (temporal) system order and construct the regressors.
   (c) When the number of regressors is large, optimize the model structure in order to simplify the model.
   (d) Estimate the parameters of the dynamic model for sensor i.
   (e) Validate the dynamic model. If the model is rejected, return to step 2a to use a different system structure or to 2b to change the set of neighbors.

To close this section, we summarize the proposed identification method in Figure 4.3. The steps are:

- Construct coupled discrete-time dynamic models, as elaborated in Section 4.3.1.
- Identify and estimate the parameters of the models, as presented in Section 4.3.2.
- Simplify the identified models to obtain simpler models, shown in Section 4.3.3.
- Sensor placement and interpolation for locations where measurements are not available, as described in Section 4.3.4.
- Validate models to assess their performance for control or estimation purpose, as described in Section 4.3.5.

Remarks:

- The proposed framework performs off-line identification for distributed-parameter systems; however, the method can be extended directly to recursive identification for the ARX structure.
- For structures that require the predicted output to compute the parameters, extension to recursive parameter estimation is possible, provided that the measurements are updated synchronously.
- The convergence for the recursive implementation of the framework can be analyzed by using the methods presented in [159].

4.4. SIMULATIONS AND APPLICATIONS

Two examples based on synthetic and real data, respectively, are considered to illustrate the effectiveness of the proposed identification approach. The synthetic data are generated from a linear two-dimensional heat conduction equation. The real-life data are temperature measurements from a small-scale real greenhouse.

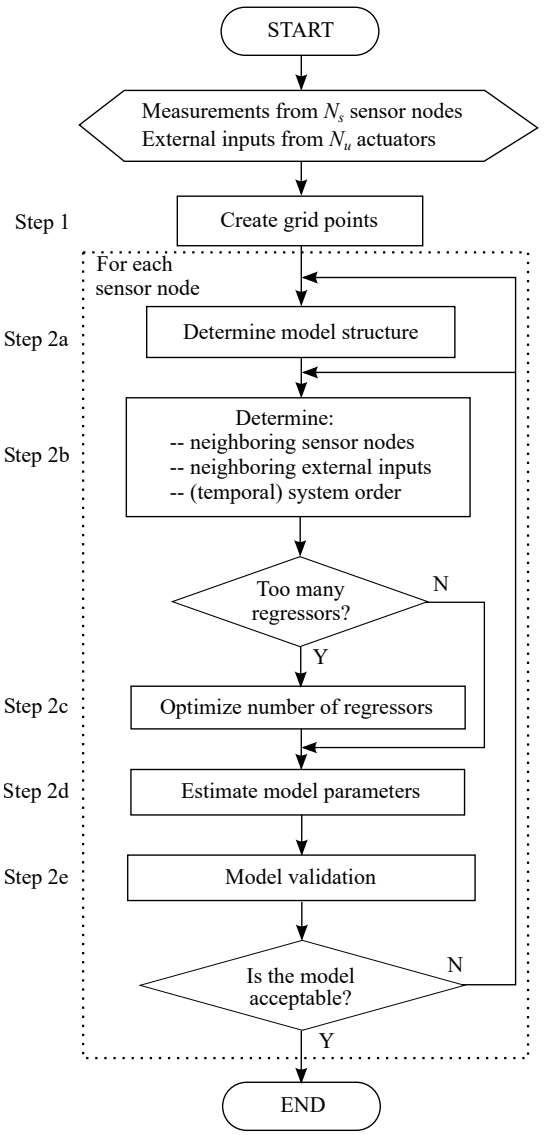


Figure 4.3: Flow chart of the proposed method.

The model structure selection step is not explicitly presented in the examples. This is because the ARX, the first structure tested, is already sufficient to obtain acceptable models, and no further model structure selection step is required.

4.4.1. HEAT CONDUCTION PROCESS

Consider the following two-dimensional heated plate conduction process:

$$\frac{\partial T(z,t)}{\partial t} = \frac{\kappa}{\rho C_\mathrm{p}} \left[\frac{\partial^2 T(z,t)}{\partial z_1^2} + \frac{\partial^2 T(z,t)}{\partial z_2^2} \right], \forall z \in \mathscr{Z} \setminus \mathscr{Z}_\mathrm{b}, \forall t \tag{4.32a}$$

$$T(z,t) = T_{\mathrm{b},i}(t), \quad \forall z \in \mathscr{Z}_{\mathrm{b},i}, \mathscr{Z}_\mathrm{b} = \cup \mathscr{Z}_{\mathrm{b},i}, \forall t \tag{4.32b}$$

$$T(z,0) = T_0, \quad \forall z \in \mathscr{Z} \tag{4.32c}$$

where $T(z,t)$ is the temperature of the plate at location z and at time t, ρ the density of the plate material, T_0 the initial temperature, C_p the heat capacity, κ the thermal conductivity, and $z = (z_1, z_2)$ the spatial coordinate on the plate. Equations (4.32b) and (4.32c) are the boundary and initial conditions, respectively. The plate's parameters are listed in Section 4.4.1. The values of the material properties are adopted from [114] and modified to speed up the simulation.

Parameter	Symbol	Value	Unit
Material density	ρ	4700	$\mathrm{kg\,m^{-3}}$
Thermal conductivity	κ	700	$\mathrm{W\,m^{-1}\,K^{-1}}$
Heat capacity	C_p	383	$\mathrm{J\,kg^{-1}\,K^{-1}}$
Plate length	L	0.7	m
Plate width	W	0.5	m
Initial temperature	T_0	35	°C
Sampling period	T_s	1	s
Grid size	$\Delta_{z_1}, \Delta_{z_2}$	0.05	m

For this example, a set of identification data is obtained by simulating the discretized version of (4.32). The central approximation of the finite-difference method is used to discretize the space and to create a grid of 14 by 10 cells; the zero-order hold method is used to discretize the time coordinate. The resulting discretized equation is simulated by letting the boundary values $T_\mathrm{b}(\cdot)$ follow pseudo-random binary signals with levels of 25 °C and 80 °C where each boundary B-1 through B-4 (as defined in Figure 4.4) is excited by a different signal u_1 through u_4. It is assumed that the excitation is uniformly distributed along the boundary for each discrete-time step k. In case a sensor node has a boundary in the neighborhood, it is taken as one input to the model. The duration of the steps is randomly selected from the set $\{80, 120, \ldots, 200\}$ seconds. The maximum value of the step duration was determined based on the largest time constant of the node responses, i.e., 180 s.

Ten sensor nodes are placed to measure the temperature of the plate as illustrated in Figure 4.4, with the exact sensor locations given in Table 4.1. The measurements are sampled with period $T_\mathrm{s} = 1$ s and Gaussian noise with zero mean and variance $0.1\,^\circ\mathrm{C}^2$ is added to the measurements. The data set is divided into an identification set and a validation set, consisting of 1500 and 740 samples, respectively.

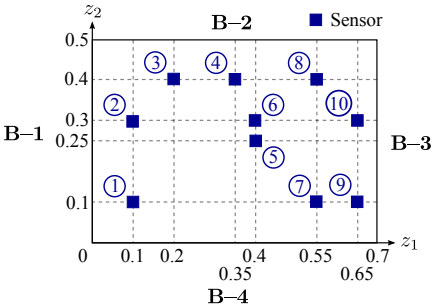


Figure 4.4: Illustration of sensor node locations for the 2D heat conduction example.

Table 4.1: Coordinates of the sensor node locations for the 2D heat conduction equation example.

Sensor #	$(\mathbf{z_1}, \mathbf{z_2})$	**Sensor #**	$(\mathbf{z_1}, \mathbf{z_2})$
Sensor 1	(0.10, 0.10)	Sensor 6	(0.40, 0.30)
Sensor 2	(0.10, 0.25)	Sensor 7	(0.55, 0.10)
Sensor 3	(0.20, 0.40)	Sensor 8	(0.55, 0.40)
Sensor 4	(0.35, 0.40)	Sensor 9	(0.65, 0.10)
Sensor 5	(0.40, 0.25)	Sensor 10	(0.65, 0.30)

The neighboring nodes are defined to be the nodes that lie within the distance ϱ = 0.35 m from a given node. The value of this neighborhood radius is set sufficiently large compared to the physical dimensions so that a sufficient number of neighboring sensors are included in the model. Typically, prior knowledge about the process can be used to determine a suitable value for the radius ϱ.

Results from two representative sensors are presented: 1 and 5. Sensor 1 is relatively close to the boundaries; it has three neighboring sensors. Since boundaries B-1 and B-4 are inside the radius ϱ, the values at boundaries B-1 and B-4 are included as inputs to the model of sensor 1. Sensor 5 is near the middle of the plate. In the radius ϱ, it has 9 neighboring sensors and three inputs from the boundary B-2, B-3, and B-4.

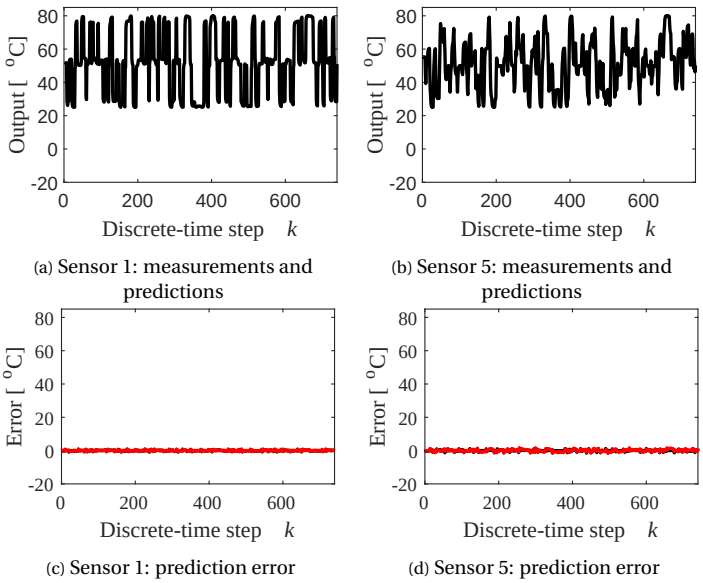
Subsequently, it is necessary to determine the order of the system. Considering that the system is slow, 10^{th}-order models with an ARX structure are used for the models. Thus, sensor 1 initially has 61 regressors for the model, and sensor 5 has 131 regressors, including the bias. Lasso is applied to reduce the number of parameters in the model, using the `lasso` function in the Statistics Toolbox of Matlab. The function requires the maximum number of parameters in the model as additional input and returns a set of models with the number of parameters varying from one to the maximum number specified. The function returns a set of reduced models for different values of the regularization parameter λ and the corresponding Mean Squared-Error (MSE) values. Then, one of those models is selected based on the smallest MSE obtained from the validation data set.

After input selection, a model with 11 parameters is obtained for sensor 1 and a model with 26 parameters for sensor 5. The reduced models are the following:

$$\begin{aligned} y_1(k+1) = {} & 0.0155\, y_1(k-1) + 0.0540\, y_3(k-1) + 0.0467\, y_5(k-1) + \\ & + 0.4118\, u_1(k-1) + 0.0173\, u_1(k-2) + 0.0026\, u_1(k-3) \\ & + 0.4134\, u_4(k-1) + 0.0244\, u_4(k-2) + 0.0034\, u_4(k-3) \end{aligned}$$

$$
\begin{aligned}
&-0.5318\\
\\
y_5(k+1) = {} & 0.0089\,y_5(k-1) + 0.0093\,y_8(k-1) + 0.0037\,y_{10}(k-2)\\
&+ 0.0145\,y_2(k-1) + 0.0050\,y_2(k-2) + 0.0035\,y_2(k-3)\\
&+ 0.1352\,y_1(k-1) + 0.0241\,y_1(k-2) + 0.0073\,y_1(k-3)\\
&+ 0.1640\,u_2(k-1) + 0.1074\,u_2(k-2) + 0.0350\,u_2(k-3)\\
&+ 0.0131\,u_2(k-4) + 0.0016\,u_2(k-5) + 0.0002\,u_2(k-6)\\
&+ 0.0831\,u_3(k-1) + 0.0627\,u_3(k-2) + 0.0221\,u_3(k-3)\\
&+ 0.0063\,u_3(k-4) + 0.0006\,u_3(k-5) + 0.1646\,u_4(k-1)\\
&+ 0.0588\,u_4(k-2) + 0.0245\,u_4(k-3) + 0.0094\,u_4(k-4)\\
&+ 0.0039\,u_4(k-5) - 0.8707
\end{aligned}
$$

where $y_i(k)$ is the measurement from sensor i, and $u_j(k)$ is the input from boundary j. From the above models, it can be seen that the model for sensor 5 uses more parameters with larger lags of inputs and neighboring measurements; this indicates that more time is needed to propagate those inputs and neighboring measurements to influence sensor 5. This is different in the case of sensor 1, which is closer to the boundaries and for which the resulting model is mainly influenced by the inputs, which yields a simpler model. The models also have constant/bias terms, which can be interpreted as heat transferred between the adjacent nodes.



(a) Sensor 1: measurements and predictions

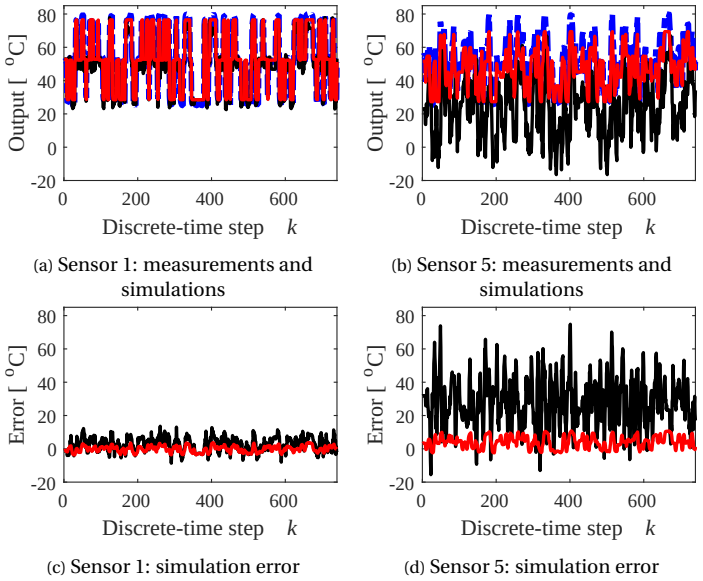
(b) Sensor 5: measurements and predictions

(c) Sensor 1: prediction error

(d) Sensor 5: prediction error

Figure 4.5: Measurements (blue, invisible due to the overlap) and one-step ahead prediction for the models with full inputs (black curves) and the ones with reduced inputs (red curves) using the validation data set for the two-dimensional heat conduction example. Note that the prediction errors of the full and the reduced input models are overlapping.

Figure 4.5 and 4.6 show the one-step ahead predictions, the free-run simulation, and their corresponding errors in comparison with the validation part of the data. As one can expect, for the



(a) Sensor 1: measurements and simulations

(b) Sensor 5: measurements and simulations

(c) Sensor 1: simulation error

(d) Sensor 5: simulation error

Figure 4.6: Measurements (blue) and free-run simulation for the models with full inputs (black) and the ones with reduced inputs (red) using the validation data set for the two-dimensional heat conduction example.

validation data, the one-step-ahead prediction error is much lower than the error for the free-run simulation. In addition, it can also be seen that the free-run prediction errors are smaller for the reduced input models than those of the full input models. This is more obvious for the model of sensor 5. As one would expect that the full models would deliver smaller errors, this means the full models suffer from overfitting. In general, the proposed identification approach works well in this case and delivers sufficiently good models.

The figures also show that the output error of the model using measurements from sensor 1 is generally smaller than that of sensor 5. This can be explained as follows: Figure 4.4 shows that sensor 5 has more neighboring sensors than sensor 1. This means the identification for measurements of sensor 5 involves more noise from measurements of neighboring sensors than in the case of sensor 1.

Figure 4.7 shows contour plots of the temperature distribution of the validation data at discrete-time step $k = 90$; this time step value has been selected arbitrarily. The one-step-ahead predictions for full and reduced-order models are shown in Figure 4.8, and the free-run predictions are shown in Figure 4.9. The sensor locations are marked with black square boxes where sensor numbers are placed on the left-hand side of the markers.

It can be seen the contour of the one-step ahead prediction is similar to that from the validation data. This is confirmed by the error contour, which is almost uniformly colored around the zero value. Note that the contours look relatively coarse because they are plotted based on sparse measurement locations using ordinary kriging, implemented in the ooDACE toolbox [51, 52], to interpolate temperature at locations that are not measured.

The R^2 fit for the full models and reduced models of sensor 1 and sensor 5 is shown in Table 4.2. The table shows that the R^2 fit of the identified models is accurate. It can also be seen that for the free-run simulation prediction, the reduced input models have a better R^2 fit than the full models. This shows that, in this case, the full models are over-parameterized and that an input reduction

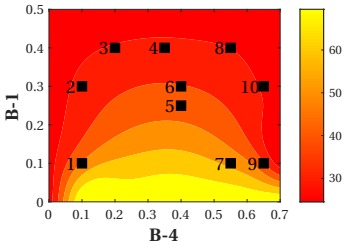
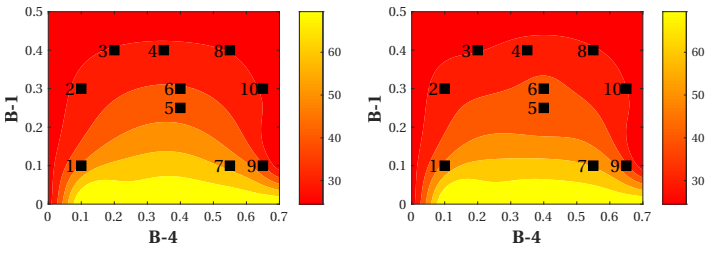
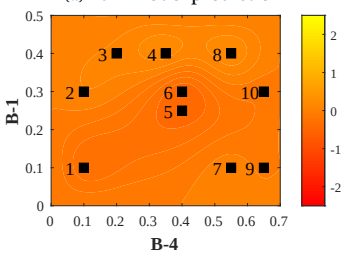


Figure 4.7: Contours of the heated plate model at discrete-time step $k = 90$ of the validation data. The black square markers are the sensor locations with their corresponding sensor numbers left of the markers.

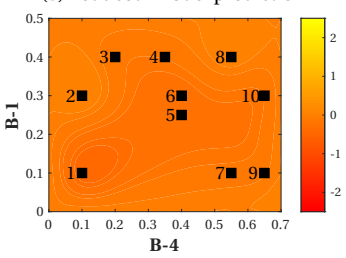


(a) Full-model prediction

(b) Reduced-model prediction

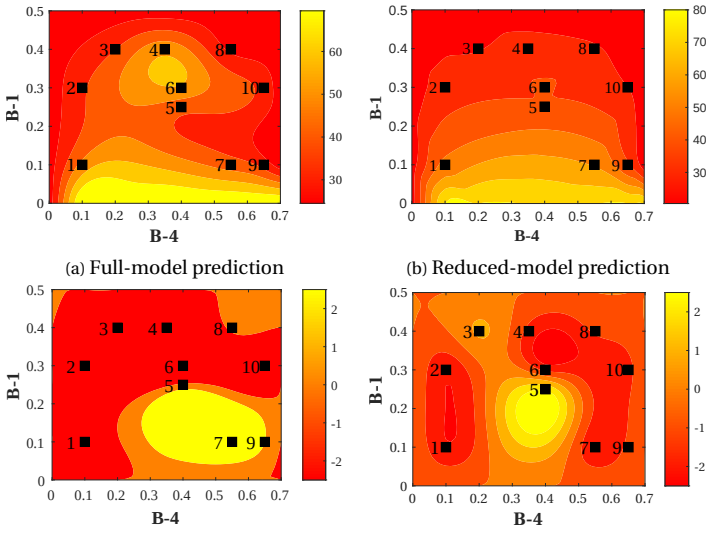


(c) Full-model prediction error



(d) Reduced-model prediction error

Figure 4.8: Contours of the one-step ahead prediction of the heated plate model at discrete-time step $k = 90$ for full and reduced-order models. The black square markers are the sensor locations with their corresponding sensor numbers left of the markers.



(a) Full-model prediction

(b) Reduced-model prediction

(c) Full-model prediction error

(d) Reduced-model prediction error

Figure 4.9: Contours of the free-run simulation of the heated plate model at discrete-time step $k = 90$ for full and reduced-order models. The black square markers are the sensor locations with their corresponding sensor numbers left of the markers.

results in better models.

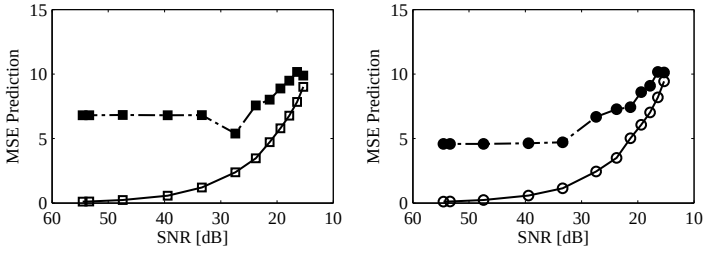
Table 4.2: The R^2 fit of the full and reduced models for one-step ahead and free-run simulation predictions of the heat equation example.

	One-step ahead		Free-run simulation	
Sensor #	**Full model**	**Reduced model**	**Full model**	**Reduced model**
1	99.9807%	99.9784%	94.9872%	99.0116%
5	99.9168%	99.8016%	-15.5446%	93.2703%

Figure 4.10 shows the change of the mean square one-step-ahead prediction error. It can be seen that a decrease in the signal-to-noise (SNR) ratio increases the prediction error. The figures also show that the full models have a better prediction performance than the reduced ones, but the difference decreases as the SNR decreases, i.e., an increase in the noise level. For the full models, the error increases exponentially, while for the reduced models, it is relatively constant for larger SNR values and increases almost linearly for smaller ones. It can also be seen in Figure 4.10 that for a relatively narrow range of low noise levels, the reduced models exhibit better robustness than the full ones.

In the numerical examples presented in this chapter, the location of the sensors is fixed. However, using the same methodology, it is possible to decide on the best sensor locations by determining the most appropriate number and locations of the sensors. Assume that the number of sensors is not fixed but that they can be located only within a finite number of possible places. For the 2D heat conduction example, the grid configuration as shown in Figure 4.11 represents the potential locations for sensor placement.

The same experiment with the same setup as in the beginning of this section is performed,



(a) Sensor 1: increasing noise (b) Sensor 5: increasing noise

Figure 4.10: One-step ahead prediction error of sensors 1 and 5 for the increasing noise variance. The solid lines correspond to the full models, and the dashed lines correspond to the reduced models.

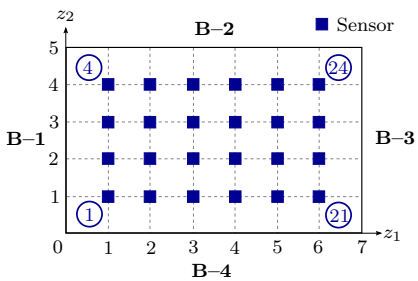
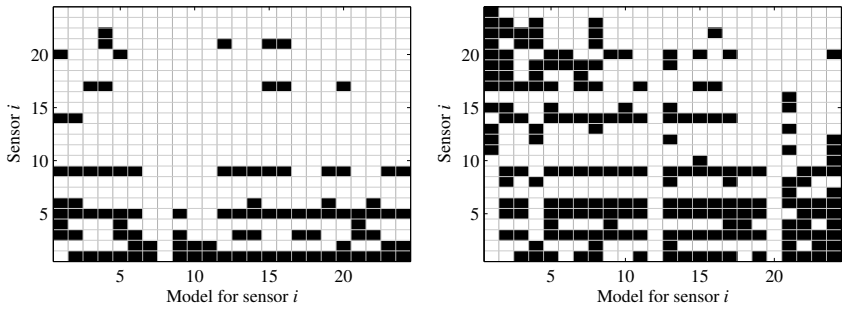


Figure 4.11: A grid of sensors for the 2D heat conduction example set up in order to determine appropriate sensor locations.

except that there are now more sensors involved, and also, the model for measurements from sensor i uses measurements from all other sensors as neighbors. After model reduction, we check the regressors from sensor i whose non-zero parameters indicate that sensor i is used in the model. For the experiment, models with 11 and 25 parameters are selected to be analyzed. The importance of the sensors is represented in the diagrams in Figure 4.12. In the figures, the horizontal axis represents the sensor models and the vertical axis represents the sensors involved in the models. The sensors used in a model are indicated by black squares in the vertical direction. If the sensor is not used in the model, a white square is drawn.



(a) Models with 11 parameters

(b) Models with 25 parameters

Figure 4.12: Sensors whose measurements that are used in the model.

A consideration for placing a sensor at a certain location is that the measurements from the sensor are used to build models of measurements from the other sensors. The more models use measurements from the sensor, the more important the sensor is. This is indicated by a large number of black squares in the horizontal direction in Figure 4.12. In Figure 4.12a, it can be seen that sensors 1, 5, and 9 are important because they are used by the majority of the models. This means that the locations of these sensors are high-potential candidates for the measurement locations. Figure 4.12a shows several white columns. These columns mean that a model with the specified number of parameters, in this case 11, cannot be found. The white column locations are different for different numbers of model parameters as shown in Figure 4.12b for models with 25 parameters. The figure also shows that sensors 3, 6, and 14 become more important by increasing the number of parameters to 25.

4.4.2. GREENHOUSE TEMPERATURE MODEL IDENTIFICATION

The proposed approach is also used to identify a model based on data from a small-scale greenhouse setup shown in Figure 4.13. The setup was built at TNO in the Netherlands. Its length is 4.6 m, its width 2.4 m, while the height of the wall and the roof are 2.4 m and 2.9 m, respectively. Six 400 W convection heaters, each of 0.6×0.6 m, are placed on the floor of the setup. This gives an average of 200 W m^{-2} irradiance. The heaters are meant to mimic the convective effect caused by the absorption of solar energy by the floor during the day [29]. The coordinates of the centers of the heaters are shown in Table 4.3.

The temperature measurements are collected using wireless sensors, which is a promising technology, with some applications in production greenhouses already reported [231]. For the experiments, a total of 68 sensor nodes have been installed to measure the temperature inside the greenhouse. Out of these, 45 sensor nodes are arranged on a grid with the spacing along the z_1, z_2, and z_3 axes equal to 0.3000 m, 0.7667 m, and 0.5500 m, respectively. Additionally, 5 sensor nodes are placed below the roof, 6 sensor nodes are right at the center of the heaters, and 12 sensors are

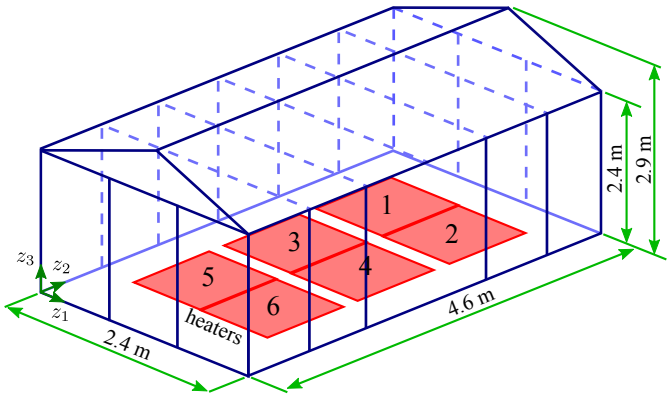


Figure 4.13: A schematic representation of the greenhouse with its physical dimensions.

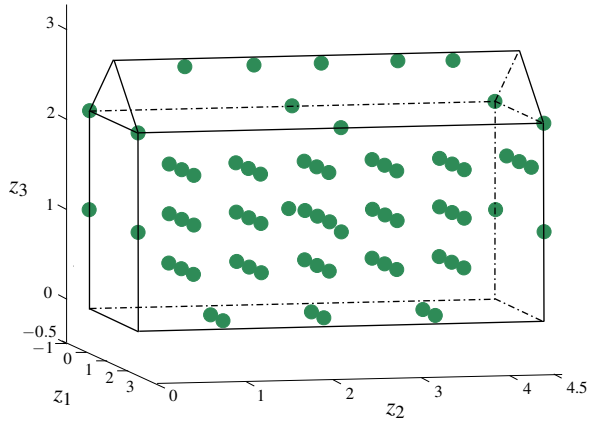


Figure 4.14: A schematic representation of the sensor locations in the greenhouse.

Figure 4.15: A photograph of the greenhouse setup used in the case study.

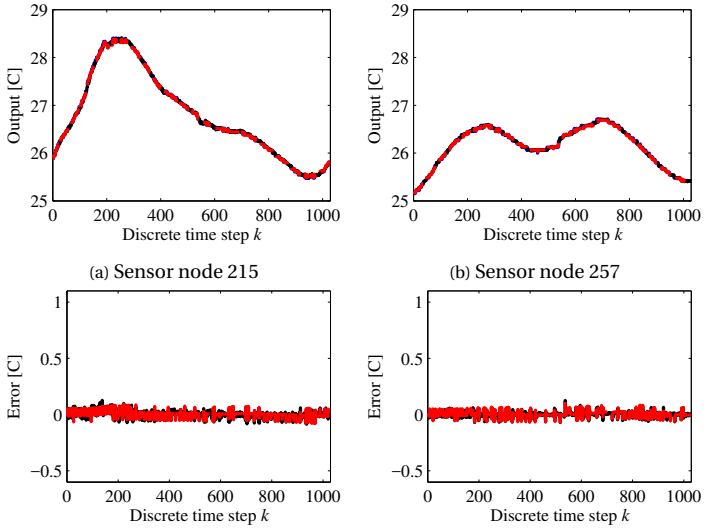
attached to the four walls of the greenhouse. Figure 4.14 shows the sensor locations and Figure 4.15 gives a photo of the setup.

Table 4.3: The center coordinates of the convection heaters in the greenhouse.

Heater #	$(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3)$	**Heater #**	$(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3)$
Heater 1	(0.90, 3.45, 0.00)	Heater 4	(1.50, 3.45, 0.00)
Heater 2	(0.90, 2.30, 0.00)	Heater 5	(1.50, 2.30, 0.00)
Heater 3	(0.90, 1.15, 0.00)	Heater 6	(1.50, 1.15, 0.00)

Throughout the identification experiments, the heaters were turned on and off in pairs: heater 1 paired with heater 4, heater 2 with heater 5, and heater 3 with heater 6 so that there are three different input signals. In total 3179 data samples have been acquired, of which 2149 samples are used for identification and 1030 samples for validation. The data sets are centered by subtracting their means before the identification and model reduction with Lasso is applied.

Among all sensors, identification results from two sensor nodes are presented: sensor node 215, located at position (1.80, 3.83, 1.10), and sensor node 257, located at (0.00, 0.00, 2.20). The neighborhood radius selected is $\varrho = 1.25\,\text{m}$, which gives 19 neighbors for sensor node 215 and 7 neighbors for sensor node 257.

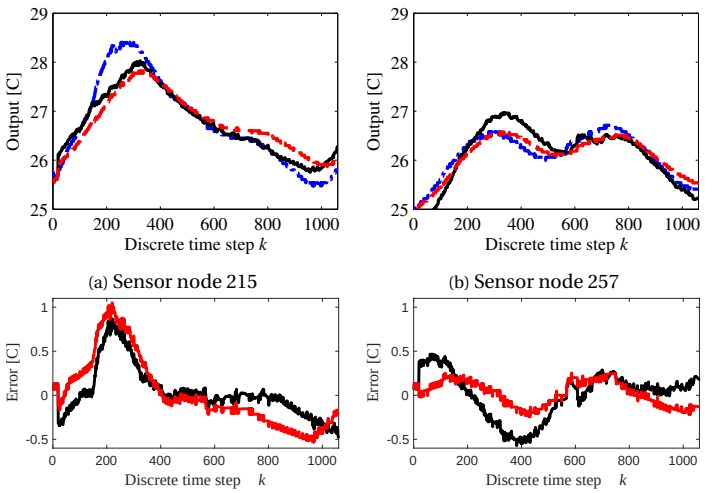


(a) Sensor node 215 (b) Sensor node 257

(c) Sensor node 215 error (d) Sensor node 257 error

Figure 4.16: Greenhouse setup measurements (blue) and one-step-ahead predictions for the model with the full set of inputs (black) and for the model with the reduced set of inputs (red) using the centered validation data set and their corresponding prediction error, i.e., error for the full model (black) and the reduced model (red). Note that the outputs and the corresponding errors of the full and the reduced input models are overlapping.

A 10^{th}-order linear ARX structure is selected for the model so that initially, there are 570 parameters and 280 parameters for respectively sensor nodes 215 and 257. The measurements, full input model output, and reduced input model simulation output, as well as the corresponding one-step estimation errors for the validation data, are shown in Figure 4.16. Similar plots for 20-step ahead prediction are shown in Figure 4.17. Setting the maximum number of parameters to 10, the



(a) Sensor node 215 (b) Sensor node 257

(c) Sensor node 215 error (d) Sensor node 257 error

Figure 4.17: Greenhouse setup measurements (blue) and 20-step-ahead predictions for the model with the full set of inputs (black) and for the model with the reduced set of inputs (red) using the centered validation data set and their corresponding prediction error, i.e., error for the full model (black) and for the reduced model (red).

Table 4.4: Mean square error (MSE) and R^2 fit of the full and reduced models for the greenhouse validation data example in the case of one-step and 20-step ahead predictions.

	Sensor 215		**Sensor 257**	
Performance	**Full**	**Reduced**	**Full**	**Reduced**
1-step prediction MSE	0.0310	0.0261	0.0229	0.0243
20-step prediction MSE	0.2826	0.3970	0.2600	0.1412
1-step prediction R^2 fit (in %)	99.8772	98.9155	99.6821	99.6515
20-step prediction R^2 fit (in %)	90.0368	80.0343	64.7745	89.8994

following models are obtained:

$$\begin{aligned} y_{215}(k+1) = {} & 0.8241\, y_{215}(k-1) + 0.1332\, y_{215}(k-2) + 0.0065\, y_{215}(k-3) \\ & + 0.0037\, y_{215}(k-5) + 0.0041\, y_{215}(k-7) + 0.0043\, y_{215}(k-8) \\ & + 0.0113\, y_{206}(k-1) + 0.0048\, y_{218}(k-1) + 0.0027\, y_{218}(k-2) \\ & + 0.0043\, y_{218}(k-3) \end{aligned} \tag{4.33}$$

$$\begin{aligned} y_{257}(k+1) = {} & 0.6206\, y_{257}(k-1) + 0.2410\, y_{257}(k-2) + 0.0093\, y_{257}(k-3) \\ & + 0.0368\, y_{257}(k-4) + 0.0092\, y_{8}(k-1) + 0.0480\, y_{220}(k-1) \\ & + 0.0064\, y_{234}(k-1) + 0.0198\, y_{264}(k-1) + 0.0003\, y_{20}(k-1) \\ & + 0.0036\, y_{20}(k-2) \end{aligned} \tag{4.34}$$

where $y_i(k)$ is the measurement at sensor node i. It can be seen that the neighboring measurements contribute to the identified model. The MSE and the R^2 fit for the validation data are shown in Table 4.4. For sensor 215, it can be seen that the MSE is smaller and the R^2 fit is larger for the reduced input model compared to the full model; while for sensor 257, the MSE increases slightly and the R^2 fit decreases slightly. For the case of sensor 215, the reduction of the R^2 fit suggests the full model is over-parameterized. Increasing the prediction horizon to 20 steps reduces the prediction performance significantly; however, the R^2 shows that the models are still stable for prediction up to 20-step ahead. Generally, it can be said that reducing the number of inputs in the models does not significantly decrease the performance of the models. This also indicates that the proposed identification framework works well in this example.

A set of simulations were performed to assess the performance — in terms of the one-step ahead prediction MSE — of the models for different numbers of neighbors for sensor 238. This sensor is located about the middle of the setup and has 8 neighbors with the same height z_3. For neighbor visualization ease, the labeled sensors are shown in Figure 4.18. The identification is performed for 2, 4, 6, and 8 neighbors, excluding sensor 238 itself. The performance of the full and the reduced models is compared for the validation data. The neighbors and the performance comparison are shown in Table 4.5. From the table, it can be seen that the one-step ahead prediction errors hardly differ for different numbers of neighbors. This shows the proposed framework is not sensitive to the number of neighboring sensors.

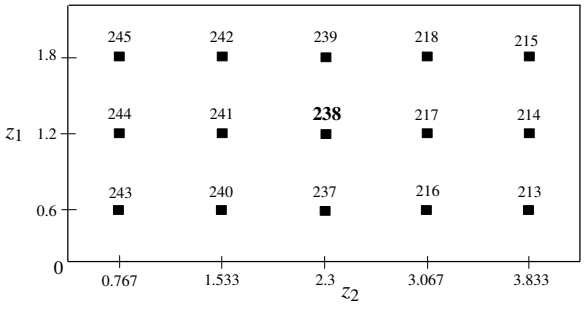


Figure 4.18: Sensors at $z_3 = 1.1$ with sensor id labels.

An experiment to estimate values at locations that are not measured is also performed for sensors shown in Figure 4.19. In this experiment, data from sensors 217, 238, and 241 are not identified, and their estimates for the validation data are obtained by using ordinary kriging. The

Table 4.5: The coordinates of sensor 238, its neighbors, and its performance for different numbers of neighboring sensors. The X symbol indicates that the sensor is used as a neighbor.

	Number of neighbors							
Sensor #	**2**		**4**		**6**		**8**	
213		X		X	X		X	
214		X			X		X	X
215				X		X		X
216							X	
217					X		X	X
218						X		X
237						X		
239						X		
240	X		X			X		X
241	X		X	X	X		X	X
242				X	X		X	
243			X			X		X
244			X		X		X	
245							X	X
MSE full	0.0215	0.0215	0.0216	0.0211	0.0220	0.0208	0.0220	0.0222
MSE red.	0.0236	0.0230	0.0238	0.0233	0.0235	0.0239	0.0235	0.0239

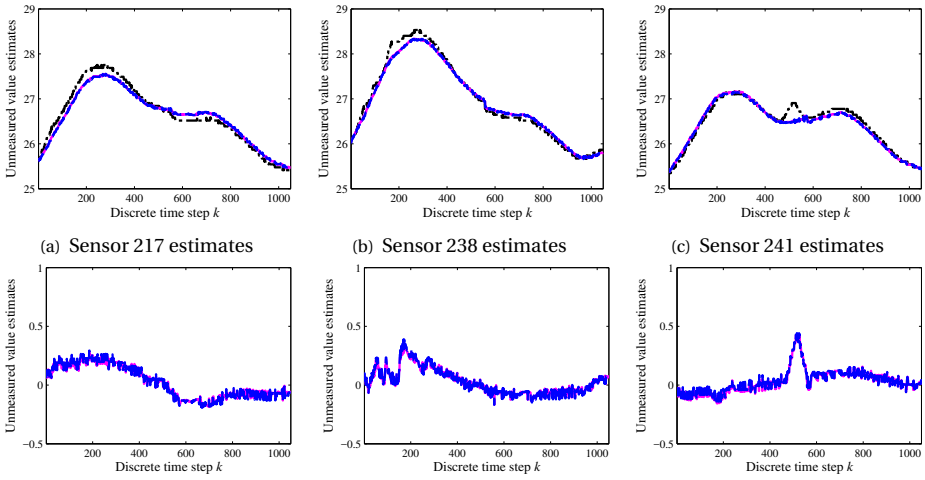
experiment is performed for both full and reduced models. The kriging models are developed by using the estimates of the validation data of the remaining sensors. The results are shown in Figure 4.19 for the estimates and their corresponding error, respectively. The experiment is repeated by omitting sensors 216, 217, 218, 240, 241, and 242. The estimates are shown in Figure 4.20 and their corresponding errors in Figure 4.21.

The figures show that ordinary kriging estimates sufficiently well the values at locations that are not measured. Furthermore, estimation differences between the full and the reduced models are not significant. For the second experiment, it can be seen that the kriging estimates for sensors 216, 217, and 218 look similar; and so do those for sensors 240, 241, and 242. This can be explained by looking at the validation data from sensors 216, 217, and 218 plotted as a group in Figure 4.22a and those from sensors 240, 241, and 242 as the other group in Figure 4.22b. From the figure, it can be seen that the temperature difference within a group is small and this creates kriging estimates with insignificant differences among them.

Contour plots of the greenhouse temperature from the validation data for $0.6 \geq z_1 \geq 1.8$, $0.767 \geq z_2 \geq 3.833$, and fixed $z_3 = 1.1$ at discrete-time step $k = 400$ are shown in Figure 4.23. The contour of the estimated temperature from the full models for one-step ahead and 20-step ahead prediction are shown in Figure 4.24, while contours from the reduced models are in Figure 4.25. The contours The plots are in 2D because the ooDACE toolbox is only able to build kriging models from 2D data. In the same way, as for the heated plate example, the plots show the contour of the validation data and the one-step ahead prediction of the full and reduced models. It can be seen that the error is larger with the reduced models than with the full model.

4.5. SUMMARY

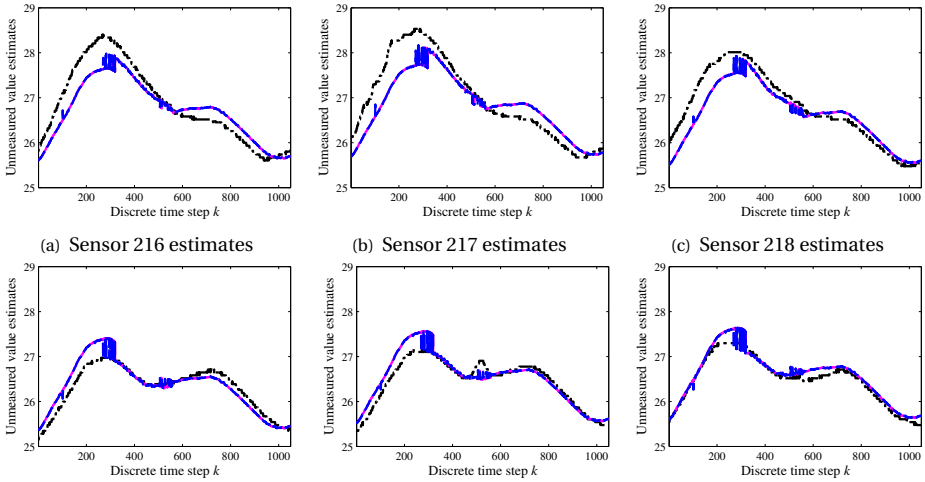
In this chapter, a method for the identification of distributed-parameter systems was presented. The method is a finite-difference based method that takes into account inputs from neighboring measurements and actuators into the model. The method assumes that the underlying partial diffe-



(a) Sensor 217 estimates (b) Sensor 238 estimates (c) Sensor 241 estimates

(d) Sensor 217 estimation error (e) Sensor 238 estimation error (f) Sensor 241 estimation error

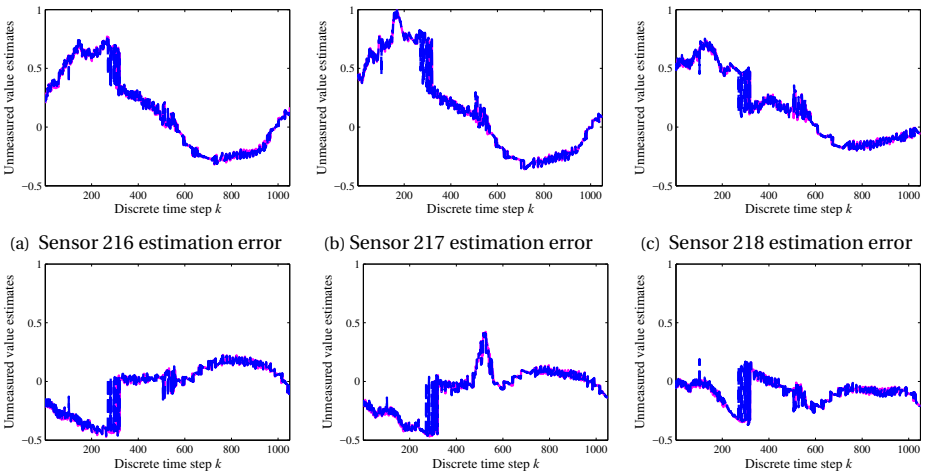
Figure 4.19: Validation data estimates for sensors 217, 238, and 241 by using ordinary kriging and their corresponding error. For (a), (b), and (c), black lines are the validation data, magenta lines are estimates from the full models, and blue lines are estimates from the reduced models. For (d), (e), and (f), magenta lines are errors from the full models, and blue lines are errors from the reduced models.



(a) Sensor 216 estimates (b) Sensor 217 estimates (c) Sensor 218 estimates

(d) Sensor 240 estimates (e) Sensor 241 estimates (f) Sensor 242 estimates

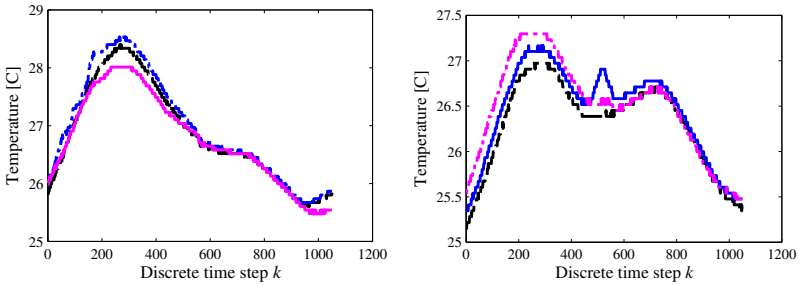
Figure 4.20: Validation data estimates for sensors 216, 217, 218, 240, 241, and 242 by using ordinary kriging. Black lines are the validation data, magenta lines are estimates from the full models, and blue lines are estimates from the reduced models.



(a) Sensor 216 estimation error (b) Sensor 217 estimation error (c) Sensor 218 estimation error

(d) Sensor 240 estimation error (e) Sensor 241 estimation error (f) Sensor 242 estimation error

Figure 4.21: Validation data estimation error for sensors 216, 217, 218, 240, 241, and 242 by using ordinary kriging. Magenta lines are errors from the full models and blue lines are errors from the reduced models.



(a) Sensor 216, 217, and 218 (b) Sensor 240, 241, and 242

Figure 4.22: Validation data plot from the sensor: (a) 216 in black, 217 in blue, 218 in magenta (b) 240 in black, 241 in blue, and 242 in magenta.

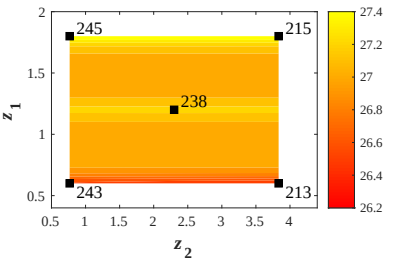
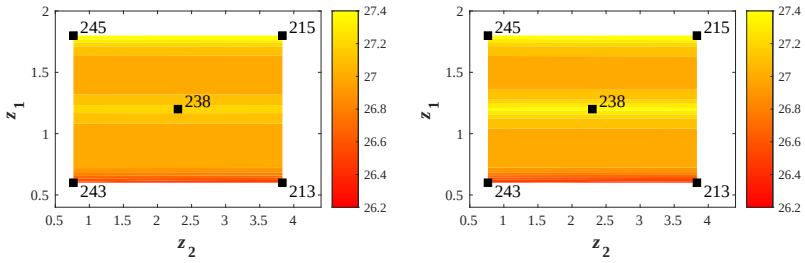
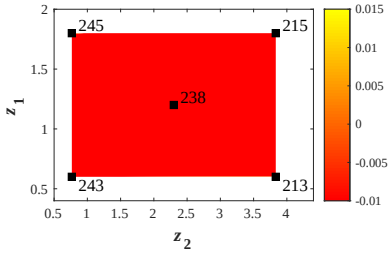


Figure 4.23: Contours of the greenhouse temperature validation at discrete-time step $k = 400$ for $0.6 \geq z_1 \geq 1.8, 0.767 \geq z_2 \geq 3.833$ and fixed $z_3 = 1.1$. The black square markers are the sensor locations, and the labeled sensors are used to build the kriging model.

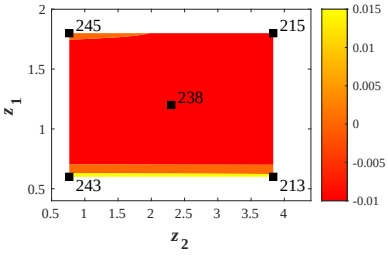


(a) One-step ahead prediction

(b) 20-step ahead prediction

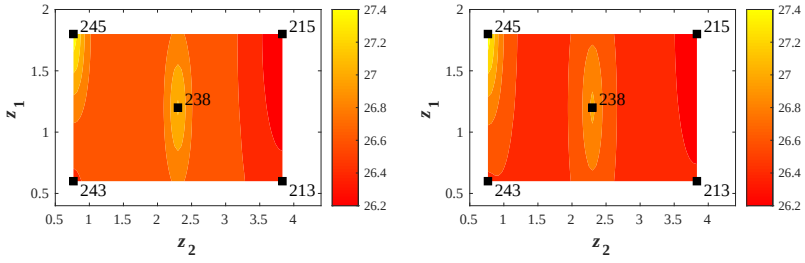


(c) One-step-ahead prediction error



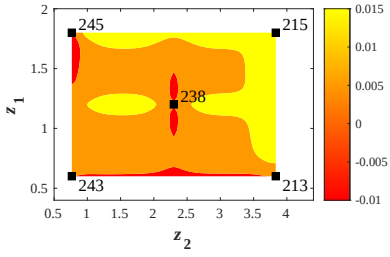
(d) 20-step-ahead prediction error

Figure 4.24: Contours of the greenhouse temperature from the full models at discrete-time step $k = 400$ of the validation data for $0.6 \geq z_1 \geq 1.8, 0.767 \geq z_2 \geq 3.833$ and fixed $z_3 = 1.1$. The contours are from one-step and 20-step ahead predictions. The black square markers are the sensor locations, and the labeled sensors are used to build the kriging model.

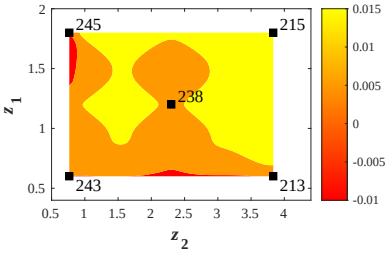


(a) One-step ahead prediction

(b) 20-step ahead prediction



(c) One-step-ahead prediction error



(d) 20-step-ahead prediction error

Figure 4.25: Contours of the greenhouse temperature the reduced models at discrete-time step $k = 400$ of the validation data for $0.6 \geq z_1 \geq 1.8, 0.767 \geq z_2 \geq 3.833$ and fixed $z_3 = 1.1$. The contours are from one-step and 20-step ahead prediction. The black square markers are the sensor locations, and the labeled sensors are used to build the kriging model.

rential equation is not known. Although a finite-difference based method is proposed, the method does not require dense measurement locations in the system. This feature allows the applicability of the method to real-life systems, which generally have a limited number of measurements. In addition, model reduction methods were applied to reduce the complexity of the model in case a large number of inputs are involved in the model. The effectiveness of the method has been shown with the help of two examples, a simulated heated plate and a real greenhouse.

5

SEARCH TREE BASED REGRESSOR SELECTION FOR NONLINEAR SYSTEM IDENTIFICATION

5.1. INTRODUCTION

Nonlinear system identification methods have been developed and applied to further understand real-life complex systems [148, 211]. In those methods, different models can be used, e.g., TS fuzzy models [214], networks of wavelets [245], networks of FIR systems [173], among others. Nonlinear system identification can also be performed online during the operation of the system [9]. Models obtained from data through system identification may involve a large number of potential regressors or inputs, but not all of them may be equally important. Simple models are usually desired because they ease the interpretation and analysis of the model and reduce the computational load, especially in applications where limited computing resources are available, e.g., in model-based real-time control. The use of simple models also reduces the danger of overfitting, which might otherwise result in unreliable parameter estimates and prediction errors.

Complex models can be simplified by keeping only the set of inputs that significantly influence the output. When reducing the number of inputs, the main question is how to reach a good trade-off between the complexity and accuracy of the model[1]. Besides providing a simplified model with sufficient accuracy, a regressor selection method should be robust, i.e., for different data sets from the same process, similar results should be obtained [106].

Regressor selection is an essential topic in statistics. Three main approaches can be used [72]: exhaustive search, stepwise regression (which includes forward regression, backward elimination, and their combinations), and optimization-based approaches, also called shrinkage methods. Exhaustive search is a simple regressor selection method that is guaranteed to give the best model, given the specified performance measure. However, for a large number of regressor candidates, the method is computationally intractable. Stepwise regression commonly combines forward regression and backward elimination in an automated regressor search. The regressor inclusion and exclusion are based on the statistical F-test, χ^2-test, or other criteria, e.g., the Akaike Information

[1] In this article, the terms inputs and regressors are used interchangeably because in a dynamic system model, the regressors consist of lagged inputs and outputs.

Criterion [2]. In optimization-based formulations, regressor selection is included by adding a regularization term in the objective function so that the least-squares optimization suppresses parameters of unimportant regressors to zero. Examples of methods using the optimization-based approach are in, e.g., [31, 185, 217].

The use of statistical regressor selection methods for linear system identification is quite common [159]. However, regressor selection for nonlinear system identification is not as established as for linear system identification since nonlinear models can be represented in various forms, e.g., polynomial models, neural networks, fuzzy models, etc. In the literature, there are different regressor selection methods for identification of nonlinear systems that can be considered as stepwise regression because they evaluate one regressor at a time and select it if it fulfills the selection criterion [15, 26, 95, 101, 126, 140, 153, 165, 187, 193, 195, 247]. Billings et al. [26] employed orthogonal least squares to compute the error reduction ratio for nonlinear output-affine models that are linear in their parameters. Zheng and Billings [247] studied the use of the mutual information criterion to configure radial basis function networks. Autin et al. [15] defined an input-output matrix from a linearized nonlinear SISO system and used that matrix to estimate the order of the system. Rhodes and Morari [193] extended the false nearest neighbors method [126] to include exogenous inputs. He and Asada [101] proposed a Lipschitz quotient to compute an index that can be used to determine important regressors and also to define the order of a nonlinear system. Krishnaswami et al. [140] proposed the use of the coherence function of input-output data to improve the approach of He and Asada [101] for MIMO systems. Mendez and Billings [165] developed a two-step method by firstly using orthogonal least-squares forward regression to reduce the number of regressor candidates for nonlinear ARMAX structures. In the second step, all possible combinations from the reduced number of regressor candidates are exhaustively searched. Lind and Ljung [153] investigated an interaction test between regressors and dependent variables based on the analysis of the variance called Test of Interactions using Layout for Intermixed ANOVA (TILIA). Šindelář and Babuška [229] developed a fuzzy-clustering-based method to measure the similarity of regressors in both input and output space. Regarding Takagi-Sugeno (TS) fuzzy models, Chiu [46] employed a backward search approach to exclude the antecedent clauses of a TS fuzzy model generated from a full set of regressors. However, the final model has to be selected manually by the user. Sáez and Zúñiga [195] proposed a sensitivity analysis to select important regressors, while Hadjili and Wertz [95] selected regressors based on greedy search. Zhang et al. [244] enhanced the firefly optimization approach by simulated annealing for feature selection of classification and regression models. Kim and Boukouvala [129] surveyed the literature for machine learning-based regressor selection for data-driven-based models. Curreri et al. [55] studied methods for input selection in soft sensor applications and compared the methods using a real-application data set. Valente and Maldonado [221] developed a forward regressor selection approach for a support vector regression model of time-series data. Sun et al. [212] proposed a selection method based on the unique causal effect of a regressor to the key performance indicator of a soft sensor.

There are also evolutionary-based algorithms that are related to regressor selection and system identification. Askari and Crevecoeur [10] developed an evolutionary symbolic sparse regression approach for the system identification of multibody systems, aiming to discover the equations of motion and system parameters from time-series data using genetic programming. Khandelwal et al. [128] presented the use of genetic programming to automatically select model structures and parameter estimation formulated as a bi-level optimization problem based on measured data. Huang et al. [118] proposed a method for model order selection in nonlinear systems using a false nearest neighbor algorithm based on Gaussian mixture model clustering and a genetic version of the expectation-maximization algorithm. Zhang et al. [243] introduced a method for constructing a diverse ensemble of decision trees using genetic programming and the Euclidean distance. Zhang et al. [244] enhanced firefly optimization using simulated annealing for the feature selection of

classification and regression models. Lagos-Eulogio et al. [145] designed a method for identification of adaptive IIR by hybridizing cellular particle swarm optimization and differential evolution. Singh et al. [203] combined particle swarm optimization and the Dingo optimizer for the parameter estimation of proton exchange membrane fuel cells. Guo et al. [90] applied biogeography-based learning particle swarm optimization to wearable robots to assist human motion.

The aforementioned methods from the literature tackle the regressor selection problem by testing a regressor candidate to see whether the candidate effectively reduces the error of the model output estimates. Moreover, these methods are developed for a specific model structure, and most of them are for models that are linear in their parameters. This limits the applicability of the methods for other model structures.

The contribution of this chapter is the development of a generic regressor selection method for nonlinear system identification. The method searches a regressor combination set based on combinatorial optimization. In particular, given a finite set of regressors and a cost function, combinatorial optimization is used to find a subset of the regressors that corresponds to the optimal value of the cost function. The method directly evaluates model candidates using different regressor combinations given the performance measure, the model structure, and the number of regressors in the model.

The search is performed by constructing a growing search tree to evaluate models with different regressor combinations. The tree is grown by creating several regressor combinations with an incremental number of regressors directed toward high-performance models. At each increment, the search compares several models simultaneously to guide the tree growth. The expansion is performed until the maximum number of regressors in the combinations or the stopping criterion is met.

A comparison with lasso, the stepwise regression, and the exhaustive search is conducted. We consider examples of models with linear-in-the-parameters, TS fuzzy, artificial neural network structures, and support vector machine regression. The results show that the proposed method is suitable for regressor selection for a variety of model structures.

5.2. PROBLEM STATEMENT

In this section, the regressor selection problem for nonlinear system identification is introduced. Firstly, the identification problem for nonlinear systems based on input-output data is described. Secondly, the formulation to include the regressor selection in the identification problem is presented.

Consider a set of input-output measurements $\{u(k), y(k)\}_{k=1}^{N} = \{\boldsymbol{u}, \boldsymbol{y}\}$ from a nonlinear system, where $u(k)$ is the input sample, $y(k)$ is the output sample, $\boldsymbol{u}$ is the vector of input samples, $\boldsymbol{y}$ is the vector of output samples, and k is the discrete-time index. The goal is to build a nonlinear model whose parameters are estimated from the data. The parameters of the model are obtained by solving the following optimization problem:

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{y}, \hat{\boldsymbol{y}}) \tag{5.1a}$$

$$\text{s.t.} \quad \hat{\boldsymbol{y}} = f(\boldsymbol{\Phi}, \boldsymbol{\theta}) \tag{5.1b}$$

where $L(\cdot)$ is the cost function that describes the difference between the data vector $\boldsymbol{y}$ and the model output vector $\hat{\boldsymbol{y}}$, the function $f(\cdot)$ is determined by the selected nonlinear model, $\boldsymbol{\Phi}$ is the regressor matrix, and $\boldsymbol{\theta}$ is the model parameter vector. The function $L(\cdot)$ sometimes is also referred to as the loss function.

The regressor matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N_\mathrm{P} \times n_\mathrm{r}}$ is constructed from vectors of lagged input-output measurements, where:

$$N_\mathrm{P} = N - \max(n_\mathrm{u}, n_\mathrm{y}) \tag{5.2}$$

is the number of rows of the regressor matrix, with n_y and n_u, respectively, the maximum lags of output and input measurement data. The regressor matrix can be expressed as:

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_{n_r} \end{bmatrix} \tag{5.3}$$

where $\boldsymbol{\phi}_i \in \mathbb{R}^{N_P}$ is the regressor vector with $\boldsymbol{\phi}_i = \begin{bmatrix} \phi_i(1) & \cdots & \phi_i(n_r) \end{bmatrix}^\top$ where $\top$ denotes the transpose operation.

The model parameter vector $\boldsymbol{\theta}$ is commonly estimated by a least-squares method to minimize the difference between the measured output and the model output. In that case, the function L in (5.1) is defined as:

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \left(\boldsymbol{y} - \hat{\boldsymbol{y}}\right)^\top \left(\boldsymbol{y} - \hat{\boldsymbol{y}}\right) \tag{5.4}$$

For input-output data from an unknown system, the identification begins by selecting the model structure, i.e., $f(\cdot)$, and the regressors of the model from a set of delayed input-output signals. The regressors that might be chosen depending on the selected model structure, including, e.g.:

- Linear regressors, e.g., $\phi_i(k) = y(k-1)$ with nonlinear $f(\cdot)$ in (5.1b). Artificial neural networks and TS fuzzy models belong to this class of models.
- Nonlinear regressors by multiplication of delayed signals, e.g., $\phi_i(k) = y(k-1)u(k-1)$ with linear $f(\cdot)$ in (5.1b). This class of models is called linear-in-the-parameters [31].
- Nonlinear regressors by mapping linear regressors through a nonlinear function, e.g., $\phi_i(k) = \kappa\left(y(k-1)\right)$ with $\kappa(\cdot)$ a kernel function, as in support vector regression models [205].

The model parameters are then estimated by using (5.1). The performance of the resulting model will depend on the model structure and the estimated parameters. One may test models with a rather large delay value for unknown systems to capture the relevant dynamics. In that case, the model will involve a large number of regressors. However, from a practical point of view, it is preferred to obtain models that consider only the most essential regressors that effectively reduce the prediction error. Note that the optimal number of regressors is generally also unknown in advance.

The regressor selection problem is to find a column-wise submatrix of $\boldsymbol{\Phi}$ that results in a high-performance model. The measure of the model performance is usually based on (5.4), prediction errors, and may include a penalty for a large number of parameters in the model as in Akaike's information criterion [2] that is used in this chapter. For the regressor selection problem, an indicator vector $\boldsymbol{\varphi}_s \in \{0,1\}^{n_r}$ is defined as follows:

$$\left[\phi_i^*\right] = \begin{cases} 1 & \text{if } \boldsymbol{\phi}_i, \text{ the } i\text{-th column of } \boldsymbol{\Phi}, \text{ is used in the model (5.1b)} \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

Using this indicator vector, the model output (5.1b) can be written as:

$$\hat{\boldsymbol{y}} = g(\boldsymbol{\Phi}, \boldsymbol{\theta}, \boldsymbol{\varphi}_s) \tag{5.6}$$

where g is determined by the selected nonlinear model. From (5.5), the number of regressors in the model, denoted by n_d, can be obtained by summing up the elements of $\boldsymbol{\varphi}_s$. As the optimal number of regressors in the model is typically unknown, it is common to search models with several different values of n_d and treat n_d as an optimization variable.

The optimization problem (5.1) is modified to become the following:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\varphi}_s, n_d} \quad J_{\mathcal{M}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \tag{5.7a}$$

$$\text{s.t.} \quad \hat{\boldsymbol{y}} = g(\boldsymbol{\Phi}, \boldsymbol{\theta}, \boldsymbol{\varphi}_s) \tag{5.7b}$$

$$\left[\phi_i^*\right] \in \{0,1\}, \quad i = 1,\ldots,n_\mathrm{r} \tag{5.7c}$$

$$n_\mathrm{d} = \sum_{i=1}^{n_\mathrm{r}} \left[\phi_i^*\right] \tag{5.7d}$$

where $J_{\mathscr{M}}$ is the cost function for the chosen model structure $\mathscr{M}$, and n_d is the number of columns of the regressor matrix $\boldsymbol{\Phi}$ selected to be used, i.e., the number of regressors in the model. In practice, one may have several candidates of n_d and select one of the resulting models.

The problem of selecting the regressors and, at the same time, finding the optimal $\boldsymbol{\theta}$ can be classified as a mixed-integer nonlinear programming (MINLP) problem. Moreover, in general, these problems are NP-hard [34]. Analytical solutions to general MINLP problems do not exist. In this chapter, a tree search algorithm is proposed to solve the regressor selection problem with applications to linear-in-the-parameter, TS fuzzy, and artificial neural network models.

5.3. EXHAUSTIVE SEARCH REGRESSOR SELECTION

Before discussing the proposed method, a simple solution to the regressor selection problem by exhaustive search is briefly presented.

Typically, the problem (5.7) is solved by firstly fixing the model structure and the maximum number n_d regressors in the model. Given n_d, there are a number of regressor combinations. Let $\mathscr{P}$ denote the set of indicator vectors $\boldsymbol{\varphi}_\mathrm{s}$. The cardinality $|\mathscr{P}| = n_\varphi$, where:

$$n_\varphi = \binom{n_\mathrm{r}}{n_\mathrm{d}} = \frac{n_\mathrm{r}!}{n_\mathrm{d}!(n_\mathrm{r}-n_\mathrm{d})!}$$

Hence, there are n_φ possible models, from which a combination of regressors that results in the model with the best performance is sought. In general, $n_\mathrm{d} \ll n_\mathrm{r}$ and consequently the value of n_φ is large.

Each regressor combination $\boldsymbol{\varphi}_{\mathrm{s},i} \in \mathscr{P}$ corresponds to a model $m_i \in \mathscr{M}$, with $\mathscr{M}$ the set of possible models. The relation between $\mathscr{P}$ and $\mathscr{M}$ is one-to-one, i.e., for a regressor combination indicator $\boldsymbol{\varphi}_{\mathrm{s},i} \in \mathscr{P}$, there is one associated model $m_i \in \mathscr{M}$ whose performance is denoted by $J_{\varphi,i}$. The elements of $\mathscr{P}$ and $\mathscr{M}$ can be sorted according to their performance, i.e., $J_{\varphi,i} \leq J_{\varphi,i+1}$, $i = 1,\ldots,n_\varphi - 1$. The solution to the regressor selection problem is the model parameter $\boldsymbol{\theta}^*$ associated with the regressor combination indicator $\boldsymbol{\varphi}_\mathrm{s}^*$ with the optimal performance J_φ^*.

Searching for the model with the best performance by computing and then sorting the performance of models with all possible regressor combinations is called exhaustive search. The method is a brute-force method and practically gives the exact solution to the regressor selection problem. However, as the number of regressors n_r increases while the desired number of regressors n_d is fixed, the computation load increases factorially. In the following section, an alternative method based on a search tree representation is presented.

5.4. SEARCH-TREE-BASED REGRESSOR SELECTION

In this section, the regression search method proposed in the chapter is presented by specifying the terminology used for the search tree, how the search tree is built, and some remarks. A short description of the Akaike's information criterion is given at the end of this section.

5.4.1. SEARCH TREE TERMINOLOGY

In this section, the proposed regressor selection method is detailed by first introducing the search tree terminology and the corresponding notation related to the regressor selection. In general, a search tree is a tree consisting of connected nodes at different levels with specified rules to grow by

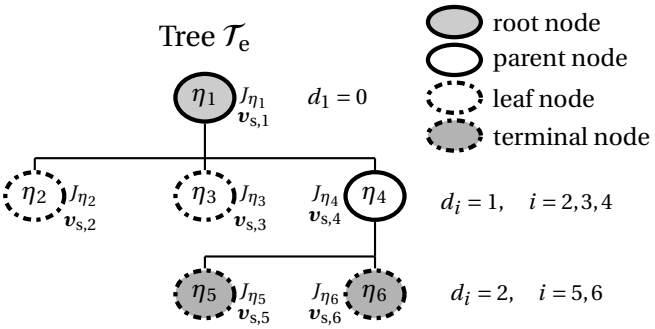


Figure 5.1: Illustration of a search tree $\mathcal{T}_e$. Parent nodes are solid line ellipses and leaf nodes are dash-dotted ellipses. The root is a grey-shaded solid line ellipse, and the terminal node is a grey-shaded dash-dotted ellipse. For the model corresponding to node η_i, the regressor indicator vector is $\boldsymbol{\varphi}_{s,i}$, and the performance is denoted by J_{η_i}. The depth of the node η_i is denoted by d_i. For the given tree $\max(d_i) = 2$.

5

adding more nodes into the tree by expanding a node that fulfills a specified criterion, e.g., Akaike's information criterion, which penalizes models that are too complex.

The simple tree $\mathcal{T}_e$ in Figure 5.1 visually illustrates the main terms used for search trees. Assume that a model with two out of three regressors is searched, namely $\boldsymbol{\phi} = \begin{bmatrix}\phi_1 & \phi_2 & \phi_3\end{bmatrix}$. The *root* of the tree is labeled by η_1 in the figure, and it has *depth* $d_1 = 0$. The root is not associated with a set of regressors and, consequently, a model either. Expanding the root gives three *children nodes*: η_2, η_3, and η_4 with depth $d_2 = d_3 = d_4 = 1$. These nodes are also called *leaf nodes* and the root is called the *parent node* of η_2, η_2, and η_3. The nodes η_2, η_3, and η_4 represent models with a single regressor indicated by the *regressor indicator vectors* $\boldsymbol{\varphi}_{s,2}$, $\boldsymbol{\varphi}_{s,3}$, and $\boldsymbol{\varphi}_{s,4}$, respectively. The value of the *cost functions* associated with the model represented by node η_2, η_3, η_4 are respectively J_{η_2}, J_{η_3}, J_{η_4}. Expanding node η_4 results in two children nodes: η_5 and η_6, which are associated with models with two regressor indicator sets $\boldsymbol{\varphi}_{s,5}$ and $\boldsymbol{\varphi}_{s,6}$ and the cost function value J_{η_5} and J_{η_6}. *Terminal nodes* are nodes that cannot be expanded anymore, e.g., η_5 and η_6, as the maximum depth of the tree is 2.

The terms can be described in detail as follows:

Regressor set $\boldsymbol{\varphi}$: A set of regressors from which a subset of it is sought for inclusion in the model.

Node $\eta_i, i \in \mathbb{N}$: A node represents a model that uses a subset of regressors from $\boldsymbol{\varphi}$. The node η_i has the following attributes:

- **Regressor subset $\boldsymbol{v}_{s,i}$:** The subset of regressors that are used in the model represented by η_i.
- **Indicator vector $\boldsymbol{\varphi}_{s,i}$:** A binary vector that indicates which regressors are used in the model represented by node η_i. The vector components are defined in (5.5).
- **Node depth d_i:** The level of node η_i in the tree. The level also shows the number of regressors used in the model.
- **Performance indicator J_{η_i}:** The performance of a model that corresponds to the node η_i. As the model of η_i corresponds to indicator vector $\boldsymbol{\varphi}_{s,i}$ in (5.5), hence the performance of the model corresponds to the cost function defined in (5.7).

As the node η_i represents a model that uses regressors indicated by $\boldsymbol{\varphi}_{s,i}$, one can refer the model to the model of node η_i.

Leaf nodes $\mathcal{L}$: The set of nodes that have no children.

Terminal leaf nodes $\mathcal{L}_T$: A set of leaf nodes that cannot be expanded to have children because they are at the maximum depth allowed in the tree (introduced to limit the number of parameters in the model).

Root η_1: The initial node in the tree with $d_1 = 0$ that does not represent a model, because the root is associated to the empty regressor subset.

Parent nodes: Nodes that have been expanded to have nodes at lower depths during the search. Nodes, except the root, are expanded based on the performance of the node.

Children nodes $\mathcal{C}_i$: The set of nodes that are built from the expansion of parent node η_i. The children of node η_i inherit regressors from their parent, η_i, with an additional regressor that is different from one child to another. Children nodes are connected to their parents by arcs.

Node expansion: Adding children nodes to nodes. Nodes cannot be expanded if they have been expanded previously or if they are terminal nodes.

The rules for expanding nodes to grow the tree are presented in the next section.

5.4.2. SEARCH TREE EXPANSION

In this section, it is shown how to select a leaf to expand. This is the main feature of the algorithm, because it is related to how to develop the tree such that it consists of nodes that correspond to high-performance models.

The search space for the optimization of the cost function (5.7) consists of all possible combinations of $\boldsymbol{\varphi}_s$ having at most n_d regressors in the model. At a certain depth, the number of regressors will be the same for all nodes. According to the search tree expansion rule, the tree grows to a maximum depth of n_d.

In principle, developing a search tree consists of the following two steps:

1. Select leaf nodes
2. Expand the leaf nodes

The second step is trivial; adding an available regressor and computing the performance associated with the nodes. The key to obtaining a good search is the first step in selecting leaf nodes to expand. The first step guides the search for a set of high-performance regressors. The search space related to the first step is to optimize the cost function (5.7) consisting of all possible combinations of $\boldsymbol{\varphi}_s$ with at most n_d regressors in the model.

Given the number of regressors n_r and the desired number of regressors in model n_d, the search tree expansion can be described as follows:

1. Initially, the root node η_1 of the tree is added. This node corresponds to the regressor combination $\boldsymbol{v}_{s,1} = \emptyset$. In other words, no model corresponds to the root node.
2. As the only node in the tree, the root η_1 is directly expanded to have child nodes η_i, $I \in \mathcal{C}_1$ with depth $d_i = 1$, and their performance is computed.
3. Mark the children nodes as leaf nodes.

The following steps are repeated until the stopping criterion is met:

4. Find all non-terminal leaf nodes at depth $d = \delta_{max}$ where $\delta_{max} < n_d$ is the current maximum depth of the tree. If no non-terminal leaf is found, then search at a lower depth, $d = \delta_{max} - 1$. The search is repeated until $d = 2$.
5. Select the node that has the lowest cost value. This node is the one that will be expanded.
6. Expand the node found in the previous step to obtain child nodes. Compute the cost function associated with the child nodes. Mark the child nodes as leaf nodes if they are at depth $d < n_d$; otherwise, mark them as terminal leaf nodes.

The above steps describe the search tree expansion method, which is equivalent to greedy search. The greedy search is achieved by constraining the expansion of only one node to the maximum depth $d = \delta_{\max}$. Within this constraint, the computational cost is the lowest, and a shortlist of models is created. This may result in a regressor combination with sub-optimal performance, similar to the proposed method.

However, there are variations in the proposed algorithm, which is mainly based on the rule of node expansion. Consider a case in which no limitation is imposed on node expansion. In other words, the nodes are expanded to allow all possible regressor combinations at each depth to have a tree to the maximum depth of n_{d}. Expanding all nodes at the current maximum depth $d = \delta_{\max}$ implies creating all possible models that use $\delta_{\max}$ regressors. This is equivalent to an exhaustive search to determine the best model from 1 to $\delta_{\max}$ regressors.

Users can consider multiple node expansions at a certain depth, as shown in Example 2 of Section 5.4.3 denoted by n_{e}. A longer list of possible models is available in this case than in the second case. This multiple node expansion avoids greedy behavior but should also be limited to avoid exhaustive search. In other words, users can define their own rules of searching. There are two search directions for a given n_{d}:

- Vertical search, that is, each expansion repetition increases the depth of the tree as long as the depth of the parent node is less than n_{d}.
- Horizontal search, that is, expansion is repeated to leaf nodes at a certain depth, but without limitation only to those at depth $n_{\mathrm{d}} - 1$.

In addition, the search may be performed by combining both directions. In general, there is no single rule for developing a search tree that fits the regressor search problems. Experience or prior knowledge of the problem may help in designing rules for search tree development.

The search time constraint can hypothetically be associated with the computation resource allocated to build the tree, i.e., the number of computation units allocated to construct the tree. Furthermore, different depths of the nodes are related to different complexities of the models. Consequently, nodes at various depths require different computation efforts.

The proposed method is developed based on the observation that adding a regressor into a model will lower the cost function value. However, the cost function reduction effect differs from one regressor to the other. Adding a correct regressor will lower the cost function more significantly. The purpose of building the search tree is to direct the search to find the best-performing node.

One can see the characteristics of the proposed method to solve a MINLP optimization problem as follows. Each step of tree building incrementally creates new combinations of regressors, and at the same time, their performance is evaluated and used to determine the next formation of new combinations with more or less regressors in the combination. Note that the best-performing regressor combination is not always obtained at the end of the search because the combination is always created until a stopping criterion is met or the tree cannot be grown anymore.

The algorithm of the tree expansion is shown in Algorithm 1. In Algorithm 1, we define the following function and variables:

$f_{\mathrm{d}}(\cdot)$**:** a function that returns the depths of an ordered set of nodes. As an example, $f_{\mathrm{d}}(\mathscr{L}) = \{2,2,2,3,3,3\}$ for $\mathscr{L} = \{2,4,5,6,7,8\}$ as in Figure 5.2b

$f_{\mathrm{i}}(\cdot)$**:** a function that returns the regressor indices from the indicator vector

$\mathscr{L}_d$**:** the set of leaf nodes at depth d

$\mathscr{L}_{\mathrm{E}}$**:** the set of n_{e} leaf nodes that will be expanded

q_d^***:** the index of node that has the least J_{AIC}, the value of Akaike information criterion, at depth d

In order to have a model with n_{d} regressors, it is clear that nodes at depth $d = n_{\mathrm{d}}$ should be marked as terminal leaf nodes. This way, no model with more than n_{d} regressors is built. The

algorithm returns regressors that deliver the least J_{AIC} of models with 2 to n_{d} parameters.

Algorithm 1 Search-tree-based regressor selection for nonlinear systems

Input: $\boldsymbol{y}$, $\boldsymbol{\Phi}$, n_{d}, model class, J_η, $f_{\text{d}}(\cdot)$, $\mathscr{L} = \mathscr{L}_{\text{T}} = \emptyset$, n_{e}, Stopping criterion
  Stop ← **false**
  Initialize tree: $\mathscr{T} \leftarrow \{\eta_1\}$
  $\mathscr{C}_1 \leftarrow$ expand η_1
  $J_{\eta_i} \leftarrow$ compute the J_{AIC} of node $\eta_i, i \in \mathscr{C}_1$
  $\mathscr{L} \leftarrow \mathscr{L} \cup \mathscr{C}_1$
  {Expand tree}
  while Stop = **false do**
    $\delta_{\max} \leftarrow \max\big(f_{\text{d}}(\mathscr{L})\big)$
    if $0 < \delta_{\max} < n_{\text{d}}$ **or** Stop = **false then**
      if $\mathscr{L}_{\delta_{\max}} \neq \emptyset$ **then**
        $\mathscr{L}_{\text{E}} \leftarrow$ nodes $\eta_i, i \in \mathscr{L}_{\delta_{\max}}$ that have the lowest J_{AIC}, $|\mathscr{L}_{\text{E}}| = n_{\text{e}}$
        $\mathscr{C}_i \leftarrow$ expand $\eta_i, i \in \mathscr{L}_{\text{E}}$
        $J_{\eta_i} \leftarrow$ compute the J_{AIC} of node $\eta_i, i \in \mathscr{C}_i$
        if $f_{\text{d}}(\mathscr{C}_i) < n_{\text{d}}$ **then**
          $\mathscr{L} \leftarrow \mathscr{L} \cup \mathscr{C}_i$
        else
          $\mathscr{L}_{\text{T}} \leftarrow \mathscr{L}_{\text{T}} \cup \mathscr{C}_i$
        end if
        $\mathscr{L} \leftarrow \mathscr{L} \setminus \mathscr{L}_{\text{E}}$
        $\mathscr{L}_{\text{T}} \leftarrow \mathscr{L}_{\text{T}} \cup \mathscr{L}_{\text{E}}$
      else
        $\delta_{\max} \leftarrow \delta_{\max} - 1$
        if $\delta_{\max} = 0$ **then**
          Stop ← **true**
        end if
      end if
      if Stopping criterion fulfilled **then**
        Stop ← **true**
      end if
    end if
  end while
  {Select the best performing node for each depth d, from $d = 2$}
  $\mathscr{L}_{\text{A}} \leftarrow \mathscr{L} \cup \mathscr{L}_{\text{T}}$
  for $d = 2$ **to** n_{d} **do**
    $q_d^* \leftarrow$ node that has the least J_{AIC} at depth d
    $\boldsymbol{v}_{\text{s},q_d^*} \leftarrow f_{\text{i}}\big(\boldsymbol{\varphi}_{\text{s},q_d^*}\big)$
  end for
Output: $\boldsymbol{v}_{\text{s},q_d^*}, \quad d = 2, \ldots n_{\text{d}}$

5.4.3. SEARCH TREE FOR REGRESSOR SELECTION

In this section, the rule for node expansion is presented by two examples to build a tree $\mathcal{T}$ for a set of four regressors, where the discrete-time indicator k is omitted to simplify the notation:

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_1 & \phi_2 & \phi_3 & \phi_4 \end{bmatrix}$$

In the examples, the goal is to find the best performance model with no more than three regressors, $n_{\mathrm{d}} = 3$.

EXAMPLE 1

The first example shows the proposed algorithm presented step-by-step.

Step 1. Initially, the tree $\mathcal{T}$ has only the root η_1. The attributes of the root are:

- Depth: $d_1 = 0$
- Regressors: $\boldsymbol{v}_{\mathrm{s},1} = \emptyset$

As the only node in the tree, the root is then expanded. The root now currently has four children nodes $\eta_i, i \in \mathscr{C}_1 = \{1,2,3,4\}$.

The depth of the children is $d_i = 1, i \in \mathscr{C}_1$

The models $\eta_i, i \in \mathscr{C}_1$ have one regressor

$\boldsymbol{v}_{\mathrm{s},2} = \{\phi_1\}$ $\boldsymbol{v}_{\mathrm{s},4} = \{\phi_3\}$
$\boldsymbol{v}_{\mathrm{s},3} = \{\phi_2\}$ $\boldsymbol{v}_{\mathrm{s},5} = \{\phi_4\}$

The tree is illustrated in Figure 5.2a.

The tree has four leaf nodes at depth $d_i = 1, i \in \mathscr{L}_1 = \{2,3,4,5\}$. The performance of the leaf nodes, $J_{\eta_i}, i \in \mathscr{L}_1$, is then evaluated, and the values are sorted in ascending order. For instance, the order of performance after sorting is $J_{\eta_3} \leq J_{\eta_5} \leq J_{\eta_4} \leq J_{\eta_2}$. As J_{η_3} has the lowest value, node η_3 is expanded.

Step 2. Node η_3 has three children nodes $\eta_i, i \in \mathscr{C}_3 = \{6,7,8\}$.

The depth of the children is $d_i = 2, i \in \mathscr{C}_3$.

The models of the node $\eta_i, i \in \mathscr{C}_3$ have two regressors:

$\boldsymbol{v}_{\mathrm{s},6} = \{\phi_1, \phi_2\}$ $\boldsymbol{v}_{\mathrm{s},8} = \{\phi_2, \phi_3\}$
$\boldsymbol{v}_{\mathrm{s},7} = \{\phi_2, \phi_4\}$

Until this point, the tree is shown in Figure 5.2b.

The tree has now three leaf nodes at depth $d_i = 2, i \in \mathscr{L}_2 = \{6,7,8\}$. After evaluating the leaf nodes $\mathscr{L}_2$, assume the ascending performance values are $J_{\eta_8} \leq J_{\eta_6} \leq J_{\eta_7}$. This means that node η_8 is expanded.

Step 3. Node η_8 has two children nodes $\eta_i, i \in \mathscr{C}_8 = \{9,10\}$.

The depth of the children is $d_i = 3, i \in \mathscr{C}_8$.

The models of the nodes $\eta_i, i \in \mathscr{C}_8$ have three regressors:

$\boldsymbol{v}_{\mathrm{s},9} = \{\phi_1, \phi_2, \phi_4\}$ $\boldsymbol{v}_{\mathrm{s},10} = \{\phi_2, \phi_3, \phi_4\}$

The tree at this step is illustrated in Figure 5.2c.

The regressors selected for the model with three parameters are based on the sorted performance of the nodes η_i at depth $d_i = 3$. Assuming $J_{\eta_9} \leq J_{\eta_{10}}$, the regressor combination that should be selected is $\boldsymbol{v}_{\mathrm{s},9} = \{\phi_1, \phi_2, \phi_4\}$.

One can also find the best-performing model with one parameter from the nodes at $d_i = 1$, resulting in $\boldsymbol{v}_{\mathrm{s},3} = \{\phi_2\}$, and with two parameters from nodes at $d_i = 2$, $\boldsymbol{v}_{\mathrm{s},8} = \{\phi_2, \phi_4\}$.

In the algorithm, a mechanism is included to avoid two or more nodes representing the same model, i.e., node duplication. From the example above, one may see that the algorithm is similar to stepwise regression by using tree representation.

The second example illustrates the ability of the proposed algorithm beyond the stepwise regression.

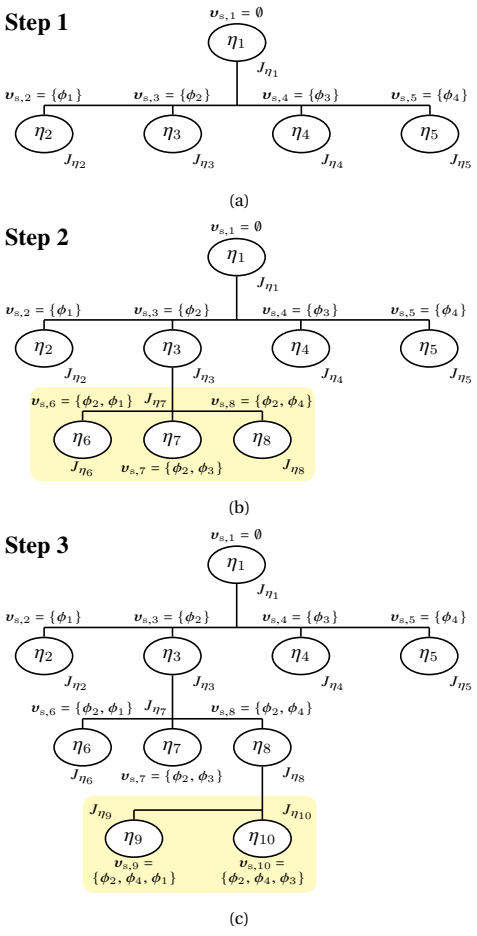


Figure 5.2: Illustration of a search tree to find a model from four regressors given that only one node is expanded at each depth (resembling stepwise regression). In (a), the selected node is the root, and no regressor is selected. The root is expanded to have four children, and the corresponding cost function values $J_{\eta_i}, i = 1,\ldots,4$ are computed. Assuming model η_3 has the smallest performance value at depth 1, and it is expanded, the resulting tree structure is shown in (b). In (b), the yellow shaded nodes are the children of η_3 with depth 2. After computing the performance of the children, the best model is expanded. Assume that model η_8 is the one with the smallest performance value. Expanding node η_8 gives the tree shown in (c).

EXAMPLE 2

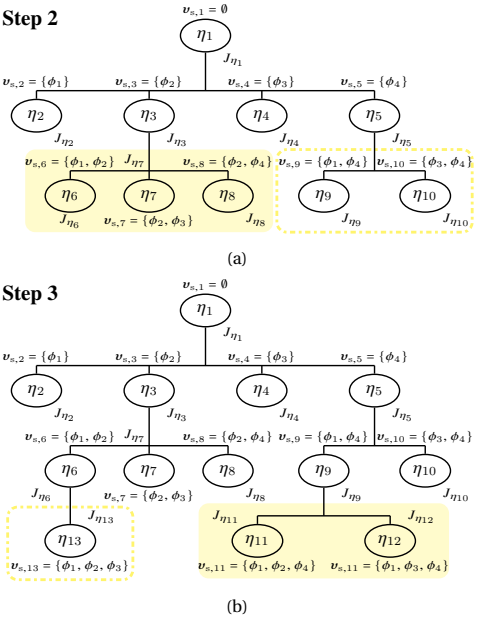


Figure 5.3: Illustration of a search tree to find a model from four regressors, allowing expansion of two nodes with the smallest performance values at each depth. Step 1 is omitted because it is the same as in Figure 5.2a. In (a) assuming the model nodes η_3 and η_5 have the best performance values, and both are expanded. In (b), both nodes η_9 and η_6 have the best performance values, and they are expanded. However, expanding node η_6 results in only one child because the other child is the same as that of the node η_9

.

The search can be performed by allowing a number of simultaneous node expansions denoted by n_e. This number is the key parameter of the proposed method. The second example shows a variation with $n_e = 2$ to show the flexibility of the proposed algorithm. The following are the steps:

Step 1. This step is the same as Step 1 in the previous example (see Figure 5.2a). The sorted node performance values are: $J_{\eta_3} \leq J_{\eta_5} \leq J_{\eta_4} \leq J_{\eta_2}$. As J_{η_3} and J_{η_5} are the two smallest values, the corresponding nodes are expanded, namely, node η_3 and η_5.

Step 2. The children of the node η_3 are the same as in Step 2 of the previous example, i.e., $\eta_i, i \in \mathscr{C}_3 = \{6,7,8\}$. Node η_5 now has two children nodes $\eta_i, i \in \mathscr{C}_5 = \{9,10\}$.

The depth of the children is $d_i = 2, i \in \mathscr{C}_5$.

The models of the nodes $\eta_i, i \in \mathscr{C}_5$ have two regressors as well:

$\boldsymbol{\upsilon}_{s,9} = \{\phi_2, \phi_1\}$ $\quad \boldsymbol{\upsilon}_{s,10} = \{\phi_2, \phi_3\}$

The tree is shown in Figure 5.3a.

Sorting the performance values of the leaf nodes at depth $d_i = 2$ gives, e.g., $J_{\eta_9} \leq J_{\eta_6} \leq J_{\eta_8} \leq J_{\eta_7} \leq J_{\eta_{10}}$. Hence, nodes η_9 and η_6 are selected for expansion. However, with node duplication check, node η_6 has only one child as the other child in the same as that of node η_9.

Step 3. Node η_9 has two children nodes $\eta_i, i \in \mathscr{C}_9 = \{11,12\}$.

The depth of the children is $d_i = 3, i \in \mathscr{C}_9$.

The models of the nodes $\eta_i, i \in \mathscr{C}_9$ have three regressors:
$\boldsymbol{v}_{s,11} = \{\phi_1, \phi_2, \phi_4\}$ $\boldsymbol{v}_{s,12} = \{\phi_2, \phi_4, \phi_3\}$
Node η_6 has one child node $\eta_i, i \in \mathscr{C}_6 = \{13\}$.
The depth of the child is $d_i = 3, i \in \mathscr{C}_6$.
The model of the node $\eta_i, i \in \mathscr{C}_6$ has three regressors: $\boldsymbol{v}_{s,12} = \{\phi_1, \phi_2, \phi_3\}$
Assuming $J_{\eta_{11}} < J_{\eta_{13}} < J_{\eta_{12}}$, the node η_{11} that corresponds to the model using the regressors $\boldsymbol{v}_{s,11} = \{\phi_1, \phi_2, \phi_4\}$ is selected.

5.4.4. PERFORMANCE MEASURE: AKAIKE'S INFORMATION CRITERION

To evaluate the performance of a regression model, different measures that consider model complexity have been suggested in the literature [159]. In this chapter, Akaike's information criterion [2] is applied because it is commonly used in system identification. The Akaike's information criterion was derived using the approximation of the Kullback-Leibler (KL) distance based on data only. AIC is one of the measures that can be used to penalize models that are too complex, among other measures, e.g., Bayesian information criterion (BIC). Users may select any other measure that suits their preference, and the proposed method provides flexibility for such selection.

For a model with n_d regressors, Akaike's information criterion is formulated as[2]:

$$J_{\text{AIC}} = -2\ln\left(\mathscr{L}_h(\hat{\boldsymbol{\theta}}|y)\right) + 2n_d$$

where $\mathscr{L}_h(\hat{\boldsymbol{\theta}}|y)$ is the maximized value of the likelihood function for a model where the parameters $\hat{\boldsymbol{\theta}}$ are estimated using observation data y. The criterion uses the number of parameters to penalize models with a large number of parameters.

The value of Akaike's information criterion can be computed using [35]:

$$J_{\text{AIC}} = N\ln\left(\frac{1}{N}\boldsymbol{e}^\top\boldsymbol{e}\right) + 2n_d$$

with N the number of samples, and $\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}}$ the difference between the observed data vector $\boldsymbol{y}$ and the estimated data vector $\hat{\boldsymbol{y}}$. If the ratio N/n_d is small (i.e., less than 40), Akaike's information criterion gives strongly biased estimates and, therefore, a modified Akaike's information criterion, AIC_c criterion should be used to increase the penalty of selecting a model with more parameters [119]. The modified criterion in case of the small N/n_d is given by [119]:

$$J_{\text{AIC}} = J_{\text{AIC}} + \frac{2n_d(n_d+1)}{N-n_d-1}$$

For TS fuzzy models, the number of parameters can be computed as follows [241]. A TS fuzzy model consists of R rules, where each rule has two parts: antecedent and consequent. In the antecedent part, besides the number of antecedent variables, the number of parameters also depends on the membership function used. In the case of Gaussian membership functions, as used in this chapter, the parameters are the mean and the standard deviation. For a model with n_d regressors, each rule has $2n_d$ antecedent parameters and $n_d + 1$ consequent parameters. The total number of parameters is then $R(3n_d + 1)$. It can be seen that a TS fuzzy model is a complex model from the number of parameters in the model.

[2]Note that in the literature, the criterion is often written using log based on the natural number e. Here the notation ln is used in order to avoid confusion, where $\ln = \log_e$.

5.5. EVALUATION EXAMPLES

The effectiveness of the algorithm is evaluated using examples with both synthetic and real-life application data. The synthetic data are from the Hénon map [104], which is a chaotic time series. The real-life application data sets are from a heat transfer process [122] and a fermenter laboratory setup [222]. All experiments were performed on a Personal Computer with an Intel® i7-9700T processor and 16GB RAM.

To evaluate the proposed method, important regressors are sought for NARX models with the following structures:

1. Linear-in-the-parameter model.
2. TS fuzzy model.
3. Artificial neural networks model.
4. Support vector regression model.

Linear-in-the-parameter models are used for all data sets. The models use polynomial regressors with a maximum of second order[3]. For heat transfer data, the other three structures above are also applied. Using common practice in system identification, the data set is divided into two subsets: one subset containing two-thirds of the total samples for identification (or parameter estimation) and one subset containing the remaining data for validation.

For the proposed method, regressor selection is performed based on the performance of the model in predicting the validation data. Two prediction approaches are used to compute the model performance:

1. One-step ahead prediction.
2. Free-run simulation prediction. This prediction only uses the data for the initial conditions. The next output predictions are computed using the input and only the previous output predictions without using any output from the data set.

The proposed method is mainly benchmarked against the exhaustive search, because it allows a straightforward implementation of both prediction methods. Additional benchmarks in the case of linear-in-the-parameter models are stepwise regression and lasso [217] from the Statistics and Machine Learning™ toolbox in MATLAB®. For completeness, the performance of the free-run simulations is also computed for the models obtained by lasso and stepwise regression.

According to the documentation, the stepwise regression function does not allow the use of validation data, meaning that the function selects the best model based only on the identification data. A different case pertains to the lasso function. A K-fold cross-validation method is used. The lasso function can randomly divide the data into K subsets and use $K-1$ subsets as the identification data and one subset as the validation data. In these examples, we set $K = 3$. Hence, the lasso function obtains all data samples without manually dividing the identification and validation data. In addition, the use of lasso here is purely for regressor selection, therefore, the parameters from MATLAB toolbox are not used. Instead, new parameters are computed based on the selected regressors.

To improve the presentation, plots that are not directly related to the experiment results are put in the 5.7 at the end of the chapter. They are the plots of the input-output data and the outputs of models and their corresponding errors.

[3] Polynomial regressors are ones that are formed from the multiplications of lagged input-output data, including self-multiplication and bias.

Table 5.1: Frequency of the selected regressors for the Hénon map example. The incorrect regressors are shaded in pink. The proposed method incrementally expands 6 leaves at depth 2 to construct models with 3 parameters. The selected regressors after the 4^{th}, 5^{th}, and 6^{th} expansion are shown. They correspond to tree sizes of 885, 1056, and 1225, respectively. As a comparison, the exhaustive search has to search from 121485 combinations.

(a) One-step ahead prediction

Selected regressors	Lasso	Stepwise	Exhaustive search	Proposed method		
$y(k-4)$	100	100	100	26	66	100
$y^2(k-2)$	100	100	100	100	100	100
Constant	100	100	100	100	100	100
$y^2(k-12)$	0	0	0	72	34	0
Others	0	74	0	2	0	0

(b) Free-run prediction

Selected regressors	Exhaustive search	Proposed method		
$y(k-4)$	2	2	2	2
$y^2(k-2)$	0	0	0	0
Constant	100	100	100	100
$y^2(k-12)$	0	0	0	0
Others	198	198	198	198

5.5.1. EXAMPLE 1: HÉNON MAP

The Hénon map is a discrete-time dynamical system that exhibits chaotic behavior. The system is defined by [104]:

$$y(k) = 1 - a\big(y(k-2) - e(k-2)\big)^2 + b\big(y(k-4) - e(k-4)\big) \tag{5.8}$$

where $e(k)$ is zero-mean Gaussian random noise. For the experiment, 100 sets of data are generated to allow consistency checks of the selected regressors.

The chaotic behavior is obtained by setting $a = 1.4$ and $b = 0.3$. From the model, 100 data sets of 1000 samples are simulated using the initial condition $y(1) = 0$, $y(2) = 1$, $y(3) = 0.5$, and $y(4) = -0.5$ and uniform random noise $e(k)$ in the range $[-0.01, 0.01]$. The regressor candidates are monomials from $y(k-i), i = 1, \ldots, 12$ and multiplication of two monomials, including self-multiplication and a constant. The total number of regressor candidates is 91.

In this example, only nodes at depth $d < 3$ are expanded to result in models with 3 regressors for the proposed method. For the proposed method, we expand the leaves at depth 2 repeatedly to 6, and for each expansion, we check if the proposed method selects the correct regressors, i.e., the ones that are in (5.8). The repeated expansion is performed because the proposed method mostly selects incorrect regressors for the first 3 repetitions. The results are shown in Tables 5.1 and 5.2. Only after more than 3 repetitions the proposed method starts selecting the correct regressors, and, therefore, their results are included in the tables.

Table 5.1 shows the frequency of the selected regressors for the compared methods. The first column of the table lists the regressors selected by the compared methods, where the incorrect regressors are shaded in pink. The other columns show the number of times a regressor is chosen from 100 experiments with different noise realizations.

For the one-step ahead prediction, the table shows that lasso and exhaustive search select all correct regressors, and stepwise regression frequently selects more regressors in addition to the

Table 5.2: Mean and standard deviation of the J_{AIC} values and computation time for the Hénon map example

(a) J_{AIC}

Methods		One-step ahead prediction	Free-run prediction
Lasso		-2551.0 ± 18.0	-191.1 ± 14.3
Stepwise		-2857.2 ± 24.1	-22.9 ± 21.9
Exhaustive search		-2863.1 ± 23.5	-225.4 ± 10.8
Proposed method	4	-1543.4 ± 784.2	-225.1 ± 10.8
	5	-2259.6 ± 848.0	-225.1 ± 10.8
	6	-2863.1 ± 23.5	-225.1 ± 10.8

(b) Computation time in seconds

Methods		One-step ahead prediction	Free-run prediction
Lasso		0.015 ± 0.004	–
Stepwise		0.009 ± 0.003	–
Exhaustive search		83.551 ± 1.230	251.746 ± 0.310
Proposed method	4	0.582 ± 0.004	1.742 ± 0.003
	5	0.673 ± 0.002	2.073 ± 0.008
	6	0.764 ± 0.002	2.393 ± 0.005

correct ones. The table also shows the results of the proposed method with 4, 5, and 6 expansions of leaf nodes at depth 2. The expansions correspond to tree sizes 885, 1056, and 1225. It can be seen that after 6 leaf node expansions at depth 2, the proposed method can find all correct regressors.

Table 5.2 shows the mean and the standard variance of the J_{AIC} values for the compared methods and the corresponding computation time to obtain the regressors. For the one-step-ahead prediction approach, the exhaustive search and the proposed method with 6 expansions yield models with the lowest mean and standard deviation of J_{AIC}, followed by stepwise regression, which delivers a model with more regressors and lasso. The lower performance shows that stepwise regression is penalized by selecting more regressors.

The lower performance of lasso comes from the fact that lasso computes the parameters in a different way compared to the least-squares method, resulting in models with lower performance. The proposed method correctly selects $y^2(k-8)$ and the constant in all runs, while the frequency of choosing $y(k-4)$ increases with more expansions of leaves at depth 2. The table also shows that choosing incorrect regressors results in a higher standard deviation of J_{AIC} than when selecting correct regressors.

In case of the free-run prediction approach, the exhaustive search and the proposed method cannot find the correct regressors. While both methods can consistently find the constant, a different regressor combination is yielded for each data set. It is presumed that the chaotic nature of the system results in data that cannot be used for long-term prediction. Consequently, the identified models return a poor long-term prediction performance.

With respect to the computation time for the one-step ahead prediction, the proposed method is much faster than the exhaustive search but slower than lasso and stepwise regression. In the case of free-run prediction, the proposed method is still faster than the exhaustive search.

Table 5.3: Regressor candidate indices of the polynomial model for the heat transfer process example.

Index	Regressor	Index	Regressor	Index	Regressor
1	$y(k-1)$	10	$y(k-1)y(k-3)$	19	$y(k-3)u(k-3)$
2	$y(k-2)$	11	$y(k-2)y(k-3)$	20	$u(k-2)u(k-3)$
3	$y(k-3)$	12	$y^2(k-3)$	21	$u^2(k-3)$
4	$u(k-2)$	13	$y(k-1)u(k-2)$	22	$y(k-1)u(k-4)$
5	$u(k-3)$	14	$y(k-2)u(k-2)$	23	$y(k-2)u(k-4)$
6	$u(k-4)$	15	$y(k-3)u(k-2)$	24	$y(k-3)u(k-4)$
7	$y^2(k-1)$	16	$u^2(k-2)$	25	$u(k-2)u(k-4)$
8	$y(k-1)y(k-2)$	17	$y(k-1)u(k-3)$	26	$u(k-3)u(k-4)$
9	$y^2(k-2)$	18	$y(k-2)u(k-3)$	27	$u^2(k-4)$
				28	1

5.5.2. EXAMPLE 2: HEAT TRANSFER PROCESS

The data set in this example comes from a simple heat transfer experiment [122]. The process consists of a 30 cm tube with a fan, a heating resistor, and a temperature sensor as illustrated in Figure 5.4. The resistor and the fan are located at one end, and the sensor is at the other end. The input is the voltage over the heating resistor $u(k) \in [0\text{V}, 12\text{V}]$ and the output is the voltage that corresponds to the air temperature measurement $y(k) \in [0\text{V}, 10\text{V}]$. The airflow of the fan can be controlled manually by a valve, considered as an independent measurable variable $v(k) \in [0°, 180°]$. The value of $v(k)$ is set constant. A more detailed description and a first-principles model of this process can be found in [122].

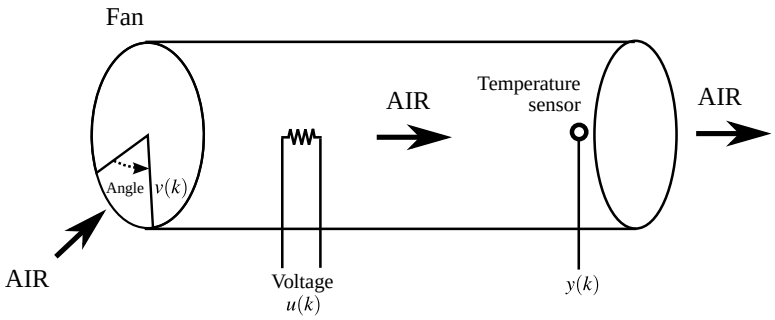


Figure 5.4: Schematic illustration of the heat transfer setup adapted from [122].

The data set contains 1309 samples, from which 873 samples are used for identification. The data is plotted in Figure 5.13. The data was centered to remove the offset before the search. For this data set, monomial regressors with the maximum degree of 2 from $y(k-i)$ and $u\big(k-i-1\big)$, $i = 1,2,3$ are created so that there are 27 regressor candidates. These regressors are indexed and shown in Table 5.3. Regressors of models with 2 to 10 parameters are searched by using exhaustive search, lasso, stepwise regression, and the proposed method. The search tree is grown by expanding three nodes at each level.

The chosen regressors are shown in Table 5.4, using the same format as in the previous examples, and the corresponding J_{AIC} values are demonstrated in Figure 5.5. Comparing the J_{AIC} performance, it can be seen that the proposed method outperforms lasso, and it performs slightly worse than exhaustive search and stepwise regression for the same number of regressors. The

Table 5.4: Indices of the selected regressors for models of the heat transfer data with 2 up to 10 parameters obtained by the exhaustive search, lasso, stepwise regression, and the proposed method with three node expansions in each depth. The dashes indicate that a model with the corresponding number of regressors could not be found.

(a) One-step ahead prediction

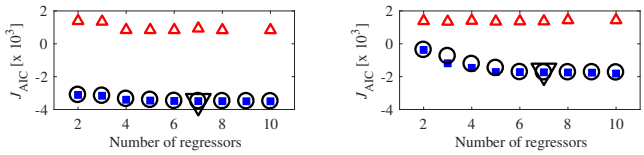
Number of parameters	Lasso	Stepwise	Exhaustive search	Proposed method
2	7, 28	—	1, 2	1, 2
3	7, 21, 28	—	1, 2, 7	1, 2, 7
4	7, 19, 21, 28	—	1, 2, 7, 28	1, 2, 7, 28
5	7, 9, 26, 27, 28	—	1, 2, 6, 7, 28	1, 2, 6, 7, 28
6	7, 9, 21, 26, 27, 28	—	1, 2, 3, 6, 7, 28	1, 2, 3, 6, 7, 28
7	7, 9, 12, 21, 26, 27 28	1, 2, 3, 6, 7, 9 28	1, 2, 3, 6, 7, 10 28	1, 2, 3, 6, 7, 10 28
8	7, 8, 9, 12, 21, 26 27, 28	—	1, 2, 6, 7, 13, 17 22, 28	1, 2, 3, 6, 7, 8 10, 28
9	—	—	1, 2, 6, 7, 10, 13 17, 22, 28	1, 2, 3, 5, 6, 7 8, 10, 28
10	7, 8, 9, 12, 21, 23 24, 26, 27, 28	—	1, 2, 6, 7, 8, 10 13, 17, 22, 28	1, 2, 3, 4, 5, 6 7, 8, 10, 28

(b) Free-run prediction

Number of parameters	Exhaustive search	Proposed method
2	13, 28	13, 28
3	1, 7, 28	1, 13, 28
4	1, 3, 8, 28	1, 7, 13, 28
5	1, 3, 5, 8, 28	1, 3, 7, 13, 28
6	1, 3, 6, 8, 17, 28	1, 3, 6, 7, 13, 28
7	1, 3, 4, 5, 8, 17, 28	1, 3, 4, 6, 7, 13, 28
8	1, 3, 6, 8, 10, 17, 24 28	1, 3, 4, 6, 7, 13, 22 28
9	1, 3, 5, 6, 9, 10, 17 24, 28	1, 3, 4, 6, 7, 12, 13 22, 28
10	1, 4, 5, 6, 8, 9, 14 16, 19, 22	1, 3, 4, 6, 7, 10, 12 13, 22, 28

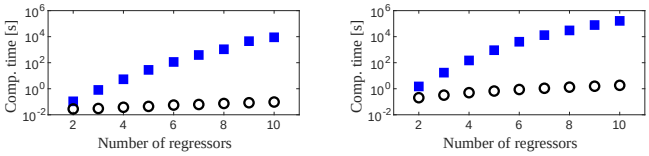
performance difference between lasso and the other methods is relatively large. It can also be seen that for the one-step ahead prediction approach, the proposed method returns models that are close to those obtained by exhaustive search. Note that lasso does not give results for models with 9 parameters. The cause is unknown because no error message is returned. The stepwise regression is only able to return a model with 7 parameters, which performs close to the models returned by exhaustive search.

To visually assess the fit of the models, the one-step ahead predictions and their corresponding errors for the validation data for models with 7 parameters are plotted in Figures 5.14 and 5.15. Predictions for models with 7 parameters have been chosen because the stepwise regression results in a model with 7 parameters only. The predictions from the exhaustive search overlap with those



(a) One-step ahead prediction (b) Free-run prediction

Figure 5.5: Values of J_{AIC} of the heat transfer validation data from models with different numbers of parameters: exhaustive search (blue-filled □), lasso (red △), stepwise regression for 6 parameters (black ▽), and the proposed method (black ∘) for the heat transfer data.



(a) One-step ahead prediction (b) Free-run prediction

Figure 5.6: Computation time, in seconds, for exhaustive search (blue-filled □) and the proposed method (black ∘) for the heat transfer data.

from the stepwise regression because both methods select mostly the same regressors and produce the same models. As Figure 5.5 suggests, the prediction error from the model obtained by lasso is the largest, while the errors from the models obtained by the proposed method, exhaustive search and stepwise regression are close each other. *Remark*: it can be seen that using the one-step ahead prediction, the proposed method and exhaustive search use linear regressors, that is, $y(k-1)$ and $y(k-2)$ in all sets of selected regressors. This may raise an idea of the sufficiency of using linear models instead of nonlinear ones. The discussion about this observation is detailed in the discussion section.

The computation times required for the method proposed in this chapter and the exhaustive search are shown in Figure 5.6. It can be seen that the proposed method is faster than the exhaustive search. As the number of regressors in the model increases, the log computation time of the proposed method increases significantly smaller than that of exhaustive search. Note that lasso and stepwise regression are omitted in Figure 5.6 as the lasso function is implemented to find several models in a run, while the stepwise regression function does not allow to specify the number of regressors.

5.5.3. EXAMPLE 3: LABORATORY FERMENTER

In this example, a laboratory fermenter is considered. The setup is the one used in [222], and it is illustrated in Figure 5.7. The capacity of the fermenter tank is 40 liter, and it is filled with 25 liter water. Air is fed from the bottom of the tank, controlled by a local mass-flow controller. The air pressure in the head space is determined by the position of an outlet valve at the top of the fermenter.

The system has two inputs: the position of the outlet valve and the position of the inlet air flow rate. The output is the pressure in the head space. To obtain a SISO system, which is used in this example, the inlet flow rate is kept constant. The time constant of 45 s allows for a sample period of 5 s. For identification, the system is excited with multi-sinusoidal signals that cover the amplitudes and the frequencies of interest. The amplitude is between 0 and 100 % valve opening,

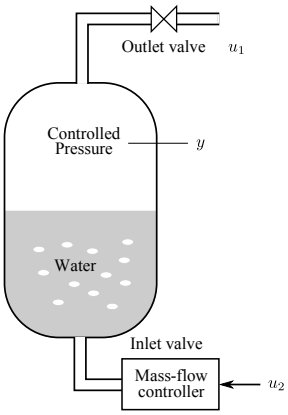


Figure 5.7: Illustration of the laboratory fermenter.

and the frequencies are 1000, 500, 250, and 100 Hz. There are two data sets for identification and also two sets for validation. The identification data sets consist of 757 and 480 samples, and they are plotted in Figure 5.16, while the validation data sets consist of 185 and 281 samples. More details on the setup are given in [222].

In this example, the effectiveness of the proposed method is tested further by using structures that are not linear-in-the-parameters. For that purpose, this section is divided into two parts. In the first part, the same experiments as for Example 2 are performed. In the second part, the proposed method is compared to a TS fuzzy model, an artificial neural networks (ANN) model, and a support vector regression (SVR) model.

PART 1: LINEAR-IN-THE-PARAMETERS

In this part, the regressor candidates are the same as in Example 2 and shown in Table 5.3. For the proposed method, the search tree is grown by expanding three nodes at each depth of the tree. The selected regressors from the one-step ahead prediction and the free-run simulation are shown in Table 5.5 for models consisting of two to ten regressors. As shown in the table, the proposed method selects different regressors compared to the ones of exhaustive search, except for two regressors. For the one-step ahead prediction, the proposed method does not select the constant, the regressor 28. This shows that the short horizon of the one-step-ahead prediction may not be preferable to select regressors for the model. This is different from the free-run simulation experiment, where the constant is consistently chosen by both the exhaustive search and the proposed method. The result from stepwise regression is omitted in Table 5.5 as it selected 15 regressors, which is larger than the desired maximum number of regressors.

It can also be observed that lasso delivers models that have a relatively constant performance for different numbers of regressors based on the one-step ahead prediction. Using the models for the free-run simulation, the performance of lasso shows relatively small differences across different numbers of regressors.

The results in J_{AIC} performance are shown in Figure 5.8. Exhaustive search selects regressors for models with better performance, but this comes at the cost of high computation times, as shown in Figure 5.9. To assess the performance difference with respect to the prediction by the model, the model output and the corresponding error output for the validation data are shown in Figures 5.18 and 5.19. From the figures, it can be seen that the models with regressors selected by the exhaustive search and the proposed method result in relatively small prediction errors.

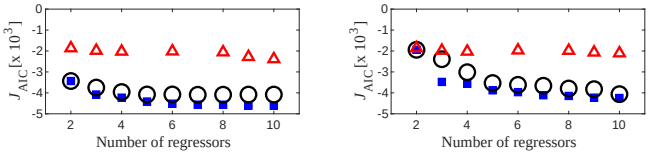
Table 5.5: Indices of the selected regressors for models of the laboratory fermenter data with 2 up to 10 parameters obtained by the exhaustive search, lasso, and the proposed method with three node expansions in each depth. The dashes indicate that a model with the corresponding number of regressors could not be found.

(a) One-step ahead prediction

Number of parameters	Lasso	Exhaustive search	Proposed method
2	7, 28	1, 3	1, 3
3	7, 8, 28	4, 24, 28	1, 12, 13
4	7, 8, 9, 28	1, 4, 24, 28	1, 4, 12, 13
5	–	1, 4, 12, 24, 28	1, 5, 12, 13, 22
6	7, 8, 9, 13, 14, 28	3, 4, 12, 15, 24, 28	1, 3, 5, 12, 13, 22
7	–	3, 4, 12, 15, 16, 24, 28	1, 5, 6, 12, 13, 14, 22
8	7, 8, 9, 11, 13, 14, 15 28	3, 4, 12, 14, 15, 23, 24 28	1, 2, 3, 6, 13, 14, 18 22
9	7, 8, 9, 10, 11, 13, 14 15, 28	3, 4, 12, 14, 15, 16, 23 24, 28	1, 3, 5, 6, 12, 13, 14 18, 22
10	7, 8, 9, 10, 11, 12, 13 14, 15, 28	3, 4, 9, 12, 14, 15, 16 23, 24, 28	1, 2, 3, 5, 6, 12, 13 14, 18, 22

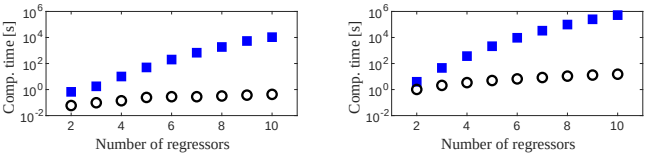
(b) Free-run prediction

Number of parameters	Exhaustive search	Proposed method
2	13, 28	13, 28
3	4, 24, 28	6, 17, 28
4	4, 17, 24, 28	5, 11, 14, 28
5	4, 11, 16, 24, 28	5, 11, 14, 23, 28
6	3, 4, 12, 15, 24, 28	5, 7, 10, 14, 23, 28
7	4, 10, 19, 20, 24, 27, 28	5, 6, 7, 10, 14, 23, 28
8	3, 4, 7, 10, 15, 24, 25 28	4, 5, 7, 9, 10, 14, 23 28
9	4, 8, 10, 14, 19, 21, 24 26, 28	4, 5, 7, 10, 13, 14, 23 24, 28
10	4, 8, 10, 14, 15, 18, 20 24, 25, 28	3, 4, 5, 7, 10, 13, 14 23, 24, 28



(a) One-step ahead prediction (b) Free-run prediction

Figure 5.8: Values of J_{AIC} of the laboratory fermenter validation data from models with different numbers of parameters: exhaustive search (blue-filled □), lasso (red △), and the proposed method (black ∘) for the heat transfer data.



(a) One-step ahead prediction (b) Free-run prediction

Figure 5.9: Computation time for the exhaustive search (blue-filled □) and the proposed method (black ∘) for the laboratory fermenter data.

Table 5.6: Regressor candidates indices for the laboratory fermenter example.

Index	Regressor	Index	Regressor
1	$u(k-1)$	5	$y(k-1)$
2	$u(k-2)$	6	$y(k-2)$
3	$u(k-3)$	7	$y(k-3)$
4	$u(k-4)$	8	$y(k-4)$

5

PART 2: TS FUZZY, NEURAL NETWORK, AND SUPPORT VECTOR REGRESSION MODELS

The regressor candidates in this example are $u(k-i)$ and $y(k-i)$, $i = 1,\ldots,4$ and they are indexed as shown in Table 5.6. In this example, it is intended to find models with two to five regressors.

Different from the previous examples, the proposed method is evaluated for three models with a NARX structure: Takagi-Sugeno (TS) fuzzy models, Neural Networks (NN), and Support Vector Machine for Regression (SVR). In this example, the feature of the proposed method to work with models that are not linear-in-the-parameters is shown. Here, the search by the proposed method is compared to that of the exhaustive search. The other methods considered before cannot be included in the comparison as no result has been found with lasso and forward selection for the TS, NN, and SVR models.

In this example, the TS fuzzy models are constructed by using product space clustering [16], which involves fuzzy clustering with the Gustafson-Kessel algorithm [91] followed by consequent parameter estimation. For the clustering, the fuzziness index of the clusters is set to 2. This value is typical for system identification [16]. A larger fuzziness index means more fuzzier or overlapping clusters. The TS fuzzy models have 3 rules, where the antecedent membership functions are obtained by projection and the degree of fulfillment of the antecedent is computed by using the product of the individual components. The consequent parameters are computed by using the least-squares method. The NN models use a hidden layer with 20 neurons, a hyperbolic tangent activation function, and the Levenberg–Marquardt algorithm for training. The SVR models are computed using the `fitrsmv` of the Statistics and Machine Learning™ toolbox in MATLAB®.

Table 5.7: Selected regressors for the TS fuzzy model of the laboratory fermenter data.

	One-step ahead prediction		Free-run prediction	
Number of regressors	Exhaustive search	Proposed method	Exhaustive search	Proposed method
2	1, 5	1, 5	2, 5	1, 5
3	1, 5, 6	1, 5, 6	1, 3, 5	1, 3, 5
4	1, 5, 6, 7	1, 5, 6, 7	1, 3, 5, 7	1, 3, 5, 7
5	1, 2, 5, 6, 8	1, 2, 5, 6, 7	1, 3, 5, 7, 8	1, 3, 5, 7, 8

Table 5.8: Selected regressors for the neural network model of the laboratory fermenter data.

	One-step ahead prediction		Free-run prediction	
Number of regressors	Exhaustive search	Proposed method	Exhaustive search	Proposed method
2	1, 5	1, 5	2, 6	1, 8
3	1, 5, 6	1, 5, 6	1, 5, 8	1, 5, 8
4	1, 5, 6, 7	1, 5, 6, 7	1, 5, 6, 8	1, 5, 6, 8
5	1, 2, 5, 6, 7	1, 2, 5, 6, 7	1, 5, 6, 7, 8	1, 5, 6, 7, 8

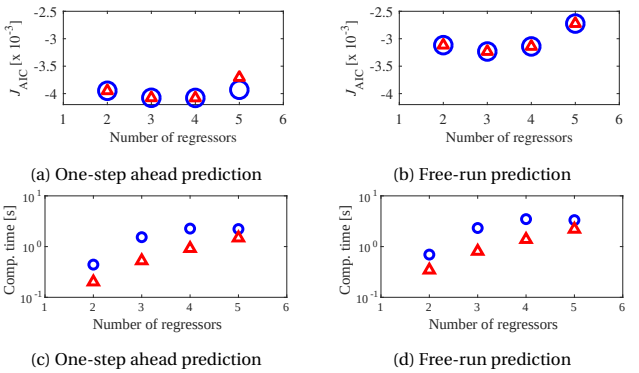
Table 5.9: Selected regressors for the support vector regression model of the laboratory fermenter data.

	One-step ahead prediction		Free-run prediction	
Number of regressors	Exhaustive search	Proposed method	Exhaustive search	Proposed method
2	1, 3	1, 3	5, 6	5, 6
3	1, 3, 4	1, 3, 4	2, 4, 7	5, 6, 8
4	1, 2, 3, 4	1, 2, 3, 4	2, 4, 7, 8	5, 6, 7, 8
5	1, 2, 3, 4, 6	1, 2, 3, 4, 6	1, 3, 4, 7, 8	2, 4, 5, 7, 8

Table 5.10: Number of models evaluated to find regressors of the highest performing models.

Number of regressors	Exhaustive search	Proposed method
2	28	16
3	56	28
4	70	43
5	56	60

The results of the regressor selection are shown in Tables 5.7, 5.8, and 5.9 for, respectively, the TS models, NN models, and SVR models. For the TS and NN models, it can be seen that the proposed method selects the same regressor sets as the exhaustive search. There are only two cases out of 16 that the proposed method give different results from the exhaustive search, i.e., the 5-regressor model based on the one-step ahead prediction and the 2-regressor model based on the free-run simulation. For the SVR models and the search based on the one-step ahead prediction, the proposed method results in the same selected regressors as the exhaustive search. However, the results are different for the search based on the free-run simulation. The proposed method delivers a different regressor selection for models with more than two regressors, but the performance of the model with three regressors is not significantly different from the one obtained from the exhaustive search.



(a) One-step ahead prediction (b) Free-run prediction

(c) One-step ahead prediction (d) Free-run prediction

Figure 5.10: Values of J_{AIC} (a and b) and the computation time (c and d) for the laboratory fermenter validation data prediction for models with different numbers of parameters: exhaustive search (blue ∘) and proposed method (red △) for the TS fuzzy models

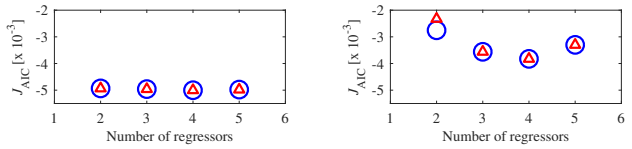
The performance of models using the selected regressors and the corresponding computation time to find the regressors are shown in Figure 5.10 for the TS fuzzy models, Figure 5.11 for the NN models, and Figure 5.12 for the SVR models. It can be seen that the performance of models selected by the proposed method is generally comparable to the performance of models from the exhaustive search for both types of prediction in the case of TS and NN models. Indeed, from Figures 5.10a and 5.11b, there are two cases where the proposed method delivers lower performance models compared to those obtained from the exhaustive search. For the SVR models, the J_{AIC} performance of models obtained by the proposed method with the free-run simulation search is generally lower than the performance of models obtained by the exhaustive search. This can also be seen based on the different regressor selections shown in Table 5.9 for models with more than 2 regressors.

Table 5.10 shows the number of models evaluated to find the regressors of the highest-performing models. It can be shown that the exhaustive search evaluates fewer models than the proposed method to find the best-performing models with 5 regressors.

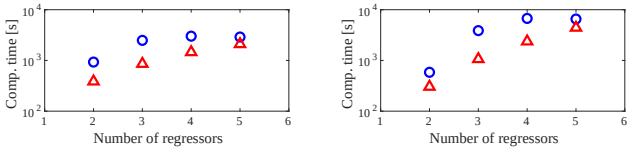
With respect to computation load, the proposed method has a computation load advantage over the exhaustive search until searching models with 4 regressors, as shown in Figures 5.10c and 5.10d for the search for TS fuzzy models, in Figures 5.11c and 5.10d for the NN models, and in Figures 5.12c and 5.12d for the SVR models. This is visible for a small number of candidates, as in this example. The computation advantage begins to decrease as soon as the number of parameters is one-half of the number of regressor candidates. Note that for the exhaustive search in this example, there are 70 candidates for the model with 4 regressors and 56 candidates for the

model with 5 regressors.

To see the difference in performance due to the different regressor selections, the free-run prediction of the validation data and the corresponding error of the models are plotted. For the 2-regressor TS fuzzy models, these are shown in Figure 5.20, for the 2-regressor NN models, they are in Figure 5.21, and for 3-regressor SVR models, they are in Figure 5.22. The prediction error from the TS fuzzy model is the smallest, while from the NN model is larger, and from the SVR model is the largest.

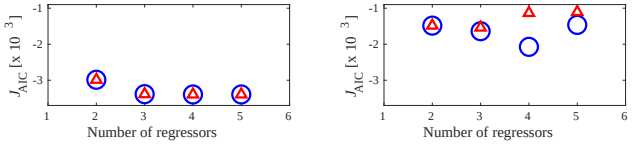


(a) One-step ahead prediction (b) Free-run prediction

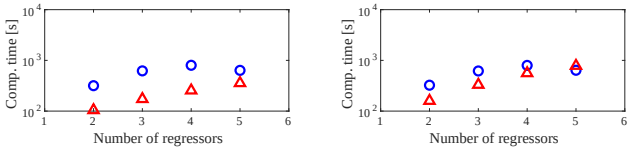


(c) One-step ahead prediction (d) Free-run prediction

Figure 5.11: Values of J_{AIC} (a and b) and the computation time (c and d) for the laboratory fermenter validation data prediction for models with different numbers of parameters: exhaustive search (blue ∘) and proposed method (red △) for the NN models



(a) One-step ahead prediction (b) Free-run prediction



(c) One-step ahead prediction (d) Free-run prediction

Figure 5.12: Values of J_{AIC} (a and b) and the computation time (c and d) for the laboratory fermenter validation data prediction for models with different numbers of parameters: exhaustive search (blue ∘) and proposed method (red △) for the SV models

5.5.4. DISCUSSION

From all data sets with different prediction approaches, searching from all possible regressor combinations is the best way to find the set of important regressors for a given model structure as demonstrated in the previous section. The disadvantage of exhaustive search is the high computation load to find the best model from all possible combinations. The enormous increase of the

number of combinations as the number of regressors increases restrains the use of this method in practical real-life applications.

LINEAR-IN-THE-PARAMETER STRUCTURE

In the case of using the one-step-ahead prediction selection, the performance of the proposed method is generally better than that of lasso or stepwise regression with respect to selecting regressors that form models with good performance. This is demonstrated in Example 1 and Example 2. The price for good performance and flexibility is the longer computation time required to examine regressor combination candidates in the tree. However, the computation time is lower than the exhaustive search.

For the Hénon map example, the chaotic characteristic of the system that generates the data, forces the proposed method to build more regressor combinations to include the correct combination. In this case, there exist local minima in the search space, whereas the global minimum is not known because it is not yet on the search tree. Expanding the search space by growing the search tree for several expansion rounds yields a correct combination.

However, lasso can find the correct regressor for the Hénon map example in every run. This can be explained by using the least-angle regression, an interpretation of lasso based on the correlation analysis between the regressors and the residuals [77]. This method uses the fact that the smallest angle between the regressors and the corresponding residuals is related to the largest correlation between the regressors and residuals. The method works by selecting regressors with the highest correlation with the residuals, determined by the least angle between the regressors and residuals. The coefficients of the selected regressors are moved; that is, they increase or decrease in the direction that preserves the angle. If another regressor is correlated with the residuals, it is added to the selected regressor group. If the coefficient of a regressor becomes zero, then the regressor is removed from the selected group. These steps are repeated until the number of regressors is reached or the coefficients no longer change.

With the correlation approach, the lasso works well with the Hénon map example. However, it does not work equally well for data from dynamic systems in other examples. The lower J_{AIC} performance in Examples 2 and 3, suggests that lasso might not be suitable for system identification, especially if the system involves external inputs as in the examples here. This might be because the correlations between external inputs and outputs are not as strong as those between outputs. However, further experiments are required to draw a definitive conclusion. Based on J_{AIC} performance, the performance of the lasso does not change significantly with different numbers of regressors and prediction horizons. This is not in line with the common intuition that more regressors will result in better performance and that a larger prediction horizon will yield worse performance. However, this may require further investigation.

The other method used in the chapter is the stepwise regression. With regard to performance, stepwise regression selects regressors that provide models with a good performance that is comparable to that obtained with the exhaustive search. However, the results from the synthetic data show that stepwise regression always selects more regressors than the correct ones. Another problem is that the stepwise regression based on the F statistics test does not provide the freedom to determine the number of regressors. Hence, this method is fully automatic.

Using orthogonal regressors as in, e.g., [24] is also of interest for this particular structure class. The orthogonalization of regressors requires additional computational efforts; however, it simplifies the search process. As reported by Billings and Voon [24], greedy search is typically sufficient to obtain a set of high-performance regressor sets.

OTHER NONLINEAR STRUCTURES

The proposed method can also be used for models with structures other than those with the linear-in-the-parameters. In this case, the proposed method is comparable only to exhaustive search as

illustrated in Example 3. In this example, the number of regressors is smaller than that of the linear-in-the-parameter; therefore, the number of combinations is also smaller. However, the computation time depends on the selected structure. In general, the ANN structure with back-propagation takes more time than the TS fuzzy and SVR models.

From the second part of Example 3, it can be observed that the proposed method delivers regressors that are mostly equal to the exhaustive search for all TS fuzzy, NN, and SVR models for the search based on one-step-ahead prediction. This shows that the proposed method works independently of the model structure in this particular case.

Structures other than the linear-in-the-parameters provide a lower number of regressors to select. This simplifies the search using a smaller number of possible regressor combinations. Using the exhaustive search is likely to be possible for SISO nonlinear systems. In the second part of Example 3, the exhaustive search is compared with the greedy search. In this case, the advantage of the proposed method is no longer significant.

In these examples, the nodes in the proposed method represent sets of regressors with a NARX structure. However, other structures can be used, such as nonlinear output error structure or nonlinear Autoregressive moving-average structure. The decision on which model structure is selected to compute the performance of the node can be selected freely by the user. This flexibility is possible using a combinatorial method of selecting important regressors.

PREDICTION METHODS

Comparing the results obtained by the one-step-ahead prediction approach with those obtained by the free-run prediction approach reveals that the first prediction approach is preferred in experiments where there are high chances to include in the set of regressors the true ones (leading to the true model). Free-run prediction is favored for real-life application data where there is no guarantee to find the true regressors, and also for applications that require long-term prediction as in optimal control.

From Examples 2 and 3, the proposed method performs less well in terms of model output accuracy compared to the exhaustive search when using the free-run simulation prediction. However, the proposed method is better based on the computational load. So the proposed method is preferred for applications with limited computational resources owing to its low computation requirement.

Also from Example 2, the proposed method and exhaustive search select $y(k-1)$ and $y(k-2)$ in all selected regressor sets when using the one-step ahead prediction. Using this prediction method, the current prediction is computed using known input and past output measurements. In other words, the predictions are linear combinations of input and output measurements. Therefore, the linear regressors, that is $y(k-1)$ and $y(k-2)$, are dominant in the model; whereas nonlinear regressors are less important for fitting the model to the data. On the other hand, using free-run simulation shows that the nonlinear regressors are more influential than the linear ones because the current prediction depends on previous predictions instead of measurements.

This shows that using real-application data, a model whose parameters are estimated using free-run simulation is closer to the true unknown model than that using one-step-ahead prediction. A sufficiently good prediction accuracy of the model from the one-step-ahead prediction is obtained when using past measurements that can be seen as information from the true unknown model.

From the second part of Example 3, it can be observed that the proposed method delivers regressors that are mostly equal to the ones obtained using the exhaustive search for all TS fuzzy, NN, and SVR models for the search based on one-step-ahead prediction. This shows that the proposed method works independently of the model structure in this particular case.

Using the free-run simulation prediction in the search, the resulting SVR models are less accurate than the TS and NN models. This is due to the fact that the parameter computations of the SVR models use one-step-ahead prediction while the node expansions use the free-run prediction.

Because the accuracy of predictions does not depend on past measurements, in general, models obtained using free-run simulation are better than those obtained using one-step-ahead prediction. The models obtained by the free-run simulation delivered sufficiently accurate predictions without using past output measurements. This allows the models to be used in other applications such as soft sensors [155].

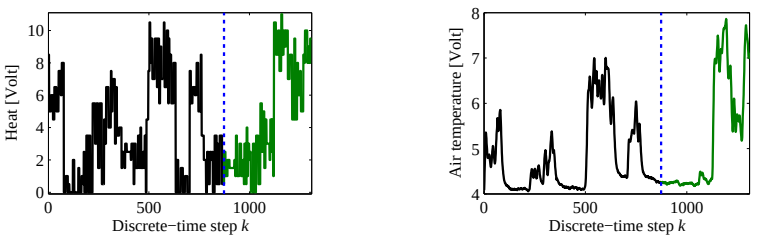
5.6. SUMMARY

In this chapter, a search-tree-based method is proposed to solve the regressor selection problem for nonlinear system identification. The method solves the problem by directly searching for the best regressor combination for a given model performance measure, which in this case is Akaike's information criterion, by building a search tree. The tree consists of hierarchically expanded nodes that represent different regressor combinations. The tree is formed with the aim of models with good performance.

The tree in the proposed method represents regressor combinations. The proposed method is independent of the model structure, although the corresponding performance computed will depend on the model structure selected. The proposed method can be applied to a large class of nonlinear models, and we have shown examples using the TS fuzzy models, ANN models, and SVR models.

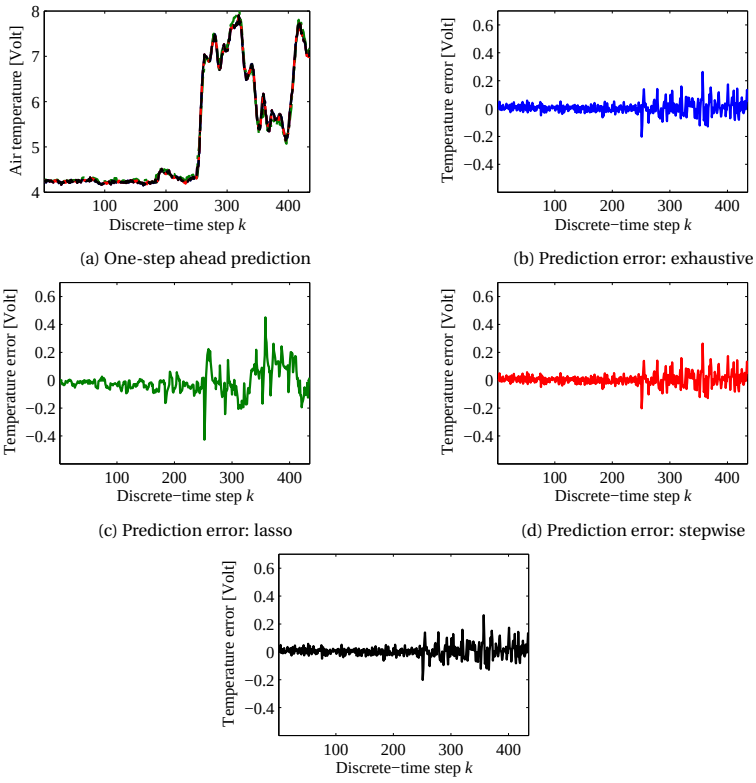
The proposed method can be seen as an extension of the stepwise regression method by performing multiple stepwise regressions at a time. This is done by allowing multiple node expansions at each level of the tree. In addition, the proposed method can also be considered as a simplification of exhaustive search, because not all possible regressor combinations are evaluated. Only combinations resulting in models with good performance are evaluated. Consequently, the proposed method is faster than the exhaustive search.

5.7. INPUT-OUTPUT DATA AND MODEL OUTPUT PLOTS



(a) Input (b) Output

Figure 5.13: Input-output data set of the heat transfer process. The vertical line divides the data set into two parts, with the identification data on the left-hand side and the validation data on the right-hand side.



(a) One-step ahead prediction

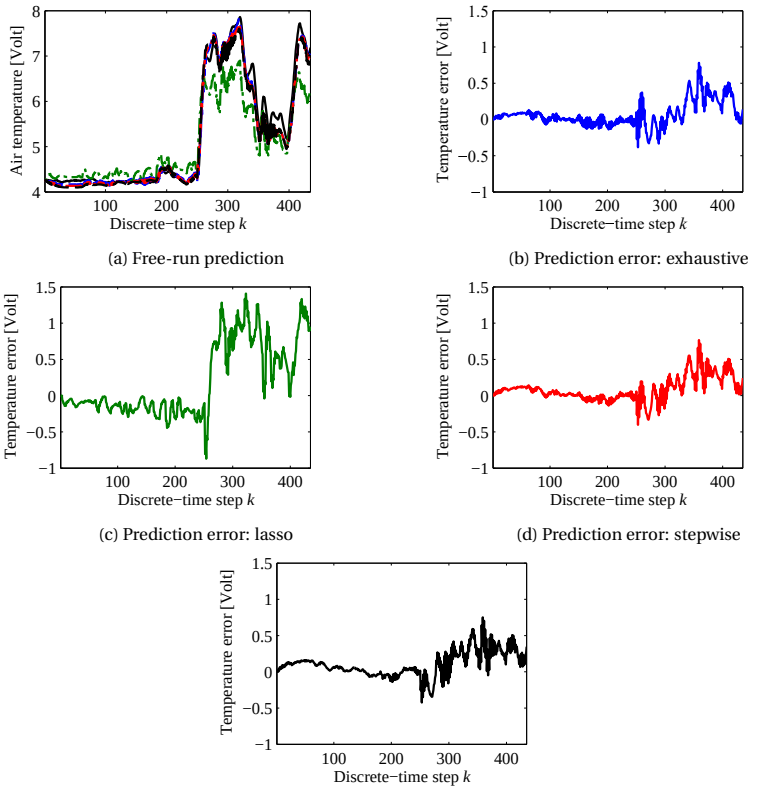
(b) Prediction error: exhaustive

(c) Prediction error: lasso

(d) Prediction error: stepwise

(e) Prediction error: proposed

Figure 5.14: One-step ahead predictions of the heat transfer validation data for models with 7 parameters and their corresponding errors: measurements (black solid line), exhaustive search (blue dashed line), lasso (green dash-dotted line), stepwise (red dash-dotted line), and proposed method (black dashed line)



(a) Free-run prediction

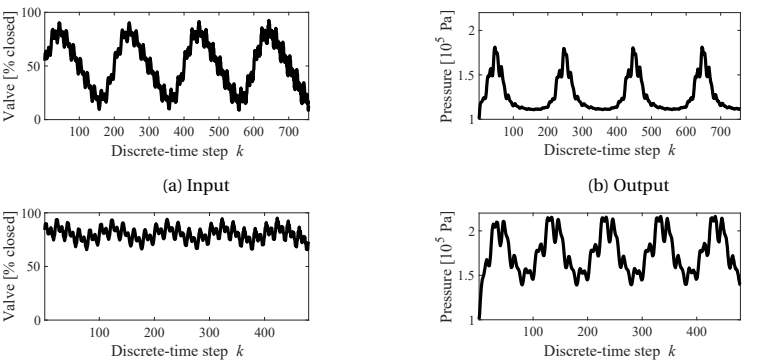
(b) Prediction error: exhaustive

(c) Prediction error: lasso

(d) Prediction error: stepwise

(e) Prediction error: proposed

Figure 5.15: Free-run simulation predictions of the heat transfer validation data for models with 7 parameters and their corresponding errors: measurements (black solid line), exhaustive search (blue dashed line), lasso (green dash-dotted line), stepwise (red dash-dotted line), and proposed method (black dashed line)



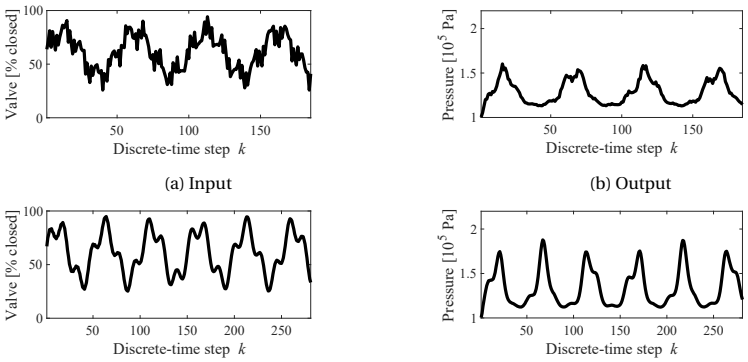
(a) Input

(b) Output

(c) Input

(d) Output

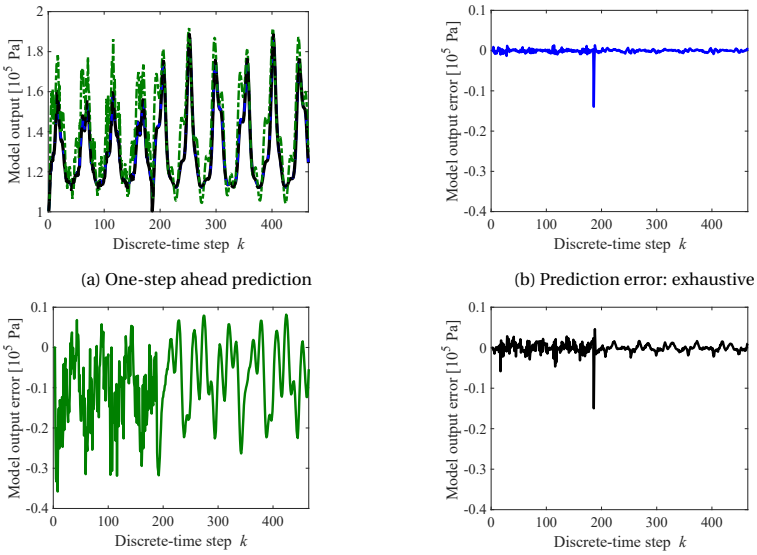
Figure 5.16: Identification data sets for the laboratory fermenter.



(a) Input (b) Output

(c) Input (d) Output

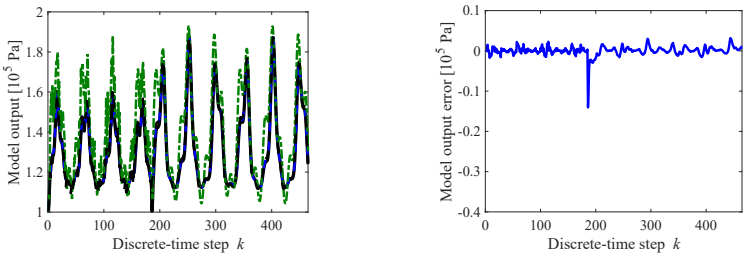
Figure 5.17: Validation data sets for the laboratory fermenter.



(a) One-step ahead prediction (b) Prediction error: exhaustive

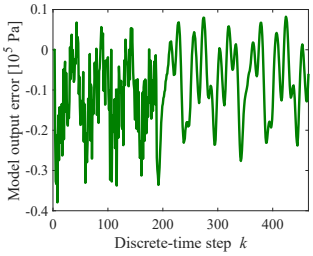
(c) Prediction error: lasso (d) Prediction error: proposed

Figure 5.18: One-step ahead predictions of the laboratory fermenter validation data for models with 7 parameters and their corresponding errors: measurements (black solid line), exhaustive search (blue dashed line), lasso (green dash-dot line), and the proposed method (black dashed line)

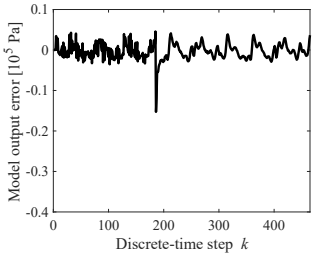


(a) Free-run prediction

(b) Prediction error: exhaustive

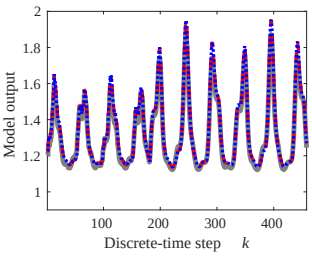


(c) Prediction error: lasso

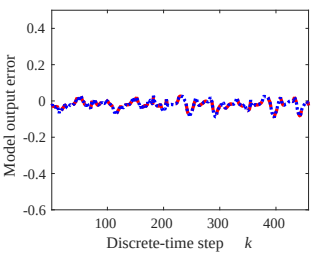


(d) Prediction error: proposed

Figure 5.19: Free-run simulation predictions of the laboratory fermenter validation data for models with 7 parameters and their corresponding errors: measurements (black solid line), exhaustive search (blue dashed line), lasso (green dash-dotted line), stepwise (red dash-doted line), and proposed method (black dashed line)

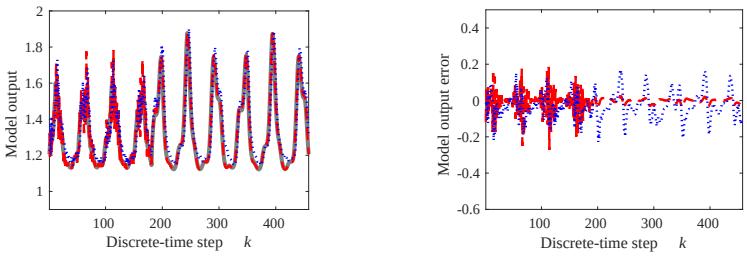


(a) Prediction



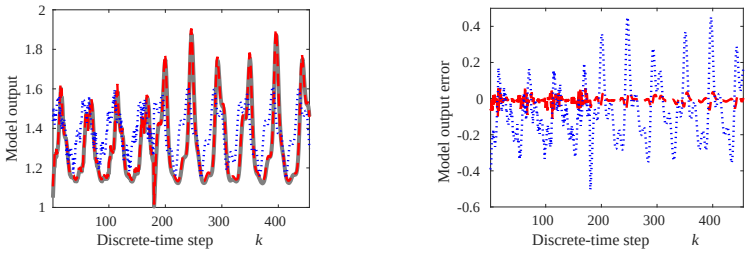
(b) Prediction error

Figure 5.20: (a) one-step ahead and free-run prediction, and (b) prediction error from the TS fuzzy models of the laboratory fermenter with 5 regressors: measurement (dark grey solid line), one-step ahead prediction from the exhaustive search (red dashed line), and from the proposed method (blue dotted line).



(a) Prediction (b) Prediction error

Figure 5.21: (a) one-step ahead and free-run prediction, and (b) prediction error from the NN models of the laboratory fermenter with 2 regressors: measurement (dark grey solid line), one-step-ahead prediction (red dashed line), and free-run prediction (blue dotted line) from the proposed search.



(a) Prediction (b) Prediction error

Figure 5.22: (a) free-run prediction, and (b) prediction error from the SVR models of the laboratory fermenter with 3 regressors: measurement (dark grey solid line), one-step-ahead prediction (red dashed line), and free-run prediction (blue dotted line) from the proposed method .

6

TAKAGI-SUGENO FUZZY SYSTEM IDENTIFICATION WITH INCOMPLETE DATA

6.1. INTRODUCTION

Data-driven system modeling is becoming more important because of the need to better represent complex systems for control and estimation purposes. As the complexity of the modeled systems increases, applying first-principles modeling becomes more difficult. In this case, a data-driven approach is preferred, especially for modeling nonlinear systems [94, 117]. In addition, one of the features of data-driven modeling is its independence from the physical system; consequently, the resulting model may not allow direct physical interpretations.

In general, the availability of the data plays an important role in data-driven modeling. However, data acquisition can suffer from problems that prevent the availability of a complete data set. The problems, e.g., might be due to the following issues:

1. Intermittent measurements caused by sensor faults or transmission failures.
2. Asynchronous sampling due to the absence of globally synchronized clocks.
3. Storage failure to retrieve recorded input-output data.

This requires the development of methods that can cope with missing measurements. The problem becomes more complex when dealing with nonlinear systems. For the identification of such systems, a Takagi-Sugeno (TS) fuzzy model representation [214] can be employed. TS fuzzy models have been successfully applied in control and estimation applications [188]. Moreover, methods have been developed for various TS fuzzy model-based control and estimation applications [82]. This is done by exploiting the capability of TS fuzzy models as universal function approximators for nonlinear functions [33], which implies that TS fuzzy models can approximate a large class of nonlinear systems with an arbitrary degree of accuracy. In this chapter, TS fuzzy system identification using incomplete data is studied by assuming that the input-output sample pairs are synchronized and the unavailable output samples are detected when the input samples are not accompanied by the output samples.

To the best knowledge of the author, the incomplete data problem for TS fuzzy system identification has not yet been addressed in the literature. Most algorithms have been developed for classification and regression problems. Although both regression and system identification are closely related, the difference in the data set structure means that the regression solutions cannot always be applied to system identification.

In this chapter, the problem of TS fuzzy identification with incomplete data is addressed. In particular, two methods are proposed to enable system identification with incomplete data. In the first method, the fuzzy c-means clustering approach for incomplete data called the optimal completion strategy [99] is extended for system identification by fusing different approximations of a missing measurement to produce a single value. The second method is an iteration of the identification procedure in which the missing samples are treated as optimization variables. These missing values are optimized (and consequently approximated) by the model obtained in each identification iteration. The approximation is terminated when a stopping criterion is satisfied.

6.2. RELATED WORK

In general, there are two approaches to dealing with system identification with incomplete data:

1. Direct approach. The algorithms of this approach can process the data without approximating the missing data.
2. Indirect approach. This approach involves an approximation of the missing samples in the identification procedure.

In the first approach, deleting the set of data pairs related to unavailable samples is a simple and obvious method. This method is not preferred if the number of missing samples is high. Hansson and Wallin [97] proposed several equivalent optimization formulations to directly estimate maximum-likelihood models from incomplete data. For the indirect approach, there are methods in the literature for linear and nonlinear systems. For the identification of linear systems, the nuclear norm [28] is used, e.g., in [158]. The Expectation Maximization (EM) method [66] can also be applied for linear systems as in, e.g., [69]. For nonlinear system identification with incomplete data, EM-based methods have also been developed. Gopaluni [87] proposed an EM-based method for nonlinear state-space system identification with incomplete data. More specifically, in the Expectation step, a particle filter and a particle smoother are used to perform pointwise state estimation to reduce the variance of the states. Gopaluni [88] developed another EM-based identification method for nonlinear systems approximated by using radial basis functions centered around the maximum a posteriori estimate of the state trajectory. At the same time, a particle filter is employed to estimate the states in the Expectation step. Deng and Huang [67] formulated an EM-based method for the identification of nonlinear parameter-varying systems modeled by multiple models with different working points. Similar to the previous methods, a particle filter is employed in the Expectation step. Yang et al. [237] developed an EM-based method for state-space model identification where a particle filter is used to numerically approximate the expected log-likelihood value of the complete data. Patel et al. [186] adapted nonlinear iterative partial least-squares to minimize the impact of missing data values.

A disadvantage in using the EM algorithm for nonlinear identification is that the Expectation step requires the smoothed estimates of the states, which are approximated using particle filters and smoothers. Given that particle filters are computationally intensive [198], their use in the Expectation step further increases the computational load during the optimization in the Maximization step.

Nonlinear system identification with incomplete data with artificial neural networks has also been reported in the literature. Tanaka and Dai [216] represented a Gaussian mixture model using a neural network structure. This network is trained using the EM method and applied to identify

a nonlinear model. Salini et al. [197] applied an evolving neural model for wastewater treatment data. The missing data are predicted by the output of a feedforward network trained by removing incomplete samples. Qiao et al. [189] employed a sensor evaluation and restoration scheme to deal with missing measurements in a power plant. The scheme contains an auto-encoder consisting of a neural network and a particle swarm optimizer fed with input-output data for the power plant operation. The auto-encoder is trained to capture the correlation between its complete inputs. Missing measurements are determined by particle swarm optimization based on the correlation between the available and the missing data.

Imputation of missing data is also a subject in other fields, e.g., machine learning. Imputation methods to approximate missing samples are one of the popular topics. There have already been some literature surveys on the topic, e.g., by Emmanuel et al. [79], Alabadla et al. [4] and Miao et al. [167]. As they are developed for applications in machine learning, these imputation methods may need adjustment for the identification of nonlinear systems.

A publication that is close to this chapter is by Almeida et al. [5], who solved a nonlinear regression problem with incomplete data with a TS fuzzy model using product space clustering [16]. Almeida et al. [5] used the partial distance strategy, one of the strategies proposed in [99]. Specifically, Hathaway and Bezdek [99] proposed four modifications of fuzzy c-means clustering to allow processing incomplete data:

1. Whole data strategy. This strategy removes incomplete data so that only complete data are processed.
2. Partial distance strategy. In this strategy, the distance from a prototype to incomplete data is computed based on the available elements of the data.
3. Optimal completion strategy. In this strategy, the unavailable elements of the incomplete data are approximated during the clustering process.
4. Nearest prototype strategy. This is a modification of the optimal completion strategy by substituting the unavailable elements of data with the nearest prototype and computing the distance based on the partial distance.

In a regression problem, a regressor matrix is constructed by appending regressor vectors. In this case, missing samples in the regressor vectors are uniquely located according to the position of the regressor vectors in the regressor matrix. In system identification, however, the regressor matrix is built by appending delayed input-output vectors so that a missing sample is in the output vector. For instance, the missing sample is propagated to the delayed output vectors located in different rows, depending on the order of the system. In this chapter, the optimal completion strategy is extended for system identification by allowing a unique estimation of missing regressor matrix elements that represent a missing sample.

6.3. PROBLEM FORMULATION

The problem of system identification with incomplete data is introduced in this section.

Consider a pair of input-output data vectors: an input vector $\boldsymbol{u} \in \mathbb{R}^N$ and the corresponding output vector $\boldsymbol{y} \in \mathbb{R}^N$:

$$\boldsymbol{u} = \begin{bmatrix} u(1) & \cdots & u(N) \end{bmatrix}^\top$$
$$\boldsymbol{y} = \begin{bmatrix} y(1) & \cdots & y(N) \end{bmatrix}^\top$$

where N is the number of data samples, $u(k)$ and $y(k)$ are scalars, $\top$ is the transpose operation, and k is the discrete-time step. Note that in this chapter, a SISO system is considered, but the proposed

approach can be straightforwardly extended to MISO and MIMO systems. For system identification purposes, a lagged input-output vector $\boldsymbol{\phi}_k$ is defined as follows:

$$\begin{aligned}\boldsymbol{\phi}_k &= \begin{bmatrix}\phi_{k,1} & \cdots & \phi_{k,n_\mathrm{a}} & \phi_{k,n_\mathrm{a}+1} & \cdots & \phi_{k,n_\mathrm{a}+n_\mathrm{b}}\end{bmatrix}^\top \\ &= \begin{bmatrix}y(k-1) & \cdots & y(k-n_\mathrm{a}) & u(k-1) & \cdots & u(k-n_\mathrm{b})\end{bmatrix}^\top\end{aligned} \tag{6.1}$$

where n_a is the maximum output lag, and n_b is the maximum input lag where typically, and also assumed here, $n_\mathrm{a} > n_\mathrm{b}$. The vector $\boldsymbol{\phi}_k$ is also called the regressor vector. Collecting these vectors for different time steps gives the regressor matrix $\boldsymbol{\Phi} = \begin{bmatrix}\boldsymbol{\phi}_{n_\mathrm{a}+1} & \cdots & \boldsymbol{\phi}_N\end{bmatrix}$ such that $\boldsymbol{\Phi} \in \mathbb{R}^{n_\mathrm{r} \times N_\phi}$ with:

$$\begin{aligned}N_\phi &= N - n_\mathrm{a} \\ n_\mathrm{r} &= n_\mathrm{a} + n_\mathrm{b}\end{aligned}$$

Note that the systems considered in this chapter are nonlinear ARX (NARX) systems that use the regressor vector (6.1).

For identification, this chapter uses the input-output data to parameterize a model given by

$$\hat{y}(k) = f\left(\boldsymbol{\phi}_k, \boldsymbol{\theta}\right) \tag{6.2}$$

with $\hat{y}$ the model output, $\boldsymbol{\theta}$ the vector of model parameters, and $f(\cdot)$ a nonlinear function. The parameter vector $\boldsymbol{\theta}$ is estimated by minimizing the squared difference between the output data and the model output $e(k) = y(k) - \hat{y}(k)$, i.e., by solving the following optimization problem:

$$\underset{\boldsymbol{\theta}}{\arg\min} \sum_{k=n_\mathrm{a}+1}^{N} \left(y(k) - f\left(\boldsymbol{\phi}_k, \boldsymbol{\theta}\right)\right)^2 \tag{6.3}$$

Using the regressor matrix $\boldsymbol{\Phi}$, the optimization formulation (6.3) can also be written as:

$$\underset{\boldsymbol{\theta}}{\arg\min} \left(\boldsymbol{\phi}_y - g(\boldsymbol{\Phi}, \boldsymbol{\theta})\right)^\top \left(\boldsymbol{\phi}_y - g(\boldsymbol{\Phi}, \boldsymbol{\theta})\right) \tag{6.4}$$

for $\boldsymbol{\phi}_y = \begin{bmatrix}y(n_\mathrm{a}+1) & \cdots & y(N)\end{bmatrix}^\top$ the regressand vector, $g(\cdot)$ a nonlinear function similar to (6.2) but accepting a vector and a matrix as parameters.

In case of incomplete data, some regressor vectors and, consequently, the regressor matrix contain unavailable elements. The following are some observations on the relation between the missing output samples and the regressor matrix:

- Depending on the order of the system, a missing output sample is propagated to different rows and columns of the regressors matrix.
- From a missing output sample, a larger n_a creates more rows with incomplete elements in the regressor matrix $\boldsymbol{\Phi}$.

Introducing missing output samples means that (6.4), in general, cannot be solved as the output model $g(\boldsymbol{\Phi}, \boldsymbol{\theta})$ cannot be evaluated due to the unknown values of the missing elements.

6.4. TS FUZZY SYSTEM IDENTIFICATION: COMPLETE MEASUREMENT CASE

The TS fuzzy model [214] is a mathematical model that uses fuzzy if-then rules with local linear or affine consequents to represent nonlinear systems. Identification of such a model takes the

regressor vector (6.1). Each rule $i = 1, \ldots, c$ of the affine TS fuzzy model is defined as:

$$\begin{aligned} &\text{Rule } i\text{:} \\ &\textbf{IF } \phi_{k,1} \text{ is } M_{i,1} \textbf{ and} \ldots \textbf{ and } \phi_{k,n_\text{a}} \text{ is } M_{i,n_\text{a}} \textbf{ and} \\ &\ \phi_{k,n_\text{a}+1} \text{ is } M_{i,n_\text{a}+1} \textbf{ and } \ldots \textbf{ and } \phi_{k,n_\text{r}} \text{ is } M_{i,n_\text{r}} \\ &\textbf{THEN } y_i(k) = \boldsymbol{\phi}_k^\top \boldsymbol{\theta}_i + \theta_{0i} \end{aligned} \tag{6.5}$$

where c is the number of rules, M_{ij} is a fuzzy set of rule i, regressor j with membership function $\mu_{M_{ij}} : \mathbb{R} \to [0,1]$, and $\boldsymbol{\theta}_i \in \mathbb{R}^{n_\text{r} \times 1}$ and θ_0 are respectively the vector of parameters and the scalar offset of the i-th rule [1].

The global model, as a blend of local models represented by the rules, is computed by using the weighted average equation:

$$y(k) = \frac{\sum_{i=1}^{c} \beta_i(\boldsymbol{\phi}_k) y_i(k)}{\sum_{i=1}^{c} \beta_i(\boldsymbol{\phi}_k)} = \frac{\sum_{i=1}^{c} \beta_i(\boldsymbol{\phi}_k)(\boldsymbol{\phi}_k^\top \boldsymbol{\theta}_i + \theta_{0,i})}{\sum_{i=1}^{c} \beta_i(\boldsymbol{\phi}_k)} \tag{6.6}$$

where $\beta_i(\boldsymbol{\phi}_k)$ is the degree of fulfillment of the i-th rule's antecedent obtained from the membership degree of all variables in the antecedent computed by using:

$$\beta_i(\boldsymbol{\phi}_k) = \mu_{M_{i,1}}(\phi_{k,1}) * \mu_{M_{i,2}}(\phi_{k,2}) * \quad \cdots \quad * \mu_{M_{i,n_\text{r}}}(\phi_{k,n_\text{r}})$$

where $*$ denotes the t-norm [251]. For instance, one can use the minimum t-norm defined by

$$\beta_i(\boldsymbol{\phi}_k) = \min\left(\mu_{M_{i,1}}(\phi_{k,1}), \mu_{M_{i,2}}(\phi_{k,2}), \quad \cdots \quad , \mu_{M_{i,n_\text{r}}}(\phi_{k,n_\text{r}})\right)$$

For identification using product space clustering, the data matrix $\boldsymbol{\Phi}_\text{A}$ is constructed as follows:

$$\boldsymbol{\Phi}_\text{A} = \begin{bmatrix} \boldsymbol{\phi}_y & \boldsymbol{\Phi} \end{bmatrix} \tag{6.7}$$

The TS fuzzy identification using product space clustering consists of two steps [16]:

1. Applying fuzzy clustering to the data matrix $\boldsymbol{\Phi}_\text{A}$. This step includes determining the cluster parameters and the number of clusters. This step results in a fuzzy parameter matrix $\boldsymbol{U} = [\mu_{i,j}]$ with $\boldsymbol{U} \in \mathbb{R}^{N_\phi \times c}$, where $\mu_{i,j}$ is the fuzzy membership value of the i-th row of $\boldsymbol{\Phi}_\text{A}$ to the cluster j. A commonly used clustering method is the Gustafson-Kessel algorithm [91]. The ability to construct ellipsoidal clusters makes this clustering algorithm suited for TS fuzzy system identification.
2. Constructing the rules. This includes antecedent parameter estimation from the clusters and consequent parameter estimation. This is done by solving a set of linear equations using the ordinary least-squares or the total least-squares methods, where each linear equation is weighted by its fuzzy membership value.

6.5. TS FUZZY IDENTIFICATION: INCOMPLETE DATA CASE

In this section, TS fuzzy identification in case of missing measurements is discussed, and two possible methods to deal with the problem are proposed.

In general, the matrix $\boldsymbol{\Phi}_\text{A}$ can be viewed as a set of information required to build a model. Unavailable samples reduce the information to build the model in comparison with the complete

[1] The rule i basically tells to use the model $y_i(k) = \boldsymbol{\phi}_k^\top \boldsymbol{\theta}_i + \theta_{0i}$ with a certain weight calculated by combining of the weights of the antecedent variables $\phi_{k,j}$

data. The remaining measurements can still be used to build a model, although the model might not be as accurate as that with complete data. We explore methods to make better use of the available measurements and to complete the data set.

One possible way to tackle the incomplete data problem is by removing rows containing unavailable samples. This is called the whole data strategy [99] or row deletion strategy. This approach is feasible when the number of missing samples N_{m} is relatively small so that there is still an adequate number of rows left to obtain an acceptable model. It should be noted that deleting rows containing missing samples also removes elements available in the other columns of the removed rows. The row deletion approach might not be preferred, especially for higher-order systems, because the number of incomplete rows increases with the increase of the maximum order of the input n_{b} and the output n_{a}.

The other approach is by imputing the missing measurements, called the optimal completion strategy [99]. A typical method of imputation is by considering the missing measurements as parameters and including them in the model parametrization. This method can be viewed as an estimator of the missing elements by utilizing the available measurements in the incomplete rows as they still have information about the system, but they are disturbed by the unavailable values.

In the following sections, two methods belonging to the second approach to perform TS fuzzy identification are presented:

1. Clustering-based imputation in Section 6.5.1
2. Iterative imputation in Section 6.5.2

6.5.1. CLUSTERING-BASED IMPUTATION

Fuzzy clustering methods for incomplete data in the literature were developed for classification and regression purposes, see, e.g., [99, 149, 218]. In principle, a system identification problem can be seen as a regression problem. The difference appears only in the regressor matrix, as the regressor matrix in system identification consists of delayed input-output data. Indeed, as it has been noted in Section 6.3, a missing sample will be propagated to different rows and columns of the regressor matrix. As an illustration, consider the matrix $\boldsymbol{\Phi}_{\mathrm{A,m}}$ below, an implementation of (6.7), which contains a missing sample. In (6.8) the output $y(5)$ is not available, which is marked by the subscript * as follows:

$$\boldsymbol{\Phi}_{\mathrm{A,m}} = \begin{bmatrix} y(4) & y(3) & y(2) & y(1) & u(3) \\ y_*(5) & y(4) & y(3) & y(2) & u(4) \\ y(6) & y_*(5) & y(4) & y(3) & u(5) \\ y(7) & y(6) & y_*(5) & y(4) & u(6) \\ y(8) & y(7) & y(6) & y_*(5) & u(7) \end{bmatrix} \tag{6.8}$$

Approximations of the missing elements in (6.8) using algorithms developed for classification or/and regression will result in different values of $y_*(5)$, while in system identification, the value of $y_*(5)$ should be the same everywhere in the matrix. To achieve that, a set of discrete-time indices at which the measurements are unavailable, denoted by $\mathbb{K}_{\mathrm{m}}$, is defined. In addition, for each member of $\mathbb{K}_{\mathrm{m}}$, a set containing the approximation of the unavailable measurement, denoted by $\boldsymbol{\phi}_{y_*(k)}$, is also defined. In (6.8), for instance, $\mathbb{K}_{\mathrm{m}} = \{5\}$ and $\boldsymbol{\phi}_{y_*(5)} = \{\phi_{2,1}, \phi_{3,2}, \phi_{4,3}, \phi_{5,4}\}$.

Applying the optimal completion strategy [99] to $\boldsymbol{\Phi}_{\mathrm{A,m}}$ typically results in the different approximation of the unavailable elements in $\boldsymbol{\phi}_{y_*(k)}$. To obtain a single value of the unavailable sample, the elements of $\boldsymbol{\phi}_{y_*(k)}$ are fused by using a functional that takes $\boldsymbol{\phi}_{y_*(k)}$ as its parameter:

$$\hat{y}(k) = h\left(\boldsymbol{\phi}_{y_*(k)}\right) \tag{6.9}$$

with $h(\cdot)$ the functional that fuses the elements of $\boldsymbol{\phi}_{y_*(k)}$. This functional may perform, e.g., averaging of all elements of $\boldsymbol{\phi}_{y_*(k)}$. In the case of the example, the estimate of the missing $y(5)$ is then calculated by averaging the members of $\boldsymbol{\phi}_{y_*(5)}$:

$$\hat{y}(5) = \frac{1}{4}\left(\phi_{2,1} + \phi_{3,2} + \phi_{4,3} + \phi_{5,4}\right)$$

where $\hat{y}(5)$ is the approximation of $y_*(5)$. The algorithm is shown in Algorithm 2. In the algorithm, the missing samples can be initialized, e.g., by interpolation from the available samples.

It is necessary to evaluate the convergence of the algorithm because of the addition of the fusion step in the optimal completion strategy [99]. Hathaway and Bezdek [99] stated that the convergence of the optimal completion strategy is guaranteed, as shown in [23]. This means the members of $\boldsymbol{\phi}_{y_*(k)}$, the estimated value $y_*(k)$ in $\boldsymbol{\Phi}_{\mathrm{A,m}}$, will also converge. Assuming

$$\min\left(\boldsymbol{\phi}_{y_*(k)}\right) \leq h\left(\boldsymbol{\phi}_{y_*(k)}\right) \leq \max\left(\boldsymbol{\phi}_{y_*(k)}\right) \tag{6.10}$$

we can also say that $\hat{y}(k)$ will converge following the convergence of $\boldsymbol{\phi}_{y_*(k)}$.

6.5.2. ITERATIVE IMPUTATION

The iterative imputation method treats missing measurements as optimization variables to obtain the optimal values of the imputed measurements besides the model parameters. The problem is formulated as:

$$\underset{\boldsymbol{\theta}_{\mathrm{F}},\boldsymbol{\theta}_{\mathrm{m}}}{\arg\min} \sum_{k \notin \mathbb{K}_{\mathrm{m}}} \left(y(k) - f_{\mathrm{F}}\left(\boldsymbol{\theta}_{\mathrm{F}},\boldsymbol{\theta}_{\mathrm{m}}\right)\right)^2 \tag{6.11}$$

with $f_{\mathrm{F}}(\cdot)$ the TS fuzzy model whose parameter $\boldsymbol{\theta}_{\mathrm{F}}$ consists of antecedent parameters and consequent parameters, and the missing data treated as model parameters $\boldsymbol{\theta}_{\mathrm{m}}$. Because the missing samples are optimization variables, the performance of the model in the identification step is computed from the available measurements.

In this method, no missing data-specific technique is used to find the optimal imputation values. The approach purely focuses on using optimization to search for imputation values with which a model with the best performance is obtained.

Solutions to (6.11) are combinations of imputation values $\boldsymbol{\theta}_{\mathrm{m}}$, together with the model parameters $\boldsymbol{\theta}_{\mathrm{F}}$, that minimize the cost function. The combination might not be unique because (6.11) is, in general, a non-convex optimization problem. This problem is typically hard to solve because of the local minima.

There are some methods to solve non-convex optimization problems of the form (6.11), e.g., meta-heuristic methods such as genetic algorithms [85] or particle swarm optimization [125]. In addition, these methods have also been extended to involve multiple initial search locations [235]. These methods approximate the solution of a global optimization problem. The optimization problem (6.11) has two sets of optimization variables; we may use the alternating optimization approach to solve the problem. The alternating optimization alternatingly performs the following steps, given the initial value of $\boldsymbol{\theta}_{\mathrm{m}}$:

1. Solve for $\boldsymbol{\theta}_{\mathrm{F}}$ by performing TS fuzzy identification to obtain a TS fuzzy model.
2. Solve for $\boldsymbol{\theta}_{\mathrm{m}}$ based on the prediction obtained by the obtained TS fuzzy model.

The steps above are repeated until a termination criterion is met. It should be noted, however, that the iteration may not always converge. In addition, the non-convex nature of the optimization may also lead to a local minimum. In this chapter, the iteration process is terminated when the performance change of the identified model gets small enough.

This method is simple because no special algorithm is needed for the clustering and consequent estimation steps to deal with the missing measurements. In general, the performance of the

Algorithm 2 Fuzzy c-means clustering for incomplete system identification data

Input: The input-output data vector $\boldsymbol{u}$, $\boldsymbol{y}$, the fuzziness cluster constant m, the number of clusters c, the stopping criterion threshold ε, the discrete-time indices of missing samples $\mathbb{K}_{\mathrm{m}}$, the initial cluster centers $\boldsymbol{V}^{(0)} = \begin{bmatrix} \boldsymbol{v}_1^{(0)} & \cdots & \boldsymbol{v}_c^{(0)} \end{bmatrix}$

Initially impute missing samples
Create augmented regressor matrix $\boldsymbol{\Phi}_{\mathrm{A}}$
Loop $\leftarrow$ true
$l \leftarrow 1$
while Loop **do**
  { *Compute square of distance* }
  $D_{i,j} \leftarrow \left(\boldsymbol{\phi}_i - \boldsymbol{v}_j^{(l)}\right)\left(\boldsymbol{\phi}_i - \boldsymbol{v}_j^{(l)}\right)^{\top}, \quad i = 1,\ldots,N_\phi, j = 1,\ldots,c$
  {*Compute fuzzy partition matrix*}
  $\mu_{i,j}^{(l)} \leftarrow \frac{D_{i,j}^{-(1-m)}}{\sum_{k=1}^{c} D_{i,k}^{-(1-m)}}$
  { *Update prototype centers* }
  $\boldsymbol{v}_j^{(l)} \leftarrow \frac{\sum_{i=1}^{N_\phi}\left(\mu_{i,j}^{(l)}\right)^m \boldsymbol{\phi}_i}{\sum_{i=1}^{N_\phi}\left(\mu_{i,j}^{(l)}\right)^m}, \quad j = 1,\ldots,c$
  if $\left\|\boldsymbol{V}^{(l)} - \boldsymbol{V}^{(l-1)}\right\|_{\mathrm{F}} < \varepsilon$ **then**
    Loop $\leftarrow$ false
  else
    { *Compute estimates of unavailable elements in* $\boldsymbol{\Phi}_{\mathrm{A}}$ *}
    for all $\phi_{i,j}^* \in \boldsymbol{\phi}_{u_*(k)}$ and $\phi_{i,j}^* \in \boldsymbol{\phi}_{y_*(k)}$ **do**
      $\phi_{i,j}^* \leftarrow \frac{\sum_{i=1}^{N_\phi}\left(U_{in}^{(l)}\right)^m \phi_{nj}}{\sum_{i=1}^{N_\phi}\left(U_{in}^{(l)}\right)^m}$
    end for
    for all $k \in \mathbb{K}_{\mathrm{m}}$ **do**
      $\varphi_{u(k)} \leftarrow \mathrm{fuse}\left\{\phi_{ij}^* | \phi_{ij}^* \in \boldsymbol{\phi}_{u_*(k)}\right\}$
      Replace $\left\{\phi_{ij}^* | \phi_{ij}^* \in \boldsymbol{\phi}_{u_*(k)}\right\}$ with $\varphi_{u(k)}$
      $\varphi_{y(k)} \leftarrow \mathrm{fuse}\left\{\phi_{ij}^* | \phi_{ij}^* \in \boldsymbol{\phi}_{y_*(k)}\right\}$
      Replace $\left\{\phi_{ij}^* | \phi_{ij}^* \in \boldsymbol{\phi}_{y_*(k)}\right\}$ with $\varphi_{y(k)}$
    end for
  end if
  $l \leftarrow l + 1$
end while

Output: $\boldsymbol{U}, \boldsymbol{V}$

resulting model is expected to decrease with the increasing number of missing measurements. On the other hand, the required computation resources are higher with the increase in the number of missing measurements due to the increasing number of optimization variables.

Algorithm 3 Method 2: Iterative Imputation

Input: $\boldsymbol{u}$, $\boldsymbol{y}$, c, m, termination criterion
  Initially impute missing measurements samples in $\boldsymbol{y}$
  Create augmented regressor matrix $\boldsymbol{\Phi}_{\mathrm{A}}$
  $\ell \leftarrow 0$
  repeat
    $\ell \leftarrow \ell + 1$
    $\boldsymbol{\theta}_{\mathrm{F}} \leftarrow$ TS fuzzy system identification with product space clustering
    $\hat{\boldsymbol{y}} \leftarrow$ run prediction with $f_{\mathrm{F}}(\boldsymbol{\theta}_{\mathrm{F}})$
    $\boldsymbol{\theta}_{\mathrm{m}}^{(\ell)} \leftarrow \hat{\boldsymbol{y}}$
  until termination criterion is met
  Insert $\boldsymbol{\theta}_{\mathrm{m}}^{(\ell)}$ into the corresponding missing measurements
  $\boldsymbol{\theta}_{\mathrm{F}} \leftarrow$ TS fuzzy system identification with product space clustering
Output: $f_{\mathrm{F}}(\boldsymbol{\theta}_{\mathrm{F}})$

6.6. SIMULATION EXPERIMENTS

In this section, the methods of Section 6.5.1 and 6.5.2 are applied. There are three data sets that are used for the experiments. The first data set is from a toy problem, the second is from a simulation of a pH neutralization process, and the third is from the measurements of a simple real-life heat transfer process. The second and the third data sets are obtained from [122]. All data sets are complete data sets.

The synthetic data set with the corresponding time stamps is recorded. The missing sample instants are generated from n different randomly selected randomized time stamps. This allows experiments with an incrementally increasing number of missing samples.

In this example, it is assumed that the missing values occur randomly during an input-output data acquisition. These random events can be modeled as Bernoulli trials, namely, independent trials with only two possible outcomes: success or failure. The probability of success for each trial is assumed to be the same throughout the trials [132]. Seeing data acquisition as a random experiment, the possible outcomes at each sampling instant are "unavailable sample" with probability p and "available sample" with probability $1-p$. From N samples of input-output data acquisition, one can expect to have $N_{\mathrm{m}} = N \cdot p$ output samples missing. In other words, p can be interpreted as the fraction of missing samples and therefore expressed as a percentage of missing samples denoted by m_{p} in percent, $m_{\mathrm{p}} = p \cdot 100$. In practice, the value of p is generally unknown until the end of the data acquisition experiment.

Four fractions of missing identification data samples m_{p} are simulated: 10%, 20%, 30%, 40%. No missing samples are assumed in the validation data. This allows for a fair performance evaluation of the models obtained from incomplete measurements. In addition, in the simulation, only output measurements are assumed to be missing, as the input samples are typically given in the data acquisition experiments. The experiments are repeated with 100 different randomized time stamp sets to allow statistical analysis of the performance of the algorithm.

To obtain the initial imputed values, the available samples are fitted to a piecewise cubic spline. There are some advantages of using cubic spline interpolation, e.g., differentiability, integrability,

Table 6.1: Consequent parameters of the toy example

i	a_i	b_i
1	0.9	−0.1
2	0.2	0.7

and smoothness. It has also been used in system identification [65]. The resulting spline is used to interpolate the unavailable samples. When the missing samples are at the beginning or the end of the data set, extrapolation is used instead of interpolation.

The R^2 values of the validation data free-run predictions[2] are presented to show the performance of the algorithms. This measure indicates how well the variance of the output/dependent variable is predicted by the input/independent variable. The statistics of the computation are also shown. The mean and standard deviation are also presented in the tables, where 'Stdv' stands for standard deviation.

In this section, the imputation with the extended fuzzy c-means is called the Clustering Method, and the iterative imputation is called the Iterative Method. As a benchmark, the identification by removing regressors containing unavailable samples is also shown. This is called Deletion Method.

The experiment is simulated using MATLAB™, where the clustering and deletion methods are self-scripted. The iterative method uses the `lsqnonlin` function from the Optimization toolbox of MATLAB™.

6

6.6.1. TOY PROBLEM

A synthetic data set for identification and validation is generated from the following TS fuzzy model:

Rule 1:

IF $u(k-1)$ is *LOW* **and** $y(k-1)$ is *LOW*

THEN $y(k) = a_1 y(k-1) + b_1 u(k-1) + c_1$

Rule 2:

IF $u(k-1)$ is *HIGH* **and** $y(k-1)$ is *HIGH*

THEN $y(k) = a_2 y(k-1) + b_2 u(k-1) + c_2$

for the values of the consequent parameters given in Table 6.1 and the fuzzy membership functions of the antecedent shown in Figure 6.1.

There are 800 samples generated from the model, 400 of them are for identification, and the rest are for validation. The data are shown in Figure 6.2, where the identification data is on the left side of the vertical blue dotted line and the validation data is on the right side. The input samples are staircase signals of random step heights in the set $\{-0.4, -0.2, 0, 0.2, 0.4\}$. The duration of the steps is also random in the set $\{3, 6, 9, \ldots, 24\}$.

The performance of the model with different percentages of missing data is shown in Table 6.2. As the number of missing data increases, the means of the R^2 performance shows the model performance degradation for all schemes, but the standard deviations have the opposite trend.

The R^2 performance of the model with different percentages of missing samples and the computation time is plotted in Figure 6.3. The numerical results, i.e., the mean and the standard deviation, of the results are shown in Table 6.2.

[2] A free-run prediction uses the identified model by using the output data for the initial condition only. After the initial condition, the simulation uses the measured input and the predicted output to compute the next predictions.

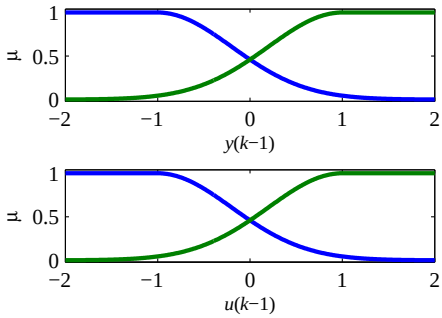
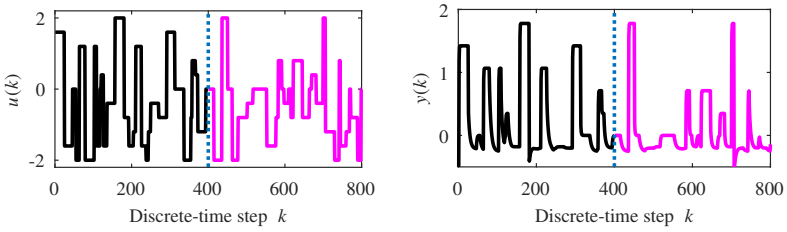


Figure 6.1: Membership functions of the antecedent for the toy problem.



(a) Input (b) Output

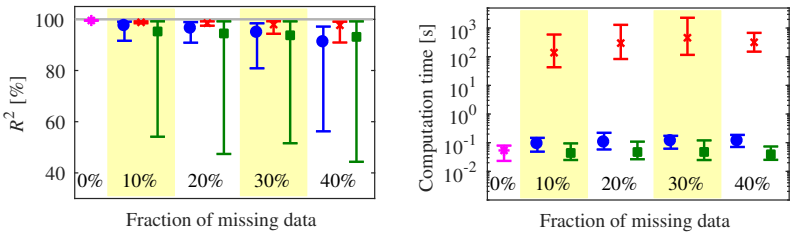
Figure 6.2: Input-output data set of the toy problem. The identification data are on the left of the vertical dotted line, while the validation data are on the right.

Table 6.2: The mean and the standard deviation of the R^2 performance for the toy problem in percent.

	Clustering		**Iterative**		**Deletion**	
m_p (%)	Mean	Stdv	Mean	Stdv	Mean	Stdv
0			99.53	2.45×10^{-4}		
10	97.42	1.67	**98.29**	**1.03**	95.25	8.87
20	96.41	1.83	**97.91**	**1.14**	94.48	9.30
30	94.79	2.90	**97.65**	**0.99**	93.75	7.80
40	91.08	6.15	**97.56**	**1.41**	93.14	8.87

Table 6.3: The mean and standard deviation of the computation time for the toy problem in seconds

	Clustering		**Iterative**		**Deletion**	
m_p (%)	Mean	Stdv	Mean	Stdv	Mean	Stdv
0			0.05	0.21		
10	0.09	0.02	138.63	87.69	**0.04**	**0.02**
20	0.10	0.03	297.82	194.37	**0.05**	**0.02**
30	0.11	0.03	458.81	386.31	**0.05**	**0.02**
40	0.11	0.03	317.43	134.50	**0.04**	**0.01**



(a) R^2 performance

(b) Computation time

Figure 6.3: The R^2 performance and the computation time for the toy example. The blue lines and markers are from the clustering method, the red ones are from the iterative method, and the green ones are from the deletion method. In Figure 6.3a, the markers (circles, x, and squares) indicate the mean of the R^2, and the upper and the lower horizontal bars are, respectively, the maximum and the minimum values.

As shown in the plots, the mean of the R^2 values increases with the increasing fraction of missing samples. The iterative method results in relatively smaller mean and minimum performance values compared to the clustering method and the deletion method. The minimum values are comparable to those of the iterative method for 10% and 20% of missing data but clearly worse for larger fractions of missing data. This is also evident from Table 6.2, where the iterative method performs better than the clustering method, while the deletion method has the lowest performance with respect to performance mean and standard deviation.

For the computation time, Figure 6.3b and Table 6.3 show that the iterative method is generally much slower than the clustering method and the deletion method, while the clustering method and the deletion method do not have a significant difference in computation time. The high computation load of the iterative method is expected as it iteratively performs a full TS fuzzy identification to optimize the missing samples.

The pattern of computation time, however, is not regular for increasing missing data fractions. One may expect an increase in computation time due to the higher number of variables to optimize to compensate for the larger missing data fraction. For all methods, the mean and the standard deviation do not continuously increase or decrease with the increase of missing data fractions. This observation will be discussed in Section 6.6.4.

6.6.2. pH NEUTRALIZATION

The identification of a neutralization tank is presented in this section. The tank has three influent streams and one effluent stream. The influent streams are acid, buffer, and base. The data are obtained by simulating the model of [96] by varying the base stream flow rate between 2 and 35 $\mathrm{mL\,s^{-1}}$, while the other influent streams are kept constant at 16.6 $\mathrm{mL\,s^{-1}}$ and 0.55 $\mathrm{mL\,s^{-1}}$. The output is the pH in the tank. The identification data set contains 480 samples, and the validation set contains 500 samples. As there are two data sets already, we just took one data set for identification and the other for validation. No technical consideration was taken in dividing the data sets for identification and validation. The input-output data are shown in Figure 6.5. These data sets were obtained from [123].

For identification, the TS fuzzy model consists of rules with cluster fuzziness 1.2 [16], and the consequent parameters are computed by the least-squares method for each rule. The regressor vector used in the model is $\boldsymbol{\phi}(k) = \begin{bmatrix} y(k-1) & y(k-2) & u(k-1) \end{bmatrix}^\top$. The TS fuzzy model parameters are taken from [16].

The results of this example are shown in Figure 6.6b to 6.6d and Table 6.4 for the R^2 perfor-

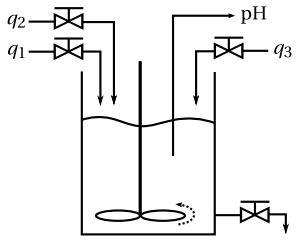
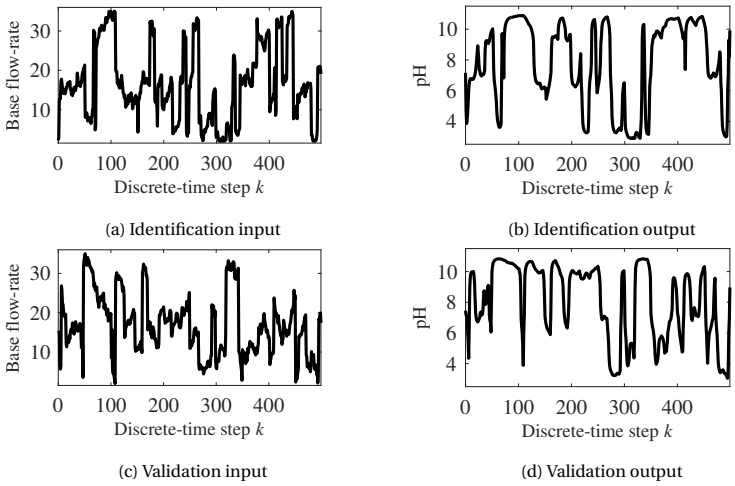


Figure 6.4: Schematic diagram of the pH neutralization setup.



(a) Identification input

(b) Identification output

(c) Validation input

(d) Validation output

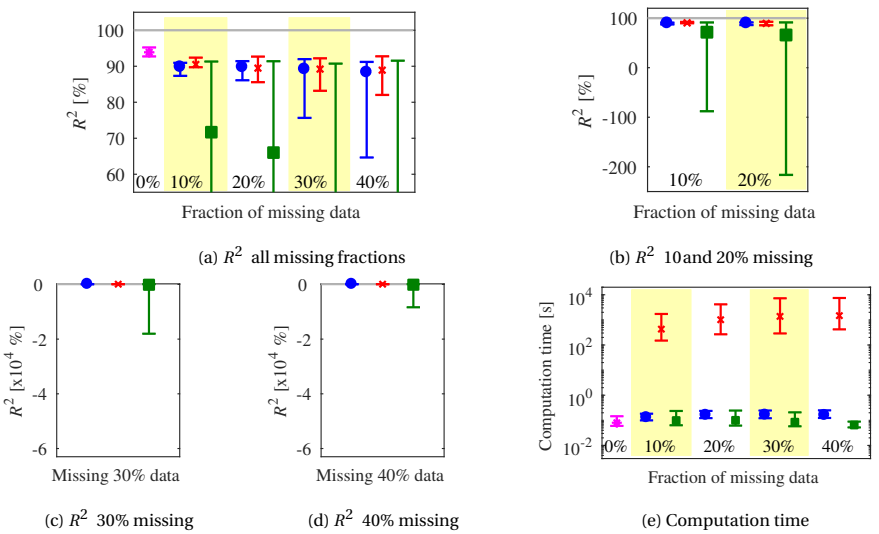
Figure 6.5: Input-output data sets for the pH neutralization example. Figure 6.5a and 6.5b are the identification data and Figure 6.5c and 6.5d are the validation data

Table 6.4: The mean and the standard deviation of the R^2 performance for the pH example in percent.

	Clustering		**Iterative**		**Deletion**	
m_p (%)	Mean	Stdv	Mean	Stdv	Mean	Stdv
0			99.87	0.81		
10	89.79	0.66	**90.49**	**0.70**	71.68	42.89
20	**89.75**	**0.82**	89.47	1.35	66.01	51.08
30	89.18	1.77	**89.21**	**1.56**	−126.56	1808.62
40	88.31	3.56	**88.91**	**1.64**	−106.52	912.33

Table 6.5: The mean and standard deviation of the computation time for the pH example in seconds.

	Clustering		**Iterative**		**Deletion**	
m_p (%)	Mean	Stdv	Mean	Stdv	Mean	Stdv
0			0.08	0.01		
10	0.13	0.02	428.58	257.48	**0.10**	**0.03**
20	0.16	0.02	1012.70	752.89	**0.10**	**0.03**
30	0.16	0.03	1379.07	1418.39	**0.08**	**0.03**
40	0.16	0.02	1474.77	1367.76	**0.06**	**0.01**



(a) R^2 all missing fractions

(b) R^2 10and 20% missing

(c) R^2 30% missing

(d) R^2 40% missing

(e) Computation time

Figure 6.6: The R^2 performance and the computation time for the pH example. The blue lines and markers are from the clustering method, the red ones are from the iterative method, and the green ones are from the deletion method.

mance of the identified models and Figure 6.6e and Table 6.5 for the computation time. The R^2 performance is presented in different figures to accommodate the large performance discrepancy for the 30% and 40% of missing data.

Figure 6.6b to 6.6d show that the clustering method and the iterative method produce models with better performance than those produced by the deletion method. As can be seen from Table 6.4, for the models from the clustering method and the iterative method, the R^2 performance has a mean of more than 88% with a standard deviation of less than 5%, while some of the models obtained using the deletion method have a negative R^2 performance mean and a standard deviation larger than 40%. The performance of models using the clustering method and the iterative method decreases as the fraction of missing data increases, but this does not occur for the deletion method.

The computation time for the clustering method and deletion method is generally much lower than that for the iterative method, with the clustering method being the fastest. Similar to the previous example, a larger fraction of missing data does not always lead to a higher computation time. For this example, however, the iterative method shows that the mean and standard deviation of the computation time increase with the fraction of missing data.

6.6.3. HEAT TRANSFER

In this section, the input-output data is from a simple heat transfer system. The data are obtained from a lab-scale real setup consisting of a 30 cm tube with a fan and an air valve, a heating resistor, and a temperature sensor as illustrated in Figure 6.7. The heating resistor and the fan are located at one end and the sensor is at the other end of the tube. The input voltage over the resistor is $u(k) \in [0\,\text{V}, 12\,\text{V}]$ and the output voltage that corresponds to the air temperature is $y(k) \in [0\,\text{V}, 10\,\text{V}]$. The airflow inside the tube is manually controlled by the opening of the valve $v(k) \in [0°, 180°]$. In this example, the valve opening is set to a constant value of 20°, and the input is random, covering the full range of operation. The first-principles derivation and the details of the process are described in [122].

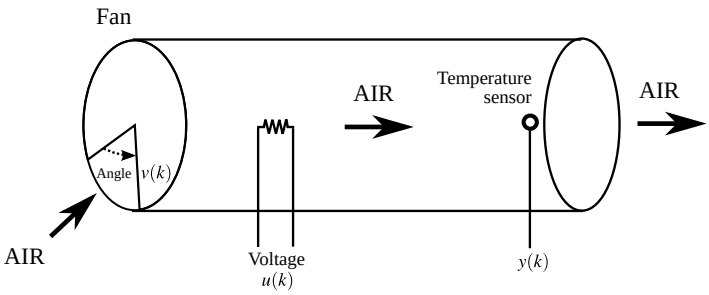
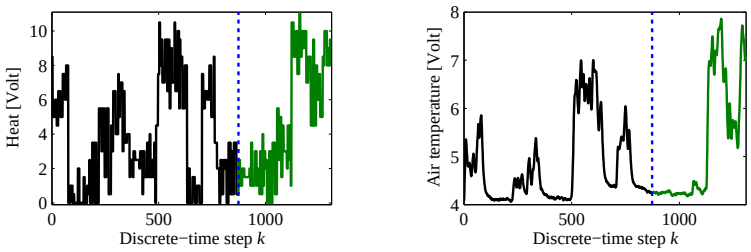


Figure 6.7: Schematic illustration of the heat transfer setup [122].



(a) Input (b) Output

Figure 6.8: Input-output data set of the heat transfer example. The identification data are to the left of the vertical dotted line, while the validation data are to the right.

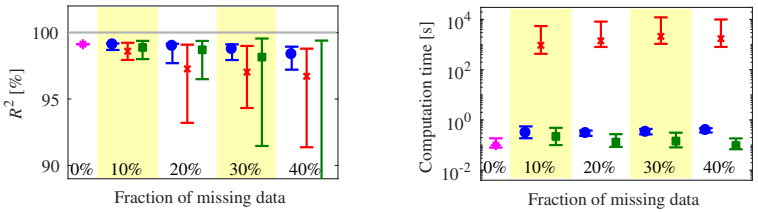
The data set shown in Figure 6.8 contains 1300 samples. The first two-thirds of them, which are 800 samples, are used for identification, and the rest are for validation. The TS fuzzy model has 3 rules with cluster fuzziness 2 [16]. Antecedent membership functions are computed by projecting the fuzzy partition onto the antecedent variables, and consequent parameters are computed by the least-squares method for each rule with regressor vector $\boldsymbol{\phi}(k) = \begin{bmatrix} y(k-1) & y(k-2) & y(k-3) & u(k-4) \end{bmatrix}^\top$ [16].

Figure 6.9a and Table 6.6 show the R^2 values for the validation data. All methods result in models with a sufficiently good mean and standard deviation, except for the deletion method with a mean R^2 value of 28.76% and a standard deviation of 650.14%. It can also be seen that increasing the missing data fraction results in deterioration of the performance of the models, although the reduction seems small except for the deletion method with a missing data level of 40%. In this example, the clustering method outperforms the iterative and deletion methods for all fractions of missing data. The deletion method performs better than the iterative method except for a missing data fraction of 40%.

The computation time is shown in Figure 6.9b and Table 6.7. Similar to the previous examples, the iterative method generally needs much more time than the clustering and iterative methods. It is also observed that a larger missing data fraction does not necessarily imply a larger computation time.

6.6.4. DISCUSSION OF THE RESULTS

The results and observations are discussed in this section. Recall that the deletion method is used as a benchmark for the other methods, as this is a simple solution obtained by removing rows with



(a) R^2 performance

(b) Computation time

Figure 6.9: The R^2 performance and the computation time for the heat example. The blue lines and markers are from the clustering method, the red ones are from the iterative method, and the green ones are from the deletion method.

Table 6.6: The mean and the standard deviation of the R^2 performance for the heat example in percent

	Clustering		**Iterative**		**Deletion**	
m_p (%)	Mean	Stdv	Mean	Stdv	Mean	Stdv
0			99.12	0.00		
10	99.09	0.06	98.59	0.36	**98.88**	**0.29**
20	**98.97**	**0.19**	97.26	0.85	98.70	0.57
30	**98.74**	**0.24**	97.02	0.95	98.15	1.10
40	**98.35**	**0.37**	96.71	1.25	28.76	650.14

6

missing elements from the regressor matrix. By using the deletion method, one can inspect the implications of removing rows containing missing samples on the performance of the resulting models.

As commonly known, missing measurements cause information decline for building models. By applying different realizations (simulations) of missing measurements, the resulting models might be different from one realization to the other. This is clearly shown in Example 3 for the deletion method, where 40% of missing samples gives a relatively large difference between the R^2 performance for the complete data and the incomplete data. The standard deviation of the R^2 performance is then also very large. Using this view for other examples, the data in the toy problem might be highly redundant and taking a relatively small number of measurements might be sufficient to have a model that is comparable to the one obtained with a large number of samples. The heat example shows the same pattern until 30% of missing data.

In case a sufficiently good model can be obtained by using a data set with a large number of missing samples, it can be assumed that the complete data set contains redundant information

Table 6.7: The mean and standard deviation of the computation time for the heat example in seconds

	Clustering		**Iterative**		**Deletion**	
m_p (%)	Mean	Stdv	Mean	Stdv	Mean	Stdv
0			0.10	0.01		
10	0.30	0.07	934.81	920.46	**0.22**	**0.06**
20	0.29	0.03	1408.54	1202.65	**0.13**	**0.03**
30	0.33	0.03	2118.35	2332.64	**0.14**	**0.05**
40	0.39	0.03	1718.41	1444.28	**0.10**	**0.02**

Table 6.8: The R^2 performance and computation time with complete data for 100 different clustering initialization.

	R^2				**Computation time [s]**			
Example	Mean	Stdv	Max	Min	Mean	Stdv	Max	Min
Toy	99.53	2.45×10^{-4}	99.53	99.53	0.05	0.21	2.15	0.02
pH	93.87	0.81	95.22	92.72	0.08	0.01	0.15	0.06
Heat	99.12	2.00×10^{-3}	99.12	99.12	0.01	0.01	0.19	0.08

for performing identification. This can be seen in the toy problem example. In that example, the performance difference between 10% and 40% missing data is less than 3%. The proposed methods, i.e., the clustering method and the iterative method, can be seen as an approach to recover the missing samples and the information by using the available data. This is done by adjusting the value of the missing samples to solve the optimization problem (6.11) for the available samples.

With respect to the computation time, it is clear from all examples that a larger missing data level does not necessarily imply a longer optimization duration. The mean of the computation time and the standard deviation as well do not always increase when the level of missing data increases. Intuitively, an increase in the number of missing data enlarges the number of optimization variables of (6.11) and consequently increases the optimization duration. For this situation, one should realize that the TS system identification with product space clustering employs the clustering algorithm in which the regressor matrix undergoes an iterative process to minimize a class of fuzzy c-means cost functions. Although the convergence of the algorithm has been proven [116], global optimality is not guaranteed. In addition, the number of iterations for convergence depends on the initial condition, which is typically calculated from the regressor matrix. By selecting the initial condition based on the regressor matrix, different missing sample sets generate different initial conditions. Consequently, clustering a regressor matrix involving different sets of missing samples takes different computation times to converge. In addition, the performance of the resulting models is different as well. Hypothetically, some missing sample sets might cause the clustering of the regressor matrix to require a higher number of iterations, thus increasing the computation time. Indeed, different clustering initializations also influence the identification process of the complete data. Table 6.8 shows the identification of the complete data with 100 different clustering initializations for all examples. It is evident that the different clustering initializations result in models with different performances that need slightly different computation times.

Based on the examples in the previous sections, one may see the trade-off between the performance of the resulting models and the computation time in the identification of a TS fuzzy system by using an incomplete data set. This leads to the following order of algorithms is suggested:

1. The deletion method. This method should first be applied, especially for data sets with a small fraction of missing samples. The simple algorithm requires low computation time but can result in sufficiently good models. Different initial conditions for the clustering step should be examined to search for the best model.
2. The clustering method. This method should be employed in case the deletion method does not return an acceptable model. The computation time of this method is slightly higher than that of the deletion method. However, the resulting models are somewhat better for a larger missing data fraction. Trials with different clustering initial conditions should also be performed to get better results.
3. The iterative method. This method is the last choice in case the clustering method cannot return an adequate model either. This method requires the highest computation time of all, but the resulting models would have a better performance. Experiments with different

clustering initial conditions are also suggested.

6.7. SUMMARY

The problem of the TS fuzzy system identification using the product space clustering method when the identification data set contains missing samples has been presented and discussed in this chapter. Two methods are introduced to overcome the problem of missing samples in system identification data. The proposed methods approximate the missing samples to enable the use of identification methods with complete data. The first method includes an extension to fuzzy c-means clustering for classification of incomplete data to use the data for system identification. The second proposed method uses the idea of alternating optimization to minimize the prediction error of the measured samples. The effectiveness of both methods has been demonstrated on three examples: a toy data set, pH neutralization data, and a real-application data set of a heat transfer process. The method of removing the incomplete regressor matrix rows is used as a benchmark. The experiment shows that the proposed methods are able to deliver better R^2 performance models. With respect to computation time, the first method requires a lower computation time than that of the second one. Analysis and recommendations have also been presented for those who want to implement the method for their own applications.

7

CONCLUSIONS AND RECOMMENDATIONS

In this thesis, several problems related to state estimation and system identification have been presented and discussed. This chapter summarizes the contributions of the thesis in Section 7.1 followed by the details of the contributions. Recommendations and some directions to extend the thesis results are indicated in Section 7.2.

7.1. CONTRIBUTIONS AND CONCLUSIONS

The main contributions of this thesis on system identification can be summarized as follows:

1. We have developed a practical identification method for spatiotemporal systems when only sparse measurements are available (Chapter 4). Using the developed method, one can build a finite-dimensional model from data from a small number of sensors compared to the spatial size.
2. We have proposed a method to select important regressors using tree representations for linear and nonlinear system identification (Chapter 5). The proposed method examines the regressor selection problem as a combinatorial optimization and searches for the solution based on the tree representation.
3. We have introduced techniques for TS fuzzy system identification in case of incomplete identification data (Chapter 6). The first technique solves the missing samples in the antecedent step by improving the clustering algorithm to estimate the missing samples. The second technique takes an optimization view by treating the missing samples as optimization variables and alternatingly optimizing the estimation of missing samples and the model output.

There are also additional results in state estimation:

1. We have reviewed the literature on state observer design for linear distributed-parameter systems (Chapter 2). Different state observer designs found in the literature have been presented. We also identified open problems for future research on the topic.
2. We have compared some implementations of the distributed Kalman filter applied to a linear distributed-parameter system (Chapter 3).

3. We have developed a discrete-time TS fuzzy observer for the METANET traffic model (Appendix A).

STATE ESTIMATION

The observer design problem for distributed-parameter systems is a more complicated task compared to that for lumped-parameter systems. The infinite-dimensional nature of the problem leads to two fundamental approaches to design observers: early lumping and late lumping. Utilizing any of the two approaches, several observer design methods have been derived in literature, with some of them extending to lumped-parameter systems. Therefore, based on the two approaches, techniques can be developed to design an observer for distributed-parameter systems.

Spatial discretization of distributed-parameter systems results in coupled lumped-parameter systems. We have compared methods in distributed Kalman filters and used the centralized method as a benchmark. The advantage of a distributed Kalman filter lies in the distribution of the estimate computation among different nodes of filters. The computation distribution increases the robustness to the failure of filter nodes. The other advantage is the flexibility in designing the connection topology. In addition, extension to an adaptive topology is also possible, for instance, by dynamically moving the sensors to cover the system spatially. However, only when the Kalman filter nodes are fully connected is the performance equal to that of the centralized Kalman filter.

SYSTEM IDENTIFICATION

Identification of distributed-parameter systems can be performed with early lumping by finite-difference method. The spatial discretization results in grid points and a set of coupled models corresponding to the grid points. A grid point at which a sensor or an actuator is deployed is called a node. The lumped-parameter system model at each grid point is coupled with the models at the adjacent grids. It is common to create a relatively small grid size to increase the accuracy of the model. Assuming all grid points are nodes, model reduction is performed to obtain simple models, models that depend on a small number of nodes.

In the case of a small number of nodes at irregular locations, a model corresponding to a node can be identified by also employing the measurements from the surrounding nodes with a larger lag to compensate for the spatial propagation of the system variable. With this approach, the resulting models have a large number of parameters because of both the number of surrounding nodes and the time lag. Subsequently, the number of parameters is reduced to obtain reasonable complex models. This identification method, described in Chapter 4, is a pragmatic approach. It can be applied to non-sophisticated systems where investment in a large number of sensors is not viable. For instance, the system spatially covers a wide area.

A model with a large number of parameters may contain parameters with low significance to the performance of the model. In most cases, simplifying the model to decrease the number of parameters is desired to simplify the model. Model simplification selects a subset of regressors that gives a higher model performance than other subsets. In Chapter 5 of this thesis, the selection is performed incrementally by building a tree consisting of regressor combinations with different subset sizes. The tree is built greedily, and in each step, several combinations can be added to the tree simultaneously.

The tree representation allows a flexible way of combinatorial regressor selection. Depending on how the tree is built, one can mimic a typical regressor selection approach such as stepwise regression or exhaustive search. One can adjust how the tree is built to control the selection process. In addition, the search is not limited to using a certain model structure. As the tree represents subsets of regressors, their performance can be evaluated using any pre-selected structure. The ability to assess the potential of the regressors for a given structure is the advantage of the combinatorial approach for regressor selection.

The regressor selection in this thesis assumes that the important regressors in a model follow the pre-selected structure. Compared to linear systems, nonlinear systems have more model structures to apply. This gives more flexibility in selecting which structure is used in the model, but this may lead to structure-dependent important regressors. In this case, the method in Chapter 5 is useful, especially for nonlinear system identification.

Missing samples in identification data create missing information in the data and can be seen as missing pieces of information to build the model. The unavailable pieces might be insignificant in case the samples are abundant, and one can just ignore them. In this thesis, TS fuzzy system identification is extended to use incomplete identification data. The missing samples are imputed in the clustering step by modifying a fuzzy c-means clustering algorithm. The modified algorithm results in a single estimate of a missing sample at a different location of the regressor matrix.

An optimization approach for identification using an incomplete data set is also presented in this thesis. Missing samples can then be viewed as optimization variables. The optimization problem is then solved by alternating between the missing sample estimation and model parameter estimation. In other words, the missing samples are estimated using the model obtained in the parameter estimation step. These two steps are iterated until the model performance is convergent.

7.2. RECOMMENDATIONS FOR FUTURE RESEARCH

There are several ways to extend the methods presented in this thesis. One of them is the application to different real-world problems. Examples include large-scale infrastructure networks that cover a wide area with only a few available measurements, such as smart grid networks [240]. Connecting different types of power generators, this system involves a number of states that are essential to monitor the power delivery from generators to end consumers. Another possible application is pipeline networks, e.g., gas transmission [1].

The following subsections provide further recommendations and point out potential research directions.

STATE ESTIMATION

Observer design heavily depends on the plant model, while it is known that the plant model always contains uncertainty due to modeling limitations. In this case, observer design methods for distributed-parameter systems that are robust to model uncertainty are important for reliable state estimates. Robust observer design methods for distributed-parameter systems might be developed by extending the methods for the lumped-parameter system, e.g., the methods of [48] or [227] for robust observer design; or they can be developed directly based on the distributed-parameter model.

Distributed-parameter systems that have been discretized both in time and space can be considered high-dimensional lumped-parameter systems. Moreover, for a spatially large system, the spatially temporally discretized system will have a large number of elements in the state vector. Therefore, fast and reliable numerical computation methods to estimate the state of the system are of great importance to this field.

From a computational point of view, several aspects affect the computation load; the number of sensor nodes, the network topology, and the quality of the sensors. Adding more sensors means adding more local measurements, which requires more computing resources for estimation. A complex topology requires more communication among sensors and, consequently, increasing computation to process the data from the neighboring sensors. Adaptive topology switching by considering computation and/or communication load will ease the implementation of the state estimator using sensors with different sophistication level.

The quality of the data communicated from neighboring sensors influences the quality of the

resulting estimates. In real-life cases, noisy data and missing data are not uncommon, especially for low-quality sensing devices. Highly noisy data introduce high uncertainty and, therefore, require a more robust estimation algorithm. Missing or unavailable data further complicates the estimation process by requiring data imputation. A robust algorithm to tackle these issues together is essential to widen the application spectrum of distributed state estimation.

From the distributed Kalman filter comparison, an important future research topic concerns spatially dynamic measurements. In this case, mobile measurement devices move cooperatively across a large spatial system. The movements should ensure that all the states, or at least those of some important locations, are estimated. The movement should consider the observability and controllability of the system and the spatial dynamics as well. Another relevant topic for moving sensors involves selecting the number of optimal measurements and the corresponding locations. The selection should also consider, e.g., the analysis of the system's observability.

SYSTEM IDENTIFICATION

Typically, an important purpose of system identification is to control the system. After obtaining the model, one should be able to design a controller on top of it. From the approach in Chapter 4, one may ask how to use the identified model to design a controller or an observer. Models from each sensor can be stacked to form a state space representation, where the measurements at sensor locations represent the states of the system. From the fact that the states are coupled across different measurement locations and the number of states may be very large, the question is how straightforward it is to apply available control design methods for the identified model while preserving the control performance and computational practicability.

Another related question concerns the choice of dominant neighbors, the measurements that highly influence the model and, consequently, the measurements from other neighbors that can possibly be neglected. The neighbor selection can be made dynamic by directly selecting dominant neighbors when the dynamics at a certain location is high, e.g., due to disturbances, and online control/estimation is required. In this case, we can directly simplify the model from the selection of neighbors, after which the model can even be simplified further by a regressor selection step. The greenhouse example considered in Chapter 4 is a limited implementation of this framework with a spatial stationary assumption.

For the regressor selection problem, the current implementation requires much of memory. It cannot handle the number of regressors that go significantly beyond the numbers presented in the example of Chapter 5, i.e., about 91 regressors. Therefore, one should consider designing software with efficient memory management to implement the tree with low memory resources. More experiments with real-life applications and different tree-building rules and comparisons with other methods will be useful to further evaluate the effectiveness of the method. Moreover, since the regressor selection method in this thesis is a heuristic method, a theoretical foundation is needed to support the validity of the proposed method analytically.

For the identification of the fuzzy system TS with an incomplete data set, for the antecedent identification step, the hyper-ellipsoid clusters give a better model performance than the hyper-sphere clusters resulting from fuzzy *c* means clustering. An extension of fuzzy clustering to handle incomplete samples beyond the fuzzy *c*-means method is still not explored, especially for algorithms that build hyper-ellipsoid clusters.

The convergence of alternating optimization for fuzzy *c*-means clustering has been proven; see, e.g. [23]. Although it worked well in the example of Chapter 6, it is necessary to provide a theoretical argumentation proof that the extension used for system identification keeps the convergence property. Furthermore, it could be assessed whether the same extension also works for other clustering algorithms employed in classification and system identification applications.

Another open problem is to establish a general theoretical framework for TS fuzzy identification

with incomplete data. The methods in Chapter 6 can be classified as two-step methods as they include the estimation of the missing samples; so they will generally be slow. A clustering algorithm that can handle missing samples and, at the same time, results in hyper-ellipsoid clusters may improve the computation time significantly.

A

FUZZY OBSERVER FOR STATE ESTIMATION OF THE METANET TRAFFIC MODEL

A.1. INTRODUCTION

Traffic jams waste significant amounts of time and fuel and contribute to the deterioration of air quality and the environment. Hence, effective traffic control on freeways is necessary to reduce congestion. In this context, traffic control is an important component of the traffic management system that aims to make better use of the available infrastructure.

Appropriate traffic control actions must be based on the actual traffic state, which, however, is not always available at any point in the traffic network. Not all relevant state variables can be measured due to technical limitations, such as the sparse arrangement of sensors or the occurrence of sensor failures. Moreover, the available measurements are corrupted by noise. For these reasons, traffic state estimation is a very relevant topic regarding effective traffic control.

Designing a state estimator requires a traffic model. Traffic models are generally classified into microscopic, mesoscopic, and macroscopic models [115]. In the case of online model-based traffic control, it is common to use a macroscopic traffic flow model, see, e.g., [136, 166, 223]. Macroscopic models express the average behavior of vehicles at specific locations and time instants. Such a model is typically nonlinear and captures the average traffic behavior through aggregated variables at different locations in the network [115]. The variables used in macroscopic models include flow, density, and speed. The model used in this appendix is the well-known METANET model [183], selected because it is frequently employed in model-based freeway traffic control [204, 239].

Among the methods applied to traffic state estimation are the extended Kalman filter (KF) [232], the unscented Kalman filter [168], and the particle filter [169]. In [233], an adaptive approach to the extended Kalman filter was used. In [102], different filter configurations are compared for the case of traffic flow estimation and parameter estimation.

A common limitation of the above approaches is the lack of convergence guarantees. Although a well-tuned extended KF, unscented KF, or particle filter can perform well in simulations, there is no guarantee that they will perform equally well in real-life situations. In this appendix, we propose an alternative approach that is based on transforming the METANET traffic model into a Takagi–Sugeno

A

(TS) fuzzy model representation and consequently applying a systematic observer design method with stability guarantees. The TS model [214] is a general function approximator that can exactly represent or approximate to an arbitrary degree of accuracy a large class of nonlinear systems. The TS model consists of fuzzy if-then rules. The rule antecedents partition a given subspace of the model variables into fuzzy regions. The consequents of the rules are linear or affine models that are valid locally in the corresponding fuzzy region.

In the literature, there are several approaches to designing TS fuzzy observers for continuous-time systems [21] and for discrete-time systems [215]. The design of TS fuzzy observers is formulated as a feasibility problem of a Linear Matrix Inequality (LMI), which can be solved by convex optimization algorithms.

In this appendix, we develop a TS fuzzy observer for the METANET traffic flow model. The design starts by transforming the METANET model into a TS fuzzy model. Then a discrete-time fuzzy observer is designed by applying stability and robustness conditions. This method is the discrete-time counterpart of the approach proposed in [146]. While in [146] the METANET model was first transformed into a continuous-time model, here we design the TS observer directly in the discrete-time setting. This is a much more realistic approach, as the METANET model is essentially a discrete-time model validated for sampling times that are typically in the order of 10 s. In addition, the measurements are available at discrete time instants as well.

A.2. PRELIMINARIES

In this section, we briefly review the METANET model.

A.2.1. THE METANET TRAFFIC MODEL

In this section, we present the macroscopic traffic flow model METANET developed in [183]. In METANET, three state variables reflect the behavior of the traffic, namely, [183]

- Traffic density ρ: the number of vehicles per length unit and per lane in a freeway segment,
- Space-mean speed v: the instantaneous average speed of vehicles per length unit in a freeway segment,
- Traffic volume or flow q: the number of vehicles leaving a freeway segment per time unit.

The METANET model represents a freeway network as a directed graph whose links are associated with stretches in the freeway network. Each link in the graph corresponds to a stretch that has uniform characteristics. A node is placed in the graph when there is a change in the geometry, such as an on-ramp or a split.

The METANET model is discrete in time and space. In the model, the m-th link of a freeway is divided into N segments of length L_m. For each link m and segment I, the state variables of the traffic as described above are expressed as the average density $\rho_{m,i}(k)$, the space-mean speed $v_{m,i}(k)$, and flow $q_{m,i}(k)$. The definitions of the variables that are used in the METANET model are in Table A.1 while the parameters and their typical values (as used in the case study in Section A.5) are listed in Table A.2. The values of the parameters have been adapted from [137].

In the segment I of link m, the flow at time step k is determined by the speed, density, and the number of lanes:

$$q_{m,i}(k) = \rho_{m,i}(k) \cdot v_{m,i}(k) \cdot \lambda_m \tag{A.1}$$

where λ_m is the number of lanes in the corresponding segment. At time step $k+1$, the density of segment I is influenced by the density at time step k, the number of vehicles entering from segment $i-1$ (inflow), and the number of vehicles leaving segment I (outflow). This relationship can be expressed as

$$\rho_{m,i}(k+1) = \rho_{m,i}(k) + \frac{T}{L_m \lambda_m}\big(q_{m,i-1}(k) - q_{m,i}(k)\big) \tag{A.2}$$

Table A.1: Variables in the traffic model.

Symbol	Variable	Units
k	time step	–
i	segment index	–
$\rho_{m,i}(k)$	traffic density	veh/km/lane
$v_{m,i}(k)$	space-mean speed	km/h
$q_{m,i}(k)$	traffic volume or flow	veh/h

where T is the sampling time, which typically has a value of 10 s. The space-mean speed of the segment i at time step $k+1$ is influenced by three terms, expressing relaxation, convection, and anticipation. The relaxation term expresses the speed change to achieve a desired equilibrium speed $V(\rho_{m,i}(k))$ corresponding to the density $\rho_{m,i}(k)$. This term is proportional to the difference between the current space-mean speed and $V(\rho_{m,i}(k))$. The convection term expresses the speed difference between the segment i and the upstream segment $i-1$. The anticipation term is the speed change due to the density change when moving from the upstream segment $i-1$ to the downstream segment i. Using these terms, the space-mean speed at time step $k+1$ can be written as

$$\begin{aligned} v_{m,i}(k+1) = v_{m,i}(k) &+ \frac{T}{\tau}\left[V(\rho_{m,i}(k)) - v_{m,i}(k)\right] \\ &+ \frac{T}{L_m} v_{m,i}(k)\big(v_{m,i-1}(k) - v_{m,i}(k)\big) \\ &- \frac{\nu \cdot T}{\tau \cdot L_m} \frac{\rho_{m,i+1}(k) - \rho_{m,i}(k)}{\rho_{m,i}(k) + \kappa_{\mathrm{M}}} \end{aligned} \tag{A.3}$$

where

$$V(\rho_{m,i}(k)) = v_{\mathrm{f},m} \cdot \exp\left[-\frac{1}{a_m}\left(\frac{\rho_{m,i}(k)}{\rho_{\mathrm{cr},m}}\right)^{a_m}\right] \tag{A.4}$$

Note that the METANET model (A.1)–(A.4) presented above is the basic model without geometry changes such as on-ramp, off-ramp, split, or merge. However, the model can be extended to include those cases (see [183] for details). In the sequel, for the sake of simplicity but without loss of generality, we consider only one link, and therefore the index m is dropped.

A.3. TS FUZZY REPRESENTATION OF THE METANET MODEL

In this section, a TS fuzzy model is developed that exactly represents the METANET model presented in the previous section. Before the TS model is derived, the set of equations of the METANET model, i.e., (A.1), (A.2), (A.3), and (A.4), have to be written as a state space representation of a nonlinear system. Since, in general, TS fuzzy models do not use algebraic equations, $q_i(k)$ in (A.1) is eliminated by substituting it into (A.2). The information needed from the neighboring segments, namely $\rho_{i+1}(k)$ and $v_{i-1}(k)$, is treated as input to the model. After some algebraic manipulations, the METANET model can be rewritten as a state equation with an affine term as follows:

$$\begin{aligned} \begin{pmatrix} \rho_i(k+1) \\ v_i(k+1) \end{pmatrix} = &\begin{pmatrix} 1 - \frac{T}{L} v_i(k) & 0 \\ \frac{\nu T}{\tau \cdot L} \frac{1}{\rho_i(k) + \kappa_{\mathrm{M}}} & 1 - \frac{T}{\tau} - \frac{T}{L} v_i(k) \end{pmatrix} \begin{pmatrix} \rho_i(k) \\ v_i(k) \end{pmatrix} \\ &+ \begin{pmatrix} \frac{T}{L} \rho_{i-1}(k) & 0 \\ \frac{T}{L} v_i(k) & -\frac{\nu T}{\tau \cdot L} \frac{1}{\rho_i(k) + \kappa_{\mathrm{M}}} \end{pmatrix} \begin{pmatrix} v_{i-1}(k) \\ \rho_{i+1}(k) \end{pmatrix} \\ &+ \begin{pmatrix} 0 \\ \frac{T}{\tau} V(\rho_i(k)) \end{pmatrix} \end{aligned} \tag{A.5}$$

Table A.2: Parameters of the traffic model and the values used in the case study of Section A.5.

Symbol	Parameter	Value	Units
L_m	length of segment	0.5	km
λ_m	number of lanes	3	–
$v_{\mathrm{f},m}$	free-flow speed	102	km/h
$\rho_{\mathrm{cr},m}$	critical density	30	veh/km/lane
τ	time constant	18	s
ν	anticipation constant	60	km^2/h
κ_{M}	constant	40	veh/km
a_m	parameter	2.34	–
$v_{\min}$	minimum velocity	7.4	km/h
$v_{\max}$	maximum velocity	200	km/h
$\rho_{\min}$	minimum density	0	veh/km/lane
$\rho_{\max}$	maximum density	150	veh/km/lane
T	sampling time	10	s

$$V(\rho_i) = v_{\mathrm{f}} \cdot \exp\Big[-\frac{1}{a_m}\Big(\frac{\rho_i}{\rho_{\mathrm{cr}}}\Big)^{a_m}\Big]$$

The development of the TS fuzzy representation of this model is presented in the next section.

A.3.1. TS FUZZY MODEL CONSTRUCTION

Consider a nonlinear system described by the following state space model[1]:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{f}(\boldsymbol{z})\boldsymbol{x}(k) + \boldsymbol{g}(\boldsymbol{z})\boldsymbol{u}(k) + \boldsymbol{a}(\boldsymbol{z}) \\ \boldsymbol{y}(k) &= \boldsymbol{\hbar}(\boldsymbol{z})\boldsymbol{x}(k) \end{aligned} \tag{A.6}$$

Here, $\boldsymbol{f}$, $\boldsymbol{g}$, $\boldsymbol{a}$, and $\boldsymbol{\hbar}$ are smooth nonlinear matrix and vector functions, respectively, $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u} \in \mathbb{R}^m$ the input vector, $\boldsymbol{y} \in \mathbb{R}^q$ the measurement vector, and $\boldsymbol{z} = \begin{bmatrix} z_1(k) & \cdots & z_p(k) \end{bmatrix}^\top$ a given vector function[2] of $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{u}$; $\boldsymbol{z}$ is called the vector of scheduling variables. All variables $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{u}$ are assumed to be bounded and to belong to a compact set $\mathbb{C}_{xyu}$.

An approximation of TS representation to a nonlinear system can be obtained using the sector nonlinearity approach [176]. The basic idea of the sector nonlinearity approach is to represent each of the nonconstant terms in the matrix functions $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{\hbar}$, and the vector function $\boldsymbol{a}$ of the model as the convex combination of two constant terms and to build the set of fuzzy rules as all combinations of the so-obtained terms. Therefore, in this approach, the number of rules is determined by the number of nonconstant terms in the matrix functions.

First, consider the state equation of the nonlinear system (A.6). The sector nonlinearity approach requires the nonlinear functions to be bounded. Therefore, we consider the nonconstant terms in either $\boldsymbol{f}$, $\boldsymbol{g}$, or $\boldsymbol{a}$ of (A.6), and we represent them by $\mathrm{nl}_j(\cdot) \in [\underline{\mathrm{nl}}_j, \overline{\mathrm{nl}}_j]$, $j = 1, 2, \ldots, p$ where $\underline{\mathrm{nl}}_j$ and $\overline{\mathrm{nl}}_j$ are respectively the lower and upper bound of the j-th term. Now, for each nonlinearity

[1] In the output equation we use $\boldsymbol{\hbar}$ to denote the nonlinear function and not h as could be expected, since the symbol h is used to denote membership functions.

[2] Each element of the vector $\boldsymbol{z}$ is time-dependent, i.e., $\boldsymbol{z}$ should be denoted as $\boldsymbol{z}(k)$. For the simplicity of notation, the explicit time-dependence is omitted in this appendix.

nl_j, we construct two weighting functions as follows

$$w_0^j(\cdot) = \frac{\overline{\mathrm{nl}}_j - \mathrm{nl}_j(\cdot)}{\overline{\mathrm{nl}}_j - \underline{\mathrm{nl}}_j} \qquad w_1^j(\cdot) = 1 - w_0^j(\cdot) \qquad \text{for } j = 1, 2, \ldots, p \tag{A.7}$$

We can see that for each nonconstant term, the two weighting functions w_0^j and w_1^j are normalized, i.e., $w_0^j(\mathrm{nl}_j(\cdot)) + w_1^j(\mathrm{nl}_j(\cdot)) = 1$, for any $\mathrm{nl}_j(\cdot)$. To define the membership functions we consider all possible products of the weight functions w_ℓ^j for $j \in \{1, 2, \ldots, p\}$ and $\ell \in \{0, 1\}$. This results in 2^p membership functions of the form

$$h_i(\boldsymbol{z}) = \prod_{j=1}^{p} w_\ell^j(z_j) \tag{A.8}$$

for $i = 1, 2, \cdots, 2^p$, $\ell \in \{0, 1\}$. These membership functions are normal, i.e., $h_i(\boldsymbol{z}) \geq 0$ and $\sum_{i=1}^{r} h_i(\boldsymbol{z}) = 1$, $r = 2^p$, where r is the number of rules. Then the fuzzy representation of the state update equation in (A.6) is given as

$$\boldsymbol{x}(k+1) = \sum_{i=1}^{r} h_i(\boldsymbol{z})\big(A_i \boldsymbol{x}(k) + B_i \boldsymbol{u}(k) + \boldsymbol{a}_i\big) \tag{A.9}$$

where A_i, B_i, and $\boldsymbol{a}_i$, $i = 1, 2, \ldots, r$ are matrices and vectors of proper dimensions, obtained by substituting the nonlinear terms $\mathrm{nl}_j(\cdot)$ by either $\underline{\mathrm{nl}}_j$ or $\overline{\mathrm{nl}}_j$ depending on whether w_0^j or w_1^j is selected.

The TS representation of the output function can be obtained in a similar way.

A.3.2. TS FUZZY REPRESENTATION OF THE METANET MODEL

We can see that the METANET model (A.5) does not have the form (A.6). To construct the TS fuzzy representation of (A.5) using the sector nonlinearity approach, it is necessary to assume that the values of the variables $\rho_{i-1}(k)$, $\rho_i(k)$, and $v_i(k)$ are bounded, $\rho_{i-1}(k) \in [\rho_{i-1,\min}, \rho_{i-1,\max}]$, $\rho_i(k) \in [\rho_{i,\min}, \rho_{i,\max}]$, and $v_i(k) \in [v_{i,\min}, v_{i,\max}]$, for all k. This assumption is reasonable since a freeway segment always has capacity limits. Furthermore, when the segment is in congestion, the space-mean speed will be very small, while there is always an upper limit of the speed of a car on a freeway.

First, we consider the state equation of (A.5). There are four nonconstant terms in the matrix functions $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{a}$, based on which the weighting functions are defined as follows:

1. For the term $1 - \frac{T}{L} v_i(k)$, the space-mean speed $v_i(k)$ has a maximum and minimum value of $v_{i,\max}$ and $v_{i,\min}$ respectively. Applying (A.7), one obtains

$$w_0^1(v_i(k)) = \frac{v_{i,\max} - v_i(k)}{v_{i,\max} - v_{i,\min}} \tag{A.10}$$

and

$$w_1^1(v_i(k)) = 1 - w_0^1(v_i(k)) \tag{A.11}$$

Note that $1 - \frac{T}{L} v_i(k)$ and $1 - \frac{T}{\tau} - \frac{T}{L} v_i(k)$ lead to the same weighting function;

2. Similarly to the above, $\frac{1}{\rho_i(k) + \kappa_M}$ leads to

$$w_0^2(\rho_i(k)) = \frac{\rho_i(k) - \rho_{i,\min}}{\rho_i(k) + \kappa_M} \frac{\rho_{i,\max} + \kappa_M}{\rho_{i,\max} - \rho_{i,\min}}$$
$$w_1^2(\rho_i(k)) = 1 - w_0^2(\rho_i(k))$$

3. The term $\exp\left[-\frac{1}{a_m}\left(\frac{\rho_i(k)}{\rho_{\text{cr}}}\right)^{a_m}\right]$ appearing in $V(\rho_i)$ is expressed using the weighting functions

$$w_0^3(\rho_i(k)) = \frac{\exp\left[-\frac{1}{a_m}\left(\frac{\rho_{i,\min}}{\rho_{\text{cr}}}\right)^{a_m}\right] - \exp\left[-\frac{1}{a_m}\left(\frac{\rho_i(k)}{\rho_{\text{cr}}}\right)^{a_m}\right]}{\exp\left[-\frac{1}{a_m}\left(\frac{\rho_{i,\min}}{\rho_{\text{cr}}}\right)^{a_m}\right] - \exp\left[-\frac{1}{a_m}\left(\frac{\rho_{i,\max}}{\rho_{\text{cr}}}\right)^{a_m}\right]}$$
$$w_1^5(\rho_{i-1}(k)) = 1 - w_0^5(\rho_{i-1}(k))$$

From the above four nonconstant terms to describe all possible combinations, result in a fuzzy system with $2^4 = 16$ rules.

Consider now the output equation:

$$\begin{aligned} \boldsymbol{y}(k) &= \begin{pmatrix} q_i(k) \\ v_i(k) \end{pmatrix} \\ &= \begin{pmatrix} v_i(k)\lambda & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \rho_i(k) \\ v_i(k) \end{pmatrix} \quad . \end{aligned} \tag{A.12}$$

The measurement matrix has one nonconstant term, namely, $v_i(k)$. However, since $v_i(k)$ in the measurement matrix is the same as that of the system equation, the same weighting function as above can be used. The speed $v_i(k)$ is also assumed to be measured, which means the membership functions of the measurement do not depend on the states that have to be estimated. The space-mean speed measurements can be approximated by the time-mean speed [234], which can be measured easily by, for instance, loop detectors.

Using the weighting functions developed above, the consequent model of the fuzzy rules can be written as

$$\begin{aligned} \boldsymbol{x}(k+1) &= A_i \boldsymbol{x}(k) + B_i \boldsymbol{u}(k) + \boldsymbol{a}_i \\ \boldsymbol{y}(k) &= C_i \boldsymbol{x}(k) \end{aligned} \tag{A.13}$$

where

$$\boldsymbol{x}(k) = \begin{pmatrix} \rho_i(k) \\ v_i(k) \end{pmatrix} \quad \boldsymbol{u}(k) = \begin{pmatrix} v_{i-1}(k) \\ \rho_{i+1}(k) \end{pmatrix}$$

and A_i, B_i, $\boldsymbol{a}_i$, and C_i are obtained by substituting the minimum or maximum values corresponding to the weighting functions used in rule i into the functions $\boldsymbol{f}$, $\boldsymbol{g}$, $\boldsymbol{\hbar}$, and $\boldsymbol{a}$.

The TS fuzzy model of the METANET is then expressed as

$$\begin{aligned} \boldsymbol{x}(k+1) &= \sum_{i=1}^{r} h_i(\boldsymbol{z}) \big(A_i \boldsymbol{x}(k) + B_i \boldsymbol{u}(k) + \boldsymbol{a}_i \big) \\ \boldsymbol{y}(k) &= \sum_{i=1}^{r} h_{\text{o},i}(\boldsymbol{z}) C_i \boldsymbol{x}(k) \end{aligned} \tag{A.14}$$

where $h_{\text{o},i}(\boldsymbol{z})$ is the membership function for the output equation, the same as (A.8). This concludes the TS fuzzy representation of the METANET model.

A.4. OBSERVER DESIGN FOR THE TS METANET MODEL

In general, an observer designed for the model (A.14) has the form

$$\begin{aligned} \hat{\boldsymbol{x}}(k+1) &= \sum_{i=1}^{r} h_i(\hat{\boldsymbol{z}}) \big[A_i \hat{\boldsymbol{x}}(k) + B_i \boldsymbol{u}(k) + a_i + K_i \big(\boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k) \big) \big] \\ \hat{\boldsymbol{y}}(k) &= \sum_{i=1}^{r} h_{\text{o},i}(\hat{\boldsymbol{z}}) C_i \hat{\boldsymbol{x}} \end{aligned} \tag{A.15}$$

where $\hat{\boldsymbol{z}}$ denotes the estimated scheduling vector and $K_i, i = 1,\ldots,r$, are the observer gains. The observer design problem is to calculate the values of $K_i, i = 1,\ldots,r$ such that the estimation error converges to zero. The estimation error can be written as

$$\boldsymbol{e}(k) = \hat{\boldsymbol{x}}(k) - \boldsymbol{x}(k) \tag{A.16}$$

Substituting (A.14) and (A.15) into (A.16) yields

$$\begin{aligned}\boldsymbol{e}(k) = &\sum_{i=1}^{r} h_i(\boldsymbol{z})\big[A_i\boldsymbol{x}(k) + B_i\boldsymbol{u}(k) + \boldsymbol{a}_i\big] \\ &- \sum_{i=1}^{r} h_i(\hat{\boldsymbol{z}})\big[A_i\hat{\boldsymbol{x}}(k) + B_i\boldsymbol{u}(k) + \boldsymbol{a}_i + K_i\big(\boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k)\big)\big]\end{aligned}$$

Adding to and subtracting from the right-hand side of the above equation $\sum_{i=1}^{r} h_i(\hat{\boldsymbol{z}})\big(A_i\boldsymbol{x}(k) + B_i\boldsymbol{u}(k) + \boldsymbol{a}_i\big)$, after some algebraic manipulations we obtain

$$\begin{aligned}\boldsymbol{e}(k+1) = &\sum_{i=1}^{r} h_i(\hat{\boldsymbol{z}})\big[A_i\boldsymbol{e}(k) - K_i\big(\boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k)\big)\big] \\ &+ \sum_{i=1}^{r} \big(h_i(\boldsymbol{z}) - h_i(\hat{\boldsymbol{z}})\big)\big[A_i\boldsymbol{x}(k) + B_i\boldsymbol{u}(k) + \boldsymbol{a}_i\big]\end{aligned} \tag{A.17}$$

Since the speed is measured, the membership functions of the measurement model do not depend on the estimated states. Therefore, we can rewrite (A.17) as

$$\begin{aligned}\boldsymbol{e}(k+1) = &\sum_{i=1}^{r}\sum_{j=1}^{r} h_i(\hat{\boldsymbol{z}})h_j(\boldsymbol{z})[A_i - K_iC_j]e(k) \\ &+ \sum_{i=1}^{r} \big(h_i(\boldsymbol{z}) - h_i(\hat{\boldsymbol{z}})\big)\big[A_i\boldsymbol{x}(k) + B_i\boldsymbol{u}(k) + \boldsymbol{a}_i\big]\end{aligned} \tag{A.18}$$

In order for the estimation error to converge to zero, the observer gains K_i have to be calculated such that the first term of (A.18) converges to zero and such that the disturbance due to the second term, $h_i(\boldsymbol{z}) - h_i(\hat{\boldsymbol{z}})$ becomes zero as $\hat{\boldsymbol{z}}$ approaches $\boldsymbol{z}$.

The observer gains K_i are usually computed using the stability conditions developed for TS systems. The estimation error dynamics (A.18) is asymptotically stable, i.e., the estimation error converges to zero if there exists a positive definite matrix P such that [215]:

$$\begin{aligned} G_{ii}^T P G_{ii} - P < 0 \\ \frac{(G_{ij} + G_{ji})^T}{2} P \frac{(G_{ij} + G_{ji})}{2} - P < 0\end{aligned} \tag{A.19}$$

for all i, j such that $i < j$ and such that $\exists\boldsymbol{z}$ s.t. $h_i(\boldsymbol{z})h_j(\boldsymbol{z}) \neq 0$, where $G_{ij} = A_i - K_iC_j$. The inequalities above can be transformed into the following LMI problem:

Find a positive definite matrix P and matrices M_i, where $M_i = PK_i, i = 1,\ldots,r$, such that

$$\begin{aligned} \begin{pmatrix} P & 2L_{ii}^T \\ 2L_{ii} & P \end{pmatrix} > 0 \\ \begin{pmatrix} P & (L_{ij} + L_{ji})^T \\ (L_{ij} + L_{ji}) & P \end{pmatrix} > 0\end{aligned} \tag{A.20}$$

for all i, j such that $i < j$ and such that $\exists\boldsymbol{z}$ s.t. $h_i(\boldsymbol{z})h_j(\boldsymbol{z}) \neq 0$ where $L_{ij} = (PA_i - M_iC_j)/2$.

The condition (A.20) above ensures the asymptotic stability of the first term of the right-hand side of (A.18). The asymptotic stability of (A.18) can be guaranteed using stability conditions for uncertain fuzzy systems (see [120]). Provided the initial estimate is close enough to the true state, (A.18) is stable [22].

The LMIs above can, e.g., be solved using the Sedumi solver of YALMIP [160]. Subsequently, the values of K_i are substituted into the observer model.

The approach presented above can be extended to include the node equations of the METANET model, which implies that the proposed approach is not only applicable to freeway stretches but also to (complex) freeway networks.

Under the condition of measured flow and nonzero flow speed in a segment, it is possible to design a fuzzy observer similar to (A.15) that can estimate both the speed and the density. This indicates that it is possible to design observers in a distributed fashion for a whole stretch or even a whole network, given that the neighboring observers communicate the estimated states among them.

A.5. SIMPLE CASE STUDY

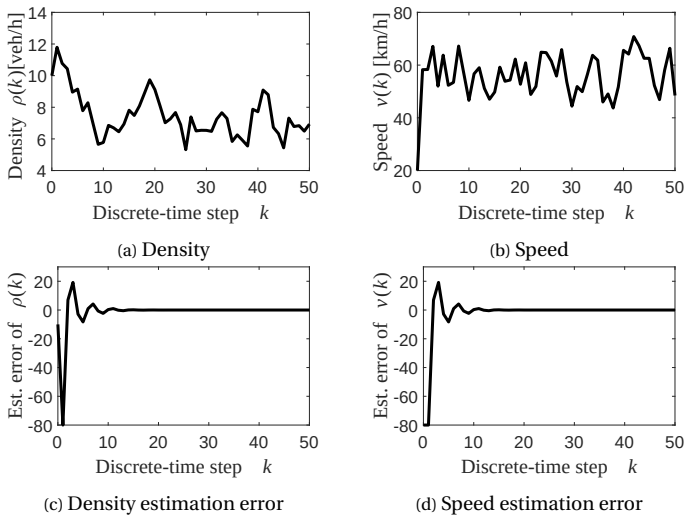
Now the proposed approach is illustrated by a simple case study in which we consider one segment i. The true initial state of the segment is $x = \begin{bmatrix} \rho & v \end{bmatrix}^\top = \begin{bmatrix} 10 & 20 \end{bmatrix}^\top$. The boundary inputs for the segment were constructed such that the downstream speed was equal to the initial speed of the segment plus uniformly distributed random noise between 0 and 15 km/h, and such that the upstream density was equal to the initial density of the segment plus uniformly distributed random noise between 0 and 15 veh/km/lane. The observer has been simulated using the initial estimate $x = \begin{bmatrix} 20 & 100 \end{bmatrix}^\top$. The output of the TS fuzzy representation of the METANET model is shown in Figure A.1a and A.1b. The estimation error using the observer is shown in Figure A.1c and A.1d. As expected, the estimation error converges to zero.

The simulation and estimation reported here have been performed on a PC with an Intel T9300 2.5 GHz processor and 3GB RAM. The total computation time, including the computation of the observer gains (1.75 s, done offline, before the actual estimation), simulation of the model, and estimation of the states was 2.26 s. Computing the estimate in one time step on average requires 0.0042 s, with 0.008 s being the maximum time that was encountered. These values are well below the typical sampling times for freeway traffic networks (which currently are typically in the range of several tens of seconds to minutes). This clearly indicates that the proposed observer is applicable online.

A.6. SUMMARY

A discrete-time Takagi–Sugeno (TS) fuzzy observer has been proposed for the METANET traffic model. An exact TS representation of the METANET model has been obtained using the sector nonlinearity approach as an illustration. The observer has been designed based on the TS fuzzy representation of the METANET model for one segment of highway stretch. The designed observer is able to estimate the non-measurable traffic states.

In our future research, we will investigate how the performance of the proposed observer compares to that of other types of observers that can be applied to the METANET model such as extended Kalman filter, unscented Kalman filters, or particle filters (see also [102, 168, 169, 232, 233]), in particular for models of real-life networks and using real measurement data as input. We will also consider a robust TS fuzzy observer design to handle uncertainties in the METANET model, as well as TS fuzzy observers for other traffic flow models.



(a) Density (b) Speed

(c) Density estimation error (d) Speed estimation error

Figure A.1: Estimation error using the TS fuzzy observer.

SYMBOLS AND NOTATIONS

This part contains a list of symbols and notations used throughout the thesis.

Symbol	Meaning
General	
$\mathbb{R}$	set of real numbers
$\mathbb{N}$	set of natural numbers
$\mathbb{N}^+$	set of positive natural numbers
$\top$	transpose operator of matrix and vector
$\mathscr{L}$	set of spatial coordinates
$\mathscr{L}_\mathrm{b}$	set of spatial coordinates at the boundaries
Chapter 2	
$\boldsymbol{A}$	continuous-time state matrix or state operator of abstract state equation
$\boldsymbol{B}$	continuous-time input matrix or input operator of abstract state equation
$\boldsymbol{A}_{\check{\mathrm{f}}}$	finite dimensional part of abstract state equation
$\boldsymbol{A}_{\check{\imath}}$	infinite dimensional part of abstract state equation
$\boldsymbol{C}$	continuous-time output matrix or output operator of abstract state equation
$\mathscr{L}$	observer gain
$\mathscr{D}$	domain
$\mathscr{P}, \mathbb{Q}$	orthogonal projection of state space in Hilbert space
$\mathscr{D}$	damping operator
$\mathscr{F}$	stiffness operator
Chapter 3	
x	state vector of systems
$\hat{x}^-$	*a priori* estimate of state x
$\hat{x}^+$	*a posteriori* estimate of state x
$\hat{x}_i^\mathrm{l}$	local estimates at node i
z	measurement vector
$\boldsymbol{F}$	discrete-time state matrix
$\boldsymbol{G}$	input matrix of discrete-time state equation
$\boldsymbol{H}$	measurement matrix of discrete-time state equation
$\boldsymbol{K}$	Kalman gain
$w, \boldsymbol{Q}$	process noise and its corresponding covariance matrix
$v, \boldsymbol{R}$	measurement noise and its corresponding covariance matrix
$\boldsymbol{P}$	estimation error covariance matrix
Δs	information vector of information filter
$\Delta \mathscr{I}$	information matrix of information filter
Chapter 4	
$\mathscr{M}_\mathrm{g}$	set of grid points from discretized space
$\mathscr{M}_\mathrm{u}$	set of inputs at grid points

A

Symbol	Meaning
$\mathcal{M}_\mathrm{s}$	set of actuators at grid points
N_u	number of actuators
N_s	number of sensors
T_s	sampling period in temporal discretization
Δ_z	inter-node distance in spatial discretization
$\mathcal{N}_{\mathrm{s},i}$	set of neighbor indices of ith sensor, including i
$\mathcal{N}_{\mathrm{u},i}$	set of neighbor indices of ith actuators, including i
ϕ_i	regressors of ith model
$\theta_i, \hat{\theta}_i$	parameter vector of ith model and its estimate
$\mathbb{E}\{\cdot\}$	expectation operator
$Y(z,t), Y(z)$	measurement equation of spatiotemporal process, and of spatial process
$L(z,t), L(z)$	spatiotemporal random process and spatial random process
$\hat{L}(z)$	estimator of $L(z)$
$V(z,t)$	noise of spatiotemporal random process and of spatial random process
g_{z_0}	true but unknown value of $L(z)$
y_z	measurement vector of random spatial process
N_{y_z}	length of vector y_z
$\mu_L, \hat{\mu_L}$	mean value of $L(z)$ and its estimate
$\boldsymbol{C}_{y_\mathrm{z}}$	covariance matrix of measurement y_z
$\mathrm{Var}(\cdot)$	variance operator
$\mathrm{Cov}(\cdot,\cdot)$	covariance operator
$V(z,t)$	spatiotemporal measurement noise
ρ	material density
κ	thermal conductivity
C_p	heat capacity
Chapter 5	
$\mathcal{T}$	search tree consisting of nodes that represent models with different numbers of regressors
$\boldsymbol{\Phi}$	regressor matrix
n_u	maximum lags of input used in the model
n_y	maximum lags of output used in the model
N_P	number of rows of the regressor matrix
$\boldsymbol{\phi}$	vector of regressor
$\boldsymbol{v}_{\mathrm{s},i}$	subset of regressors used in the node η_i
$\boldsymbol{\varphi}_\mathrm{s}$	vector that indicates the regressors used in the node η_i
$\boldsymbol{\Phi}_\mathrm{s}$	regressor matrix
n_d	number of regressors in the matrix
n_r	number of possible regressors
n_e	number of simultaneous node expansions for the propsed method
n_φ	number of possible combinations of models that takes n_d regressors out of n_r regressors
η_i	model represented by the node i
J_AIC	performance of a model based on the Akaike information criterion
J_{η_i}	performance of a model that corresponds to the node η_i
$\mathscr{C}_i$	set of nodes that are built from the expansion of parent node η_i
$\mathscr{L}$	set of nodes that do not have children nodes
$\mathscr{L}_\mathrm{T}$	set of terminal nodes that cannot be expanded to build children nodes

Symbol	Meaning
Chapter 6	
f_{F}	nonlinear function represented by a TS fuzzy model
u_*	input data containing missing samples
y_*	output data containing missing samples
$\mathbb{K}_{\mathrm{m}}$	set indices of unavailable measurements
$\boldsymbol{\Phi}_{\mathrm{A}}$	regressor matrix augmented by the target vector
$\phi_{i,j}$	element of $\boldsymbol{\Phi}$ at row i and column j
$\boldsymbol{\phi}_k$	regressor vector consisting of lagged input-output samples
N_{m}	number of missing samples
$\boldsymbol{\Phi}_{\mathrm{A,m}}$	regressor matrix containing missing elements
n_{a}	maximum delay of output samples
n_{b}	maximum delay of input samples
$\boldsymbol{\theta}_{\mathrm{F}}$	set of antecedent and consequent parameters of TS fuzzy model
m_{p}	fraction of missing samples
Appendix A	
L_m	length of segment
λ_m	number of lanes
$v_{\mathrm{f},m}$	free flow speed
$\rho_{\mathrm{cr},m}$	critical density
τ	time constant
ν	anticipation constant
κ_{M}	constant
a_m	parameter
v_{min}	minimum velocity
v_{max}	maximum velocity
ρ_{min}	minimum density
ρ_{max}	maximum density

BIBLIOGRAPHY

[1] Hesam Ahmadian Behrooz and R. Bozorgmehry Boozarjomehry. Modeling and state estimation for gas transmission networks. *Journal of Natural Gas Science and Engineering*, 22: 551–570, 2015.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, 2002.

[4] Mustafa Alabadla, Fatimah Sidi, Iskandar Ishak, Hamidah Ibrahim, Lilly Suriani Affendey, Zafienas Che Ani, Marzanah A. Jabar, Umar Ali Bukar, Navin Kumar Devaraj, Ahmad Sobri Muda, Anas Tharek, Noritah Omar, and M. Izham Mohd Jaya. Systematic review of using machine learning in imputing missing values. *IEEE Access*, 10:44483–44502, 2022.

[5] R.J. Almeida, U. Kaymak, and J.M.C. Sousa. A new approach to dealing with missing values in data-driven fuzzy modeling. In *Proceedings of the 2010 IEEE International Conference on Fuzzy Systems*, pages 1–7, Barcelona, Spain, July 2010.

[6] Peter Alriksson and Anders Rantzer. Distributed Kalman filtering using weighted averaging. In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS 2006)*, pages 2445–2450, Kyoto, Japan, July 2006.

[7] Peter Alriksson and Anders Rantzer. Model based information fusion in sensor networks. In *Proceedings of the 17th IFAC World Congress*, pages 4150–4155, Seoul, Korea, July 2008.

[8] Brian D. O. Anderson and John B. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, N.J., 1979.

[9] Goran Andonovski, Gašper Mušič, Sašo Blažič, and Igor Škrjanc. Evolving model identification for process monitoring and prediction of non-linear systems. *Engineering Applications of Artificial Intelligence*, 68:214–221, 2018.

[10] Ehsan Askari and Guillaume Crevecoeur. Evolutionary sparse data-driven discovery of multibody system dynamics. *Multibody System Dynamics*, 58(2):197–226, 2023.

[11] P. Astrid. *Reduction of Process Simulation Models : A Proper Orthogonal Decomposition Approach*. PhD thesis, Technische Universiteit Eindhoven, 2004.

[12] T. M. Atanackovic and A. Guran. *Theory of Elasticity for Scientists and Engineers*. Springer, 2000.

[13] Michael Athans. Toward a practical theory for distributed parameter systems. *IEEE Transactions on Automatic Control*, 15(2):245–247, April 1970.

[14] J.A. Atwell and B.B. King. Proper orthogonal decomposition for reduced basis feedback controllers for parabolic equations. *Mathematical and Computer Modelling*, 33(1):1–19, 2001.

[15] M. Autin, M. Biey, and M. Hasler. Order of discrete time nonlinear systems determined from input-output signals. In *Proceedings of the 1992 IEEE International Symposium on Circuits and Systems*, volume 1, pages 296–299, San Diego, CA, USA, 10-13 May 1992.

[16] Robert Babuška. *Fuzzy Modeling for Control*. Kluwer Academic Publishers, Norwell, USA, 1998.

[17] M. Balas. Feedback control of flexible systems. *IEEE Transactions on Automatic Control*, 23 (4):673–679, August 1978.

[18] Alberto Bemporad, Stefano Di Cairano, Erik Henriksson, and Karl Henrik Johansson. Hybrid model predictive control based on wireless sensor feedback: An experimental study. *International Journal of Robust and Nonlinear Control*, 20(2):209–225, 2010.

[19] A. Ben-Nakhi, M. A. Mahmoud, and A. M. Mahmoud. Inter-model comparison of CFD and neural network analysis of natural convection heat transfer in a partitioned enclosure. *Applied Mathematical Modelling*, 32:1834–1847, 2008.

[20] Peter Benner. Solving large-scale control problems. *IEEE Control Systems Magazine*, 24(1): 44–59, 2004.

[21] P. Bergsten, R. Palm, and D. Driankov. Observers for Takagi-Sugeno fuzzy systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 32(1):114–121, February 2002.

[22] Pontus Bergsten. *Observers and Controllers for Takagi-Sugeno Fuzzy Systems*. Ph.D. Thesis, Örebro University, Sweden, September 2001.

[23] James C Bezdek and Richard J Hathaway. Convergence of alternating optimization. *Neural, Parallel & Scientific Computations*, 11(4):351–368, 2003.

[24] S. A. Billings and W. S. F. Voon. A prediction-error and stepwise-regression estimation algorithm for non-linear systems. *International Journal of Control*, 44(3):803–822, 1986.

[25] Stephen A Billings. *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. John Wiley & Sons, 2013.

[26] Stephen A. Billings, Michael J. Korenberg, and Sheng Chen. Identification of non-linear output-affine systems using an orthogonal least-squares algorithm. *International Journal of Systems Science*, 19:1559–1568, 1988.

[27] Åke Björck. *Numerical Methods for Least Squares Problems*. SIAM, USA, 1996.

[28] Thomas Blumensath and Michael E Davies. Low-rank matrix recovery via convex optimization. *IEEE Signal Processing Letters*, 15(10):809–812, 2008.

[29] T. Boulard, G. Papadakis, C. Kittas, and M. Mermier. Air flow and associated sensible heat exchanges in a naturally ventilated greenhouse. *Agricultural and Forest Meteorology*, 88(1–4): 111–119, 1997.

[30] C.A. Brebbia and J. Dominguez. *Boundary Elements: An Introductory Course*. WIT Press, 2nd edition, 1994.

[31] Leo Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37(4): 373–384, 1995.

[32] Robert Grover Brown and Patrick Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley & Sons, 3rd edition, 1996.

[33] J. J. Buckley. Sugeno type controllers are universal controllers. *Fuzzy Sets and Systems*, 53(3): 299–303, 1993.

[34] Samuel Burer and Adam N. Letchford. Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, 17(2):97–106, 2012.

[35] Kenneth Burnham. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer, New York, USA, 2002.

[36] C. Byrnes, D. S. Gilliam, and V. I. Shubov. Global Lyapunov stabilization of a nonlinear distributed parameter system. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 2, pages 1769–1773, Lake Buena Vista, FL, USA, 1994.

[37] R. Carroll and D. Lindorff. An adaptive observer for single-input single-output linear systems. *IEEE Transactions on Automatic Control*, 18(5):428–435, October 1973.

[38] F.S. Cattivelli and A.H. Sayed. Diffusion strategies for distributed kalman filtering and smoothing. *IEEE Transactions on Automatic Control*, 55(9):2069–2084, September 2010.

[39] Yossi Chait and Clark J. Radcliffe. Control of distributed parameter systems with spillover using an augmented observer. In *Proceedings of the 1987 American Control Conference*, pages 1193–1198, Minneapolis, MN, USA, June 1987.

[40] Siddhartha Chakrabarty and Floyd Hanson. Cancer drug delivery in three dimensions for a distributed parameter control model using finite elements. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2088–2093, San Diego, CA, USA, January 2007.

[41] B-S Chen, C-L Lin, and F-B Hsiao. Robust observer-based control of a vibrating beam. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 205(23):77–89, 1991.

[42] Bin Chen, Wen Chen, and Xing Wei. Characterization of space-dependent thermal conductivity for nonlinear functionally graded materials. *International Journal of Heat and Mass Transfer*, 84:691–699, 2015.

[43] Bin Chen, Wen Chen, Alexander H-D Cheng, Lin-Lin Sun, Xing Wei, and Hongmei Peng. Identification of the thermal conductivity coefficients of 3D anisotropic media by the singular boundary method. *International Journal of Heat and Mass Transfer*, 100:24–33, 2016.

[44] S. Chen, S. A. Billings, and W. Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control*, 50(5):1873–1896, 1989.

[45] Meng-Bi Cheng, V. Radisavljevic, and Wu-Chung Su. Output-feedback boundary control of an uncertain heat equation with noncollocated observation: A sliding-mode approach. In *Proceedings of the 5th Industrial Electronics and Applications*, pages 2187–2192, Taichung, China, June 2010.

[46] Stephen L. Chiu. Selecting input variables for fuzzy models. *Journal of Intelligent and Fuzzy Systems*, 4(4):243–256, January 1996.

[47] Panagiotis D. Christofides. Robust control of parabolic PDE systems. *Chemical Engineering Science*, 53(16):2949–2965, 1998.

[48] G. Ciccarella, M. Dalla Mora, and A. Germani. A robust observer for discrete time nonlinear systems. *Systems & Control Letter*, 24(4):291–300, 1995.

[49] D. Coca and S. A. Billings. Direct parameter identification of distributed parameter systems. *International Journal of Systems Science*, 31(1):11–17, 2000.

[50] J. Cortes. Distributed kriged kalman filter for spatial estimation. *IEEE Transactions on Automatic Control*, 54(12):2816–2827, 2009.

[51] I. Couckuyt, F. Declercq, T. Dhaene, H. Rogier, and L. Knockaert. Surrogate-based infill optimization applied to electromagnetic problems. *International Journal of RF and Microwave Computer-Aided Engineering*, 20(5):492–501, 2010.

[52] I. Couckuyt, A. Forrester, D. Gorissen, F. De Turck, and T. Dhaene. Blind kriging: Implementation and performance analysis. *Advances in Engineering Software*, 49:1–13, July 2012.

[53] G.F. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design*. Addison-Wesley, 3 edition, 2001.

[54] Noel Cressie and Christopher K. Wikle. *Statistics for Spatio-Temporal Data*. John Wiley & Sons, USA, 2011.

[55] Francesco Curreri, Salvatore Graziani, and Maria Gabriella Xibilia. Input selection methods for data-driven soft sensors design: Application to an industrial process. *Information Sciences*, 537:1–17, 2020.

[56] R.F. Curtain and H.J. Zwart. *An Introduction to Infinite-Dimensional Linear Systems Theory*. Springer-Verlag, New York, 1995.

[57] R.F. Curtain, M.A. Demetriou, and K. Ito. Adaptive observers for structurally perturbed infinite dimensional systems. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 1, pages 509–514, San Diego, CA , USA, Dec. 1997 1997.

[58] R.F. Curtain, M.A. Demetriou, and K. Ito. Adaptive observers for slowly time varying infinite dimensional systems. In *Proceedings of the 37th IEEE Conference on Decision and Control*, volume 4, pages 4022 –4027, Tampa, FL, USA, December 1998.

[59] R. David, J.-L. Vasel, and A. Vande Wouwer. Settler dynamic modeling and MATLAB simulation of the activated sludge process. *Chemical Engineering Journal*, 146(2):174–183, 2009. ISSN 1385–8947.

[60] M.A. Demetriou and K. Ito. Adaptive observers for a class of infinite dimensional systems. In *Proceedings of the 13th IFAC World Congress*, volume K, pages 409–413, San Francisco, CA, USA, 1996.

[61] M.A. Demetriou, K. Ito, and R.C. Smith. Adaptive monitoring and accommodation of nonlinear actuator faults in positive real infinite dimensional systems. *IEEE Transactions on Automatic Control*, 52(12):2332–2338, December 2007.

[62] Michael A. Demetriou. Natural second-order observers for second-order distributed parameter systems. *Systems & Control Letter*, 51(3-4):225–234, 2004.

[63] Michael A. Demetriou. Natural consensus filters for second order infinite dimensional systems. *Systems & Control Letter*, 58(12):826–833, 2009.

[64] Michael A. Demetriou. Design of consensus and adaptive consensus filters for distributed parameter systems. *Automatica*, 46(2):300–311, 2010.

[65] E.J. Dempsey and D.T. Westwick. Identification of hammerstein models with cubic spline nonlinearities. *IEEE Transactions on Biomedical Engineering*, 51(2):237–245, 2004.

[66] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1): 1–38, 1977.

[67] Jing Deng and Biao Huang. Identification of nonlinear parameter varying systems with missing output data. *AIChE Journal*, 58(11):3454–3467, 2012.

[68] Feng Ding. Hierarchical multi-innovation stochastic gradient algorithm for Hammerstein nonlinear system modeling. *Applied Mathematical Modelling*, 37(4):1694–1704, February 2013.

[69] Feng Ding, Guangjun Liu, and Xiaoping P. Liu. Parameter estimation with scarce measurements. *Automatica*, 47(8):1646–1655, 2011.

[70] C.C. Doumanidis and N. Fourligkas. Temperature distribution control in scanned thermal processing of thin circular parts. *IEEE Transactions on Control Systems Technology*, 9(5): 708–717, September 2001.

[71] S. Drakunov and V. Utkin. Sliding mode observers. tutorial. In *Proceedings of the 34th IEEE Conference on Decision and Control*, New Orleans, LA, USA, 1995.

[72] Norman R. Draper and Harry Smith. *Applied Regression Analysis*. Wiley-Interscience, 3rd edition, 1998.

[73] Jason A. Duan, Alan E. Gelfand, and C. F. Sirmans. Modeling space-time data using stochastic differential equations. *Bayesian Analysis*, 4(4):733–758, December 2009.

[74] P. Dufour, P. Laurent, and C.Z. Xu. Observer based model predictive control of the water based painting drying using a humidity profile soft sensor and a temperature measurement. In *Proceedings of the 1st International Workshop and Symposium on Industrial Drying (IWSID)*, Mumbay, India, 2004. paper SY152.

[75] D. Edouard, D. Schweich, and H. Hammouri. Observer design for reverse flow reactor. *AIChE Journal*, 50(9):2155–2166, 2004.

[76] M.O. Efe. A finite dimensional sliding mode observer for a spatially continuous process. In *Proceedings of the International Workshop Variable Structure Systems*, pages 302–307, Antalya, Turkey, June 2008.

[77] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[78] A. El Jai and M. Amouroux. Sensors and observers in distributed parameter systems. *International Journal of Control*, 47(1):333–347, 1988.

[79] Tlamelo Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. A survey on missing data in machine learning. *Journal of Big Data*, 8(1):140, 2021. ISSN 2196-1115.

[80] Dilek Erdirencelebi and Sukran Yalpir. Adaptive network fuzzy inference system modeling for the input selection and prediction of anaerobic digestion effluent quality. *Applied Mathematical Modelling*, 35(8):3821–3832, 2011.

[81] Stanley J. Farlow. The GMDH algorithm of ivakhnenko. *The American Statistician*, 35(4): 210–215, 1981.

[82] Gang Feng. A survey on analysis and design of model-based fuzzy control systems. *IEEE Transactions on Fuzzy Systems*, 14(5):676–697, October 2006.

[83] B. Friedland. Observers. In *The Control Handbook*, volume 3, chapter 15, pages 15–1–15–23. CRC Press, 2 edition, 1996.

[84] Nobuo Fujii. Feedback stabilization of distributed parameter systems by a functional observer. *SIAM Journal on Control and Optimization*, 18(2):108–120, 1980.

[85] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Artificial Intelligence. Addison-Wesley, 1989.

[86] Arthur S. Goldberger. Best linear unbiased prediction in the generalized linear regression model. *Journal of the American Statistical Association*, 57(298):369–375, June 1962.

[87] R. B. Gopaluni. A particle filter approach to identification of nonlinear processes under missing observations. *The Canadian Journal of Chemical Engineering*, 86(6):1081–1092, 2008.

[88] R. Bhushan Gopaluni. Nonlinear system identification under missing observations: The case of unknown model structure. *Journal of Process Control*, 20(3):314–324, 2010.

[89] R. Gressang and G. Lamont. Observers for systems characterized by semigroups. *IEEE Transactions on Automatic Control*, 20(4):523–528, August 1975.

[90] Qing Guo, Zhenlei Chen, Yao Yan, Wenying Xiong, Dan Jiang, and Yan Shi. Model identification and human-robot coupling control of lower limb exoskeleton with biogeography-based learning particle swarm optimization. *International Journal of Control, Automation and Systems*, 20(2):589–600, 2022. ISSN 2005-4092.

[91] D.E. Gustafson and W.C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proceedings of the 1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, volume 17, pages 761–766, San Diego, CA, USA, 1978.

[92] Aleksandar Haber and Michel Verhaegen. Subspace identification of large-scale interconnected systems. *IEEE Transactions on Automatic Control*, 59(10):2754–2759, 2014.

[93] R. Haberman. *Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems*. Pearson, 5 edition, 2014.

[94] Maki K. Habib, Samuel A. Ayankoso, and Fusaomi Nagata. Data-driven modeling: Concept, techniques, challenges and a case study. In *Proceedings of the 2021 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1000–1007, Takamatsu, Japan, 2021.

[95] M.L. Hadjili and V. Wertz. Takagi-Sugeno fuzzy modeling incorporating input variables selection. *IEEE Transactions on Fuzzy Systems*, 10(6):728–742, 2002.

[96] Raymond C. Hall and Dale E. Seborg. Modelling and self-tuning control of a multivariable pH neutralization process part I: Modelling and multiloop control. In *Proceedings fo the American Control Conference*, pages 1822–1827, Pittsburgh, PA, USA, June 1989.

[97] Anders Hansson and Ragnar Wallin. Maximum likelihood estimation of gaussian models with missing data–Eight equivalent formulations. *Automatica*, 48(9):1955–1962, 2012.

[98] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer New York, 2009.

[99] R.J. Hathaway and J.C. Bezdek. Fuzzy *c*-means clustering of incomplete data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 31(5):735–744, 2001.

[100] Simon O. Haykin. *Neural Networks and Learning Machines*. Pearson, Upper Saddle River, USA, 3rd edition, 2009.

[101] Xiangdong He and Haruhiko Asada. A new method for identifying orders of input-output models for nonlinear dynamic systems. In *Proceedings of the 1993 American Control Conference*, pages 2520–2523, San Fransisco, CA, USA, June 1993.

[102] A. Hegyi, D. Girimonte, R. Babuška, and B. De Schutter. A comparison of filter configurations for freeway traffic state estimation. In *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC '06)*, pages 1029–1034, Toronto, Canada, September 2006.

[103] S. Henkel, A. Kharitonov, and O. Sawodny. Modelling and optimisation of a glass feeder considered as a distributed parameter system. In *Proceedings of the Annual Conference SICE 2007*, pages 2950–2954, Takamatsu, Japan, September 2007.

[104] M. Hénon. A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics*, 50(1):69–77, 1976.

[105] J.P. Hespanha, P. Naghshtabrizi, and Yonggang Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, January 2007.

[106] Tim Hesterberg, Nam Hee Choi, Lukas Meier, and Chris Fraley. Least angle and ℓ_1 penalized regression: A review. *Statistics Surveys*, 2:61–93, 2008.

[107] Z Hidayat, Zs Lendek, R Babuška, and B De Schutter. Fuzzy observer for state estimation of the metanet traffic model. In *Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems*, pages 19–24. IEEE, 2010.

[108] Z Hidayat, R Babuška, A Núñez, and B De Schutter. Identification of distributed-parameter systems from sparse measurements. *Applied Mathematical Modelling*, 51:605–625, 2017.

[109] Z Hidayat, R Babuška, A Núñez, and B De Schutter. A paper in model reduction with tree representation. *Submitted for publication*, 2023.

[110] Z Hidayat, R Babuška, A Núñez, and B De Schutter. A paper in TS fuzzy system identification with missing data. *In preparation for publication*, 2023.

[111] Zulkifli Hidayat, R Babuška, Bart De Schutter, and Alfredo Núñez. Decentralized kalman filter comparison for distributed-parameter systems: A case study for a 1d heat conduction process. In *Proceedings of the 2011 IEEE 16th Conference on Emerging Technologies & Factory Automation*, pages 1–8. IEEE, 2011.

[112] Zulkifli Hidayat, Robert Babuška, Bart De Schutter, and Alfredo Núñez. Observers for linear distributed-parameter systems: A survey. In *Proceedings of the 2011 IEEE International Symposium on Robotic and Sensors Environments*, pages 166–171. IEEE, 2011.

[113] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[114] Jack Phillip Holman. *Heat Transfer*. McGraw-Hill Higher Education, New York, NY, USA, 10th edition, 2009.

[115] S.P. Hoogendoorn and P.H.L. Bovy. State-of-the-art of vehicular traffic flow modelling. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 215(4):283–303, January 2001.

[116] F. Höppner and F. Klawonn. A contribution to convergence theory of fuzzy c-means and derivatives. *IEEE Transaction on Fuzzy Systems*, 11:682–694, 2003.

[117] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013. ISSN 0020-0255. Data-based Control, Decision, Scheduling and Fault Diagnostics.

[118] Xiaoyi Huang, Haoran Xu, and Jizheng Chu. Nonlinear model order selection: A GMM clustering approach based on a genetic version of EM algorithm. *Mathematical Problems in Engineering*, 2022:1–16, December 2022.

[119] Clifford M. Hurvich and Chih-Ling Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 1989.

[120] D. Ichalal, B. Marx, J. Ragot, and D. Maquin. State and unknown input estimation for nonlinear systems described by Takagi-Sugeno models with unmeasurable premise variables. In *Proceedings of the 17th Mediterranean Conference on Control and Automation*, pages 217–222, Thessaloniki, Greece, June 2009.

[121] Frank P. Incropera and David P. Dewitt. *Fundamentals of Heat and Mass Transfer*. John Wiley, 5th edition, 2002.

[122] Tor A. Johansen and Bjarne A. Foss. Empirical modeling of a heat transfer process using local models and interpolation. In *Proceedings of the 1995 American Control Conference*, volume 5, pages 3654–3654, 1995.

[123] Tor Arne Johansen. *Operating Regime based Process Modeling and Identication*. Ph.D. Thesis, Norwegian Institute of Technology, Trondheim, Norway, 1994.

[124] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions on the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[125] J. Kennedy, J.F. Kennedy, R.C. Eberhart, and Y. Shi. *Swarm Intelligence*. Evolutionary Computation Series. Morgan Kaufmann, 2001.

[126] Matthew B. Kennel, Reggie Brown, and Henry D. I. Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A*, 45(6):3403–3411, 1992.

[127] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, New Jersey, 3rd edition, 2002.

[128] Dhruv Khandelwal, Maarten Schoukens, and Roland Tóth. Automated multi-objective system identification using grammar-based genetic programming. *Automatica*, 154:111017, 2023.

[129] Sun Hye Kim and Fani Boukouvala. Machine learning-based surrogate modeling for data-driven optimization: A comparison of subset selection for regression techniques. *Optimization Letters*, 14(4):989–1010, 2020.

[130] S. Kitamura, S. Sakairi, and M. Nishimura. Observer for distributed-parameter diffusion systems. *Electr. Eng. Jpn*, 92(6):142–149, 1972.

[131] R. Klein, N.A. Chaturvedi, J. Christensen, J. Ahmed, R. Findeisen, and A. Kojic. State estimation of a reduced electrochemical model of a lithium-ion battery. In *Proceedings of the 2010 American Control Conference*, pages 6618–6623, Baltimore, MD, USA, July 2010.

[132] Hisashi Kobayashi, Brian L. Mark, and William Turin. *Probability, Random Processes, and Statistical Analysis*. Cambridge University Press, 2012.

[133] Toshihiro Kobayashi. Discrete-time observers and parameter determination for distributed parameter systems with discrete-time input–output data. *SIAM Journal on Control and Optimization*, 21(3):331–351, 1983.

[134] Toshihiro Kobayashi and Shigeru Hitotsuya. Observers and parameter determination for distributed parameter systems. *International Journal of Control*, 33(1):31–50, 1981.

[135] M. Köhne. Implementation of distributed parameter state observers. In *Distributed Parameter Systems: Modelling and Identification*, volume 1, pages 310–324. Springer, Berlin, 1978.

[136] Apostolos Kotsialos and Markos Papageorgiou. Motorway network traffic control systems. *European Journal of Operational Research*, 152(2):321–333, 2004.

[137] Apostolos Kotsialos, Markos Papageorgiou, Christina Diakaki, Yannis Pavlis, and Frans Middelham. Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool METANET. *IEEE Transactions on Intelligent Transportation Systems*, 3(4): 282–292, December 2002.

[138] Costas Kravaris and John H Seinfeld. Identification of spatially varying parameters in distributed parameter systems by discrete regularization. *Journal of Mathematical Analysis and Applications*, 119(1-2):128–152, 1986.

[139] G. Kreisselmeier. The generation of adaptive law structures for globally convergent adaptive observers. *IEEE Transactions on Automatic Control*, 24(3):510–513, June 1979.

[140] V. Krishnaswami, Yong Wha Kim, and G. Rizzoni. A new model order identification algorithm with application to automobile oxygen sensor modeling. In *Proceedings of the 1995 American Control Conference*, volume 3, pages 2113–2117, Seattle, WA, USA, June 1995.

[141] Miroslav Krstić, Rafael Vazquez, Antranik A. Siranosian, and Matt Bement. Sensing schemes for state estimation in turbulent flows and flexible structures. *Smart Structures and Materials 2006: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*, 6174(1):61740U, 2006.

[142] Carlos S. Kubrusly. Distributed parameter system identification: A survey. *International Journal of Control*, 26(4):509–535, 1977.

[143] C.S. Kubrusly and H. Malebranche. Sensors and controllers location in distributed systems – a survey. *Automatica*, 21(2):117–128, 1985.

[144] Walter Lacarbonara, Ali H. Nayfeh, and Wayne Kreider. Experimental validation of reduction methods for nonlinear vibrations of distributed-parameter systems: Analysis of a buckled beam. *Nonlinear Dynamics*, 17(2):95–117, 1998.

[145] Pedro Lagos-Eulogio, Juan Carlos Seck-Tuoh-Mora, Norberto Hernandez-Romero, and Joselito Medina-Marin. A new design method for adaptive IIR system identification using hybrid CPSO and DE. *Nonlinear Dynamics*, 88(4):2371–2389, 2017. ISSN 1573-269X.

[146] Zsófia Lendek, Robert Babuška, and Bart De Schutter. Fuzzy models and observers for freeway traffic state tracking. In *Proceedings of the 2010 American Control Conference*, pages 2278–2283, Baltimore, Maryland, 2010.

[147] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady State and Time Dependent Problems*. SIAM, Philadelphia, PA, USA, 2007.

[148] Chaoshun Li, Jianzhong Zhou, Jian Xiao, and Han Xiao. Hydraulic turbine governing system identification using T–S fuzzy model optimized by chaotic gravitational search algorithm. *Engineering Applications of Artificial Intelligence*, 26(9):2073–2082, 2013.

[149] Dan Li, Hong Gu, and Liyong Zhang. A fuzzy c-means clustering algorithm based on nearest-neighbor intervals for incomplete data. *Expert Systems with Applications*, 37(10):6942–6947, 2010.

[150] Han-Xiong Li and Chenkun Qi. Modeling of distributed parameter systems for applications – a synthesized review from time-space separation. *Journal of Process Control*, 20(8):891–901, 2010.

[151] Y.F. Li and X.B. Chen. End-point sensing and state observation of a flexible-link robot. *IEEE/ASME Transactions on Mechatronics*, 6(3):351–356, September 2001.

[152] John H. Lilly. Finite-dimensional adaptive observers applied to distributed parameter systems. *IEEE Transactions on Automatic Control*, 38(3):469–474, March 1993.

[153] Ingela Lind and Lennart Ljung. Regressor and structure selection in NARX models using a structured ANOVA approach. *Automatica*, 44(2):383–395, 2008.

[154] Jacques-Louis Lions. Some aspects of modelling problems in distributed parameter systems. In Prof Dr Antonio Ruberti, editor, *Distributed Parameter Systems: Modelling and Identification*, pages 11–41. Springer Berlin Heidelberg, January 1978.

[155] Jialin Liu, Shi-Shang Jang, and David Shan-Hill Wong. Developing a soft sensor with online variable selection for industrial multi-mode processes. In *26th European Symposium on Computer Aided Process Engineering*, volume 38, pages 398–403. Elsevier, 2016.

[156] M. Liu, N. Patwari, and A. Terzis. Special issue on sensor network applications. *Proceedings of the IEEE*, 98(11):1804–1807, November 2010.

[157] Y. A. Liu and Leon Lapidus. Observer theory for distributed-parameter systems. *International Journal of Systems Science*, 7(7):731–742, 1976.

[158] Zhang Liu, Anders Hansson, and Lieven Vandenberghe. Nuclear norm system identification with missing inputs and outputs. *Systems & Control Letters*, 62(8):605–612, August 2013.

[159] Lennart Ljung. *System Identification: Theory for the User*. Prentice Hall, USA, 2nd edition, 1999.

[160] J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the International Symposium on Computer Aided Control Systems Design*, pages 284–289, Taipei, Taiwan, 2004.

[161] D. Luenberger. An introduction to observers. *IEEE Transactions on Automatic Control*, 16(6): 596–602, December 1971.

[162] David G. Luenberger. Observing the state of a linear system. *IEEE Transactions on Military Electronics*, 8(2):74–80, April 1964.

[163] Simon Mandelj, Igor Grabec, and Edvard Govekar. Statistical approach to modeling of spatiotemporal dynamics. *International Journal of Bifurcation and Chaos*, 11(11):2731–2738, 2001.

[164] Leonard Meirovitch. *Dynamics and Control of Structures*. John Wiley & Sons, New York, 1990.

[165] E.M.A.M. Mendez and S.A. Billings. An alternative solution to the model structure selection problem. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 31(6):597–608, November 2001.

[166] Albert Messmer and Markos Papageorgiou. Automatic control methods applied to freeway network traffic. *Automatica*, 30(4):691–702, 1994.

[167] Xiaoye Miao, Yangyang Wu, Lu Chen, Yunjun Gao, and Jianwei Yin. An experimental survey of missing data imputation algorithms. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–20, 2022.

[168] Lyudmila Mihaylova, René Boel, and András Hegyi. An unscented Kalman filter for freeway traffic estimation. In *Proceedings of the IFAC Symposium on Control in Transportation Systems*, pages 31–36, Delft, The Netherlands, August 2006.

[169] Lyudmila Mihaylova, René Boel, and Andreas Hegyi. Freeway traffic estimation within particle filtering framework. *Automatica*, 43(2):290–300, 2007.

[170] R. Miranda, I. Chairez, and J. Moreno. Observer design for a class of parabolic PDE via sliding modes and backstepping. In *Proceedings of the International Workshop Variable Structure Systems*, pages 215–220, Mexico City, Mexico, June 2010.

[171] E. A. Misawa and J. K. Hedrick. Nonlinear observers—a state-of-the-art survey. *Journal of Dynamic Systems, Measurement, and Control*, 111(3):344–352, 1989.

[172] Kirsten Morris. Control of systems governed by partial differential equations. In *The Control Handbook*, volume 3, chapter 67, pages 67–1–67–37. CRC Press, 2 edition, 2010.

[173] Tobias Münker and Oliver Nelles. Nonlinear system identification with regularized local FIR model networks. *Engineering Applications of Artificial Intelligence*, 67:345–354, 2018. ISSN 0952-1976.

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

[175] Tu Duc Nguyen. Second-order observers for second-order distributed parameter systems in $\mathbb{R}^2$. *Systems & Control Letter*, 57(10):787–795, 2008.

[176] Hiroshi Ohtake, Kazuo Tanaka, and Hua O. Wang. Fuzzy modeling via sector nonlinearity concept. *Integrated Computer-Aided Engineering*, 10(4):333–341, January 2003.

[177] Reza Olfati-Saber. Distributed Kalman filtering and sensor fusion in sensor networks. In Panos J. Antsaklis and Paulo Tabuada, editors, *Networked Embedded Sensing and Control*, number 331 in Lecture Notes in Control and Information Sciences, pages 157–167. Springer Berlin/Heidelberg, Germany, July 2006.

[178] Reza Olfati-Saber. Kalman-consensus filter : Optimality, stability, and performance. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 7036–7042, Shanghai, China, December 2009.

[179] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, January 2007.

[180] Y. Orlov and J. Bentsman. Adaptive distributed parameter systems identification with enforceable identifiability conditions and reduced-order spatial differentiation. *IEEE Transactions on Automatic Control*, 45(2):203–216, February 2000.

[181] P. A. Orner and A. M. Foster. A design procedure for a class of distributed parameter control systems. *Journal of Dynamic Systems, Measurement, and Control*, 93(2):86–92, 1971.

[182] Y. Pan and S. A. Billings. The identification of complex spatiotemporal patterns using coupled map lattice models. *International Journal of Bifurcation and Chaos*, 18(4):997–1013, 2008.

[183] Markos Papageorgiou, Jean-Marc Blosseville, and Habib Hadj-Salem. Macroscopic modelling of traffic flow on the Boulevard Périphérique in Paris. *Transportation Research Part B: Methodological*, 23(1):29–47, February 1989.

[184] H.M. Park and D.H. Cho. The use of the Karhunen-Loève decomposition for the modeling of distributed parameter systems. *Chemical Engineering Science*, 51(1):81–98, 1996.

[185] Young Woong Park and Diego Klabjan. Subset selection for multiple linear regression via optimization. *Journal of Global Optimization*, 77(3):543–574, 2020.

[186] Nikesh Patel, Prashant Mhaskar, and Brandon Corbett. Subspace based model identification for missing data. *AIChE Journal*, 66(10):e16538, 2020.

[187] Hong Pi and Carsten Peterson. Finding the embedding dimension and variable dependencies in time series. *Neural Computation*, 6(3):509–520, 1994.

[188] Radu-Emil Precup and Hans Hellendoorn. A survey on industrial applications of fuzzy control. *Computers in Industry*, 62(3):213–226, 2011.

[189] Wei Qiao, Zhi Gao, Ronald G. Harley, and Ganesh K. Venayagamoorthy. Robust neuro-identification of nonlinear plants in electric power systems with missing sensor measurements. *Engineering Applications of Artificial Intelligence*, 21(4):604–618, 2008.

[190] B. S. Rao and H. F. Durrant-Whyte. Fully decentralised algorithm for multisensor Kalman filtering. *IEE Proceedings D: Control Theory and Applications*, 138(5):413–420, September 1991.

[191] W. Harmon Ray. *Advanced Process Control*. McGraw-Hill, New York, 1981.

[192] W.H. Ray. Some recent applications of distributed parameter systems theory – a survey. *Automatica*, 14(3):281–287, 1978.

[193] Carl Rhodes and Manfred Morari. Determining the model order of nonlinear input/output systems. *AIChE Journal*, 44(1):151–163, 1998.

[194] David L. Russell. Distributed parameter systems: An overview. In *Encyclopedia of Life Support Systems (EOLSS): Control Systems, Robotics And Automation*. EOLSS Publishers, London, 2003.

[195] D. Sáez and R. Zúñiga. Takagi-Sugeno fuzzy model structure selection based on new sensitivity analysis. In *Proceedings of the 14th IEEE International Conference on Fuzzy Systems*, pages 501–506, Reno, NV, USA, May 2005.

[196] Y. Sakawa and T. Matsushita. Feedback stabilization of a class of distributed systems and construction of a state estimator. *IEEE Transactions on Automatic Control*, 20(6):748–753, December 1975.

[197] S. Salini, T. Minerva, A. Zirilli, A. Tiano, and F. Pizzocchero. Neural networks in missing data analysis, model identification and non linear control. In Andrea Bonarini, Francesco Masulli, and Gabriella Pasi, editors, *Soft Computing Applications*, pages 177–192. Springer, Heidelberg, 2003.

[198] Simo Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[199] John H. Seinfeld and Masato Koda. Numerical implementation of distributed parameter filters with application to problems in air pollution. In *Distributed Parameter Systems: Modelling and Identification*, pages 42–69. Springer Berlin Heidelberg, 1978.

[200] Guangren Shi. *Data Mining and Knowledge Discovery for Geoscientists*. Elsevier Science Publishers B. V., NLD, 1st edition, 2013.

[201] J. Sijs, M. Lazar, P.P.J. van den Bosch, and Z. Papp. An overview of non-centralized Kalman filters. In *Proceedings of the IEEE International Conference on Control Applications (CCA 2008)*, pages 739–744, San Antonio, TX, USA, September 2008.

[202] Dan Simon. *Optimal State Estimation, Kalman, H_∞, and Nonlinear Approaches*. John Wiley & Sons, 2006.

[203] Beant Singh, Parag Nijhawan, Manish Kumar Singla, Jyoti Gupta, and Parminder Singh. Hybrid algorithm for parameter estimation of fuel cell. *International Journal of Energy Research*, 46(8):10644–10655, 2022.

[204] Silvia Siri, Cecilia Pasquale, Simona Sacone, and Antonella Ferrara. Freeway traffic control: A survey. *Automatica*, 130:109655, 2021.

[205] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[206] Andrey Smyshlyaev and Miroslav Krstic. Backstepping observers for a class of parabolic PDEs. *Systems & Control Letter*, 54(7):613–625, 2005.

[207] Torsten Söderström. Errors-in-variables methods in system identification. *Automatica*, 43 (6):939–958, 2007.

[208] Jason L. Speyer. Computation and transmission requirements for a decentralized linear-quadratic-gaussian control problem. *IEEE Transactions on Automatic Control*, 24(2):266–269, April 1979.

[209] Sarah K. Spurgeon. Sliding mode observers: A survey. *International Journal of Systems Science*, 39(8):751–764, 2008.

[210] P. Stavroulakis and P. E. Sarachik. Design of optimal controllers for distributed systems using finite dimensional state observers. In *Proceedings of the IEEE Conference on Decision and Control*, volume 12, pages 105–109, San Diego, CA, USA, December 1973.

[211] Julian Stoev and Johan Schoukens. Nonlinear system identification–application for industrial hydro-static drive-line. *Control Engineering Practice*, 54:154–165, 2016.

[212] Yan-Ning Sun, Wei Qin, Jin-Hua Hu, Hong-Wei Xu, and Poly Z.H. Sun. A causal model-inspired automatic feature-selection method for developing data-driven soft sensors in complex industrial processes. *Engineering*, 22:82–93, 2022.

[213] A Taflove and SC Hagness. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Artech House, 2006.

[214] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116–132, 1985.

[215] Kazuo Tanaka, Takayuki Ikeda, and Hua O. Wang. Fuzzy regulators and fuzzy observers: relaxed stability conditions and LMI-based designs. *IEEE Transactions on Fuzzy Systems*, 6(2): 250–265, May 1998.

[216] Masahiro Tanaka and Jianshe Dai. Identification of nonlinear systems with missing data by the EM algorithm. In *Proceedings of the 11th IFAC Symposium on System Identification*, pages 645–650, 1997.

[217] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, January 1996.

[218] Heiko Timm, Christian Döring, and Rudolf Kruse. Different approaches to fuzzy clustering of incomplete datasets. *International Journal of Approximate Reasoning*, 35(3):239–249, 2004.

[219] Marius Tucsnak and George Weiss. *Observation and Control for Operator Semigroups*. Birkhäuser, Basel, 2009.

[220] Dariusz Uciński. *Optimal Measurement Methods for Distributed Parameter System Identification*. CRC Press, USA, 2004.

[221] José Manuel Valente and Sebastián Maldonado. SVR-FFS: A novel forward feature selection approach for high-frequency time series forecasting using support vector regression. *Expert Systems with Applications*, 160:113729, 2020.

[222] H. J. L. van Can, H. A. B. Te Braake, C. Hellinga, A. J. Krijgsman, H. B. Verbruggen, K. Ch. A. M. Luyben, and J. J. Heijnen. Design and real time testing of a neural model predictive controller for a nonlinear system. *Chemical Engineering Science*, 50(15):2419–2430, 1995.

[223] M. van den Berg, A. Hegyi, B. De Schutter, and J. Hellendoorn. Integrated traffic control for mixed urban and freeway networks: A model predictive control approach. *European Journal of Transport and Infrastructure Research*, 7(3):223–250, September 2007.

[224] Sabine Van Huffel and Joos Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis*. SIAM, Philadelphia, PA, USA, 1991.

[225] A. Vande Wouwer and M. Zeitz. State estimation in distributed parameter systems. In *Encyclopedia of Life Support Systems (EOLSS): Control Systems, Robotics and Automation*. EOLSS Publishers, London, 2003.

[226] A. Vande Wouwer, C. Renotte, I. Queinnec, and Ph. Bogaerts. Transient analysis of a wastewater treatment biofilter: Distributed parameter modelling and state estimation. *Mathematical and Computer Modelling of Dynamical Systems: Methods, Tools and Applications in Engineering and Related Sciences*, 12(5):423–440, 2006.

[227] K. C. Veluvolu, Y. C. Soh, and W. Cao. Robust discrete-time nonlinear sliding mode state estimation of uncertain nonlinear systems. *International Journal of Robust and Nonlinear Control*, 17:803–828, 2007.

[228] D. Vries, K. J. Keesman, and H. Zwart. A Luenberger observer for an infinite dimensional bilinear system: A UV disinfection example. In *Proceedings of the 3rd IFAC Symposium on System, Structure and Control*, pages 667–672, Foz do Iguassu, Brazil, October 2007.

[229] R. Šindelář and R. Babuška. Input selection for nonlinear regression models. *IEEE Transactions on Fuzzy Systems*, 12(5):688–696, October 2004.

[230] Fajie Wang, Wen Chen, and Yan Gu. Boundary element analysis of inverse heat conduction problems in 2D thin-walled structures. *International Journal of Heat and Mass Transfer*, 91: 1001–1009, 2015.

[231] Ning Wang, Naiqian Zhang, and Maohua Wang. Wireless sensors in agriculture and food industry–recent development and future perspective. *Computers and Electronics in Agriculture*, 50(1):1–14, 2006.

[232] Yibing Wang and Markos Papageorgiou. Real-time freeway traffic state estimation based on extended Kalman filter: A general approach. *Transportation Research Part B: Methodological*, 39(2):141–167, 2005.

[233] Yibing Wang, Markos Papageorgiou, Albert Messmer, Pierluigi Coppola, Athina Tzimitsi, and Agostino Nuzzolohegyi. An adaptive freeway traffic state estimator. *Automatica*, 45(1):10–24, 2009.

[234] John Glen Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1(3):325–362, 1952.

[235] Zequn Wei and Jin-Kao Hao. Multistart solution-based tabu search for the set-union knapsack problem. *Applied Soft Computing*, 105:107260, 2021. ISSN 1568-4946.

[236] Christopher K. Wikle and Mevin B. Hooten. A general science-based framework for dynamical spatio-temporal models. *TEST*, 19(3):417–451, November 2010.

[237] Xianqiang Yang, Xin Liu, and Shen Yin. Robust identification of nonlinear systems with missing observations: The case of state-space model structure. *IEEE Transactions on Industrial Informatics*, 15(5):2763–2774, may 2019.

[238] I. Yaz, V. Bakke, and E. Yaz. Receding window observer and dynamic feedback control of discrete infinite dimensional systems. In *Proceedings of the 30th IEEE Conference on Decision and Control*, volume 3, pages 3031–3032, Brighton, UK, December 1991.

[239] Bao-Lin Ye, Weimin Wu, Keyu Ruan, Lingxi Li, Tehuan Chen, Huimin Gao, and Yaobin Chen. A survey of model predictive control methods for traffic signal control. *IEEE/CAA Journal of Automatica Sinica*, 6(3):623–640, 2019.

[240] Felix Ghislain Yem Souhe, Alexandre Teplaira Boum, Pierre Ele, Camille Franklin Mbey, and Vinny Junior Foba Kakeu. A novel smart method for state estimation in a smart grid using smart meter data. *Applied Computational Intelligence and Soft Computing*, 2022(1):7978263, 2022.

[241] John Yen and Liang Wang. Constructing optimal fuzzy models using statistical information criteria. *Journal of Intelligent and Fuzzy Systems*, 7(2):185–201, 1999.

[242] Y. Yüksel and Jr. Bongiorno, J. Observers for linear multivariable systems with applications. *IEEE Transactions on Automatic Control*, 16(6):603–613, December 1971.

[243] Hengzhe Zhang, Aimin Zhou, and Hu Zhang. An evolutionary forest for regression. *IEEE Transactions on Evolutionary Computation*, 26(4):735–749, 2022.

[244] Li Zhang, Kamlesh Mistry, Chee Peng Lim, and Siew Chin Neoh. Feature selection using firefly optimization for classification and regression models. *Decision Support Systems*, 106: 64–85, 2018.

[245] Q. Zhang and A. Benveniste. Wavelet networks. *IEEE Transactions on Neural Networks*, 3(6): 889–898, November 1992.

[246] D. Zheng, K. A. Hoo, and M. J. Piovoso. Low-order model identification of distributed parameter systems by a combination of singular value decomposition and the Karhunen-Loève expansion. *Industrial & Engineering Chemistry Research*, 41(6):1545–1556, 2002.

[247] Guang L. Zheng and Steve A. Billings. Radial basis function network configuration using mutual information and the orthogonal least squares algorithm. *Neural Networks*, 9(9): 1619–1637, 1996.

[248] Lincheng Zhou, Xiangli Li, and Feng Pan. Gradient based iterative parameter identification for Wiener nonlinear systems. *Applied Mathematical Modelling*, 37(16–17):8203–8209, September 2013.

[249] Xing-Gui Zhou, Liang-Hong Liu, Yin-Chun Dai, Wei-Kang Yuan, and J.L. Hudson. Modeling of a fixed-bed feactor using the K-L expansion and neural networks. *Chemical Engineering Science*, 51(10):2179–2188, 1996.

[250] O. C. Zienkiewicz, R L. Taylor, and J. L. Zu. *The finite Element Method.* Elsevier, Butterworth-Heinemann, Amsterdam, 7 edition, 2013.

[251] H.J. Zimmermann. *Fuzzy Set Theory – and Its Applications.* Springer, 4th edition, 2011.

[252] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

About the Author

Zulkifli Hidayat was born on 25 December 1970 in Jakarta, Indonesia. He finished his bachelor's degree in electrical engineering in September 1997 from Sepuluh Nopember Institute of Technology, Surabaya, Indonesia and his master's program in control engineering in February 2004 from the University of Twente, Enschede, the Netherlands. In 2009, he was awarded a scholarship from the Ministry of Communication and Information, Indoensia, for PhD research at the Delft Center for Systems and Control, Delft University of Technology, Delft, the Netherlands. He worked on the identification of finite dimensional spatio-temporal systems under the supervision of prof. dr. Robert Babuška, prof. dr. ir. Bart De Schutter, and Dr. Alfredo Núñez Vicencio.