Data Mining Techniques for Process Development

A dissertation submitted in partial fulfillment of the requirements
for the degree of D.Sc. at Hungarian Academy of Sciences

János Abonyi

Veszprém
2010

Preface

During the last decade a major shift has begun in process and chemical industry, since there was an urgent need for new tools that are able to support the optimisation of process operations and the development of new production technologies. Approaches of this shift differ from company to company but one common feature is the requirement of the intensive communication between those who conduct research, design, manufacturing, marketing and management. Such communication should be centered on modeling and simulation due to the need of the integration of the whole product and process development chain in all time and scale levels of the company.

To explore and transfer all the useful knowledge needed to operate and optimise technologies and the business processes, the research of the applicant aimed the development of a novel methodology to integrate heterogeneous information sources and heterogeneous models. The proposed methodology can be referred as model mining, since it is based on the extraction and transformation of information not only from historical process data but also from different type of process models. The introduction of this novel concept required the development of new algorithms and tools for model analysis, reduction and information integration. For this purpose fuzzy systems based modelling, clustering and visualization algorithms have been developed. To handle multi-objective optimization problems where the goals are non-commensurable and are in conflict with each other a novel interactive optimization algorithm has been worked out based on visualization tools and evolutionary algorithms.

The proposed knowledge discovery and management framework addresses important challenges in the fields of systems science, process engineering and information technology. Although the primary goal of the developments was to design new tools that can be fruitfully applied in process engineering, the development and unconventional integration of computational intelligence, data mining, and system identification algorithms resulted in useful results that can also be applied to support marketing, management, scientific research and other engineering activities.

Contents

Chapter 1

Introduction

1.1 State-of-the-art and objectives

As it is emphasized by the 7th Framework Programme, it is essential to develop new methods to increase industrial competitiveness in order to speed up the transformation of the European economy. Hence, there is an urgent need for new tools which are able to support product and process design, and the optimization of production technologies. These tools require intensive communication between design, manufacturing, marketing and management that could be centred on modelling and simulation. This goal has been already realised by engineers and directors of leading companies, e.g. DuPont and Dow Chemical, think that "model integrates the whole organization" (see Figure 1.1). They believe that the extensive use of models is the way that data, information, and knowledge should be conveyed from research to engineering, to manufacturing, and on to the business team [1].

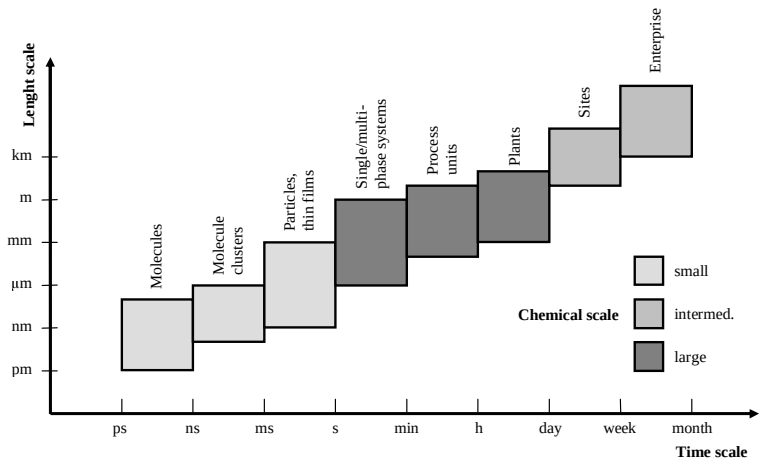


Figure 1.1: Time and scale levels in process engineering. Different scales define different engineering problems.

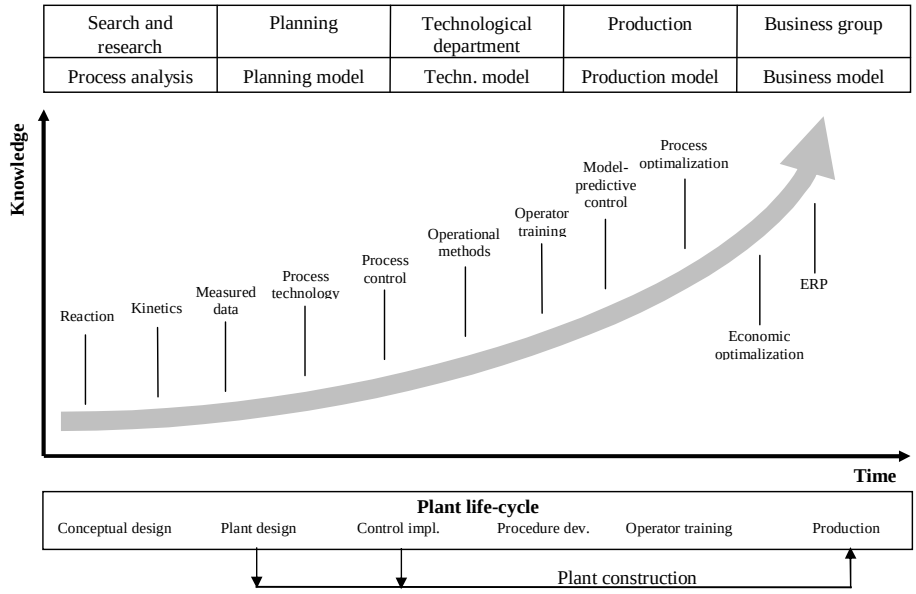


Figure 1.2: Knowledge management in "life-cycle modeling" and integrated modeling - a concept by Dow Chemical Co. Models are applied at every level of a technology and used to transfer information from conceptual design to the optimization of the production.

This led to the the idea of life-cycle modeling: integrating and connecting the model islands, which use different approaches and tools in each life-cycle phase and thus transfer the information and knowledge between the stages [2]. Figure 1.2 shows the knowledge management of life-cycle modeling by Dow Chemical, while Figure 1.3 shows its applicability in process optimization: the continuous improvement cycle centered around the integrated models [3].

The concept of life-cycle modelling is only a vision of how companies should operate in the 21st century. But instead of this there are only model islands for the time being, where isolated models are used for different and limited purposes on different levels of the technology (if they exist at all). These models are heterogeneous not only because they have different purposes, but also because information for the modelling and identification of the processes can be obtained from different sources: (i) mechanistic knowledge obtained from first-principles (physics and chemistry), (ii) empirical or expert knowledge, expressed as linguistic rules, (iii) measurement data, obtained during normal operation or from an experimental process, and different modelling paradigms should be used for an efficient utilisation of these different sources of information.

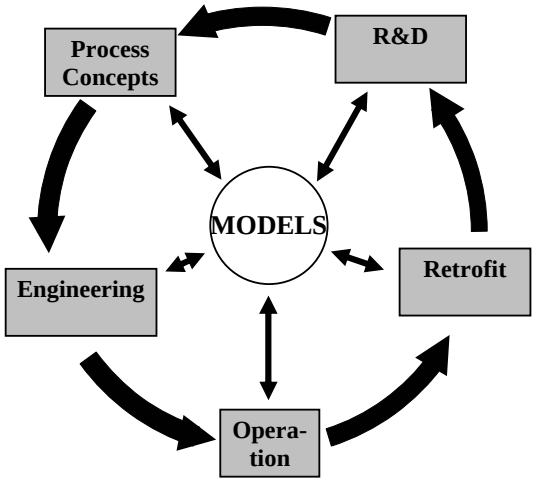


Figure 1.3: The concept of life-cycle modeling for continuous process optimization.

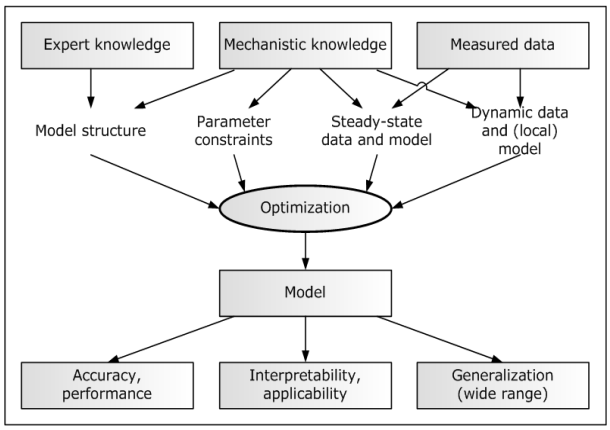


Figure 1.4: The proposed scheme combines expert and mechanistic knowledge and measured data in the form of rule-based systems, model structure, parameter constrains and local models. The model is optimized to ensure booth good prediction performance and interpretability.

Therefore, the aim of the research of the applicant is to develop novel methodologies to integrate heterogeneous information sources and heterogeneous models[1]. The key idea is the utilization of data mining and computational intelligence techniques[2] since the synergistic integration of qualitative and quantitative models require tools to handle uncertain information (by fuzzy logic), systems complexity (by neural networks and evolutionary algorithms), and expert knowledge (by rule-based systems) (see Figure 1.4).

[1]The first attempt of the author to integrate these information sources is concluded in: Abonyi J, Fuzzy Model Identification for Control, Boston: Birkhauser Verlag, 2003. 273 p.,(ISBN:978-0-8176-4238-9), Independent citatitions: 30

[2]Abonyi J, Feil B, Abraham A, Computational intelligence in data mining, INFORMATICA (LJUBLJANA) 29:(1) pp. 3-12. (2005), Independent citatitions: 8

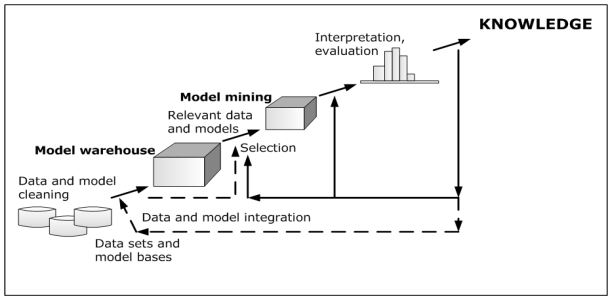


Figure 1.5: Scheme of model mining. The concept is similar to data mining, but in this schemes not only historical process data but different type of models are also analysed.

1.2 Methodology

The key of the proposed approach is the integration of the existing models and information sources to explore useful knowledge. The whole process can be called model mining. This type of knowledge discovery is relatively new, its importance has only been recognized by the Global Modelling and Assimilation Office at NASA in mining of atmospheric phenomena relationships. Hence, this research is the first concentrated attempt to develop a new methodology for model mining. The proposed approach, depicted in Figure 1.5, consists of the following steps.

1. Developing and understanding of the application domain and identifying the problem.

2. Creating and pre-processing the target set. This phase starts with activities in order to get familiar with the models and data, to identify problems, to get first insights into the data and models to detect interesting subsets, to form hypotheses. To support this step, the utilized models should be as transparent as possible since model transparency allows the user to effectively combine different types of information.

3. Model Mining. The ultimate goal of the whole process is to extract potentially useful knowledge which could not be explored from one single model or database, but only from the integrated information sources and models in the model warehouse. The goals of model mining are achieved via solving following problems:

    - *3.a Model Transformation and reduction:* information presented by different type of models can be used to transform or reduce other models to make them more precise and/or robust, or to expand their operational range (e.g. to improve the extrapolation capability of a black-box model using *a priori* knowledge based constraints on the model parameters). Some of the computational intelligence models lend themselves to be transformed into other model structures that allow information transfer between different models (e.g. radial

basis functions are functionally equivalent to fuzzy systems) which makes possible the combination of expert knowledge and data intensive modeling.

- *3.b Model Fusion:* the integration of the information content of different models is the key issue of the proposed research. During this process it should be kept in mind in what range the particular models are valid and how these local models can be combined to get global model. Information can be stored within nominal, ordinal, ratio or fuzzy data; images and pictures, multivariate time series or documents can also be information sources [4]. Besides of these several type of models should be treated, e.g. graphs, decision trees, neural networks, rule based systems, etc. Although these models need different types of reduction methods (e.g. spanning trees for graphs, pruning for decision trees, rule base reduction for fuzzy models etc.), but they can be integrated by fuzzy systems (extended Takagi-Sugeno fuzzy models).
- *3.c Visualization* is a very promising field because it makes possible merging the best of computer (calculation) capabilities with the best of human perception (cognitive) abilities. There are several methods to visualize multidimensional data [5, 6]. However, the visualization of the results of the knowledge discovery process is much more complex than the visualization of multivariate data. The world-leading companies like AT&T and Microsoft are also dealing with these problems. The common feature of the solutions is that the novel visualization methods are based on novel similarity measures. The visualization of clustering results have been also dealt with by our research group, where a similarity of the fuzzy clusters are measured. In the last three years we also developed a new measure for multivariate data based on the introduction of a particular distance measure based on the correlation of the process variables. These results illustrate that for model mining purposes there is a clear need to define tailored similarity measures according to the heterogeneous information sources (similarity of models, images, graphs etc.).
- *3.d Application of model mining algorithm(s):* Selecting algorithms for searching for patterns. It is essential to involve the user or the experts into this key step because it is often very difficult to formalize the complex and many times contradictory goals. One possible solution is to increase the degree of interactivity. Such a tool will be developed in Chapter 5.

4. Interpreting and application of the mined patterns: Whereas the "knowledge worker" judges the success of the application of modelling and discovery techniques more technically, she or he contacts domain experts later in order to discuss the results and their applicability. Based on the results it can be necessary to return to any of steps 1-3 for further iteration.

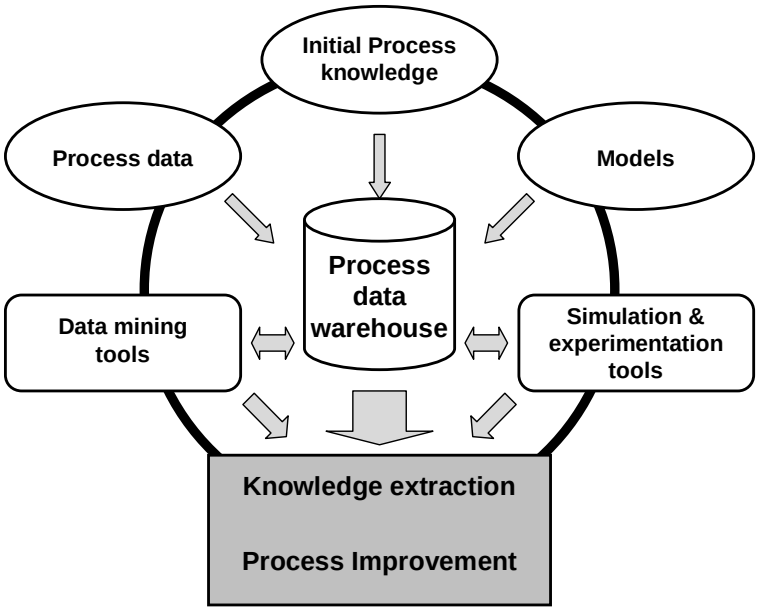


Figure 1.6: An integrated framework for process improvement.

1.3 Motivation, roadmap of the thesis

Concluding the previously described need for integration, the main motivation of the thesis is to create an integrated framework whereto data mining, modeling and simulation, experimentation tools can be incorporated. To achieve model integrity, the existing models should be reapplied, the non-existing models created, and all the models connected in an appropriate way. If it is possible to collect sufficiently large amount of data from the process, Knowledge Discovery in Databases (KDD) technique can be applied to extract information focused on the maintenance or control operation problems to make the production more efficient [7].

As I suggest in Figure 1.6, the information flow of such integrated methodologies should be centered around a process data warehouse in a process improvement cycle. Sources come from available process data, current process knowledge (rules, constraints, etc.) and an integrated global model of products, process and process control. As these information are collected in the data warehouse, data mining tools, modeling and experimentation tools can be applied to aid the improvement of the process while extracting further knowledge.

All these developments and the process data warehouse, which they are centered around, were created within the research project of the Cooperative Research Centre of Chemical Engineering Institute entitled as "Optimization of multi-product continuous technologies" with implementation at the Polypropylene plant of Tisza Chemical Group Plc., Hungary.

According to the motivations and main goals explained above, the thesis is structured as follows: Chapter 2 describes an integrated process data warehouse with applications to product quality and operating cost estimations, while Chapter 3 presents novel clustering based data analysis tools to be able to analyze data queried from or transferred to the data warehouse. As shown previously, process data and simulator models are linked together through experimentation, hence a genetic algorithm based tool for interactive process optimization was developed, which is detailed in Chapter 4. Till know some tools for process intensification, scaling up, control and monitoring have been worked out and implemented to demonstrate the potential impact of the developed approach in technology development. The details of these application examples are presented in Appendix A. As this chapter illustrates, the results presented in this thesis can be immediately applied in the field of process engineering by designing tools that can support model based process and product development. For readers not familiar with every concept and notation used in this theses the theoretical background of the contributions are briefly discussed in Appendix B.

Chapter 2

Process Data Warehousing and Mining

According to costumers' expectations and market challenge, process industry needs to have the ability to operate complex, highly interconnected plants that are profitable and that meet quality, safety, environmental and other standards. In ideal case this requires modeling and simulation tools, which integrate not only the whole product and process development chains, but all the process units, plants, and subdivisions of the company. These information islands are also formed at the level of the collection and analysis of historical process data, however the access to this data is limited due to the heterogeneity of the information sources (the data is generated in different places and stored in different type of databases) and due to the data is stored only in shorter periods of time. This chapter proposes a know-how for the design and implementation of process data warehouses that integrates plant-wide information, where integration means information, location, application and time integrity. The process data warehouse contains non-violate, consistent and preprocessed historical process data and works independently from other databases operating at the level of the control system. To extract information from historical process data stored in the process data warehouse tools for data mining and exploratory data analysis have been developed.

2.1 Motivation

The increasing automation and tighter quality constraints related to production processes make the operator's and plant engineer's job more and more difficult. The operators in the process industry have many tasks such as to keep the process condition as closely as possible to a given operating point, to preserve optimality, to detect failures and to maintain safety. The more heterogenous the units are the less transparent the system is. Hence, there is a need for integrated information system that solves these problems and supports process monitoring and development.

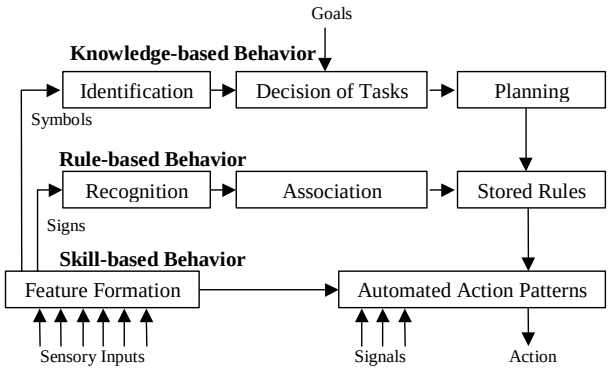


Figure 2.1: Three-level model of skilled human operator.

As the three-level model of the performance of skilled operators shown in Figure 2.1 suggests, such Operator Support Systems (OSS) should indicate intuitive and essential information on what is happening with the process and give suggestions according to operator's experience and skills [8], [9], [10]. Hence, the OSS of complex processes should be the combination of information systems, mathematical models and algorithms aimed to extract relevant information (signs, e.g. process trends and symbols) to "ease" the operator's work. In the following the main elements of this kind of system are described.

In modern industrial technologies the existence of a distributed control system (DCS) is a basic requirement. This system is responsible for the safe operation of the technology in the local level. In the coordination level of the DCS many complex tasks are handled, like controller tuning, process optimization, model identification and error diagnostic. These tasks are based on process models. As new products are required to be introduced to the market over a short time scale to ensure competitive advantage, the development of process models necessitates the use of empirical based techniques, since phenomenological model development is unrealizable in the time available [9]. Hence, the mountains of data that computer-controlled plants generate, must be effectively used. For this purpose most of the DCS systems are able to store operational process data. However, DCS has limited storage capacity because this is not its main function, only data logged in the last one or two months is stored in these computers. Since data measured in a longer time period have to be used for sophisticated process analysis, quality control, and model building, it is expedient to store data in a historical database that is useful to query, group and analyze the data related to the production of different products and different grade transitions. Today several software products in the market provides the capability of integration of historical process data of DCS's: e.g. Intellution I-historian [11], Siemens SIMATIC [12], the system of Fisher-Rosemount PlantWeb [13] and the Wonderware FactorySuite 2000 MMI software package [14].

As it will be deeply described in the the case study shown in Section A.1, there are several heterogeneous information sources that have to be integrated to support the work of engineers and operators with relevant, accurate and useful information. In case of process systems standard data warehousing and

OLAP techniques are not always suitable for this purpose because the operation units in the process industry have significant dynamical behavior that requires special attention contrary to the classical static business models. The source of this dynamical behavior is the dynamical effect of the transportation and mixing of material flows. Since process engineering systems rarely operate on steady-state manner (process transitions, product changes significantly occur), and the control and monitoring of these dynamical transitions are the most critical tasks of the process operators, the synchronization of data taken from heterogeneous information sources of process systems requires dynamical process models. These dynamic qualities of process units and the related data sources make it unsuitable to simply apply standard OLAP techniques. Hence, as it will be presented in the following section, the integration of the historical databases of DCS's into OSSs is not only a technical problem, in this process the special features of the technology have to be taken into account.

2.2 Process Data Warehouse

In case of complex processes the design of integrated information system is extremely important. This is is especially true in higher level of control of the process and enterprize. E.g. to monitor and control of the quality of the production (better product quality, less environmental pollution, less off-specification product etc.), data taken from different units of the production-line (e.g. quality of the row materials, operating parameters of the reaction system, product quality measurements) have to be analyzed. Since there is a strong but complex and dynamically changing dependency among these data, there is a need to improve the classical functions and models of standard Data Warehouses to help the work of operators.

The proposed *focus on process approach* means stronger focus on the material and information flow through the entire enterprise, where the OSS follows the process through the organization instead of focusing separate tasks of the isolated process units. This also means that most of the information moves horizontally within the organization, thus requiring a higher degree of cooperation and communication across the different divisions in the plant [10], [15].

This session proposes the integration of heterogeneous historical data taken from the various production units into a data warehouse with focus on the specialties of the technology. The resulted model-based information system contains only consistent, non-violate, pre-processed, historical data [16], and it is working independently from the distributed control system (DCS). This type of data warehouse has the features presented Table 2.1, and it is called as Process Data Warehouse.

	DCS relational database	Process Data Warehouse
Function	day to day data storage	decision supporting
Data	actual, daily, detailed, included in relation, isolated	historical, summarized, integrated, consolidated
Using	iterative	ad-hoc
Unit of work	short, general transactions	complex queries
User	operator	engineer, operator plant manager
Design	application oriented	subject oriented
Access	read-write	engineer, many queries
Number of accessed records	decimal order	million order
Size	100 MB-GB	100 GB-TB
Degree	transactional time	inquiry time
Region	process unit	whole production line
Detectable dependencies (type of analysis)	static	dynamic
Main elements	SQL database of DCS	SQL database of DCS *and* independent DW *and* computational unit of models

Table 2.1: The main differences between the DCS relational database and the Process Data Warehouse.

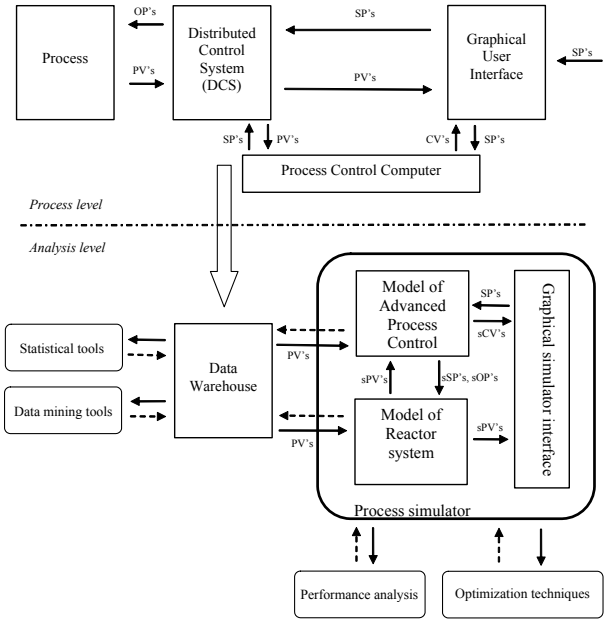


Figure 2.2: the integrated methodology of process analysis for a complex, DCS regulated process

2.3 Integrated Framework for Process Development

The developed components for an integrated information system are shown in Figure 2.2. It shows the structure of the proposed process analysis methodology for the development of a complex systems. This structure supposes that there are a DCS (with data storage functions) and a process computer in the system, so a process Data Warehouse integrated into the framework can be created and all the developed tools are centered around this data warehouse.

Process Data Warehouse. The data stored by DCS definitely have the potential to provide information for product and process design, monitoring and control. However, these data have limited access in time on the process control computers, since they are archived retrospectively, and can be unreliable because of measurement failure or inconsistent storage. Process Data Warehouse is a data analysis-decision support and information process unit, which operates separately from the databases of the DCS. It is an information environment in contrast to the data transfer-oriented environment, which contains trusted, processed and collected data for historic data analysis. The data collected into DW directly provide input for different data mining, statistical tools, like classification, clustering, association rules, etc., and visualization techniques, e.g. quantile-quantile plots, box plots, histograms, etc. Besides these tools and techniques, DW indirectly creates a basis for optimization and system performance analysis techniques through a process simulator of the process and its advanced local control system, since models can be validated based on historic data stored in DW.

(Dynamic) Data model. Actually, data is simply a record of all business activities, resources, and results of the organization. The data model is a well-organized abstraction of that data. So, it is quite natural that the data model has become the best method to understand and manage the business of the organization. Without a data model, it would be very difficult to organize the structure and contents of the data in the data warehouse [17]. The application of data and enterprize modeling (EM) is extremely important, as these models describe the organization, maps the work-processes, and thereby identifies the needs of OSS. The data model plays the role of a guideline, or plan, to implement the data warehouse. The design of a process data warehouse is based on the synchronization of the events related to the different information sources which requires the understanding the material, energy and information flow between the units of the plant. For this purpose not only classical data modeling techniques have to be used, but models related to the nonlinear functional relationships of the process and product variables and dynamic models that represent the dynamical behavior of these variables.

Process model. It is an integrated application of laboratory kinetics, thermodynamics, transport phenomena and experiments with plant scale-up parameters embedded into different process unit models. Therefore, a multi-scale model, whose complexity depends on the current technology. Its parts can be achieved by first principle, black-box or semi-mechanistic (hybrid) modeling approaches. Advanced control and monitoring algorithms of OSS are based on

state variables which are not always measurable or they are measured off-line. Hence, for the effective application of these tools there is a need for state estimation algorithms that are based on the model of the monitored and/or controlled process. In the presence of additive white Gaussian noise Kalman filter provides optimal estimates of the states of a linear dynamical system. For nonlinear processes Extended Kalman Filtering (EKF) should be used [18]. The dynamic model of EKF can be a *first-principle model* formulated by a set of nonlinear differential equations or *black-box model*, e.g. a neural network (NN). Generally, models used in the state estimation of process systems are formulated by macroscopic balance equations, for instance, mass or energy balances. In general, not all of the terms in these equation are exactly or even partially known. In semi-mechanistic modeling black-box models, like neural networks are used to represent the otherwise difficult-to-obtain parts of the model. Usually, in the modeling phase it turns out which parts of the first principles model are easier and which are more laborious to obtain and often we can get the so-called hybrid model structure that integrates a first-principle model with a NN model which serves as an estimator of unmeasured process parameters that are difficult to model from first-principles [19]. Since this seminal paper of Psichogios, many industrial applications of these semi-mechanistic models have been reported, and it has been proven that this kind of models has better properties than stand-alone NN applications, e.g. in the pyrolysis of ethane [20], in industrial polymerization [21], and or bioprocess optimization [22]. The aim of the case study of this thesis is the examination of the applicability of such semi-mechanistic models in industrial environment, namely how this model structure can be identified and applied for state estimation in OSS.

Product model i.e. inferential model. Models that are attached to process models, hence in many applications they are not considered separately from them, but inferential product models are rather closely related to product attributes than to process models. For example, if the process model defines the composition of a reactor liquid phase output stream, a possible product model can estimate boiling curve of the output mixture. They can also be modeled by different approaches for proper estimation of property relationships. Formulated products (plastics, polymer composites) are generally produced from many ingredients, and large number of the interactions between the components and the processing conditions all have the effect on the final product quality [23]. When a reliable nonlinear model is available that is able to estimate the quality of the product, it can be inverted to obtain the suitable operating conditions required for achieving the target product quality [24]. If such model is incorporated to the OSS, significant economic benefits can be realized. To estimate the product quality, an approximate reasoning system is needed which is capable of handling imperfect information. In the proposed structure with the integration of modeling and monitoring functions a new method is developed which based on semi-mechanistic modeling and nonlinear-state estimation was proposed for this purpose. For the identification of a neural network a spline-smoothing approach has been followed, where splines have been used to extract the desired outputs of the neural network from infrequent and noisy measurements. The results

show that the proposed process data warehousing and data mining methods are efficient and useful tools for data integration, decision support, state and product quality estimation, which tools can be useful to increase the productivity of complex technological processes.

Process Control model. It uses the designed structure of regulatory process control system, information about the controlled and the perturbed variables, possible states, and operation ranges. In case of a complex system, usually distributed control system (DCS) assures locally the secure and safe operating of the technology. It is extended by an advanced model based process control computer (Process Computer) that calculates among others the operation set points (OP's) to DCS.

(Graphical) Interface, Front-end Tools. It handles the input-output connections between process-product model, control model, data warehouse and the user. Complex process technologies are multivariable, exhibit nonlinear characteristics, and often have significant time delays. In this case the operator cannot easily follow and visualize what is happening in the process, so the computer should aid for visualization of the process states and their relation to the quality of the final product. As the final product quality is measured in the quality control laboratory, not only WYSIWYW (What You See Is What You Want) interfaces between the operator and the console are important but WYSIWIS (What You See Is What I See) interfaces between the operators (operators at the reactor, at the product formation process and at the laboratory) are needed to share the information horizontally in the organization. A data warehouse provides the base for the powerful data analysis techniques that are available today such as data mining and multidimensional analysis, as well as the more traditional query and reporting. Making use of these techniques along with process data warehousing can result in easier access to the information the operators need for more informed decision making.

Plant operators are skilled in the extraction of real-time patterns of process data and the identification of distinguishing features (see Figure 2.1). Hence, the correct interpretation of measured process data is essential for the satisfactory execution of many computer-aided, intelligent decision support systems (DSS) that modern processing plants require. The aim of the incorporation of multivariate statistical based approaches into the OSS is to reduce the dimensionality of the correlated process data by projecting them down onto a lower dimensional latent variable space where the operation can be easily visualized. These approaches use the techniques of principal component analysis (PCA) or projection to latent structure (PLS). Beside process performance monitoring, these tools can be used for system identification [24], [25], ensuring consistent production and product design [26]. The potential of existing approaches has been limited by its inability to handle more than one recipe/grade. There is, therefore, a need for methodologies from which process representations can be developed which simultaneously handle a range of products, grade and recipes [9].

In supervisory control, detection and diagnosis of faults, product quality control and recovery from large operation derivations, determining the mapping from process trends to operating conditions is the pivotal task. Query and reporting analysis is the process of posing a question to be answered, retrieving relevant data from the data warehouse, transforming it into the appropriate context, and displaying it in a readable format. It is driven by analysts who must pose those questions to receive an answer. These tasks are quite different from data mining, which is data driven.

For particular analysis, the applicability of the integrated model depends on the applied components. It can be a soft-sensor (e.g. used product quality estimation as it is shown in Section A.3), process monitoring (e.g. state estimation and visualization), reasoning or reverse engineering tool (e.g. production parameter estimation), operator training/qualification (e.g. state transition optimization, product classification) or decision support system application.

2.4 Conclusions

In this chapter I proposed a novel know-how for the design and implementation of process data warehouses that integrates plant-wide information, where integration means information, location, application and time integrity. The process data warehouse contains non-violate, consistent and preprocessed historical process data and works independently from other databases operating at the level of the control system [1]. I have also pointed out that such information system should also contain the model of the control system[2]. The details of the components of the presented information system define whether the resulted data warehouse supports soft sensors, process monitoring, reverse engineering or operator training/qualification. When the simulator outputs are also stored in the DW, comparison of the simulated outputs to the real industrial data can provide further information for optimization and tuning the parameters of the control systems. This scheme results in a DW-centered continuous process improvement cycle. Generally, the advantage of having an offline simulator of the system is that it can be used to predict product quality, estimate the state of the system and find new optimal operating points in a multi-objective environment, results of operability tests, effects of e.g. new recipes or catalyst can be investigated without any cost attachment or system failure, and it is easily extendable for system performance analysis tools and optimization techniques.

[1] Pach F P, Feil B, Nemeth S, Arva P, Abonyi J, Process-data-warehousing-based operator support system for complex production technologies, IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART A-SYSTEMS AND HUMANS 36: pp. 136-153. (2006), IF: 0.980, Independent citations: 4

[2] Balasko B, Nemeth S, Nagy G, Abonyi J, Integrated Process and Control System Model for Product Quality Control - Application to a Polypropylene Plant, Chemical Product and Process Modeling , 3:(1) pp. 1-12. Paper 50. (2008), DOI: 10.2202/1934-2659.1213

I have successfully applied the prototype of this system for estimation of product quality by a new semi-mechanistic product model extension and for the extraction of cost and energy consumption based on box-plots and quantile-quantile plots [3]. To extract information from historical process data stored in the process data warehouse tools for data mining and exploratory data analysis have been developed. [4]

In case of complex production processes it is often not sufficient to analyze only input-output data for process monitoring purposes. The reasons may be that historical process data alone do not have enough information content, it can be incomplete, not measured frequently or not at regular intervals. In these cases it is important to obtain information about state variables; therefore (nonlinear) state estimation algorithm is needed. This phenomenon has been proved experimentally at the time-series segmentation based process monitoring, where the result of the segmentation was much more reliable when the estimated state variables or the error covariance matrices computed by the state estimation algorithm have been also utilized by the segmentation algorithms[5]. When models of process and control systems are integrated to a process Data Warehouse the resulted structure support engineering tasks related to analysis of system performance, process optimization, operator training (OTS), reverse engineering, and form decision support (DSS) systems.

In chemical production processes it often happen that the product quality can be measured only relatively rarely or with considerable dead time (e.g. because of the time demand of laboratory tests). In these situations it would be advantageous if the product quality could be estimated using a state estimation algorithm. However, due to the complexity of the production processes there are often no enough a priori knowledge to build a proper model to estimate the important, but unknown process variables[6]. In these cases black box models are worth applying to approximate the unknown phenomena and building into the white box model of the system. These models are called semi-mechanistic models. Such semi-mechanistic model was developed for the on-line product quality estimation in an industrial polyethylene reactor. Since in the proposed semi-mechanistic model structure a neural network is designed as a part of a nonlinear algebraic-differential equation set, there were no available direct input-output data to train the weights of the network. To handle this problem a simple, yet practically useful spline-smoothing based technique has been used[7].

[3] Abonyi J, Application of Exploratory Data Analysis to Historical Process Data of Polyethylene Production, HUNGARIAN JOURNAL OF INDUSTRIAL CHEMISTRY 35: pp. 85-93. (2007)

[4] Abonyi J, Nemeth S, Vincze C, Arva P, Process analysis and product quality estimation by Self-Organizing Maps with an application to polyethylene production, COMPUTERS IN INDUSTRY 52: pp. 221-234. (2003), IF: 0.692, Independent citations: 10

[5] Feil B, Abonyi J, Nemeth S, Arva P, Monitoring process transitions by Kalman filtering and time-series segmentation, COMPUTERS & CHEMICAL ENGINEERING 29: pp. 1423-1431. (2005), IF: 1.501, Independent citations: 5

[6] Feil B, Abonyi J, Pach P, Nemeth S, Arva P, Nemeth M, Nagy G, Semi-mechanistic models for state-estimation - Soft sensor for polymer melt index prediction, LECTURE NOTES IN ARTIFICIAL INTELLIGENCE 3070: pp. 1111-1117. (2004), IF: 0.251, Independent citations: 3

[7] Abonyi J, Roubos H, Babuska R, Szeifert F, Identification of Semi-Mechanistic Models with Interpretable TS-fuzzy submodels by Clustering, OLS and FIS Model Reduction. In: J Casillas,

Similary to the design of soft-sensors, the bottleneck of nonlinear model based controller design is also the modeling of the controlled system. In practice, the effectiveness of nonlinear controllers is limited due to the uncertainties in model parameters, e.g. kinetic parameters, and in model structure. To cope with this problem, a semi-mechanistic model has been developed which combines a priori and a posteriori models in such a way that the uncertain part of the a priori model is replaced by an artificial neural network. The effectiveness of this approach has been demonstrated by the nonlinear control of a simulated continuous stirred tank reactor[8].

O Cordon, F Herrera, L Magdalena (szerk.) Fuzzy modeling and the interpretability-accuracy trade-off, Heidelberg: Physica Verlag, 2003,pp. 221-248. Independent citations: 2

[8]Madar J, Abonyi J, Szeifert F, Feedback linearizing control using hybrid neural networks identified by sensitivity approach, ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE 18: pp. 343-351. (2005), IF: 0.709, Independent citations: 7

Chapter 3

Fuzzy Clustering for Regression, System Identification, and Classifier Induction

The amount of the data stored in various information systems grows very fast. These data sets could contain hidden, potentially useful knowledge. Clustering, as a special area of data mining is, one of the most commonly used methods for discovering the hidden structure of the considered data set. The main goal of clustering is to divide objects into well separated groups in a way that objects lying in the same group are more similar to each another than to objects in other groups. In the literature several clustering and visualization methods can be found. However, due to the huge variety of problems and data sets, it is a difficult challenge to find a powerful method that is adequate for all problems. In this chapter I summarize the results I obtained at the development of problem-specific clustering algorithms.

3.1 Fuzzy Clustering for Nonlinear Regression

Fuzzy identification is an effective tool for the approximation of uncertain nonlinear systems on the basis of measured data [27]. Among the different fuzzy modeling techniques, the Takagi-Sugeno (TS) model [28] has attracted most attention. This model consists of if–then rules with fuzzy antecedents and mathematical functions in the consequent part (see Section B.2 for more details). The antecedents fuzzy sets partition the input space into a number of fuzzy regions, while the consequent functions describe the system's behavior in these regions [29]. The construction of a TS model is usually done in two steps. In the first step, the fuzzy sets (membership functions) in the rule antecedents are determined. This can be done manually, using knowledge of the process, or by some data-driven techniques. In the second step, the parameters of the consequent functions are estimated. As these functions are usually chosen to be linear in their parameters, standard linear least-squares methods can be applied.

The bottleneck of the construction of fuzzy models is the identification of the antecedent membership functions, which is a nonlinear optimization problem. Typically, gradient-decent neuro-fuzzy optimization techniques are used [30], with all the inherent drawbacks of gradient-descent methods: (1) the optimization is sensitive to the choice of initial parameters and hence can easily get stuck in local minima; (2) the obtained model usually has poor generalization properties; (3) during the optimization process, fuzzy rules may loose their initial meaning (i.e., validity as local linear models of the system under study). This hampers the a posteriori interpretation of the optimized TS model. An alternative solution are gradient-free nonlinear optimization algorithms. Genetic algorithms proved to be useful for the construction of fuzzy systems [31, 32]. Unfortunately, the severe computational requirements limit their applicability as a rapid model-development tool.

Fuzzy clustering in the Cartesian product-space of the inputs and outputs is another tool that has been quite extensively used to obtain the antecedent membership functions [33, 34, 35]. Attractive features of this approach are the simultaneous identification of the antecedent membership functions along with the consequent local linear models and the implicit regularization [36]. By clustering in the product-space, multidimensional fuzzy sets are initially obtained, which are either used in the model directly or after projection onto the individual antecedent variables. As it is generally difficult to interpret multidimensional fuzzy sets, projected one dimensional fuzzy sets are usually preferred. However, the projection and the approximation of the point-wise defined membership functions by parametric ones may deteriorate the performance of the model. This is due to two types of errors: the decomposition error and the approximation error. The decomposition error can be reduced by using eigenvector projection [35, 37] and/or by fine-tuning the parameterized membership functions. This fine-tuning, however, can result in overfitting and thus poor generalization of the identified model.

In this section, a new cluster prototype is introduced, that can easily be represented by an interpretable Takagi-Sugeno (TS) fuzzy model. Similarly to other fuzzy clustering algorithms, the alternating optimization method is employed in the search for the clusters. This new technique is demonstrated on the MPG (miles per gallon) prediction problem. The obtained results are compared with results from the literature. It is shown that with the presented modified Gath–Geva algorithm not only good prediction performance is obtained, but also the interpretability of the model improves.

Clustering based Fuzzy Model Identification

The available data samples are collected in matrix $\mathbf{Z}$ formed by concatenating the regression data matrix $\mathbf{X}$ and the output vector $\mathbf{y}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \qquad \mathbf{Z}^T = [\mathbf{X}\,\mathbf{y}]\,. \tag{3.1}$$

Each observation thus is an $n+1$-dimensional column vector $\mathbf{z}_k = [x_{1,k}, \ldots, x_{n,k}, y_k]^T = [\mathbf{x}_k^T\, y_k]^T$. Through clustering, the data set $\mathbf{Z}$ is partitioned into c clusters. The c is assumed to be known, based on prior knowledge, for instance (refer to [35] for methods to estimate or optimize c in the context of system identification). The result is a fuzzy partition matrix $\mathbf{U} = [\mu_{i,k}]_{c\times N}$, whose element $\mu_{i,k}$ represents the degree of membership of the observation $\mathbf{z}_k$ in cluster i.

Clusters of different shapes can be obtained by using an appropriate definition of cluster prototypes (e.g., linear varieties) or by using different distance measures. The Gustafson–Kessel (GK) clustering algorithm has often been applied to identify TS models. The main drawbacks of this algorithm are that only clusters with approximately equal volumes can be properly identified and that the resulted clusters cannot be directly described by univariate parametric membership functions.

To circumvent these problems, Gath–Geva algorithm [38] is applied. Since the cluster volumes are not restricted in this algorithm, lower approximation error and more relevant consequent parameters can be obtained than with Gustafson–Kessel (GK) clustering. An example can be found in [35], p. 91. The clusters obtained by GG clustering can be transformed into exponential membership functions defined on the linearly transformed space of the input variables.

Probabilistic Interpretation of Gath–Geva Clustering

The Gath–Geva clustering algorithm can be interpreted in the probabilistic framework. Denote $p(\eta_i)$ the unconditional cluster probability (normalized such that $\sum_{i=1}^{c} p(\eta_i) = 1$), given by the fraction of the data that it explains; $p(\mathbf{z}|\eta_i)$ is the domain of influence of the cluster, and will be taken to be multivariate gaussian $N(\mathbf{v}_i, \mathbf{F}_i)$ in terms of a mean $\mathbf{v}_i$ and covariance matrix $\mathbf{F}_i$. The Gath–Geva algorithm is equivalent to the identification of a mixture of Gaussians that model the $p(\mathbf{z}|\eta)$ probability density function expanded into a sum over the c clusters

$$p(\mathbf{z}|\eta) = \sum_{i=1}^{c} p(\mathbf{z}, \eta_i) = \sum_{i=1}^{c} p(\mathbf{z}|\eta_i)p(\eta_i) \tag{3.2}$$

where the $p(\mathbf{z}|\eta_i)$ distribution generated by the i-th cluster is represented by the Gaussian function

$$p(\mathbf{z}|\eta_i) = \frac{1}{(2\pi)^{\frac{n+1}{2}}\sqrt{|\mathbf{F}_i|}} \exp\left(-\frac{1}{2}(\mathbf{z}-\mathbf{v}_i)^T(\mathbf{F}_i)^{-1}(\mathbf{z}-\mathbf{v}_i)\right). \tag{3.3}$$

Through GG clustering, the $p(\mathbf{z}) = p(\mathbf{x}, y)$ joint density of the response variable y and the regressors $\mathbf{x}$ is modeled as a mixture of c multivariate $n+1$-dimensional Gaussian functions.

The conditional density $p(y|\mathbf{x})$ is also a mixture of Gaussian models. Therefore, the regression problem can be formulated on the basis of this probability as

$$\begin{aligned} y = f(\mathbf{x}) &= E[y|\mathbf{x}] = \\ &= \int yp(y|\mathbf{x})dy = \frac{\int yp(y,\mathbf{x})dy}{p(\mathbf{x})} = \\ &= \sum_{i=1}^{c} \frac{\left[[\mathbf{x}^T\ 1]\theta_i\right] p(\mathbf{x}|\eta_i)p(\eta_i)}{p(\mathbf{x})} = \sum_{i=1}^{c} p(\eta_i|\mathbf{x}) \left[[\mathbf{x}^T\ 1]\theta_i\right] . \end{aligned} \quad (3.4)$$

Here, θ_i is the parameter vector of the local models to be obtained later on (Section 3.1) and $p(\eta_i|\mathbf{x})$ is the probability that the i-th Gaussian component is generated by the regression vector $\mathbf{x}$:

$$p(\eta_i|\mathbf{x}) = \frac{\frac{p(\eta_i)}{(2\pi)^{n/2}\sqrt{|\mathbf{F}_i^{xx}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{v}_i^x)^T(\mathbf{F}_i^{xx})^{-1}(\mathbf{x}-\mathbf{v}_i^x)\right)}{\sum_{i=1}^{c} \frac{p(\eta_i)}{(2\pi)^{n/2}\sqrt{|(\mathbf{F})_i^{xx}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{v}_i^x)^T(\mathbf{F}_i^{xx})^{-1}(\mathbf{x}-\mathbf{v}_i^x)\right)} \quad (3.5)$$

where $\mathbf{F}^{xx}$ is obtained by partitioning the covariance matrix $\mathbf{F}$ as follows

$$\mathbf{F}_i = \begin{bmatrix} \mathbf{F}_i^{xx} & \mathbf{F}_i^{xy} \\ \mathbf{F}_i^{yx} & \mathbf{F}_i^{yy} \end{bmatrix} \quad (3.6)$$

where

- $\mathbf{F}_i^{xx}$ is the $n \times n$ submatrix containing the first n rows and columns of $\mathbf{F}_i$,
- $\mathbf{F}_i^{xy}$ is an $n \times 1$ column vector containing the first n elements of last column of $\mathbf{F}_i$,
- $\mathbf{F}_i^{yx}$ is an $1 \times n$ row vector containing the first n elements of the last row of $\mathbf{F}_i$, and
- $\mathbf{F}_i^{yy}$ is the last element in the last row of $\mathbf{F}_i$.

Construction of Antecedent Membership Functions

The 'Gaussian Mixture of Regressors' model [39] defined by (3.4) and (3.5) is in fact a kind of operating regime-based model where the validity function is chosen as $\phi_i(\mathbf{x}) = p(\eta_i|\mathbf{x})$. Furthermore, this model is also equivalent to the TS fuzzy model where the rule weights in (B.26) are given by:

$$w_i = \frac{p(\eta_i)}{(2\pi)^{n/2}\sqrt{|\mathbf{F}_i^{xx}|}} \tag{3.7}$$

and the membership functions are the Gaussians. However, in this case, $\mathbf{F}_i^{xx}$ is not necessarily in the diagonal form and the decomposition of $\mathbf{A}_i(\mathbf{x})$ to the univariate fuzzy sets $A_{i,j}(x_j)$ is not possible.

If univariate membership functions are required (for interpretation purposes), such a decomposition is necessary. Two different approaches can be followed.

The first one is an approximation, based on the axis-orthogonal projection of $\mathbf{A}_i(\mathbf{x})$. This approximation will typically introduce some decomposition error, which can, to a certain degree, be compensated by using global least-squares re-estimation of the consequent parameters. In this way, however, the interpretation of the local linear models may be lost, as the rule consequents are no longer local linearizations of the nonlinear system [40, 41].

The second approach is an exact one, based on eigenvector projection [35], also called the transformed input-domain approach [37]. Denote $\lambda_{i,j}$ and $\mathbf{t}_{i,j}$, $j = 1, \ldots, n$, the eigenvalues and the unitary eigenvectors of $\mathbf{F}_i^{xx}$, respectively. Through the eigenvector projection, the following fuzzy model is obtained in the transformed input domain:

$$R_i: \ \textbf{If } \widetilde{x}_{i,1} \text{ is } A_{i,1}(\widetilde{x}_{i,1}) \textbf{ and } \ldots \textbf{ and } \widetilde{x}_{i,n} \text{ is } A_{i,n}(\widetilde{x}_{i,n}) \textbf{ then } \hat{y} = \mathbf{a}_i^T\mathbf{x} + b_i \tag{3.8}$$

where $\widetilde{x}_{i,j} = \mathbf{t}_{i,j}^T\mathbf{x}$ are the transformed input variables. The Gaussian membership functions are given by

$$A_{i,j}(\widetilde{x}_{i,j}) = \exp\left(-\frac{1}{2}\frac{(\widetilde{x}_{i,j} - \widetilde{v}_{i,j})^2}{\widetilde{\sigma}_{i,j}^2}\right) \tag{3.9}$$

with the cluster centers $\widetilde{v}_{i,j} = \mathbf{t}_{i,j}^T\mathbf{v}_i^x$ and and variances $\widetilde{\sigma}_{i,j}^2 = \lambda_{i,j}^2$.

Estimation of Consequent Parameters

Two least-squares methods for the estimation of the parameters in the local linear consequent models are presented: weighted total least squares and weighted ordinary least squares.

- **Ordinary Least-Squares Estimation**

  The ordinary weighted least-squares method can be applied to estimate the consequent parameters in each rule separately, by minimizing the following criterion:

$$\min_{\boldsymbol{\theta}_i} \frac{1}{N}\left(\mathbf{y} - \mathbf{X}_e\boldsymbol{\theta}_i\right)^T \boldsymbol{\Phi}_i \left(\mathbf{y} - \mathbf{X}_e\boldsymbol{\theta}_i\right) \tag{3.10}$$

where $\mathbf{X}_e = [\mathbf{X}\ \mathbf{1}]$ is the regressor matrix extended by a unitary column and $\mathbf{\Phi}_i$ is a matrix having the membership degrees on its main diagonal:

$$\mathbf{\Phi}_i = \begin{bmatrix} \mu_{i,1} & 0 & \cdots & 0 \\ 0 & \mu_{i,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu_{i,N} \end{bmatrix} . \tag{3.11}$$

The weighted least-squares estimate of the consequent parameters is given by

$$\boldsymbol{\theta}_i = \left(\mathbf{X}_e^T \mathbf{\Phi}_i \mathbf{X}_e\right)^{-1} \mathbf{X}_e^T \mathbf{\Phi}_i \mathbf{y} \,. \tag{3.12}$$

When $\mu_{i,k}$ is obtained by the Gath–Geva clustering algorithm, the covariance matrix can directly be used to obtain the estimate instead of (3.12):

$$\begin{aligned} \mathbf{a}_i &= \left(\mathbf{F}^{xx}\right)^{-1} \mathbf{F}^{xy} \,, \\ b_i &= v_i^y - \mathbf{a}_i^T \mathbf{v}_i^x \,. \end{aligned} \tag{3.13}$$

This follows directly from the properties of least-squares estimation [42].

- **Total Least-Squares Estimation**

  As the clusters locally approximate the regression surface, they are n-dimensional linear subspaces of the $(n+1)$-dimensional regression space. Consequently, the smallest eigenvalue of the i-th cluster covariance matrix $\mathbf{F}_i$ is typically in orders of magnitude smaller than the remaining eigenvalues [35]. The corresponding eigenvector $\mathbf{u}_i$ is then the normal vector to the hyperplane spanned by the remaining eigenvectors of that cluster:

$$\mathbf{u}_i^T (\mathbf{z} - \mathbf{v}_i) = 0 \,. \tag{3.14}$$

  Similarly to the observation vector $\mathbf{z} = [\mathbf{x}^T\ y]^T$, the prototype vector and is partitioned as $\mathbf{v}_i = \left[(\mathbf{v}_i^x)^T\ v_i^y\right]$, i.e., into a vector $\mathbf{v}^x$ corresponding to the regressor $\mathbf{x}$, and a scalar v_i^y corresponding to the output y. The eigenvector is partitioned in the same way, $\mathbf{u}_i = \left[(\mathbf{u}_i^x)^T\ u_i^y\right]^T$. By using these partitioned vectors, (3.14) can be written as

$$\left[(\mathbf{u}_i^x)^T\ u_i^y\right] \left([\mathbf{x}^T\ y] - \left[(\mathbf{v}_i^x)^T\ v_i^y\right]\right)^T = 0 \tag{3.15}$$

  from which the parameters of the hyperplane defined by the cluster can be obtained:

$$y = \underbrace{\frac{-1}{u_i^y} (\mathbf{u}_i^x)^T}_{\mathbf{a}_i^T} \mathbf{x} + \underbrace{\frac{1}{u_i^y} (\mathbf{u}_i)^T \mathbf{v}_i}_{b_i} \,. \tag{3.16}$$

  Although the parameters have been derived from the geometrical interpretation of the clusters, it can be shown [35] that (3.16) is equivalent to the weighed total least-squares estimation of the consequent parameters, where each data point is weighed by the corresponding membership degree.

The TLS algorithm should be used when there are errors in the input variables. Note, however, that the TLS algorithm does not minimize the mean-square prediction error of the model, as opposed to the ordinary least-squares algorithm. Furthermore, if the input variables of the model are locally strongly correlated, the smallest eigenvector then does not define a hyperplane related to the regression problem; it may rather reflect the dependency of the input variables.

Modified Gath–Geva Clustering

The main drawback of the construction of interpretable Takagi–Sugeno fuzzy models via clustering is that clusters are generally axes-oblique rather than axes-parallel (the fuzzy covariance matrix $\mathbf{F}^{xx}$ has non-zero off-diagonal elements) and consequently a decomposition error is made in their projection. To circumvent this problem, I propose a new fuzzy clustering method in this section.

Each cluster is described by an input distribution, a local model and an output distribution:

$$\begin{aligned} p(\mathbf{x}, y) &= \sum_{i=1}^{c} p(\mathbf{x}, y, \eta_i) = \sum_{i=1}^{c} p(\mathbf{x}, y|\eta_i) p(\eta_i) \\ &= \sum_{i=1}^{c} p(y|\mathbf{x}, \eta_i) p(\mathbf{x}|\eta_i) p(\eta_i) \,. \end{aligned} \tag{3.17}$$

The input distribution, parameterized as an unconditional Gaussian [43], defines the domain of influence of the cluster similarly to the multivariate membership functions

$$p(\mathbf{x}|\eta_i) = \frac{1}{(2\pi)^{\frac{n}{2}} \sqrt{|\mathbf{F}_i^{xx}|}} \exp\left(-\frac{1}{2} (\mathbf{x} - \mathbf{v}_i^x)^T (\mathbf{F}_i^{xx})^{-1} (\mathbf{x} - \mathbf{v}_i^x) \right) . \tag{3.18}$$

The output distribution is

$$p(y|\mathbf{x}, \eta_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y - \mathbf{x}^T\theta_i)^T (y - \mathbf{x}^T\theta_i)}{2\sigma_i^2} \right) . \tag{3.19}$$

When the transparency and interpretability of the model is important, the cluster covariance matrix $\mathbf{F}^{xx}$ can be reduced to its diagonal elements similarly to the simplified axis-parallel version of the Gath–Geva clustering algorithm [44]:

$$p(\mathbf{x}_k|\eta_i) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_{i,j}^2}} \exp\left(-\frac{1}{2} \frac{(x_{j,k} - v_{i,j})^2}{\sigma_{i,j}^2} \right) . \tag{3.20}$$

The identification of the model means the determination of the cluster parameters: $p(\eta_i), \mathbf{v}_i^x, \mathbf{F}_i^{xx}, \theta_i, \sigma_i$. Bellow, the expectation maximization (EM) identification of the model is presented, followed by a re-formulation of the algorithm in the form of fuzzy clustering.

The basics of EM are the following. Suppose we know some observed values of a random variable $\mathbf{z}$ and we wish to model the density of $\mathbf{z}$ by using a

model parameterized by η. The EM algorithm obtains an estimate $\hat{\eta}$ that maximizes the likelihood $\mathcal{L}(\eta) = p(\mathbf{z}|\eta)$ by iterating over the following two steps:

- **E-step** In this step, the current cluster parameters η_i are assumed to be correct and based on them, the posterior probabilities $p(\eta_i|\mathbf{x}, y)$ are computed. These posterior probabilities can be interpreted as the probability that a particular piece of data was generated by the particular cluster's distribution. By using the Bayes theorem, the conditional probabilities are:

$$p(\eta_i|\mathbf{x}, y) = \frac{p(\mathbf{x}, y|\eta_i)p(\eta_i)}{p(\mathbf{x}, y)} = \frac{p(\mathbf{x}, y|\eta_i)p(\eta_i)}{\sum_{i=1}^{c} p(\mathbf{x}, y|\eta_i)p(\eta_i)} \,. \tag{3.21}$$

- **M-step** In this step, the current data distribution is assumed to be correct and the parameters of the clusters that maximize the likelihood of the data are sought. The new unconditional probabilities are:

$$p(\eta_i) = \frac{1}{N} \sum_{k=1}^{N} p(\eta_i|\mathbf{x}, y) \,. \tag{3.22}$$

The means and the weighted covariance matrices are computed by:

$$\mathbf{v}_i^x = \frac{\sum\limits_{k=1}^{N} \mathbf{x}_k p(\eta_i|\mathbf{x}_k, y_k)}{\sum\limits_{k=1}^{N} p(\eta_i|\mathbf{x}_k, y_k)} \,, \tag{3.23}$$

$$\mathbf{F}_i^{xx} = \frac{\sum\limits_{k=1}^{N} (\mathbf{x}_k - \mathbf{v}_i^x)(\mathbf{x}_k - \mathbf{v}_i^x)^T p(\eta_i|\mathbf{x}_k, y_k)}{\sum\limits_{k=1}^{N} p(\eta_i|\mathbf{x}_k, y_k)} \,. \tag{3.24}$$

In order to find the maximizing parameters of the local linear models, the derivative of the log-likelihood is set equal to zero:

$$\begin{aligned} 0 &= \frac{\partial}{\partial \theta_i} \ln \prod_{k=1}^{N} p(\mathbf{x}_k, y_k) = \sum_{k=1}^{N} \frac{\partial}{\partial \theta_i} \ln p(\mathbf{x}_k, y_k) \\ &= \frac{1}{N p(\eta_i)} \sum_{k=1}^{N} p(\eta_i|\mathbf{x}, y)\,(y_k - f_i(\mathbf{x}_k, \theta_i)) \frac{\partial f_i(\mathbf{x}_k, \theta_i)}{\partial \theta_i} \end{aligned} \tag{3.25}$$

Here, $f_i(\mathbf{x}_k, \theta_i)$ represents the local consequent models, $f_i(\mathbf{x}_k, \theta_i) = \mathbf{a}_i^T \mathbf{x}_k + b_i$. The above equation results in weighted least-squares identification of the local linear models (3.12) with the weighting matrix

$$\mathbf{\Phi}_j = \begin{bmatrix} p(\eta_i|\mathbf{x}_1, y_1) & 0 & \cdots & 0 \\ 0 & p(\eta_i|\mathbf{x}_2, y_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p(\eta_i|\mathbf{x}_N, y_N) \end{bmatrix} . \tag{3.26}$$

Finally, the standard deviations σ_i are calculated. These standard deviations are parameters of the $p(y|\mathbf{x}, \eta_i)$ distribution functions defined by (3.19).

$$\sigma_i^2 = \frac{\sum_{k=1}^{N} (y_k - f_i(\mathbf{x}_k, \theta_i))^T (y_k - f_i(\mathbf{x}_k, \theta_i)) p(\eta_i|\mathbf{x}_k, y_k)}{N p(\eta_i)} . \tag{3.27}$$

In this section, the EM algorithm is re-formulated to provide an easily implementable algorithm, similar to Gath–Geva clustering, for the identification of TS fuzzy models that do not use transformed input domains. See Algorithm 3.1.1.

Note that the distance measure (3.32) consists of two terms. The first one is the distance between the cluster centers and $\mathbf{x}$, while the second one quantifies the performance of the local linear models.

Algorithm 3.1.1 (Gath–Geva Clustering for Takagi–Sugeno Models)**.**

Initialization *Given the data set* $\mathbf{Z}$*, specify* c*, choose the weighting exponent* $m = 2$ *and the termination tolerance* $\epsilon > 0$*. Initialize the partition matrix such that* (B.1), (B.2) *and* (B.3) *holds.*

Repeat *for* $l = 1, 2, \ldots$ *(*l *is the iteration counter)*

Step 1 Calculate the parameters of the clusters:

- *Centers of the membership functions:*

$$\mathbf{v}_i^{x\,(l)} = \sum_{k=1}^{N} \mu_{i,k}^{(l-1)} \mathbf{x}_k / \sum_{k=1}^{N} \mu_{i,k}^{(l-1)} . \tag{3.28}$$

- *Standard deviations of the Gaussian membership functions:*

$$\sigma_{i,j}^{2\,(l)} = \sum_{k=1}^{N} \mu_{i,k}^{(l-1)} (x_{j,k} - v_{j,k})^2 / \sum_{k=1}^{N} \mu_{i,k}^{(l-1)} . \tag{3.29}$$

- *Parameters of the local models:*

$$\theta_i = \left(\mathbf{X}_e^T \mathbf{\Phi}_i \mathbf{X}_e \right)^{-1} \mathbf{X}_e^T \mathbf{\Phi}_i \mathbf{y} , \tag{3.30}$$

  where the weights are collected in the $\mathbf{\Phi}_i$ *matrix given by* (3.11).

- *A priori probabilities of the clusters:* $\alpha_i = \frac{1}{N} \sum_{k=1}^{N} \mu_{i,k}$.

- *Weights of the rules:*

$$w_i = \prod_{j=1}^{n} \frac{\alpha_i}{\sqrt{2\pi\sigma_{i,j}^2}} . \tag{3.31}$$

Step 2 Compute the distance measure $D_{i,k}^2$**:**

$$\begin{aligned} \frac{1}{D_{i,k}^2} \quad = \quad & \prod_{j=1}^{n} \frac{\alpha_i}{\sqrt{2\pi\sigma_{i,j}^2}} \exp\left(-\frac{1}{2} \frac{(x_{j,k} - v_{i,j})^2}{\sigma_{i,j}^2} \right) \cdot \\ & \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_k - f_i(\mathbf{x}_k, \theta_i))^T (y_k - f_i(\mathbf{x}_k, \theta_i))}{2\sigma_i^2} \right) . \end{aligned} \tag{3.32}$$

Step 3 *Update the partition matrix*

$$\mu_{i,k}^{(l)} = \frac{1}{\sum_{j=1}^{c} \left(D_{i,k}(\mathbf{z}_k, \eta_i) / D_{j,k}(\mathbf{z}_k, \eta_j) \right)^{2/(m-1)}}, \; 1 \le i \le c, \; 1 \le k \le N . \tag{3.33}$$

until $||\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}|| < \epsilon$.

Example 3.1. Comparative study among clustering based techniques

The example under consideration is the Automobile MPG (miles per gallon) prediction benchmark. The following methods were used and compared:

1. *GG-TLS: Gath–Geva clustering with total least-squares estimation of the consequent parameters.*
2. *GG-LS: Gath–Geva clustering with weighted ordinary least-squares estimation of the consequent parameters.*
3. *EM-TI: The presented method with transformed input variables.*
4. *EM-NI: The presented method with the original input variables.*

As some clustering methods are sensitive to differences in the numerical ranges of the different features, the data can be normalized to zero mean and unit variance:

$$\widetilde{z}_{j,k} = \frac{z_{j,k} - \bar{z}_j}{\sigma_j} \tag{3.34}$$

where $\bar{z}_j$ and σ_j are the mean and the variance of the given variable, respectively. The goal is to predict the fuel consumption of an automobile on the basis of several given characteristics, such as the weight, model year, etc. The data set was obtained from the UCI Repository of Machine Learning Databases and Domain Theories[1]*. After removing samples with missing values, the data set was reduced to 392 entries. This data set was divided into a training set and a test set, each containing 196 samples. The performance of the models is measured by the root mean squared prediction error (RMSE): RMSE $= \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2}$. The approximation power of the identified models is then compared with fuzzy models with the same number of rules obtained by the Fuzzy Toolbox for MATLAB® (the ANFIS model [45]) and the Fuzzy Model Identification (FMID) Toolbox based on Gustafson–Kessel clustering [35]. The inputs to the TS fuzzy model are: x_1: Displacement, x_2: Horsepower, x_3: Weight, x_4: Acceleration and x_5: Model year. Originally, there were six features available. The first one, the number of cylinders, is neglected here because the clustering algorithms run into numerical problems on features with only a small number of discrete values. Fuzzy models with two, three and four rules were identified with the presented method. With the two rule-model, the presented clustering method achieved RMSE values of 2.72 and 2.85 for the training and test data, respectively, which is nearly the same performance as with the three and four-rule models. The FMID Toolbox gives very similar results: RMSE values of 2.67 and 2.95 for the training and test data. Considerably worse results where obtained with the ANFIS algorithm, which gave an overtrained model with the RMSE of 1.97 on the training data but 91.35 on the test data. These results indicate that the presented clustering method has very good generalization properties. For a further comparison I also give the results of a linear regression model given in [45]. This linear model has seven parameters and six input variables (the previously given five variables and the number of cylinders). The training and test RMSE of this model are 3.45 and 3.44, respectively.*

[1] FTP address: ftp://ics.uci.edu/pub/machine-learning-databases/auto-mpg

Fuzzy models with only two input variables were also identified, where the selected features were taken from [45], where the following model structure was proposed:

$$\mathrm{MPG} = f\left(\mathrm{Weight}, \mathrm{Year}\right) . \tag{3.35}$$

As the Gath–Geva and EM-TI models capture correlations among the input variables, the TS fuzzy model extracted from the clusters should use multivariable antecedent membership functions:

$$R_i \quad : \quad \textbf{If } \mathbf{x} \textit{ is } \mathbf{A}_i \textbf{ then } \hat{y} = \mathbf{a}_i^T \mathbf{x} + b_i$$

or transformed input variables:

$$R_i \quad : \quad \textbf{If } \mathbf{t}_{i,1}^T \mathbf{x} \textit{ is } A_{i,1} \textbf{ and } \mathbf{t}_{i,2}^T \mathbf{x} \textit{ is } A_{i,2} \textbf{ then } \hat{y} = \mathbf{a}_i^T \mathbf{x} + b_i$$

where $i = 1, \ldots, c$, $\hat{y}$ *is the estimated MPG and* $\mathbf{x}^T = [\mathrm{Weight},\ \mathrm{Year}]$.

These models cannot be easily analyzed, interpreted and validated by human experts, because the fuzzy sets (linguistic terms) are defined in a multidimensional or linearly transformed space. However, the presented EM-NI method (Modified Gath–Geva clustering) results in the standard rules with the original antecedent variables in the conjunctive form:

$$R_i : \textbf{If } \mathrm{Weight} \textit{ is } A_{i,1} \textbf{ and } \mathrm{Year} \textit{ is } A_{i,2} \textbf{ then } \hat{y} = a_{i,1}\mathrm{Weight} + a_{i,2}\mathrm{Year} + b_i \quad [w_i] . \tag{3.36}$$

Table 3.1 compares the prediction performance of the obtained models:

Table 3.1: Comparison of the performance of the identified TS models with two input variables.

Method	2 rules (train)	2 rules (test)	4 rules (train)	4 rules (test)
GG-TLS	3.34	3.43	5.58	5.71
GG-TLS-N	3.25	3.57	3.43	4.00
GG-LS	2.97	2.97	2.77	2.95
EM-TI	2.97	2.95	2.62	2.93
EM-NI	2.97	2.95	2.90	2.95
ANFIS	2.67	2.95	2.37	3.05
FMID	2.96	2.98	2.84	2.94

Among the four presented approaches, only the total-least-squares identification is sensitive to the normalization of the data. Hence, in Table 3.1 GG-TLS-N denotes the results obtained by making the identification with the use of normalized data.

Normally, the model performance on the training data improves with the increasing number of clusters, while the performance on the evaluation data improves until the effect of over-fitting appears and then it starts degrading (bias-variance tradeoff). However, when the total-least squares (TLS) method is applied, the training error became larger with the increase of the model complexity. This is because the input variables of the model are strongly correlated and the smallest eigenvector does not define a hyperplane related to the regression problem, but it reflects the dependency of the input variables. Already for two clusters, the difference between the two small eigenvalues is very small (the eigenvalues are $[10.92,\ 2.08,\ 3.4 \cdot 10^5]$ *for the first cluster and* $[1.37 \cdot 10^5,\ 37.69,\ 4.93]$ *for the second one).*

The presented fuzzy clustering method showed a slightly better performance than the Gath–Geva algorithm. As these methods identify fuzzy models with transformed input variables, they have good performance because of the effective axes-oblique partition of the input domain, which can be seen in Figure 3.1.

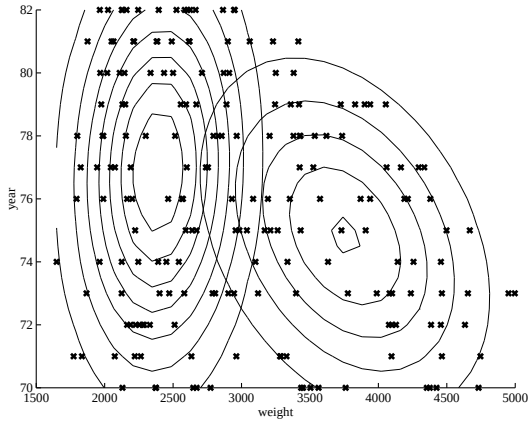


Figure 3.1: Clusters detected by GG clustering algorithm.

The EM-NI algorithm yields clusters that are not rotated in the input space (see Figure 3.2). These clusters can be projected and decomposed to easily interpretable membership functions defined on the individual features as shown in Figure 3.3 for the two-rule model and in Figure 3.4 for the four-rule model. This constraint, however, reduces the flexibility of the model, which can result in worse prediction performance. I use EMR-TI to demonstrate how much performance one has to sacrifice for the interpretability. For this example, the difference in performances turns out to be negligible (see Table 3.1).

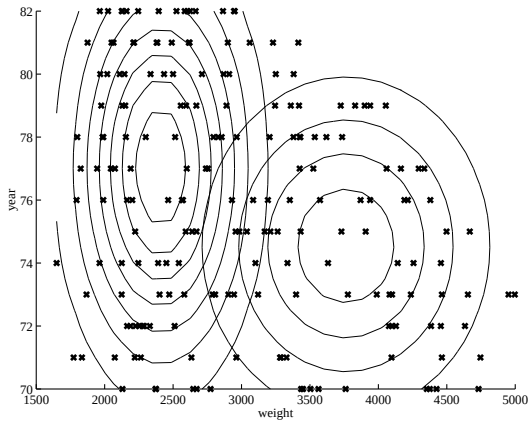


Figure 3.2: Clusters detected by the modified algorithm.

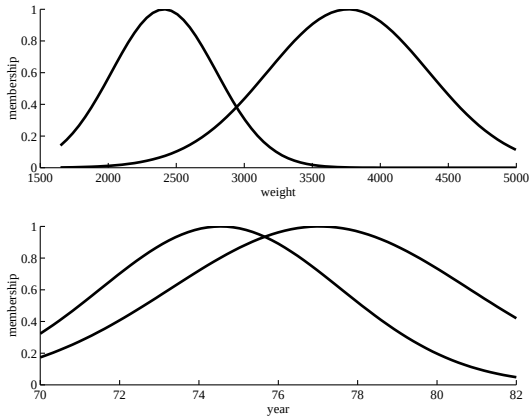


Figure 3.3: Membership functions obtained.

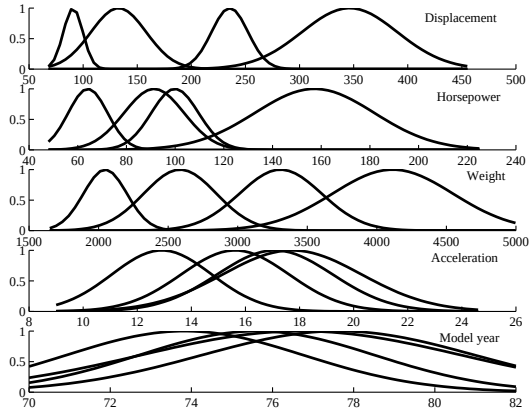


Figure 3.4: Membership functions of the TS model for MPG prediction based on five inputs.

The Fuzzy Toolbox of MATLAB® (ANFIS, neuro-fuzzy model) [46] and the Fuzzy Model Identification (FMID) Toolbox [35] were also used to identify fuzzy models for the MPG prediction problem. As can be seen from Table 3.1, the presented method obtains fuzzy models that have good performance compared to these alternative techniques.

The resulted model is also good at extrapolation. The prediction surface of the model with two inputs is shown in Figure 3.5. If this surface is compared to the prediction surface of the ANFIS generated model (see [45]), one can see that the ANFIS model spuriously estimates higher MPG for heavy cars because of lack of data due to the tendency of manufacturers to begin building small compact cars during the mid 70s. As can be seen in Figure 3.5, the obtained EM-NI model does not suffer from this problem.

□

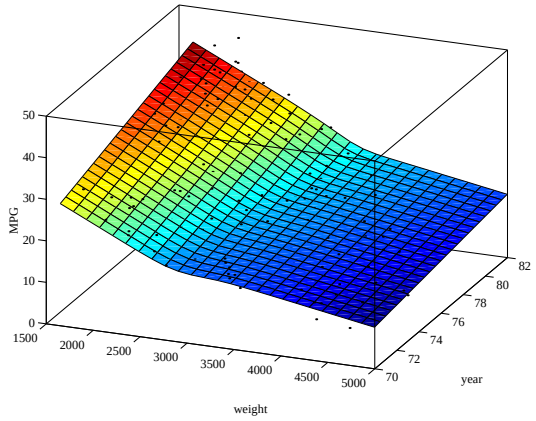


Figure 3.5: Prediction surface of the model.

Selection of the Antecedent and Consequent Variables

Using too many antecedent and consequent variables results in difficulties in the prediction and interpretability capabilities of the fuzzy model due to redundancy, non-informative features and noise. To avoid these problems in this section two methods are presented.

As the fuzzy model is linear in the parameters θ_i, the parameters can be identified by the least squares method (see (3.12)) that can be also formulated as:

$$\theta_i = \mathbf{B}^+ \mathbf{y} \sqrt{\boldsymbol{\beta}_i} \tag{3.37}$$

where $\mathbf{B}^+$ denotes the Moore-Penrose pseudo inverse of $\boldsymbol{\Phi}_e \sqrt{\boldsymbol{\beta}_i}$.

The OLS method transforms the columns of $\mathbf{B}$ into a set of orthogonal basis vectors in order to inspect the individual contribution of each rule. To do this Gram-Schmidt orthogonalization of $\mathbf{B} = \mathbf{WA}$ is used, where $\mathbf{W}$ is an orhogonal matrix $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ and $\mathbf{A}$ is an upper triangular matrix with unity diagonal elements. If $\mathbf{w}_i$ denotes the i–th column of $\mathbf{W}$ and g_i is the corresponding element of the OLS solution vector $\mathbf{g} = \mathbf{A}\theta_i$, the output variance $(\mathbf{y}\sqrt{\boldsymbol{\beta}_i})^T(\mathbf{y}\sqrt{\boldsymbol{\beta}_i})/N$ can be explained by the regressors $\sum_{i=1}^{n_r} g_i \mathbf{w}_i^T \mathbf{w}_i / N$. Thus, the error reduction ratio, ϱ, due to an individual rule i can be expressed as

$$\varrho^i = \frac{g_i^2 \mathbf{w}_i^T \mathbf{w}_i}{(\mathbf{y}\sqrt{\boldsymbol{\beta}_i})^T(\mathbf{y}\sqrt{\boldsymbol{\beta}_i})} \tag{3.38}$$

This ratio offers a simple mean for ordering the consequent variables, and can be easily used to select a subset of the inputs in a forward-regression manner.

Feature selection is usually necessary. For this purpose, I modify the Fischer interclass separability method which is based on statistical properties of the data and has been applied for feature selection of labeled data [47]. The interclass separability criterion is based on the $\mathbf{F}_B$ between-class and the $\mathbf{F}_W$ within-class covariance matrices that sum up to the total covariance of the training data $\mathbf{F}_T$, where:

$$\mathbf{F}_W = \sum_{i=1}^{c} p(\eta_i)\mathbf{F}_i, \mathbf{F}_B = \sum_{i=1}^{c} p(\eta_i) \left(\mathbf{v}_i - \mathbf{v}_0\right)^T \left(\mathbf{v}_i - \mathbf{v}_0\right)$$

$$\mathbf{v}_0 = \sum_{i=1}^{c} p(\eta_i)\mathbf{v}_i \tag{3.39}$$

The feature interclass seperatibility selection criterion is a trade-off between $\mathbf{F}_W$ and $\mathbf{F}_B$:

$$J = \frac{\det \mathbf{F}_B}{\det \mathbf{F}_W} \tag{3.40}$$

The importance of a feature is measured by leaving out the feature and calculating J for the reduced covariance matrices. The feature selection is made iteratively by leaving out the least needed feature.

Example 3.2. ***Orthogonal Least Squares and Interclass Separability***

The presented fuzzy modeling approach is applied the previously studied Automobile MPG (miles per gallon) prediction case study [45] (see Example **??***). The above presented model reduction techniques, OLS and FIS, are now applied to remove redundancy and simplify the model. First, based on the obtained fuzzy clusters, the OLS method is used for ordering of the input variables. The fuzzy model with two rules is applied. It turned out that for both clusters (local models) the "model year" and the "weight" become the most relevant input variables. Without re-estimation (re-clustering) of the model and additional removal of the other variables from the consequent part of the model, the modeling performance drops only to 3.75 and 3.55 RMSE for the training and the test data, respectively. It is worth noting, that the same features turned out to be the most important in [45], where a cross-validation based method has been used for input selection of neuro-fuzzy models.*

Based on the above considerations, the clustering has been applied again to identify a model based on the two selected attributes "weight" and "model year". This results in RMSE values of 2.97 and 2.95 for training and test data, respectively. In addition, the Fisher Interclass separability method is applied and the second attribute "model year" could be removed from the antecedent of both rules. The final result is a model with RMSE values of 2.97 and 2.98 for training and test data, respectively. The other methods are now applied with the same attributes and their performances are summarized in Table 3.2.

The clusters are much more separated on the input "weight" (see Figure 3.3). Hence, the application of Fisher Inteclass separability shows that it is enough to use only this input on antecedent part of the model:

$$\begin{gathered} R_1: \textbf{ If } Weight \textit{ is } A_{i,1} \textbf{ then } \textit{MPG} = \\ [-0.0107 \;\; 1.0036][Weight,\, Year]^T - 23.1652,\; [0.57] \\ R_2: \textbf{ If } Weight \textit{ is } A_{i,2} \textbf{ then } \textit{MPG} = \\ [-0.0038 \;\; 0.4361][Weight,\, Year]^T - 1.4383,\; [0.43] \end{gathered} \tag{3.41}$$

Concluding, the obtained model has good approximation capabilities, similar to some other advanced methods, but is also very compact. Furthermore, the methods seems to have good generalization properties because results on learning data are similar to those on training data. If the prediction surface surface of the obtained model is compared to the prediction surface of the ANFIS model (see [45]), one can see that the ANFIS approach spuriously estimates higher MPG for heavy cars because of luck of data due to the tendency of manufacturers to begin building small compact cars during the mid 70's, while the presented approach does not suffer from this problem.

Table 3.2: Performance of the identified TS models with two rules and only two input variables. HYBRID: Clustering + OLS + FIS, GG: Gath–Geva clustering, ANFIS: neuro-fuzzy model, FMID: Fuzzy Model Identification Toolbox

Method	train RMSE	test RMSE
HYBRID	2.97	2.95
GG	2.97	2.97
ANFIS	2.67	2.95
FMID	2.94	3.01

□

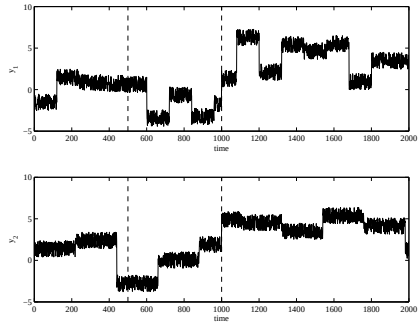
3.2 Fuzzy Clustering for Time-series Segmentation

In this section a fuzzy clustering based algorithm is presented which is useful for the fuzzy segmentation of multivariate temporal databases (these results were published in [48]). Time-series segmentation addresses the following data mining problem: given a time-series, T, find a partitioning of T into c segments that are internally homogeneous [49]. Depending on the application, the goal of the segmentation is to locate stable periods of time, to identify change points, or to simply compress the original time-series into a more compact representation [50]. Although in many real-life applications a lot of variables must be simultaneously tracked and monitored, most of the segmentation algorithms are used for the analysis of only one time-variant variable [51]. Hoverer, in some cases it is necessary to synchronously segment the time-series of the variables.

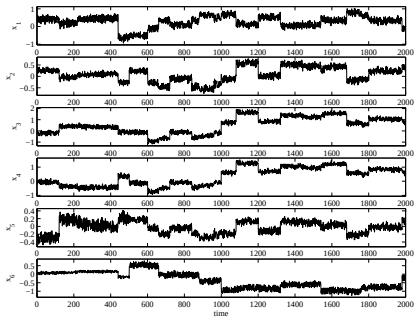
The segmentation of multivariate time-series is especially important in the data-based analysis and monitoring of modern production systems, where huge amount of historical process data are recorded with distributed control systems (DCS). These data definitely have the potential to provide information for product and process design, monitoring and control [52]. This is especially important in many practical applications where first-principles modeling of complex "data rich and knowledge poor" systems are not possible [53]. Therefore, KDD methods have been successfully applied to the analysis of process systems, and the results have been used in process design, process improvement, operator training, and so on [25]. Hence, the data mining algorithm presented in this session has been developed to the analysis of the historical process data of a medium and high-density polyethylene (MDPE, HDPE) plant. The operators of this polymerization process should simultaneously track many process variables. Of course, due to the hidden nature of the system the measured variables are correlated. Hence, it is useful to monitor only some principal components that is widely applied in advanced process monitoring. The main problem of this approach is the fact that in some cases the hidden process, which can be observed as the correlation among the variables, varies in time. In our example this phenomenon can occur when a different product is formed, and/or different catalyst is applied, or there are significant process faults, etc. The segmentation of only one measured variable is not able to detect such changes. Hence, the segmentation algorithm should be based on multivariate statistical tools.

To demonstrate this problem let us consider the synthetic dataset shown in Figure 3.6. The observed variables that can be seen in Figure 3.6(b) are not independent, they were generated by the latent variables shown in Figure 3.6(a). The correlation among the observed variables changes at the quarter of the time period, and the mean of the latent variables changes at the half of the time period. These changes are marked by vertical lines in Figure 3.6(a). As it can be seen in Figure 3.6(b), such information can be detected neither by application of univariate segmentation algorithms, nor by the visual inspection of the observed variables.

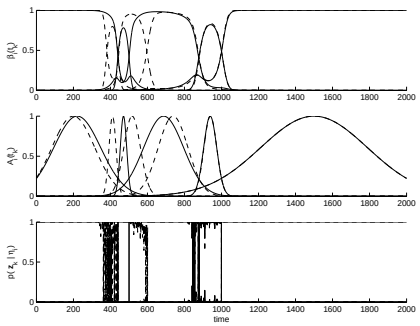
Hence, the aim of this session is to develop an algorithm that is able to handle time-varying characteristics of multivariate data: (i) changes in the mean;



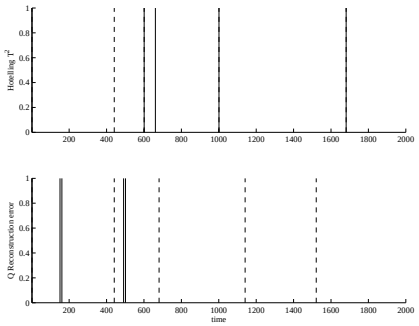
(a) Latent variables



(b) Observed variables



(c) Results obtained by fuzzy clustering, (–): $q = 2$,(- -): $q = 5$



(d) Results obtained by the bottom-up algorithm, (–): $q = 2$,(- -): $q = 5$

Figure 3.6: The synthetic dataset and its segmentation by different algorithms based on two and five principal components.

(ii) changes in the variance; and (iii) changes in the correlation structure among the variables.

To discover that type of changes of the hidden relationships of multivariate time-series, multivariate statistical tools should be applied by the segmentation algorithm. Among the wide range of possible tools, e.g. random projection, independent component analysis, the presented algorithm utilizes Principal Component Analysis (PCA). Linear PCA can give good prediction results for simple time-series, but can fail in the analysis of historical data having changes in regime or having nonlinear relations among the variables. The analysis of such data requires the detection of locally correlated clusters [54]. These algorithms do the clustering of the data to discover the local relationship among the variables similarly to mixture of Principal Component Models [55].

Time-series segmentation may be considered as clustering with a time- ordered structure. The contribution of this session is the introduction of a new fuzzy clustering algorithm which can be effectively used to segment large, multivariate time-series. Since the points in a cluster must come from successive time points, the time-coordinate of the data has to be also considered during the clustering. One possibility to deal with time is to define a new cluster prototype that uses time as an additional variable. Hence, the clustering is based on a

distance measure which consists of two terms: the first distance term is based on how the data are in the given segment defined by the Gaussian fuzzy sets defined in the time domain, while the second term measures how far the data are from the hyperplane of the PCA model of the segments.

The fuzzy segmentation of time-series is an adequate idea. The changes of the variables of the time-series are usually vague and are not focused on any particular time point. Therefore, it is not practical to define crisp bounds of the segments. For example, if humans visually analyze historical process data, they use expressions like "this point belongs to this operating point less and belongs to the other more". A good example of this kind of fuzzy segmentation is how fuzzily the start and the end of *early morning* is defined. Fuzzy logic is widely used in various applications where the grouping of overlapping and vague objects is necessary [56], and there are many fruitful examples in the literature for the combination of fuzzy logic with time-series analysis tools [57, 58, 50, 59].

The key problem of the application of fuzzy clustering for time-series segmentation is the selection of the number of segments for the clustering process. Obviously, this is a standard problem also in the classical c-means clustering. In the context of time series, however, it appears to be even more severe. For this purpose a bottom-up algorithm has been worked out where the clusters are merged during a recursive process. The cluster merging is coordinated by a fuzzy decision making algorithm which utilizes a compatibility criterion of the clusters [60], where this criterion is calculated by the similarity of the Principal Component Models of the clusters [61].

Time-series Segmentation Problem Formulation

A time-series $T = \{\mathbf{x}_k | 1 \leq k \leq N\}$ is a finite set of N samples labeled by time points $t_1, \ldots, t_N$, where $\mathbf{x}_k = [x_{1,k}, x_{2,k}, \ldots, x_{n,k}]^T$. A segment of T is a set of consecutive time points $S(a, b) = \{a \leq k \leq b\}$, $\mathbf{x}_a, \mathbf{x}_{a+1}, \ldots, \mathbf{x}_b$. The c-segmentation of time-series T is a partition of T to c non-overlapping segments $S_T^c = \{S_i(a_i, b_i) | 1 \leq i \leq c\}$, such that $a_1 = 1, b_c = N$, and $a_i = b_{i-1} + 1$. In other words, a c-segmentation splits T to c disjoint time intervals by segment boundaries $s_1 < s_2 < \ldots < s_c$, where $S_i(s_{i-1} + 1, s_i)$.

Usually the goal is to find homogeneous segments from a given time-series. In such cases the segmentation problem can be defined as constrained clustering: data points should be grouped based on their similarity, but with the constraint that all points in a cluster must come from successive time points. (See [62] for the relationship of time series and clustering from another point of view.) In order to formalize this goal, a $cost(S(a, b))$ cost function with the internal homogeneity of individual segments should be defined. The cost function can be any arbitrary function. For example in [63, 49] the sum of variances of the variables in the segment was defined as $cost(S(a, b))$. Usually, the $cost(S(a, b))$ cost function is defined based on the distances between the actual values of the time-series and the values given by a simple function (constant or linear function, or a polynomial of a higher but limited degree) fitted to the data of each segment. Hence, the optimal c-segmentation simultaneously determines the

a_i, b_i borders of the segments and the θ_i parameter vectors of the models of the segments by minimizing the cost of c-segmentation which is usually the sum of the costs of the individual segments:

$$cost(S_T^c) = \sum_{i=1}^{c} cost(S_i)\,. \tag{3.42}$$

This cost function can be minimized by dynamic programming, which is computationally intractable for many real datasets [63]. Consequently, heuristic optimization techniques such as greedy top-down or bottom-up techniques are frequently used to find good but suboptimal c-segmentations [64, 65]. In data mining, the bottom-up algorithm has been used extensively to support a variety of time-series data mining tasks [64]. This algorithm is quite powerful since the the merging cost evaluations requires simple identifications of Principal Component Analysis (PCA) models which is easy to implement and computationally cheap to calculate. Because of this simplicities and because PCA defines linear hyperplane, the presented approach can be considered as the multivariate extension of the piecewise linear approximation (PLA) based time-series segmentation and analysis tools developed by Keogh [64, 66].

Although PCA is well known tool, it is advantageous to overview this method just because of the notation as well. PCA is based on the projection of correlated high dimensional data onto a hyperplane. This mapping uses only the first few q nonzero eigenvalues and the corresponding eigenvectors of the $\mathbf{F}_i = \mathbf{U}_i \Lambda_i \mathbf{U}_i^T$, covariance matrix, decomposed to the Λ_i matrix that includes the eigenvalues $\lambda_{i,j}$ of $\mathbf{F}_i$ in its diagonal in decreasing order, and to the $\mathbf{U}_i$ matrix that includes the eigenvectors corresponding to the eigenvalues in its columns. The vector $\mathbf{y}_{i,k} = \mathbf{W}_i^{-1}(\mathbf{x}_k) = \mathbf{W}_i^T(\mathbf{x}_k)$ is a q-dimensional reduced representation of the observed vector $\mathbf{x}_k$, where the $\mathbf{W}_i$ weight matrix contains the q principal orthonormal axes in its column $\mathbf{W}_i = \mathbf{U}_{i,q}\Lambda_{i,q}^{\frac{1}{2}}$.

Based on PCA the $cost(S_i)$ can be calculated in two ways. This cost can be equal to the reconstruction error of this segment

$$cost(S_i) = \frac{1}{b_i - a_i + 1} \sum_{k=a_i}^{b_i} Q_{i,k} \tag{3.43}$$

where $Q_{i,k} = (\mathbf{x}_k - \hat{\mathbf{x}}_k)^T(\mathbf{x}_k - \hat{\mathbf{x}}_k) = \mathbf{x}_k^T(\mathbf{I} - \mathbf{U}_{i,p}\mathbf{U}_{i,p}^T)\mathbf{x}_k$. When the hyperplane of the PCA model has adequate number of dimensions, the distance of the data from the hyperplane is resulted by measurement failures, disturbances and negligible information, so the projection of the data into this p-dimensional hyperplane does not cause significant reconstruction error.

Although the relationship among the variables can be effectively described by a linear model, in some cases it is possible that the data is distributed around some separated centers in this linear subspace. The Hotelling T^2 measure is often used to calculate the distance of the data point from the center in this linear subspace. This can be also used to compute $cost(S_i)$

$$cost(S_i) = \frac{1}{b_i - a_i + 1} \sum_{k=a_i}^{b_i} T_{i,k}^2 = \frac{1}{b_i - a_i + 1} \sum_{k=a_i}^{b_i} \mathbf{y}_{i,k}^T \mathbf{y}_{i,k}. \tag{3.44}$$

When the variance of the segments are minimized during the segmentation, equation (3.42) results in the following equation:

$$cost(S_T^c) = \sum_{i=1}^{c} \sum_{k=s_{i-1}+1}^{s_i} \| \mathbf{x}_k - \mathbf{v}_i^x \|^2 = \sum_{i=1}^{c} \sum_{k=1}^{N} \beta_i(t_k) d^2(\mathbf{x}_k, \mathbf{v}_i^x) \,. \tag{3.45}$$

where $d^2(\mathbf{x}_k, \mathbf{v}_i^x)$ represents the distance between the $\mathbf{v}_i^x$ mean of the variables in the i-th segment (center of the i-th cluster) and the $\mathbf{x}_k$ data point; and $\beta_i(t_k) \in \{0, 1\}$ stands for the crisp membership of the k-th data point in the i-th segment, and:

$$\beta_i(t_k) = \begin{cases} 1 & \text{if} s_{i-1} < k \leq s_i \\ 0, & \text{otherwise.} \end{cases} \tag{3.46}$$

This equation is well comparable to the typical error measure of standard k-means clustering but in this case the clusters are limited to contiguous segments of the time-series instead of the Voronoi regions in R^n.

The changes of the variables of the time-series are usually vague and are not focused on any particular time point. As it is not practical to define crisp bounds of the segments, in this session Gaussian membership functions, $A_i(t_k)$, are used to represent the $\beta_i(t_k) \in [0, 1]$ fuzzy segments of a time-series:

$$A_i(t_k) = \exp\left(-\frac{1}{2} \frac{(t_k - v_i^t)^2}{\sigma_{i,t}^2} \right), \; \beta_i(t_k) = \frac{A_i(t_k)}{\sum_{j=1}^{c} A_j(t_k)} \tag{3.47}$$

(These terms are analogous to the Gaussian membership function and the degree of activation of the ith rule in classical fuzzy classifier as can be seen in Section B.3.) For the identification of the v_i^t centers and $\sigma_{i,t}^2$ variances of the membership functions, a fuzzy clustering algorithm is introduced. The algorithm, which is similar to the modified Gath–Geva clustering [67], assumes that the data can be effectively modelled as a mixture of multivariate (including time as a variable) Gaussian distribution, so it minimizes the sum of the weighted squared distances between the $\mathbf{z}_k = [t_k, \mathbf{x}_k^T]^T$ data points and the η_i cluster prototypes

$$J = \sum_{i=1}^{c} \sum_{k=1}^{N} (\mu_{i,k})^m \, d^2(\mathbf{z}_k, \eta_i) \tag{3.48}$$

where $\mu_{i,k}$ represents the degree of membership of the observation $\mathbf{z}_k = [t_k, \mathbf{x}_k^T]^T$ is in the i-th cluster $(i = 1, \ldots, c)$ and $m \in [1, \infty)$ is a weighting exponent that determines the fuzziness of the resulting clusters (usually chosen as $m = 2$).

The Gath–Geva clustering algorithm can be interpreted in a probabilistic framework, since the $d^2(\mathbf{z}_k, \eta_i)$ distance is inversely proportional to the probability that the $\mathbf{z}_k$ data point belongs to the i-th cluster, $p(\mathbf{z}_k|\eta_i)$. The data are assumed to be normally distributed random variables with expected value $\mathbf{v}_i$ and covariance matrix $\mathbf{F}_i$. The Gath–Geva clustering algorithm is equivalent to the identification of a mixture of Gaussians that represents the $p(\mathbf{z}_k|\eta)$ probability

density function expanded in a sum over the c clusters

$$p(\mathbf{z}_k|\eta) = \sum_{i=1}^{c} p(\mathbf{z}_k|\eta_i)p(\eta_i) \tag{3.49}$$

where the $p(\mathbf{z}_k|\eta_i)$ distribution generated by the i-th cluster is represented by the Gaussian function

$$p(\mathbf{z}_k|\eta_i) = \frac{1}{(2\pi)^{\frac{n+1}{2}}\sqrt{\det(\mathbf{F}_i)}} \exp\left(-\frac{1}{2}(\mathbf{z}_k - \mathbf{v}_i)^T \mathbf{F}_i^{-1}(\mathbf{z}_k - \mathbf{v}_i)\right) \tag{3.50}$$

and $p(\eta_i)$ is the unconditional cluster probability (normalized such that $\sum_{i=1}^{c} p(\eta_i) = 1$ holds), where η_i represents the parameters of the i-th cluster, $\eta_i = \{p(\eta_i), \mathbf{v}_i, \mathbf{F}_i | i = 1, \ldots, c\}$.

Since the time variable is independent from the $\mathbf{x}_k$ variables, the presented clustering algorithm is based on the following $d^2(\mathbf{z}_k, \eta_i)$ distance measure

$$p(\mathbf{z}_k|\eta_i) = \frac{1}{d^2(\mathbf{z}_k, \eta_i)} = \underbrace{\alpha_i}_{p(\eta_i)} \underbrace{\frac{1}{\sqrt{2\pi\sigma_{i,t}^2}} \exp\left(-\frac{1}{2}\frac{(t_k - v_i^t)^2}{\sigma_{i,t}^2}\right)}_{p(t_k|\eta_i)} \times$$
$$\underbrace{\frac{1}{(2\pi)^{\frac{r}{2}}\sqrt{\det(\mathbf{A}_i)}} \exp\left(-\frac{1}{2}(\mathbf{x}_k - \mathbf{v}_i^x)^T \mathbf{A}_i^{-1}(\mathbf{x}_k - \mathbf{v}_i^x)\right)}_{p(\mathbf{x}_k|\eta_i)} \tag{3.51}$$

which consists of three terms. The first α_i term represents the *a priori* probability of the cluster, while the second represents the distance between the k-th data point and the v_i^t center of the i-th segment in time. The third term represents the distance between the cluster prototype and the data in the feature space where $\mathbf{v}_i^x$ means the coordinate of the i-th cluster center in the feature space and r is the rank of $\mathbf{A}_i$ distance norm corresponding to the i-th cluster.

The presented cluster prototype formulated by (3.51) is similar to that used by the Gath–Geva clustering algorithm. However, it utilizes a different distance norm, $\mathbf{A}_i$. In the following section, it will be demonstrated how this norm can be based on the principal component analysis of the cluster.

PCA based Distance Measure

The $\mathbf{A}_i$ distance norm can be defined in many ways. It is wise to select this norm to scale the variables so that those with greater variability do not dominate the clustering. One can scale by dividing by standard deviations, but a better procedure is to use statistical (Mahalanobis) distance, which also adjusts for the correlations among the variables. In this case $\mathbf{A}_i$ is the fuzzy covariance matrix $\mathbf{A}_i = \mathbf{F}_i$, where

$$\mathbf{F}_i = \frac{\sum_{k=1}^{N} (\mu_{i,k})^m (\mathbf{x}_k - \mathbf{v}_i^x)(\mathbf{x}_k - \mathbf{v}_i^x)^T}{\sum_{k=1}^{N} (\mu_{i,k})^m}. \tag{3.52}$$

When the variables are highly correlated, the $\mathbf{F}_i$ covariance matrix can be ill-conditioned and cannot be inverted. Recently two methods have been worked out to handle this problem [68]. The first method is based on fixing the ratio between the maximal and minimal eigenvalues of the covariance matrix. The second method is based on adding a scaled unity matrix to the calculated covariance matrix. Both methods result in invertible matrices, but neither of them extracts the potential information about the hidden structure of the data.

One limiting disadvantage of PCA is the absence of an associated probability density or generative model which is required to compute $p(\mathbf{x}_k|\eta_i)$. Tipping and Bishop [55] developed a method called Probabilistic Principal Component Analysis (PPCA). In the PPCA the log-likelihood of the observing the data under this model is

$$\mathcal{L} = \sum_{k=1}^{N} \ln(p(\mathbf{x}_k|\eta_i)) = -\frac{N}{2}\left\{n\ln(2\pi) + \ln\left(\det(\mathbf{A}_i)\right) + \mathrm{trace}(\mathbf{A}_i^{-1}\mathbf{F}_i)\right\} \tag{3.53}$$

where $\mathbf{A}_i = \sigma_{i,x}^2\mathbf{I} + \mathbf{W}_i\mathbf{W}_i^T$ is the modified covariance matrix of the i-th cluster which can be used to compute the $p(\mathbf{x}_k|\eta_i)$ probability. The log-likelihood is maximized when the columns of $\mathbf{W}_i$ span the principal subspace of the data. Tipping and Bishop proofed that the only nonzero stationary points of the derivative of (3.53) with respect to $\mathbf{W}_i$ occur for

$$\mathbf{W}_i = \mathbf{U}_{i,q}\left(\Lambda_{i,q} - \sigma_{i,x}^2\mathbf{I}\right)^{1/2}\mathbf{R}_i \tag{3.54}$$

where $\mathbf{R}_i$ is an arbitrary $q \times q$ orthogonal rotation matrix and $\sigma_{i,x}^2$ is given by

$$\sigma_{i,x}^2 = \frac{1}{n-q}\sum_{j=q+1}^{n} \lambda_{i,j}. \tag{3.55}$$

The algorithmic description of the Expectation Maximization (EM) approach to PPCA model is given in [55] but it can also be found in the following section, where the estimation of this model is incorporated into the clustering procedure.

Modified GG-Clustering for Time-series Segmentation

One of the most important advantages of PPCA models is that it allows their combination into mixture of models. Mixtures have been extensively used as models where data can be viewed as arising from several populations mixed in varying proportions, and Expectation Maximization (EM) is widely used to estimate the parameters of the components in a mixture [69]. The clusters obtained by Gath–Geva (GG) clustering, also referred to Fuzzy Maximum Likelihood clustering, are multivariate Gaussian functions. The Alternating Optimization (AO) of these clusters is identical to the Expectation Maximization (EM) (maximum likelihood estimation) identification of the mixture of these Gaussian models when the fuzzy weighting exponent $m = 2$ [70].

Similarly to GG clustering, in the presented algorithm the optimal parameters of the $\eta_i = \{\mathbf{v}_i^x, \mathbf{A}_i, v_i^t, \sigma_{i,x}^2, \alpha_i\}$ cluster prototypes are determined by the minimization of the (3.48) functional subjected to the classical clustering constraints (B.1), (B.2) and (B.3). The Alternating Optimization results in the easily implementable algorithm described in Algorithm 3.2.1.

The usefulness and accuracy of the algorithm depends on the right choice of the q number of principal components (PCs) and the c number of the segments. Hence, the crucial question of the usefulness of the presented cluster algorithm is how these parameters can be determined in an automatic manner. This will be presented in the following two subsections.

Algorithm 3.2.1 (Clustering for Time-Series Segmentation)**.**

Initialization *Given a time-series T specify c and q, choose a termination tolerance $\epsilon > 0$, and initialize the values of $\mathbf{W}_i, \mathbf{v}_i^x, \sigma_{i,x}^2, \mu_{i,k}$.*

Repeat *for $l = 1, 2, \ldots$*

Step 1 *Calculate the η_i parameters of the clusters*

- *a priori probability of the cluster*

$$\alpha_i = \frac{1}{N} \sum_{k=1}^{N} \mu_{i,k}. \tag{3.56}$$

- *the cluster centers*

$$\mathbf{v}_i^x = \frac{\sum_{k=1}^{N} (\mu_{i,k})^m \left(\mathbf{x}_k - \mathbf{W}_i \langle \mathbf{y}_{i,k} \rangle\right)}{\sum_{k=1}^{N} (\mu_{i,k})^m} \tag{3.57}$$

where the expectation of the latent variables is $\langle \mathbf{y}_{i,k} \rangle = \mathbf{M}_i^{-1} \mathbf{W}_i^T \left(\mathbf{x}_k - \mathbf{v}_i^x\right)$ and the $q \times q$ matrix $\mathbf{M}_i = \sigma_{i,x}^2 \mathbf{I} + \mathbf{W}_i^T \mathbf{W}_i$.

- *the new values of $\mathbf{W}_i$*

$$\widetilde{\mathbf{W}}_i = \mathbf{F}_i \mathbf{W}_i \left(\sigma_{i,x}^2 \mathbf{I} + \mathbf{M}_i^{-1} \mathbf{W}_i^T \mathbf{F}_i \mathbf{W}_i\right)^{-1} \tag{3.58}$$

where $\mathbf{F}_i$ computed by (3.52)*.*

- *the new value of $\sigma_{i,x}^2$*

$$\sigma_{i,x}^2 = \frac{1}{q} trace(\mathbf{F}_i - \mathbf{F}_i \mathbf{W}_i \mathbf{M}_i^{-1} \widetilde{\mathbf{W}}_i^T). \tag{3.59}$$

- *the distance norm ($n \times n$ matrix)*

$$\mathbf{A}_i = \sigma_{i,x}^2 \mathbf{I} + \widetilde{\mathbf{W}}_i \widetilde{\mathbf{W}}_i^T. \tag{3.60}$$

- *the model parameters in time: the center and the standard deviation*

$$v_i^t = \frac{\sum\limits_{k=1}^{N} (\mu_{i,k})^m t_k}{\sum\limits_{k=1}^{N} (\mu_{i,k})^m}, \; \sigma_{i,t}^2 = \frac{\sum\limits_{k=1}^{N} (\mu_{i,k})^m \left(t_k - v_i^t\right)^2}{\sum\limits_{k=1}^{N} (\mu_{i,k})^m}. \tag{3.61}$$

Step 2 *Compute the $d^2(\mathbf{z}_k, \eta_i)$ distance measures by* (3.51)*.*

Step 3 *Update the partition matrix*

$$\mu_{i,k}^{(l)} = \frac{1}{\sum_{j=1}^{c} \left(d(\mathbf{z}_k, \eta_i)/d(\mathbf{z}_k, \eta_j)\right)^{2/(m-1)}}, 1 \leq i \leq c, \; 1 \leq k \leq N. \tag{3.62}$$

until $||\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}|| < \epsilon$.

Automatic Determination of the Number of Segments

In data mining, the bottom-up segmentation algorithm has been extensively used to support a variety of time series data mining tasks [64]. The algorithm starts by creating a fine approximation of the time series, and iteratively merges the lowest cost pair of segments until a stopping criteria is met. For the automatic selection of the number of segments, a similar approach is presented in this section. The presented recursive cluster merging technique evaluates the adjacent clusters for their compatibility (similarity) and merges the clusters that are found to be compatible. Then, after the proper initialization of the parameters of the new cluster the clustering is performed again. During this merging and re-clustering procedure the number of clusters is gradually reduced, until an appropriate number of clusters is found. This procedure is controlled by a fuzzy decision making algorithm based on the similarity between the PCA models.

Similarity of PCA Models

The similarity of two PCA models (i.e. hyperplanes) can be calculated by the PCA similarity factor, S_{PCA}, developed by Krzanowski [61, 71]. Consider two segments, S_i and S_j, of a dataset having the same n variables. Let the PCA models for S_i and S_j consist of q PC's each. The similarity between these subspaces is defined based on the sum of the squares of the cosines of the angles between each principal component of $\mathbf{U}_{i,q}$ and $\mathbf{U}_{j,q}$:

$$S_{PCA}^{i,j} = \frac{1}{q}\sum_{i=1}^{q}\sum_{j=1}^{q}\cos^2\theta_{i,j} = \frac{1}{q}\mathrm{trace}\left(\mathbf{U}_{i,q}^T\mathbf{U}_{j,q}\mathbf{U}_{j,q}^T\mathbf{U}_{i,q}\right) \tag{3.63}$$

Because the $\mathbf{U}_{i,q}$ and $\mathbf{U}_{j,q}$ subspaces contain the q most important principal components that account for the most of the variance in their corresponding datasets, $S_{PCA}^{i,j}$ is also a measure of similarity between the segments S_i and S_j.

Since the purpose of the segmentation is also to detect changes in the mean of the variables, it is not sufficient to compute only the $S_{PCA}^{i,j}$ similarity factor but the distance among the cluster centers also has to be taken into account

$$d(\mathbf{v}_i^x, \mathbf{v}_j^x) = \|\mathbf{v}_i^x - \mathbf{v}_j^x\|. \tag{3.64}$$

Hence, the compatibility criterion has to consider the $c_{i,j}^1 = S_{PCA}^{i,j}$ and $c_{i,j}^2 = d(\mathbf{v}_i^x, \mathbf{v}_j^x)$ factors.

The Decision Making Algorithm

Because the compatibility criterion quantifies various aspects of the similarity of the clusters, the overall cluster compatibility should be obtained through an aggregation procedure. A fuzzy decision making algorithm can be used for this purpose [60].

Compatibility criteria (3.63) and (3.64) are evaluated for each pair of clusters. The resulted compatibility matrix is descriptive concerning the structure of the whole time-series, e.g. repetitive motifs can be also detected from the analysis of the compatibility of the non-adjacent clusters. Following a fuzzy decision making approach, the decision goals for each criterion have to be defined using a fuzzy set. Figure 3.7 shows the triangular membership functions defined for the two criteria. The important parameters of the membership functions are the limits of their support, characterized by the ν^1 knot point for parallelism and ν^2 for closeness. The values of ν^1 and ν^2 are given by averaging compatibilities according to

$$\nu^1 = \frac{1}{c(c-1)}\sum_{i=1}^{c}\sum_{\substack{j=1\\j\neq i}}^{c} c_{i,j}^1, \tag{3.65}$$

$$\nu^2 = \frac{1}{c(c-1)}\sum_{i=1}^{c}\sum_{\substack{j=1\\j\neq i}}^{c} c_{i,j}^2. \tag{3.66}$$

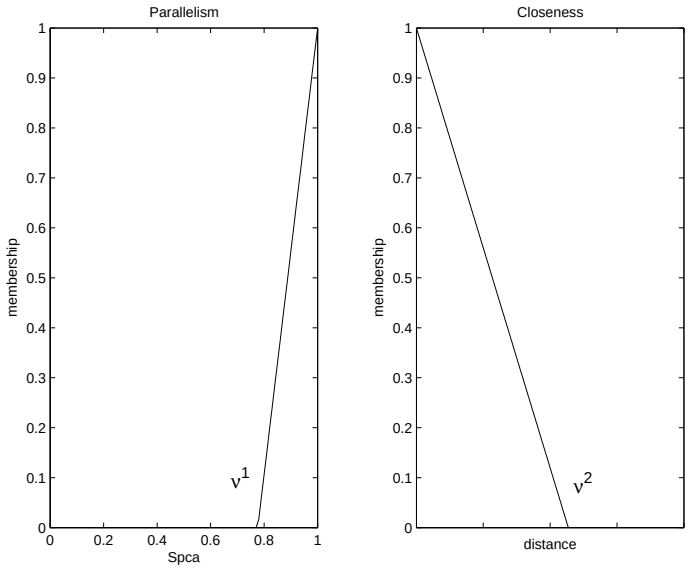


Figure 3.7: Membership functions for parallelism and closeness of clusters

Evaluating the membership functions with the values $c_{i,j}^1$ and $c_{i,j}^2$, one obtains the $\mu_{i,j}^1$ degree of parallelism and $\mu_{i,j}^2$ of closeness. The overall cluster compatibility is determined by the aggregation of the two criteria. A fuzzy aggregation operator is used for this purpose. The outcome of the decision procedure is the $\mathbf{O}$ overall compatibility matrix whose $O_{i,j}$ elements are given by

$$O_{i,j} = \left[\frac{(\mu_{i,j}^1)^2 + (\mu_{i,j}^2)^2}{2}\right]^{1/2}. \tag{3.67}$$

Given the $\mathbf{O}$ compatibility matrix, the clusters that will be merged must be identified and combined. Clusters can be merged in several ways. Our method merges the most similar pair of adjacent clusters as long as the value of the corresponding $O_{i,i+1}$ is above a threshold γ.

Cluster Merging

The applied bottom-up strategy merges two adjacent clusters in each iteration. To preserve the information represented by the clusters there is a need for a merging method that can directly compute the new initial parameters of the cluster from the merged clusters. This can be done by several ways [72]. In this session the method developed by P. M. Kelly is applied [73]. The $\mathbf{v}_{i*}^x$ mean of the resulting cluster is computed from the individual cluster means

$$\mathbf{v}_{i*}^x = \frac{N_i}{N_{i*}}\mathbf{v}_i^x + \frac{N_j}{N_{i*}}\mathbf{v}_{i+1}^x \tag{3.68}$$

where $N_i = \sum_{k=1}^{N} \mu_{i,k}, N_{i+1} = \sum_{k=1}^{N} \mu_{i+1,k}$ and $N_{i*} = N_i + N_{i+1}$, while the $\mathbf{F}_{i*}$ new covariance matrix is calculated from the $\mathbf{F}_i$ and $\mathbf{F}_{i+1}$ old covariance

matrices as

$$\mathbf{F}_{i*} = \frac{N_i - 1}{N_{i*} - 1}\mathbf{F}_i + \frac{N_{i+1} - 1}{N_{i*} - 1}\mathbf{F}_{i+1} + \frac{N_i N_{i+1}}{N_{i*}(N_{i*} - 1)}[(\mathbf{v}_i^x - \mathbf{v}_{i+1}^x)(\mathbf{v}_i^x - \mathbf{v}_{i+1}^x)^T]. \tag{3.69}$$

The new values of $\mathbf{W}_{i*}$ and $\sigma_{i*,x}^2$ can be computed by (3.54) and (3.55) based on the eigenvector-eigenvalue decomposition of $\mathbf{F}_{i*}$. Based on the obtained results the $p(\mathbf{x}_k|\eta_{i*})$ probabilities can be easily computed (see the algorithm in Section 3.2).

This method can also be applied to merge the membership functions characterized by the parameters v_i^t, $\sigma_{i,t}^2$ and v_{i+1}^t, $\sigma_{i+1,t}^2$, and the $p(t_k|\eta_{i*})$ probabilities can be determined from the new values of v_{i*}^t and $\sigma_{i*,t}^2$ by (3.51). The unconditional probability of the $i*$-th cluster is equal to the sum of the probabilities of the previous clusters $p(\eta_{i*}) = p(\eta_i) + p(\eta_{i+1})$. After the previously presented initialization of the new cluster prototype the new $\mu_{i*,k}$ membership values can be computed by (3.51) and (3.62).

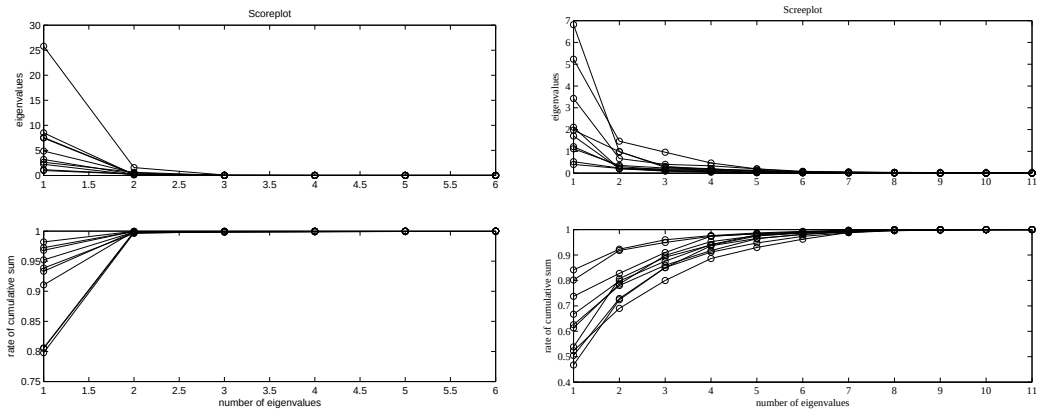
Number of Principal Components

Beside the selection of the right number of clusters, the second bottleneck of the successful application of the presented algorithm is the selection of the the right number of principal components. This can be done by the analysis of the eigenvalues of the covariance matrices of the initial segments. For this purpose a so-called screeplot can be drawn that plots the ordered eigenvalues according to their contribution to the variance of data. Another possibility is to define q based on the desired accuracy (loss of variance) of the PPCA models:

$$\sum_{j=1}^{q-1} \lambda_{i,j} / \sum_{j=1}^{n} \lambda_{i,j} < accuracy \leq \sum_{j=1}^{q} \lambda_{i,j} / \sum_{j=1}^{n} \lambda_{i,j}. \tag{3.70}$$

The syntetic dataset shown in Figure 3.6 was initially partitioned into ten segments. As Figure 3.8(a) illustrates, the magnitude of the eigenvalues and their cumulative rate to their sum show that two PCs are sufficient to approximate the distribution of the data with 98% accuracy. Obviously, this analysis can be fully automatized.

It is interesting to note that the analysis of the fuzzy hypervolume cluster validity measure is similar to the approach of the analysis of the eigenvalues, because the $V_i = (\det(\mathbf{F}_i))^{1/2}$ hypervolume of a cluster is proportional to the product of the eigenvalues. Hence, the right number of principal components can be also determined based on the product of the q largest eigenvalues.



(a) Synthetic dataset. *(b) Industrial dataset.*

Figure 3.8: Screeplots of the synthetic and the industrial data shown in Figure 3.6 and Figure A.9, respectively.

The Segmentation Algorithm

Based on the the previously presented building blocks the presented clustering based segmentation algorithm is formulated as follows:

Algorithm 3.2.2 (Time-Series Segmentation Algorithm)**.**

Step 1 *Uniformly segment the data by a large number of segments (In the examples given in this session ten segments were used as starting point). Determine the q number of the principal components based on the analysis of the eigenvalues of these segments. For this purpose screeplot or cluster validity measure can be used (see Section 3.2 for more details).*

Step 2 *The values of m fuzziness parameter, the γ threshold for the $\mathbf{O}$ compatibility matrix and the ϵ termination tolerance must be chosen. In the case studies $m = 2, \epsilon = 10^{-4}$, the value of γ is usually between $0.3 - 0.75$ depending of the homogeneity of the time-series.*

Step 3 *Execute the clustering algorithm (see Section 3.2). The cluster merging must be evaluated after predefined number of iteration steps (see Section 3.2). In all of our applications this number was 100. The algorithm stops if the termination tolerance is reached and cluster merging is not necessary.*

As this formulation of the algorithm shows, although there is a need to define some parameters of the algorithm before its application (γ, ϵ, and the initial number of clusters), it is possible to apply the method for high dimensional time-series even if almost nothing is known about the structure of these series in advance. Hence, the method is useful for knowledge discovery. Of course, data mining is an iterative procedure. The results of the segmentation should

be evaluated by human experts or by the performances of other modelling and data mining tools based on the segmented data, and if it is needed, the "knowledge worker" should return to the segmentation task with a new set of these parameters.

In the following the effectiveness of the developed algorithm will be illustrated by two examples: the synthetic dataset introduced in Section 3.2 and an industrial dataset taken from an industrial polymerization reactor, and the obtained results will be compared to the results given by the multivariate extension of the bottom-up segmentation algorithm of Keogh [64].

Example 3.3. Synthetic time-series segmentation based on GG clustering

The synthetic dataset given in Figure 3.6 is designed to illustrate how a multivariate segmentation algorithm should detect the changes of the latent process behind the high dimensional data.

In Figure 3.8 it has been shown that the screeplot of the eigenvalues suggests that the clustering algorithm should take into account two principal components. From Figure 3.6(c) – which shows the β_i normalized and the $A_i(t) = p(t_k|\eta_i)$ Gaussian membership functions, and the $p(\mathbf{z}_k|\eta_i)$ probabilities – it can be seen that with this parameter the presented method found five segments and it is able to detect the changes in the correlation structure and in the mean of the data. The $S_{PCA}^{i(i+1)}$ similarity measures of the adjacent clusters are $0.99, 0.17, 0.99$, and 0.99812, which suggest that the correlation among the variables has significantly changed between the first and the second segments, while the other segments differ mainly in their mean. These results agree with Figure 3.6(a) and justify the accuracy and the usefulness of the presented method.

To illustrate the importance of the selection of the right number of PCs the same segmentation has been performed with only one PC. In this case, the algorithm found 10 nearly symmetric segments, hence it was not able to explore the hidden information behind the data. As it is depicted in Figure 3.6(c), in case of five PCs the algorithm gave reasonable, but not so characteristic result. These results were compared to the results of the bottom-up method based on the Hotelling T^2 (top) and the reconstruction error Q (bottom) shown in Figure 3.6(d). The bottom-up algorithm based on the reconstruction error Q is sensitive to the change in the correlation structure but it was not able to find the change in the mean. The method based on the Hotelling T^2 measure is on the contrary. The method based on the Q measure is very sensitive to the number of PCs. As can be seen in Figure 3.6(d) when $q = 2$ the result is very different from that obtained by $q = 5$, but in both cases the algorithm finds the change in the correlation structure.

This comparison showed contrary to the multivariate extensions of the classical bottom-up segmentation algorithm, the developed cluster analysis based segmentation algorithm can simultaneously handle the problems of the detection of the change of the latent process and the change of the mean of the variables and it is more robust with respect to the number of principal components.

□

3.3 Fuzzy Clustering for Classifier Induction

Algorithm 3.3.1 (Supervised Fuzzy Clustering)**.**

Initialization *Given a set of data* $\mathbf{Z}$ *specify* R*, choose a termination tolerance* $\epsilon > 0$*. Initialize the* $\mathbf{U} = [\mu_{i,k}]_{R \times N}$ *partition matrix randomly, where* $\mu_{i,k}$ *denotes the membership that the* $\mathbf{z}_k$ *data is generated by the* i*th cluster.*

Repeat *for* $l = 1, 2, \ldots$

Step 1 *Calculate the parameters of the clusters*

- *Calculate the centers and standard deviation of the Gaussian membership functions (the diagonal elements of the* $\mathbf{F}_i$ *covariance matrices):*

$$\mathbf{v}_i^{(l)} = \frac{\sum\limits_{k=1}^{N} \left(\mu_{i,k}^{(l-1)}\right)^m \mathbf{x}_k}{\sum\limits_{k=1}^{N} \left(\mu_{i,k}^{(l-1)}\right)^m}, \; \sigma_{i,j}^{2\,(l)} = \frac{\sum\limits_{k=1}^{N} \left(\mu_{i,k}^{(l-1)}\right)^m (x_{j,k} - v_{j,k})^2}{\sum\limits_{k=1}^{N} \left(\mu_{i,k}^{(l-1)}\right)^m} \tag{3.71}$$

- *Estimate the consequent probability parameters,*

$$p(c_i | r_j) = \frac{\sum_{k | y_k = c_i} \left(\mu_{j,k}^{(l-1)}\right)^m}{\sum_{k=1}^{N} \left(\mu_{j,k}^{(l-1)}\right)^m}, 1 \leq i \leq C, \; 1 \leq j \leq R \tag{3.72}$$

- *A priori probability of the cluster and the weight (impact) of the rules:*

$$P(r_i) = \frac{1}{N} \sum_{k=1}^{N} \left(\mu_{i,k}^{(l-1)}\right)^m, \;\; w_i = P(r_i) \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_{i,j}^2}} \tag{3.73}$$

Step 2 *Compute the distance measure* $d^2(\mathbf{z}_k, r_i)$ *by* (3.75)*.*

Step 3 *Update the partition matrix*

$$\mu_{i,k}^{(l)} = \frac{1}{\sum\limits_{j=1}^{R} \left(d(\mathbf{z}_k, r_i)/d(\mathbf{z}_k, r_j)\right)^{2/(m-1)}}, \; 1 \leq i \leq R, \; 1 \leq k \leq N \tag{3.74}$$

until $||\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}|| < \epsilon$.

The automatic determination of compact fuzzy classifiers rules from data has been approached by several different techniques. Generally, the bottleneck of the data-driven identification of fuzzy systems is the structure identification that requires nonlinear optimization. Thus for high dimensional problems, the initialization the fuzzy model becomes very significant. Common initializations methods such as grid-type partitioning [74] and *rule generation on extrema* initialization, result in complex and non-interpretable initial models and the rule-base simplification and reduction steps become computationally demanding. To avoid these problems, fuzzy clustering algorithms [75] were put forward. However, the obtained membership values have to be projected onto the input variables and approximated by parameterized membership functions that deteriorates the performance of the classifier. This decomposition error can be reduced by using eigenvector projection [37], but the obtained linearly transformed input variables do not allow the interpretation of the model. To avoid the projection error and maintain the interpretability of the model, the presented approach is based on the Gath–Geva (GG) algorithm [38], because the simplified version of GG clustering allows the direct identification of fuzzy models with exponential membership functions [44].

Neither GG nor GK algorithm does not utilize the class labels. Hence, they give suboptimal result if the obtained clusters are directly used to formulate a classical fuzzy classifier. Hence, there is a need for fine-tuning of the model. This GA or gradient-based fine-tuning, however, can result in overfitting and thus poor generalization of the identified model. Unfortunately, the severe computational requirements of these approaches limit their applicability as a rapid model-development tool. This section focuses on the design of interpretable fuzzy rule based classifiers from data with low-human intervention and low-computational complexity. Hence, a new modelling scheme is introduced based only on fuzzy clustering (see also in [76]). The presented algorithm uses the class label of each point to identify the optimal set of clusters that describe the data. The obtained clusters are then used to build a fuzzy classifier.

The contribution of this approach is twofold.

- The classical fuzzy classifier consists of rules each one describing one of the C classes. In this section a new fuzzy model structure is presented where the consequent part is defined as the probabilities that a given rule represents the $c_1, \ldots, c_C$ classes. The novelty of this new model is that one rule can represent more than one classes with different probabilities.

- Classical fuzzy clustering algorithms are used to estimate the distribution of the data. Hence, they do not utilize the class label of each data point available for the identification. Furthermore, the obtained clusters cannot be directly used to build the classifier. In this section a new cluster prototype and the related clustering algorithm have been introduced that allows the direct supervised identification of fuzzy classifiers.

The presented algorithm is similar to the Multi-Prototype Classifier technique [77, 78]. In this approach, each class is clustered independently from the other classes, and is modeled by few components (Gaussian in general). The main difference of this approach is that each cluster represents different classes, and the number of clusters used to approximate a given class have to be determined manually, while the presented approach does not suffer from these problems.

Classical fuzzy clustering algorithms are used to estimate the distribution of the data. Hence, they do not utilize the class label of each data point available for the identification. Furthermore, the obtained clusters cannot be directly used to build the classifier. In the following a new cluster prototype and the related distance measure will be introduced that allows the direct supervised identification of fuzzy classifiers. As the clusters are used to obtain the parameters of the fuzzy classifier, the distance measure is defined similarly to the distance measure of the Bayes classifier (B.33):

$$\frac{1}{d^2(\mathbf{z}_k, r_i)} = \underbrace{P(r_i) \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\frac{(x_{j,k} - v_{i,j})^2}{\sigma_{i,j}^2}\right)}_{\text{Gath–Geva clustering}} P(c_j = y_k | r_i) \tag{3.75}$$

This distance measure consists of two terms. The first term is based on the geometrical distance between the $\mathbf{v}_i$ cluster centers and the $\mathbf{x}_k$ observation vector,

while the second is based on the probability that the r_i-th cluster describes the density of the class of the k-th data, $P(c_j = y_k|r_i)$. It is interesting to note, that this distance measure only slightly differs from the unsupervised Gath–Geva clustering algorithm which can also be interpreted in a probabilistic framework [38]. However, the novelty of the presented approach is the second term, which allows the use of class labels.

Similarly to the update equations of Gath–Geva clustering algorithm, the following equations will result in a solution using Lagrange multipliers method.

Example 3.4. Classification of the Wine data

In order to examine the performance of the presented identification method the well-known multidimensional classification benchmark problem is presented. The studied Wine data come from the UCI Repository of Machine Learning Databases[2]*. The performance of the obtained classifiers was measured by ten-fold cross validation. The data divided into ten sub-sets of cases that have similar size and class distributions. Each sub-set is left out once, while the other nine are applied for the construction of the classifier which is subsequently validated for the unseen cases in the left-out sub-set.*

For comparison purposes, a fuzzy classifier, that utilizes all the 13 information profile data about the wine has been identified by the presented clustering algorithm based on all the 178 samples. Fuzzy models with three and six rules were identified. The three rule-model gave only 2 misclassification (correct percentage 98.9%). When a cluster was added to improve the performance of this model, the obtained classifier gave only 1 misclassification (99.4%). The classification power of the identified models is then compared with fuzzy models with the same number of rules obtained by Gath–Geva clustering, as Gath–Geva clustering can be considered the unsupervised version of the presented clustering algorithm. The Gath–Geva identified fuzzy model achieves 8 misclassifications corresponding to a correct percentage of 95.5%, when three rules are used in the fuzzy model, while 6 misclassifications (correct percentage 96.6%) in the case of four rules. The results are summarized in Table 3.3. As it is shown, the performance of the obtained classifiers are comparable to those in [79] and [74], but use far less rules (3-5 compared to 60) and less features.

Table 3.3: Classification rates on the Wine data for ten independent runs.

Method	Best result	Aver result	Worst result	Rules	Model eval
Corcoran and Sen [79]	100%	99.5%	98.3%	60	150000
Ishibuchi et al. [74]	99.4%	98.5%	97.8%	60	6000
GG clustering	95.5 %	95.5 %	95.5 %	3	1
Sup (13 features)	98.9 %	98.9 %	98.9 %	3	1
Sup (13 features)	99.4 %	99.4 %	99.4 %	4	1

These results indicate that the presented clustering method effectively utilizes the class labels. As can be seen from Table 3.3, because of the simplicity of the presented clustering algorithm, the presented approach is attractive in comparison with other iterative and optimization schemes that involves extensive intermediate optimization to generate fuzzy classifiers.

[2]http://www.ics.uci.edu

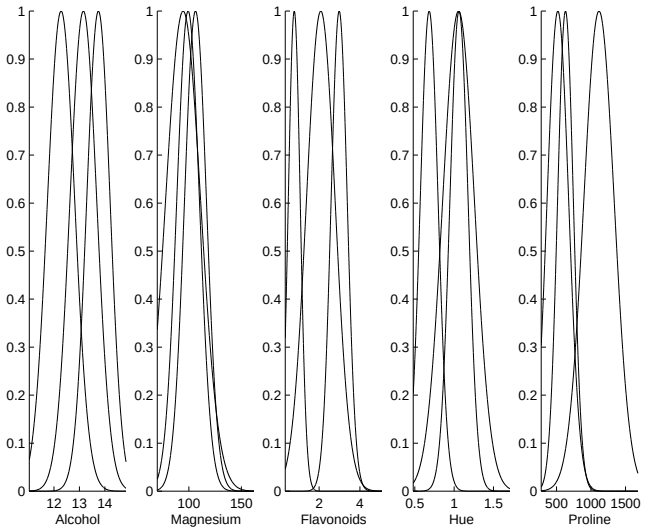


Figure 3.9: Membership functions obtained by fuzzy clustering.

Table 3.4: Classification rates and model complexity for classifiers constructed for the Wine classification problem. Results from averaging a ten-fold validation.

Method	$\min$Acc.	meanAcc.	$\max$Acc.	$\min \sharp$Feat.	mean$\sharp$Feat.	$\max \sharp$Feat.
GG: $R = 3$	83.33	94.38	100	10	12.4	13
Sup: $R = 3$	88.88	97.77	100	12	12.6	13
GG: $R = 3$	88.23	95.49	100	4	4.8	5
Sup: $R = 3$	76.47	94.87	100	4	4.8	5
GG: $R = 6$	82.35	94.34	100	4	4.9	5
Sup: $R = 6$	88.23	97.15	100	4	4.8	5

The ten-fold validation is a rigorous test of the classifier identification algorithms. These experiments showed 97.77% *average classification accuracy, with* 88.88% *as the worst and* 100% *as the best performance (Table 3.4). The above presented automatic model reduction technique removed only one feature without the decrease of the classification performance on the training data. Hence, to avoid possible local minima, the feature selection algorithm is used to select only five features, and the presented scheme has been applied again to identify a model based on the selected five attributes. This compact model with average 4.8 rules showed* 97.15% *average classification accuracy, with* 88.23% *as the worst and* 100% *as the best performance. The resulted membership functions and the selected features are shown in Figure 3.9. Comparing the fuzzy sets in Figure 3.9 with the data shows that the obtained rules are highly interpretable. For example, the Flavonoids are divided in Low, Medium and High, which is clearly visible in the data.*

□

3.4 Fuzzy Clustering for Model Order Selection

Most data-driven identification algorithms assume that the model structure is *a priori* known or that it is selected by a higher-level 'wrapper' structure-selection algorithm. Several information-theoretic criteria have been proposed for structure selection in linear dynamic input–output models. Examples of the classical criteria are the Final Prediction-Error (FPE) and the Akaike Information Criterion (AIC) [80]. Later, the Minimum Description Length (MDL) criterion developed by Schwartz and Rissanen was proven to produce consistent estimates of the structure of linear models [81]. With these tools, determining the structure of linear systems is a rather straightforward task. However, relatively little research has been done into the structure selection for nonlinear models. In the paper of Aguirre and Billings [82], the concepts of term clusters and cluster coefficients are defined and used. It is argued that if a certain type of term in a nonlinear model is spurious, the respective cluster coefficient is small compared with the coefficients of the other clusters represented in the model. In [83], this approach is used to the structure selection of polynomial models. In [84] an alternative solution to the model structure selection problem is introduced by conducting a forward search through the many possible candidate model terms initially and then performing an exhaustive all subset model selection on the resulting model. A backward search approach based on orthogonal parameter-estimation is also applied [56, 85]. As can be seen, these techniques are 'wrapped' around a particular model construction method. Hence, the result of the estimate can be biased due to the particular construction method used. To avoid this problem a 'model free' approach is followed where no particular model needs to be constructed in order to select the order of the model. The advantage of this approach is that this estimate is based on geometrical/embedding procedures and does not depend on the model representation that will be used a posteriori, i.e. the results would have a rather general character. This is important advantage, as the construction of a NARX model consists of the selection of many structural parameters which have significant effect to the performance of the designed model: e.g. the model order, type of the nonlinearity (Hammerstein or Wiener type system) [86], scheduling variables, number of neurons in a neural network, etc. The simultaneous selection of these structural parameters is a problematic task. The primary objective of this section is to decompose this complex problem by providing some useful guidance in selecting a tentative model order. However, it should be bore in mind that there is no escape of performing a model-driven structure selection, once a certain model representation is chosen. For instance, suppose a model-free model order selection algorithm is used to determine the correct model order. If a neural network is used to model the process, the designer still need to decide on the activation function, the number of nodes etc. Therefore, the model order selection method that will be presented definitely not spare the user of having to go through some sort of structure selection. Indeed, if the orders of the nonlinear input-output model are well chosen, then structure selection will be much facilitated.

Deterministic suitability measures [87] and false nearest neighbor (FNN) algorithms [88] have already been proposed for data-based selection of the model order. These methods build upon similar methods developed for the analysis of chaotic time series [89]. The idea behind the FNN algorithm is geometric in nature. If there is enough information in the regression vector to predict the future output, then for any two regression vectors which are close in the regression space, the corresponding future outputs are also close in the output space. The structure is then selected by computing the percentage of false neighbors, i.e., vectors that violate the above assumption. A suitable threshold parameter must be specified by the user. For this purpose, heuristic rules have been proposed [87]. Unfortunately, for nonlinear systems the choice of this parameter will depend on the particular system under study [88]. The computational effort of this method also rapidly increases with the number of data samples and the dimension of the model.

To increase the efficiency of this algorithm, I present two clustering-based algorithms, which can be found in one of our previous work [90] as well. The main idea of these algorithms is the following. When the available input–output data set is clustered in the product space of the regressors and the model output, the obtained clusters will approximate the regression surface of the model. Although a clustering algorithm is utilized, the method does not assume that the data exhibit a cluster substructure. Clustering is used to detect clusters that are local linear approximations of the regression surface. In the first algorithm the threshold constant that is used to compute the percentage of the false neighbors is estimated from the shape of the obtained clusters. Departing from the theory behind the MDL algorithm, a new direct model structure selection algorithm is also developed. If the right input variables are used, because of the functional relationship between the regressors and the model output, the data are locally highly correlated and the obtained clusters are flat. In this way, the problem of determining the appropriate regressors is transformed into the problem of checking the flatness of the clusters, similarly to [35, 91]. The main advantage of the presented solution is that it is model-free. This means that no particular model needs to be constructed in order to select the model structure. There is also no need for finding the nearest neighbors for each data point, which is a computationally expensive task.

FNN Algorithm

Many non-linear static and dynamic processes can be represented by the following regression model

$$y_k = f(\boldsymbol{\phi}_k) \tag{3.76}$$

where $f(.)$ is a nonlinear function and $\boldsymbol{\phi}_k$ represents its input vector and $k = 1, \ldots, N$ represents the index of the k-th available input-output data.

Among this class of models, the identification of discrete-time, Non-linear Auto-Regressive models with eXogenous inputs (NARX) is considered. In the NARX model, the model regressors are past values of the process outputs y_k

and the process inputs u_k.

$$\boldsymbol{\phi}_k = [y_{k-1}, \ldots, y_{k-n_a}, u_{k-1}, \ldots, u_{k-n_b}]^T \tag{3.77}$$

while the output of the model is the one-step ahead prediction of the process, y_k. The number of past outputs used to calculate y_k is n_a, and the number of past inputs is n_b. The values n_a and n_b are often referred to as model orders. The above SISO system representation can be assumed without a loss of generality since the extension to MISO and MIMO systems is straightforward.

The method of false nearest neighbors (FNN) was developed by Kennel [89] specifically for determining the minimum embedding dimension, the number of time-delayed observations necessary to model the dynamic behavior of chaotic systems. For determining the proper regression for input-output dynamic processes, the only change to the original FNN algorithm involves the regression vector itself [88].

The main idea of the FNN algorithm stems from the basic property of a function. If there is enough information in the regression vector to predict the future output, then any of two regression vectors which are close in the regression space will also have future outputs which are close in some sense. For all regression vectors embedded in the proper dimensions, for two regression vectors that are close in the regression space and their corresponding outputs are related in the following way:

$$y_k - y_j = df\left(\boldsymbol{\phi}_k^{n_a,n_b}\right)\left[\boldsymbol{\phi}_k^{n_a,n_b} - \boldsymbol{\phi}_j^{n_a,n_b}\right] + o\left(\left[\boldsymbol{\phi}_k^{n_a,n_b} - \boldsymbol{\phi}_j^{n_a,n_b}\right]\right)^2 \tag{3.78}$$

where $df\left(\boldsymbol{\phi}_k^{n_a,n_b}\right)$ is the Jacobian of the function $f(.)$ at $\boldsymbol{\phi}_k^{n_a,n_b}$.

Ignoring higher order terms, and using the Cauchy-Schwarz inequality the following inequality can be obtained:

$$|y_k - y_j| \leq \left\|df\left(\boldsymbol{\phi}_k^{n_a,n_b}\right)\right\|_2 \left\|\boldsymbol{\phi}_k^{n_a,n_b} - \boldsymbol{\phi}_j^{n_a,n_b}\right\|_2 \tag{3.79}$$

$$\frac{|y_k - y_j|}{\left\|\boldsymbol{\phi}_k^{n_a,n_b} - \boldsymbol{\phi}_j^{n_a,n_b}\right\|_2} \leq \left\|df\left(\boldsymbol{\phi}_k^{n_a,n_b}\right)\right\|_2 \tag{3.80}$$

If the above expression is true, then the neighbors are recorded as true neighbors. Otherwise, the neighbors are false neighbors.

Based on this theoretical background, the outline of the FNN algorithm is the following.

1. Identify the nearest neighbor to a given point in the regressor space. For a given regressor:

   $$\boldsymbol{\phi}_k^{n_a,n_b} = [y_{k-1}, \ldots, y_{k-n_a}, u_{k-1}, \ldots, u_{k-n_b}]^T$$

   find the nearest neighbor $\boldsymbol{\phi}_j^{n_a,n_b}$ such that the distance d is minimized:

   $$d = ||\boldsymbol{\phi}_k^{n_a,n_b} - \boldsymbol{\phi}_j^{n_a,n_b}||_2$$

2. Determine if the following expression is true or false

$$\frac{|y_k - y_j|}{||\boldsymbol{\phi}_k^{n_a,n_b} - \boldsymbol{\phi}_j^{n_a,n_b}||_2} \leq \mathcal{R}$$

   where $\mathcal{R}$ is a previously chosen threshold value. If the above expression is true, then the neighbors are recorded as true neighbors. Otherwise, the neighbors are false neighbors.

3. Continue the algorithm for all times k in the data set.

4. Calculate the percentage of points in the data that have false nearest neighbors $J(n_a, n_b)$.

5. Continue the algorithm for increasing n_a and n_b using the percentage of false nearest neighbors drops to some acceptably small number.

The FNN algorithm is sensitive to the choice of the $\mathcal{R}$ threshold. In [87] the threshold value was selected by trial and error method based on empirical rules of thumb, $10 \leq \mathcal{R} \leq 50$. However, choosing a single threshold that will work well for all data sets is impossible task. In this case, it is advantageous to estimate $\mathcal{R}$ based on (3.80) using the the maximum of the Jacobian, $\mathcal{R} = max_k \left\| df\left(\boldsymbol{\phi}_k^{n_a,n_b}\right) \right\|$, as it was suggested by Rhodes and Morari [88].

Since this method uses the Jacobian of the identified models, the performance and the flexibility of these models can deteriorate the estimate of the model order. When the Jacobian is overestimated, the algorithm underestimates the order of the system. Contrary, if the model estimates smaller Jacobian than the real Jacobian of the system, the model order selection algorithm overestimates the order of the model. Hence, the modeler should be careful at the construction of the model used to estimate the Jacobian of the system (e.g. the model can be over or under parameterized, etc.). To increase the efficiency of the FNN based structure selection, a clustering-based algorithm will be introduced in the following section.

Fuzzy Clustering based FNN

The main idea of this section is the following. When the available input-output data set is clustered in the product space of the regressors and the model output and when the appropriate regressors are used, the collection of the obtained clusters will approximate the regression surface of the model. In this case the clusters can be approximately regarded as local linearizations of the system and can be used to estimate $\mathcal{R}$. Clusters of different shapes can be obtained by different clustering algorithms by using an appropriate definition of cluster prototypes (e.g., points vs. linear varieties) or by using different distance measures. The Gustafson–Kessel clustering algorithm [92] has been often applied to identify Takagi–Sugeno fuzzy systems that are based on local linear models [35]. The main drawback of this algorithm is that only clusters with approximately

equal volumes can be properly identified which constrain makes the application of this algorithm problematic for the task of this section. To circumvent this problem, in this section Gath–Geva algorithm is applied [67, 38].

The clustering algorithm has only one parameter: c, the number of the clusters. In general, the increase of c increases the accuracy of the model. However, to avoid overfitting and the excessive computational costs, it is recommended to determine the number of the clusters automatically. For this purpose various methods can be applied [35, 38].

The applied validity measure is based on the hyper-volume index:

$$V_{fc} = \sum_{i=1}^{c} \det(\mathbf{F}_i)\,. \tag{3.81}$$

This index represents the volume of the clusters. When the nonlinear hypersurface of the identification data is correctly approximated by the clusters, this volume should be small. I scale this index by the volume of covariance matrix of the data

$$I = \frac{V_{fc}}{\det(\mathrm{cov}(\mathbf{X}))}\,. \tag{3.82}$$

Estimation of the $\mathcal{R}$ Threshold Coefficient

The collection of c clusters approximates the regression surface. Hence, the clusters can be approximately regarded as local linear subspaces described by the cluster ellipsoids. The smallest eigenvalues λ_{i,n_b+n_a+1} of the cluster covariance matrices $\mathbf{F}_i$ are typically in orders of magnitude smaller than the remaining eigenvalues [67, 35].

The eigenvector corresponding to this smallest eigenvalue, $\mathbf{t}^i_{n_b+n_a+1}$, determines the normal vector to the hyperplane spanned by the remaining eigenvectors of that cluster

$$(\mathbf{t}^i_{n_b+n_a+1})^T(\mathbf{x}_k - \mathbf{v}_i) = 0 \tag{3.83}$$

Similarly to the observation vector $\mathbf{x}_k = [\boldsymbol{\phi}_k^T\, y_k]^T$, the prototype vector is partitioned as $\mathbf{v}_i = \begin{bmatrix} \left(\mathbf{v}_i^{\phi}\right)^T & v_i^y \end{bmatrix}$ into a vector $\mathbf{v}^{\phi}$ corresponding to the regressor $\boldsymbol{\phi}_k$, and a scalar v_i^y corresponding to the output y_k. The smallest eigenvector is partitioned in the same way, $\mathbf{t}^i_{n_a+n_b+1} = \begin{bmatrix} \left(\mathbf{t}^{i,\phi}_{n_a+n_b+1}\right)^T & t^{i,y}_{n_a+n_b+1} \end{bmatrix}^T$. By using this partitioned vectors (3.83) can be written as

$$\begin{bmatrix} \left(\mathbf{t}^{i,\phi}_{n_a+n_b+1}\right)^T & t^{i,y}_{n_a+n_b+1} \end{bmatrix}^T \left([\boldsymbol{\phi}_k^T\, y_k]^T - \begin{bmatrix} \left(\mathbf{v}_i^{\phi}\right)^T & v_i^y \end{bmatrix} \right) = 0 \tag{3.84}$$

from which the parameters of the hyperplane defined by the cluster can be obtained:

$$y_k = \underbrace{\frac{-1}{t^{i,y}_{n_a+n_b+1}} \left(\mathbf{t}^{i,\phi}_{n_a+n_b+1}\right)^T}_{\mathbf{a}_i^T} \boldsymbol{\phi}_k + \underbrace{\frac{1}{t^{i,y}_{n_a+n_b+1}} \left(\mathbf{t}^{i}_{n_a+n_b+1}\right)^T \mathbf{v}_i}_{b_i} = \mathbf{a}_i^T \boldsymbol{\phi}_k + b_i \tag{3.85}$$

Although the parameters have been derived from the geometrical interpretation of the clusters, it can be shown [35] that (3.85) is equivalent to the weighed total least-squares estimation of the consequent parameters, where each data point is weighed by the corresponding $\sqrt{\mu_{i,k}}$.

The main contribution of this section is that it suggests the application of an adaptive threshold function to FNN, which takes into account the nonlinearity of the system. This means, based on the result of the fuzzy clustering, for all input-output data pairs different $\mathcal{R}_k$ values are calculated. Since, the optimal value of $\mathcal{R}_k$ is $\mathcal{R}_k = \|df\left(\boldsymbol{\phi}_k^{n_a,n_b}\right)\|$ and the $df\left(\boldsymbol{\phi}_k^{n_a,n_b}\right)$ partial derivatives can be estimated based on the shape of the clusters from (3.85)

$$df\left(\boldsymbol{\phi}_k^{n_a,n_b}\right) \approx \sum_{i=1}^{c} \mu_{i,k} \frac{-1}{t^{i,y}_{n_a+n_b+1}} \left(\mathbf{t}^{i,\phi}_{n_a+n_b+1}\right)^T \tag{3.86}$$

the threshold can be calculated as

$$\mathcal{R}_k = \left\| \sum_{i=1}^{c} \mu_{i,k} \frac{-1}{t^{i,y}_{n_a+n_b+1}} \left(\mathbf{t}^{i,\phi}_{n_a+n_b+1}\right)^T \right\|_2 \tag{3.87}$$

Cluster Analysis based Direct Model Order Estimation

In the previous section a new cluster analysis based approach to the adaptive choice of the $\mathcal{R}$ threshold value of the FNN algorithm has been presented. Based on the geometric idea behind this algorithm, in this section an alternative structure selection algorithm will be presented that does not require the time-consuming calculation of the nearest neighbors in the identification data.

The idea of this second algorithm is based on the well known fact that in the absence of the observation noise, the covariance matrix of the identification data generated by a linear system has a zero eigenvalue with multiplicity s given by

$$s = 1 + \min(n_a - n_{a,l}, n_b - n_{b,l}) \tag{3.88}$$

when the selected model orders n_b and n_a are greater than or equal to the true orders of the linear system, i.e., $n_b \geq n_{b,l}$ and $n_a \geq n_{a,l}$ [93]. This relationship between the parameters n_a, n_b and the eigenvalues of the covariance matrix can be used to select the model order.

In [81] it has been shown that the widely applied Minimum Description Length (MDL) model order selection criterion can be expressed based on the smallest eigenvalue of the data covariance matrix:

$$J^{n_a,n_b}_{MDL} = \frac{N}{2}\log(\lambda_{min}) + \frac{1}{2}(n_a + n_b)\log N \tag{3.89}$$

Multiplying both sides by $2/N$ and combining the terms results in

$$\frac{2}{N} J^{n_a,n_b}_{MDL} = \log\left(\lambda_{min}\left(N^{1/N}\right)^{n_a+n_b}\right) \tag{3.90}$$

As $\log(.)$ is a monotonically increasing, in [81] a criterion containing exactly the same information as MDL has been proposed:

$$J^{n_a,n_b} = \lambda_{min} \left(N^{1/N}\right)^{n_a+n_b} \tag{3.91}$$

Since $N^{1/N} \approx 1$ for large N, one can see that the MDL criterion asymptotically provides the same information as the minimum eigenvalue of the covariance matrix. The advantage of the above formulation is that it can also be applied to noise-free data, in which λ_{min} is zero and where the logarithm in (3.89) thus cannot be calculated.

The utilized fuzzy clustering obtains local linear approximation of the nonlinear system, (3.91) can be easily modified for cluster-based model order estimation by weighting the values of this simplified cost functions calculated from the cluster covariance matrices with the a priori probability of the clusters:

$$J^{n_a,n_b} = \sum_{i=1}^{c} \alpha_i \lambda_{i,min} \tag{3.92}$$

Because the model order is determined by finding the number of past outputs n_a and past inputs n_b, the $J(n_a, n_b)$ indices form a table in these two dimensions. It is possible to find a 'global' solution (or solutions) for the model orders by computing the index over all values of n_a and n_b in a certain range and search for a rapid decrease of $J(n_a, n_b)$. The indices that have the smallest $J(n_a, n_b)$ relative to $J(n_a - 1, n_b)$ and $J(n_a, n_b - 1)$ represents the 'best' estimate of the model order. Hence, each row of the table is divided by the previous row to form a row ratio table, and each column is divided by the previous column to create a column ratio table. With the use of these ratios the model order can be determined:

$$[n_a, n_b] = \underset{n_a,n_b}{\arg\min} \left\{ \max \left(\frac{J(n_a, n_b)}{J(n_a - 1, n_b)}, \frac{J(n_a, n_b)}{J(n_a, n_b - 1)} \right) \right\} . \tag{3.93}$$

Example 3.5. Direct Model order selection of a polymerization reactor

In this example, taken from [88], I use data generated by a simulation model of a continuous polymerization reactor. This model describes the free-radical polymerization of methyl methacrylate with azobisisobutyronitrile as an initiator and toluene as a solvent. The reaction takes place in a jacketed CSTR. Under some simplifying assumption, the first-principle model is given by:

$$\begin{aligned}
\dot{x}_1 &= 10(6 - x_1) - 2.4568 x_1 \sqrt{x_2} \\
\dot{x}_2 &= 80u - 10.1022 x_2 \\
\dot{x}_3 &= 0.024121 x_1 \sqrt{x_2} + 0.112191 x_2 - 10 x_3 \\
\dot{x}_4 &= 245.978 x_1 \sqrt{x_2} - 10 x_4 \\
y &= \frac{x_4}{x_3}
\end{aligned}$$

The dimensionless state variable x_1 is the monomer concentration, and x_4/x_3 is the number-average molecular weight (the output y). The process input u is the dimensionless volumetric flow rate of the initiator. For further information on this model and its derivation, see [94]. According to [88], I apply a uniformly distributed random input over the range 0.007 to 0.015 with the sampling time of 0.2 s.

With four states, a sufficient condition for representing the dynamics is a regression vector that includes four delayed inputs and outputs. In this case, however, the system has two states that are weakly observable. This can be observed by linearizing the system and performing balanced realization, which shows that two of the Hankel singular values are larger than the remaining singular values. This week observability leads to the system can be approximated by a smaller input–output description [95]. Obviously, the results may change depending on where the system is linearized. Although, in this case such effect have not occurred, the local linear behavior of a nonlinear system can significantly vary, even if the system is analyzed around off-equilibrium operating points [56, 96]. The main advantage of the presented clustering based approach is that the clusters are the local linear approximations of the nonlinear system, so they can be directly used to estimate the operating regions and the orders of the local linear models [67].

The presented clustering-based algorithm was applied to 960 data points. The indices $J(n_a, n_b)$ obtained by using the direct model order estimation (see (3.92)*) are given in Table 3.5.*

Table 3.5: Polymerization data: $J(n_a, n_b)$ values obtained based on the eigenvalues of 3 clusters.

Input lags (n_b)	*Output lags (n_a)*				
	0	1	2	3	4
0	-	5.55	4.84	4.80	4.81
1	4.28	1.28	0.54	0.43	0.42
2	1.23	0.30	0.44	0.41	0.37
3	0.37	0.34	0.31	0.33	0.34
4	0.29	0.35	0.30	0.32	0.32

With the use of (3.93)*, the ratios of the $J(n_a, n_b)$ values were computed and tabulated in Table 3.6. One can see that the structure with $n_a = 1$ and $n_b = 2$ is clearly indicated. This result is in agreement with the analysis of Rhodes [88] who showed that a nonlinear model with these orders is appropriate.*

Table 3.6: Polymerization data: Ratios obtained from Table 3.5.

Input lags (n_b)	*Output lags (n_a)*			
	1	2	3	4
1	0.30	0.42	0.80	0.98
2	0.24	1.47	0.95	0.90
3	1.13	0.91	1.06	1.03
4	1.21	0.97	1.07	1.00

The clustering based false nearest neighbor (FNN) algorithm is also applied to the data, and the results are given in Table 3.7. The model structure with $n_a = 1$ and $n_b = 2$ is indicated, which confirms the above results. The main drawback of the FNN algorithm, however, is that it requires demanding calculations of the nearest neighbors for each data point.

Table 3.7: Polymerization data: results obtained with the FNN method.

Input lags (n_b)	*Output lags (n_a)*				
	0	1	2	3	4
0	100.00	99.59	92.32	53.64	0.40
1	99.46	69.54	10.24	0.94	0.27
2	73.18	3.10	2.69	0.40	0.00
3	8.76	0.81	0.13	0.00	0.00
4	0.54	0.00	0.00	0.00	0.00

Table 3.8 shows results obtained for the linear ARX model structure. Note that for this particular process, the linear method also works satisfactorily, although the decrease is less sharp.

Table 3.8: Polymerization data: results obtained with a linear model (smallest eigenvalue of the covariance matrix of the data).

Input lags (n_b)	*Output lags (n_a)*				
	0	1	2	3	4
0	-	19.65	16.62	14.98	14.04
1	10.02	3.33	2.14	2.00	1.99
2	2.91	1.94	1.93	1.91	1.87
3	1.93	1.91	1.82	1.81	1.80
4	1.88	1.82	1.81	1.75	1.75

I can allocate that this linear model-based method does not give conspicuously incorrect results, as it behaves similarly to the method presented in [88]. The only difference is that the linear model-based approach applies the "average" gain of the system, while the method of Rhodes and Morari utilizes the maximal gain of the nonlinear system [88]. For highly nonlinear systems both approaches can induce large model order estimation error, as the linear model-based approach can over-, while the maximum gain-based approach can under-estimate the order of the system.

□

3.5 Conclusions

I developed a novel fuzzy clustering algorithm that is able to simultaneously fit local linear models of fuzzy models and determine their operating regions[3],[4]. Comparing with well-known methods it can be determined that the developed method gives the most transparent results at similar accuracy to these methods.

The classical fuzzy classifier consists of rules each one describing one of the classes. In this chapter I proposed a novel fuzzy model structure where each rule can represent more than one classes with different probabilities. The obtained classifier can be considered as an extension of the quadratic Bayes classifier that utilizes mixture of models for estimating the class conditional densities. A supervised clustering algorithm has been worked out for the identification of this fuzzy model. The relevant input variables of the fuzzy classifier have been selected based on the analysis of the clusters by Fisher's interclass separability criteria. This new approach is applied to the well-known wine and Wisconsin Breast Cancer classification problems[5]. The proposed method can be used for the discretization of continuous features to form efficient fuzzy decision tree based classifiers. The resulted fuzzy classifiers are very compact and well interpretable while the accuracy is still comparable to the best results reported in the literature[6].

To develop a method that is able to detect changes in the hidden relationship between variables and/or in their average I worked out a novel method based on mixtures of probabilistic principal component analysis (PPCA) models. PPCA model parameters and the borders of the segments are determined by a fuzzy clustering based algorithm. The developed tool is able to determine the number of principal components and segments automatically. The effectiveness of the developed method was demonstrated by synthetic and also industrial data sets[7].

[3]Abonyi J, Babuska R, Szeifert F, Modified Gath-Geva fuzzy clustering for identification of Takagi-Sugeno fuzzy models, IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART B-CYBERNETICS 32: pp. 612-621. (2002), IF: 0.630, Independent citations: 97

[4]Abonyi J, Roubos JA, Oosterom M, Szeifert F, Compact TS-fuzzy models through clustering and OLS plus FIS model reduction, IEEE International Conference on Fuzzy System., 2001. pp. 1420-1423. Independent citations: 18

[5]Abonyi J, Szeifert F, Supervised fuzzy clustering for the identification of fuzzy classifiers, PATTERN RECOGNITION LETTERS 24: pp. 2195-2207. (2003), IF: 0.809, Independent citations: 54

[6]Abonyi J, Supervised Fuzzy Clustering Based Initialization of Fuzzy Partitions for Decision Tree Induction. The 14th Online World Conference on Soft Computing in Industrial Applications WWW, 2009.11.17-2009.11.29,pp. 1-10.

[7]Abonyi J, Feil B, Nemeth S, Arva P, Modified Gath-Geva clustering for fuzzy segmentation of multivariate time-series, FUZZY SETS AND SYSTEMS 149: pp. 39-56. (2005), IF: 1.039, Independent citations: 16

I recognized that the false nearest neighbor method (FNN) usually used for model order estimation can be improved using clustering because time demanding model identification steps can be omitted. I proposed an algorithm that can be used to accurately estimate model orders without FNN on the only basis of clustering results with the analysis of cluster covariance matrices, so other time consuming computational steps, searching for nearest neighbors, can be avoided. The developed algorithms gave accurate results linear as well as nonlinear cases (polymerization and Van der Vusse reactors)[8]. There is an analogous problem related to autonomous systems as well. The proper selection of the state space dimension is a challenge by chaotic systems. As a solution, a novel method has been worked out with its help three tasks can be solved simultaneously utilizing clustering results: (1) selection of the dimension of the state space, (2) estimation of the dimension of the manifold that data extend, and (3) identification of a model that can be used for prediction. It can be determined from the examples that the developed method gave accurate results by the autonomous systems with known dimensions[9].

[8]Feil B, Abonyi J, Szeifert F, Model order selection of nonlinear input-output models - a clustering based approach, JOURNAL OF PROCESS CONTROL 14: pp. 593-602. (2004), IF: 1.241, Independent citations: 20

[9]Balazs Feil, Balazs Balasko, Janos Abonyi, Visualization of Fuzzy Clusters by Fuzzy Sammon Mapping Projection – Application to the Analysis of Phase Space Trajectories, special issue of "Soft Computing" on "Soft Computing for Information Mining", 11, (5), 479-488,. 2006 (IF: 0.33)

Chapter 4

Prior Knowledge based Constraints in Parameter Identification

In many practical situations, the involvement of laboratory and industrial experiments are expensive and time consuming and accurate measurements cannot be made. This problem results in a small number of data points that can be used for parameter identification. Explicit regularization, like penalties on non-smooth behavior of the model and application of *a priori* knowledge-based parameter constraints, can dramatically improve the robustness of the identification algorithm, eventually leading to more accurate parameter estimates [97]. In this chapter two algorithms will be presented based on this concept.

4.1 Grey-Box Fuzzy Model Identification

Constrained Identification of TS Fuzzy Models

Fuzzy models are often overparametrized, even with the implicit regularization effect of the local identification [98]. In this sectoion a new approach to data-driven identification of TS fuzzy models will be presented based on constrained identification of fuzzy models. Therefore, in the following the generating element of this algorithm, the constrained identification of the consequent parameters, is presented. When linear equality and inequality constraints are defined on these parameters, quadratic programming (QP) can be applied instead of the previously presented least-squares method, which can be solved effectively compared to other constrained nonlinear optimization algorithms [99].

The possible parameter constraints can be grouped into three categories:

- **Local constraints** are valid only for the parameters of a regression model, $\mathbf{\Lambda}_i\theta_i \leq \omega_i$.
- **Global constraints** are related to all of the regression models, $\mathbf{\Lambda}_{gl}\theta_i \leq \omega_{gl},\ i = 1, \ldots, c$.

- **Relative constraints** define the relative magnitude of the parameters of two or more regression models,

$$\mathbf{\Lambda}_{rel,i,j} \begin{bmatrix} \theta_i \\ \theta_j \end{bmatrix} \leq \omega_{rel,i,j}$$

.

These constraints can be written in a compact form:

$$\mathbf{\Lambda}\theta \leq \omega \tag{4.1}$$

with

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{\Lambda}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{\Lambda}_c \\ \mathbf{\Lambda}_{gl} & 0 & \cdots & 0 \\ 0 & \mathbf{\Lambda}_{gl} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{\Lambda}_{gl} \\ \{\mathbf{\Lambda}_{rel}\} & & & \end{bmatrix}, \ \omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_c \\ \omega_{gl} \\ \omega_{gl} \\ \vdots \\ \omega_{gl} \\ \{\omega_{rel}\} \end{bmatrix}, . \tag{4.2}$$

An example of these types of constraints is illustrated in Figure 4.1.

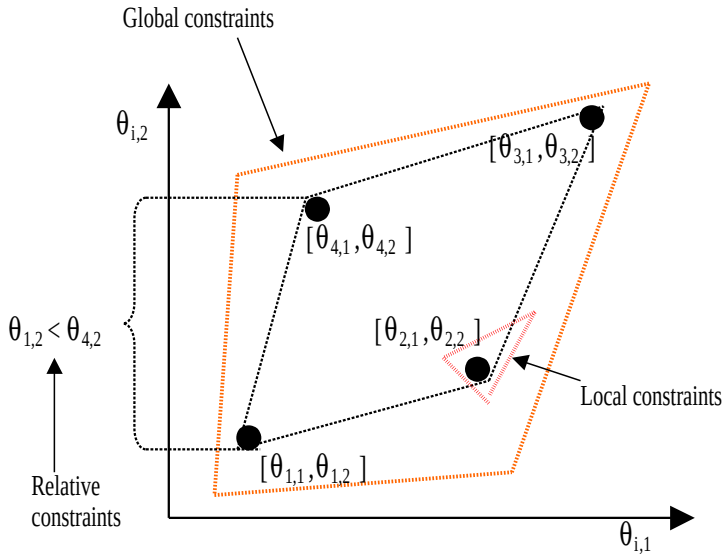


Figure 4.1: Example for local, global and relative constraints.

With the use of (4.1), a constrained global optimization problem can be formulated:

$$\min_{\theta'} \left\{ \frac{1}{2}\theta^T \mathbf{H}\theta + \mathbf{c}^T\theta \right\} \tag{4.3}$$

with $\mathbf{H} = 2{\mathbf{X}_e}^T\mathbf{X}_e \quad \mathbf{c} = -2{\mathbf{X}_e}^T\mathbf{y}$ subject to (4.1), where $\mathbf{X}_e$ is defined by the method described in [100].

When relative constraints are present at local identification, the set of the weighted optimization problems has to be solved simultaneously. In this case $\mathbf{H}$ and $\mathbf{c}$ are formulated as follows: $\mathbf{H} = 2\mathbf{X}^T\mathbf{\Phi}\mathbf{X}$, $\mathbf{c} = -2\mathbf{X}^T\mathbf{\Phi}\mathbf{y}$, where

$$\mathbf{y} = \begin{bmatrix} \mathbf{y} \\ \mathbf{y} \\ \vdots \\ \mathbf{y} \end{bmatrix}, \ \mathbf{X}_e = \begin{bmatrix} \mathbf{X} & 0 & \cdots & 0 \\ 0 & \mathbf{X} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{X} \end{bmatrix}, \ \Phi = \begin{bmatrix} \Phi_1 & 0 & \cdots & 0 \\ 0 & \Phi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Phi_{N_r} \end{bmatrix}. \quad (4.4)$$

Prior Knowledge based Parameter Constraints

The proposed grey-box identification method is based on the constrained data-driven identification of fuzzy systems presented in the previous subsection. The linear inequality constraints can be based on prior knowledge about the dynamic behavior of the process. The aim of this section is to show how such knowledge can be transformed into the constraints.

It has been shown in [101, 102, 103] that certain types of a priori knowledge about the dynamic behavior of a linear system can be expressed in the form of linear inequality constraints on the parameters of a linear time-invariant (LTI) model:

$$\Lambda_{LTI}\theta_{LTI} \leq \omega_{LTI} \quad (4.5)$$

where $\theta_{LTI} = [a_1, a_2, \ldots, a_{n_a}, b_1, \ldots, b_{n_b}, c]$ denotes the parameters of the LTI model

$$\hat{y}(k) = \sum_{i=1}^{n_a} a_i y(k-i) + \sum_{i=1}^{n_b} b_i u(k-i-n_d) + c\,. \quad (4.6)$$

These constraints on the LTI model parameters define a convex parameter set Ω:

$$\Omega = \{\theta_{LTI} | \Lambda_{LTI}\theta_{LTI} \leq \omega_{LTI}\}\ . \quad (4.7)$$

The aim of this session is to incorporate important prior knowledge into the fuzzy model. The parameter set, θ_{LTI}, of the LTI model (4.6) has to be a subset of Ω.

Because of the convexity of Ω and the convexity of the applied fuzzy inference method, if we would like to use the fuzzy model in LPV interpreted mode, it is sufficient to check the constraints for the rule consequents. This means that the constraints can easily be adapted to the TS fuzzy model:

$$\Lambda^*\theta_j \leq \omega^* \quad (4.8)$$

where $\theta_j = \left[a_1^j, \ldots, a_{n_a}^j, \ldots, b_{n_b}^j, c^j\right]$ denotes the parameters of the j-th local model, $\mathbf{\Lambda}^*$ and ω^* represent global constraints on the fuzzy model.

In the following, it will be shown how prior knowledge about the stability, stationary gains and the settling time of the process can be transformed into the linear inequalities (4.8).

- *Prior Knowledge About Sampling* It is well known that the poles of a stable discrete-time model that emerge from a correctly sampled, continuous-time system cannot be situated in $\mathcal{C}^-$, the left half of the complex plane. This knowledge on sampling can be translated into inequality constraints on the model parameters:

$$(-1)^i \left(-a_i^j\right) \geq 0,\ 1 \leq i \leq n_y\,. \tag{4.9}$$

- *Prior Knowledge About Process Stability* Additional constraints can be derived from stability considerations. Let $\mathcal{C}(m,R)$ denote the set of complex numbers within or at a circle with a real-valued center m and radius R,

$$\mathcal{C}(m,R) = \{z \in \mathcal{C} \mid \|z-m\| \leq R;\ m \in \mathcal{R};\ R \in \mathcal{R}+\}\,. \tag{4.10}$$

Tulleken [103] has shown that for the poles of a linear discrete-time system to be in $\mathcal{C}(m,R)$, the following linear constraints on the model parameters must be satisfied:

$$\Re \cdot \left[1, \left(-a_1^j\right), \left(-a_2^j\right), \ldots, \left(-a_{n_a}^j\right)\right]^T \ \geq\ 0\,, \tag{4.11}$$

$$\Re \cdot \Im \cdot \left[1, \left(-a_1^j\right), \left(-a_2^j\right), \ldots, \left(-a_{n_a}^j\right)\right]^T \ \geq\ 0\,, \tag{4.12}$$

where the nonzero elements of the upper and lower triangular matrices $\Re$ and $\Im$ are defined by

$$[\Re]_{ij} = (-1)^i \binom{j}{i},\ i = 0,1,\ldots,n_y,\ j = 0,1,\ldots,n_a\,, \tag{4.13}$$

$$[\Im]_{ij} = (m-R)^{i-j} \binom{n_a-j}{i-j} (2R)^{n_a-i}\,. \tag{4.14}$$

If $m=0$ and $R=1$, the previous equation constitutes the smallest convex hull of the admissible parameter region corresponding to the stable system model having all poles in the unit circle, $\mathcal{C}(0,1)$, in the complex plane. Therefore, these are necessary conditions for asymptotic stability. More details about the derivation of the above constraints can be found in [103].

Example 4.1. *Stability of a second-order system. Let us consider a second-order system with $n_a = 2$ and $n_b = 1$: $y(k+1) = a_1 y(k) + a_2 y(k-1) + b_1 u(k) + c$. In this case $\Re \cdot \Im$ becomes:*

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 \\ -4 & 2 & 0 \\ 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 0 & -2 \\ 1 & -1 & 1 \end{bmatrix}$$

It can be seen from the coefficients of the above matrices and equation (4.11) *that for the given second-order system the following holds:*

$$\begin{aligned} 1 + a_1 + a_2 &\geq 0, \\ -a_1 - 2a_2 &\geq 0, \\ a_2 &\geq 0\,. \end{aligned}$$

From (4.12) *and* (4.1)*, the following inequality constraints are obtained:*

$$\begin{aligned} 1 + a_1 + a_2 &\geq 0, \\ 2 - 2a_2 &\geq 0, \\ 1 - a_1 + a_2 &\geq 0\,. \end{aligned}$$

□

- *Knowledge About Process Gain* Often not only the stability of the plant is known beforehand, but also the admissible intervals or at least the signs of the static gains are known. Since the steady-state gain is only defined for stable systems, open-loop stability (4.11) – (4.12) must also be imposed on the estimates. From equation (4.6), the restrictions on the gain of the local model are given by

$$K^j_{\min} \leq \frac{\sum_{i=1}^{n_b} b_i^j}{1 - \sum_{i=1}^{n_a} a_i^j} \leq K^j_{\max}\,. \tag{4.15}$$

  This knowledge can be represented by the following inequality constraints:

$$K^j_{\min}\left(1 - \sum_{i=1}^{n_a} a_i^j\right) - \sum_{i=1}^{n_b} b_i^j \leq 0\,, \tag{4.16}$$

$$-K^j_{\max}\left(1 - \sum_{i=1}^{n_a} a_i^j\right) + \sum_{i=1}^{n_b} b_i^j \leq 0\,. \tag{4.17}$$

- *Knowledge About Settling Time* The approximate settling time around a given operating point (the time required for a step response to settle within a band around its final value) is usually known. For second-order discrete-time systems, a known maximum settling time τ_s^j approximately translates to all poles lying within a circle, centered at the origin in the z-plane with the radius $r^j = \exp\left(-4.6\frac{T_s}{\tau_s^j}\right)$, where T_s denotes the sampling time. Based on this consideration, the following inequality constraints can be developed [102]:

$$\begin{aligned} -\left(r^j_{\max}\right)^2 a_1 + \left(r^j_{\min} - 2r^j_{\max}\right) a_2^j &\leq -\left(r^j_{\max}\right)^2 r^j_{\min}\,, \\ r^j_{\max} a_1^j + a_2^j &\leq \left(r^j_{\max}\right)^2\,, \\ a_2^j &\leq 0\,. \end{aligned} \tag{4.18}$$

- *Knowledge About the Nonlinearity* For nonlinear processes, besides the bounds on the gain and/or the settling time, the tendency of the change of these parameters may be known as well.

By using local constraints, the prior information on the nonlinear behavior around a given operating point can be used in the identification procedure, while the global constraints (4.8) represent knowledge that is independent of the operating point.

Example 4.2. Identification of a liquid level process.

The following example is used to illustrate the advantages of the proposed identification method. The laboratory process consists of two cascaded tanks depicted in Figure 4.2. Water is supplied into the upper tank through a controlled peristaltic pump. A pressure transmitter attached to the bottom of the lower tank measures the level of the liquid in this tank. The process is connected to a personal computer through a data acquisition board. The identification and control software is written under Matlab*, using the Real-Time Toolbox to collect data and to control the process. The sampling time is 2.5 seconds.*

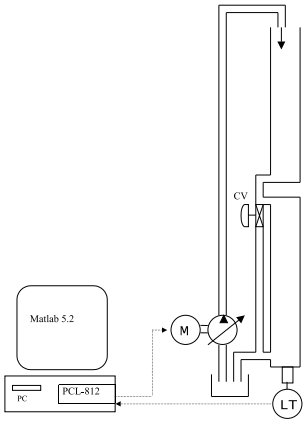


Figure 4.2: The cascaded-tanks setup.

The TS model was constructed from process measurements. The input variables of the fuzzy model were selected on the basis of prior knowledge about the process. Since the system can be modelled approximately as a second-order system with a time delay, and the source of the nonlinearity is the level-dependent outflow from the tank, the antecedent variable of the fuzzy model is chosen to be $\mathbf{z}(k) = y(k)$*. This results in the following TS fuzzy model structure:*

$$R_j: \textbf{ If } y(k) \textit{ is } A_{1,j} \textbf{ then} \qquad (4.19)$$
$$\hat{y}(k+1) = a_1^j y(k) + a_2^j y(k-1) + b_1^j u(k-2) + c^j \,.$$

Six triangular fuzzy sets were used on the antecedent universe $y(k)$*. Based on the range of the liquid level and after some manual optimization, the cores of the fuzzy sets were selected to be* $a_{1,j}$ *={ 0.0781, 0.2000, 0.4000, 0.5000, 0.6000, 0.8081}. The identification data set contains* $N = 1400$ *samples, using an input signal shown in Figure 4.3. In this example, the process input and output signals are plotted in percentage of their full range. The signal was designed to contain the important frequencies in the* expected *range of the process dynamics. Later on, we will see that the frequency content of this signal is not ideal, which is typical for real-life experiments.*

Three different models were identified:

- *Model 1: No prior knowledge was used during the identification.*
- *Model 2: Prior knowledge on process stability was used and the minimal and maximal steady-state gain were assumed:* $K_{\min} = 0$ *and* $K_{\max} = 2.5$*.*
- *Model 3: The second case was extended by the following constraints on the settling time of each submodel (Table 4.2).*

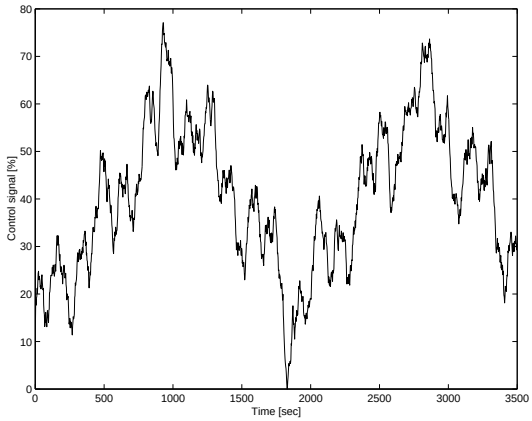


Figure 4.3: The training control sequence.

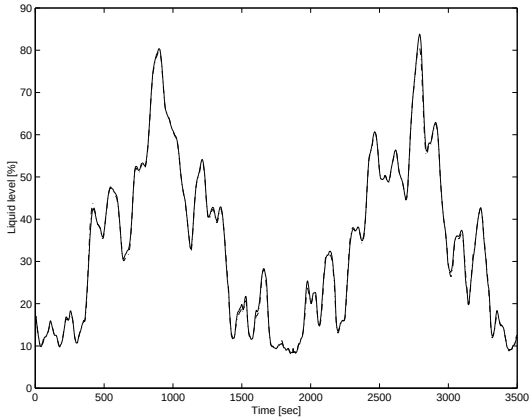


Figure 4.4: The measured (—) and simulated ($-\ -$) process output.

rule index	1	2	3	4	5	6
$\tau_{\min}$	15	40	60	70	80	80
$\tau_{\max}$	30	60	100	120	140	140

Table 4.1: Settling times of the local models.

The developed fuzzy models were validated by means of (recurrent) simulation on a separate validation data set. The simulated and the measured liquid levels are shown in Figure 4.4. The two curves can hardly be distinguished from each other.

The performance of the obtained models was measured by the variance accounted for (VAF) index.

Models 1 and 2 both achieve the performance VAF $=$99.89 and Model 3 has VAF$=$99.9. Clearly, there is no significant difference between the performance of the fuzzy models identified with and without the use of a priori knowledge. It is interesting to note that even though the modeling performance is almost unchanged, the consequent parameters of the three models differ considerably (Table 4.2). This effect is caused by the fact that the unconstrained global least-squares estimation method biases the estimate of the local model parameters, which may hamper the local interpretation of the TS fuzzy model, while the prior knowledge-based constraints regularize the model parameters. If the incorporated constraints are adequate, this regularization does not result in worse modeling performance. Moreover, if the identification data do not contain all information about the process (which is the usual case), the constraints can even improve the modeling performance.

Figure 4.5 shows three measured step responses of the process and the step responses of the fuzzy model at these operating points. The generated step responses clearly show the above-mentioned regularization effect of the constraints. One can see that by using more prior knowledge, more accurate local models can be identified from the global process data. The large deviation in the step response and the small difference in the modeling performance is mainly caused by the offset parameters, c^j, of the rules. By using the proposed constrained identification method, the offset parameters influence the steady-state behavior of the plant only (as one would normally expect). The reason why the constraints on the settling time give the most significant improvement is that the identification data do not contain enough information on low-frequency and steady-state behavior of the system. This is a typical real-life situation, as it is often difficult to design identification experiments that yield data with an appropriate balance between low and high frequencies. An advantage of the proposed method is that the information on low-frequency behavior can originate from other sources or separate experiments (step responses) and it can easily be integrated with the information coming from the dynamic data.

□

Model 1

	1	2	3	4	5	6
a_1^i	-0.2419	-0.0398	-0.0217	-0.0238	0.0375	0
a_2^i	-0.4630	-0.6574	-0.6499	-0.6470	-0.7032	-0.6680
b_1^i	0.0217	0.0691	0.0350	0.0392	0.0325	0.0384
c^i	0.1315	0.3112	0.6487	0.8094	0.9753	1.3141

Model 2

	1	2	3	4	5	6
a_1^i	1.1271	1.3291	1.3472	1.3452	1.4064	1.3689
a_2^i	-0.4630	-0.6574	-0.6499	-0.6470	-0.7032	-0.6680
b_1^i	0.0217	0.0691	0.0350	0.0392	0.0325	0.0384
c^i	0.0246	0.0374	0.1011	0.1250	0.1539	0.2079

Model 3

	1	2	3	4	5	6
a_1^i	1.3632	1.5495	1.6560	1.6134	1.6827	1.6422
a_2^i	-0.4646	-0.6047	-0.6815	-0.6407	-0.7049	-0.6668
b_1^i	0.0297	0.0718	0.0333	0.0394	0.0324	0.0384
c^i	0.0055	-0.0182	-0.0088	-0.0124	-0.0107	-0.0139

Table 4.2: Parameters of the local models obtained rby different identification methods.

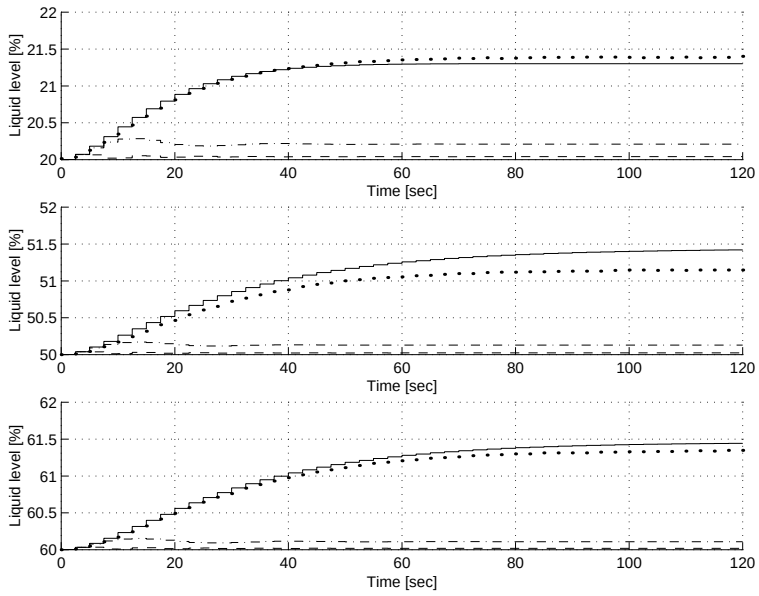


Figure 4.5: Step responses of the fuzzy models (- - model 1, - · model 2, — model 3) and the measured step response at the corresponding operating points (· · · · · ·).

4.2 Prior Knowledge based Spline Smoothing

In many practical situations, laboratory and industrial experiments are expensive and time consuming and accurate measurements cannot be made. This problem results in a small number of data points that are often noisy and obtained at irregular time intervals. Hence, data smoothening and re-sampling are often required to handle such data sets. A good method for this purpose is the cubic spline approximation [104, 105] that provides acceptable results and has practical relevance [106], e.g. it was used for the estimation of reaction kinetic models [107, 108].

The identification of reaction rates is an important problem, as chemical process models usually contain reaction networks, and the kinetic parameters of these models often cannot be identified on the basis of *a priori* knowledge alone. Therefore, process modeling is generally supported by experiments to identify the kinetic parameters. Many methods have been suggested to obtain reasonable estimates for these rate coefficients from experimental data [109]. For example, an interesting method for estimating rate constants of complex kinetic models from isothermal, batch or plug flow reactor data was described in the well-known paper of Himmelblau at al. [110]. This method, similarly to other approaches, is applicable when large numbers of data points along the concentration trajectories are available. If there are few data points, it is worth fitting individual splines for each measured variable and re-sampling the data based on these splines [106, 107, 108]. Unfortunately, the low number of the measured data points and the measurement noise also affect the quality of a spline approximation. Hence, there is a need for a new approach that can handle this problem.

The main goal of this section is to develop a multivariate spline approximation method that employs *a priori* knowledge in the approximation. It will be shown that *a priori* knowledge, e.g. assumed material balance or the visual inspection of the data, can be easily incorporated into the spline approximation. It should be noted that, though, this idea is used for the cubic spline approximation in this session, it can be applied to other approximation techniques too.

Standard Cubic Spline Approximation

Cubic splines are piecewise third-order polynomials. The polynomials are defined such that their values and first derivatives are continuous at so-called knots where the individual polynomials are interconnected. So when a cubic spline is identified, a continuous function is fitted to the available measured data, $\mathbf{x} = [x_1, \ldots, x_n]^T$ given at time instants $\mathbf{t} = [t_1, \ldots, t_n]^T$. An efficient way to calculate the coefficients of the cubic spline is given by Horiuchi [107].

To formulate this algorithm, let us define a cubic spline for a knot sequence: $t_1 = k_1 < k_2 < k_3 < \ldots < k_{n-1} < k_n = t_N$. The cubic spline is a sequence of cubic polynomials defined for each interval, $[k_1, k_2], [k_2, k_3], \ldots, [k_{n-1}, k_n]$, by the combination of the function-values and the first-order derivatives at the knots

$$S(t) = s'_i a_i(t) + s'_{i+1} b_i(t) + s_i c_i(t) + s_{i+1} d_i(t) \tag{4.20}$$

for $k_i \leq t < k_{i+1}$, where

$$s_i = S(k_i),\ s'_i = \frac{dS(t)}{dt}|_{t=k_i} \tag{4.21}$$

$$\begin{aligned} a_i(t) &= \tfrac{(k_{i+1}-t)^2(t-k_i)}{h_i^2},\ b_i(t) = -\tfrac{(k_{i+1}-t)(t-k_i)^2}{h_i^2} \\ c_i(t) &= \tfrac{(k_{i+1}-t)^2(2(t-k_i)+h_i)}{h_i^3},\ d_i(t) = \tfrac{(t-k_i)^2(2(k_{i+1}-t)+h_i}{h_i^3}, \end{aligned} \tag{4.22}$$

where $h_i = k_{i+1} - k_i$. As can be seen from (4.21), the spline is linear in the parameters

$$\theta = [s_1, s'_1, s_2, s'_2, \ldots, s_n, s'_n]^T. \tag{4.23}$$

Hence, the θ parameter vector can be determined by minimizing the following quadratic cost function

$$\min_\theta Q(\theta) \quad \text{where} \quad Q(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - S(t_i)\right)^2 = \|\mathbf{x} - \mathbf{\Psi}\theta\|_2, \tag{4.24}$$

where

$$\mathbf{\Psi} = \begin{pmatrix} c(t_1) & a(t_1) & d(t_1) & b(t_1) & 0 & 0 & \ldots & & & & 0 \\ c(t_2) & a(t_2) & d(t_2) & b(t_2) & 0 & 0 & \ldots & & & & 0 \\ & & & \vdots & & & \vdots & & & & \\ 0 & 0 & c(t_j) & a(t_j) & d(t_j) & b(t_j) & 0 & \ldots & & & 0 \\ & & & & \vdots & & \vdots & & & & \\ & & & & & \ddots & & & & & \\ 0 & 0 & \ldots & & & & & c(t_n) & a(t_n) & d(t_n) & b(t_n) \end{pmatrix}_{N\times 2n}. \tag{4.25}$$

This optimization problem can be solved analytically by the ordinary linear least squares (LS) method

$$\theta = \left(\mathbf{\Psi}\mathbf{\Psi}^T\right)^{-1}\mathbf{\Psi}^T\mathbf{x}. \tag{4.26}$$

The advantage of the utilized cubic spline is that the integral and derivative of the spline is also linear in the parameters of the spline, that is

$$\frac{dS(t)}{dt} = s'_i a'_i(t) + s'_{i+1} b'_i(t) + s_i c'_i(t) + s_{i+1} d'_i(t), \tag{4.27}$$

where

$$\begin{aligned} a'_i(t) &= \frac{(-2(k_{i+1}-t)(t-k_i) + (k_{i+1}-t)^2}{h_i^2} \\ b'_i(t) &= -\frac{2(k_{i+1}-t)(t-k_i) - (t-k_i)^2}{h_i^2} \\ c'_i(t) &= \frac{-2(k_{i+1}-t)(2(t-k_i)+h_i) + 2(k_{i+1}-t)^2}{h_i^3} \\ d'_i(t) &= \frac{2(t-k_i)(2(k_{i+1}-t)+h_i - 2(t-k_i)^2)}{h_i^3}. \end{aligned} \tag{4.28}$$

Simultaneous Spline Approximation

The basic idea of this section is the simultaneous spline-smoothing of multivariate data. In this case, the unknown parameter vector contains the parameters of l individual splines, $\tilde{\theta} = [\theta_1^T, \ldots, \theta_l^T]^T$, where $\theta_j = [s_{j,1}, s'_{j,1}, s_{j,2}, s'_{j,2}, \ldots, s_{j,n_j}, s'_{j,n_j}]^T$ is the parameter vector of j-th spline. The available measured data are arranged into vectors, $\tilde{\mathbf{x}} = [\mathbf{x}_1^T, \ldots, \mathbf{x}_l^T]^T$ and $\tilde{\mathbf{t}} = [\mathbf{t}_1^T, \ldots, \mathbf{t}_l^T]^T$, where $\mathbf{x}_j = [x_{j,1}, x_{j,2}, \ldots, x_{j,N_j}]^T$ is the data vector for j-th spline. (Note that N_j does not have to be identical to N_i if $i \neq j$). The $\tilde{\theta}$ parameter vector can also be estimated by minimizing the following quadratic cost function

$$\min_{\tilde{\theta}} Q(\tilde{\theta}),$$

where

$$Q(\tilde{\theta}) = \sum_{j=1}^{l} \left(\frac{1}{N_j} \sum_{i=1}^{N_j} \Big(x_{j,i} - S_j(t_{j,i}) \Big)^2 \right) = \|\tilde{\mathbf{x}} - \tilde{\mathbf{\Psi}}\tilde{\theta}\|_2. \tag{4.29}$$

This LS formulation allows for the incorporation of the available *a priori* knowledge into the simultaneous spline fitting procedure. Because the parameters of the splines are function values and function derivatives at the knots, *a priori* knowledge can be easily represented by equality and inequality constraints defined on the parameters of the model. It is advantageous to apply linear inequality and equality constraints, that is

$$\min_{\tilde{\theta}} Q(\tilde{\theta}) \quad \text{subject to} \quad \mathbf{G}\tilde{\theta} \leq \mathbf{g} \text{ and } \mathbf{H}\tilde{\theta} = \mathbf{h}, \tag{4.30}$$

as the obtained quadratic programming (QP) optimization problem has a global optimum, and can be effectively solved by standard numerical packages, such as MATLAB [111].

Such *a priori* knowledge can be based on the visual inspection of the data, for example, one might observe that the concentration of a component is monotonously decreasing or it can come from balance equations such as

$$\mathbf{W}\mathbf{x} = \mathbf{0}, \; \mathbf{V}\frac{d\mathbf{x}}{dt} = \mathbf{0} \tag{4.31}$$

The drawback of this hard-constrained method is that it constrains the values and the derivatives of the splines only at the knot points. To constrain the complete spline, a "soft-constrained" approach has been developed that handles equality constraints formulated as

$$\begin{aligned} A_{j,i} S_j(\hat{t}_i) &= b_{j,i} \\ C_{j,i} \frac{dS_j(\hat{t}_i)}{dt} &= d_{j,i}, \end{aligned} \tag{4.32}$$

where S_j is the j-th spline function and $\hat{t}_i$ is an arbitrary time instant used for the evaluation of the constrains, $\hat{\mathbf{t}} = [\hat{t}_1, ..., \hat{t}_{\hat{N}}]^T$. With the use of these additional

equations, the resulting LS estimation problem is formulated as

$$\begin{aligned} Q(\tilde{\theta}) &= \sum_{j=1}^{l}\left(\frac{1}{N_j}\sum_{i=1}^{N_j}\left(x_{j,i}-S_j(t_{j,i})\right)^2\right) \\ &+ \lambda_1\sum_{j=1}^{l}\left(\frac{1}{\hat{N}}\sum_{i=1}^{\hat{N}}\left(b_{j,i}-A_{j,i}S_j(\hat{t}_i)\right)^2\right) \\ &+ \lambda_2\sum_{j=1}^{l}\left(\frac{1}{\hat{N}}\sum_{i=1}^{\hat{N}}\left(d_{j,i}-C_{j,i}\frac{dS_j(\hat{t}_i)}{dt}\right)^2\right) \end{aligned} \tag{4.33}$$

$$Q(\tilde{\theta}) = \|\tilde{\mathbf{x}}-\tilde{\mathbf{\Psi}}\tilde{\theta}\|_2 + \lambda_1\|\tilde{\mathbf{y}}_1-\tilde{\mathbf{\Psi}}_1\tilde{\theta}\|_2 + \lambda_2\|\tilde{\mathbf{y}}_2-\tilde{\mathbf{\Psi}}_2\tilde{\theta}\|_2. \tag{4.34}$$

In general, several constraints can be defined in the form

$$Q(\tilde{\theta}) = \|\tilde{\mathbf{x}}-\tilde{\mathbf{\Psi}}_0\tilde{\theta}\|_2 + \sum_{i=1}^{p}\lambda_i\|\tilde{\mathbf{y}}_i-\tilde{\mathbf{\Psi}}_i\tilde{\theta}\|_2, \tag{4.35}$$

where the λ_1, λ_1, ..., λ_p regularization parameters determine the weight of the soft-constraints. There is no general recipe for the design of these regularization parameters. In this session, a heuristic approach is followed, where the parameters are set according to the modeler's belief in the accuracy and the importance of the corresponding constraints. Certainly, the soft-constrained method can be combined with hard-constraints, for example, by using the soft constrains for regularizing (smooth) the spline, while using the hard-constraints to incorporate the *a priori* knowledge into the parameter identification.

Because the spline approximation is black-box modeling approach, when *a priori* knowledge is incorporated into the identification of the parameters, a grey-box modeling scheme is obtained. Hence, the above-presented method can be called as grey-box spline approximation.

Two examples are presented to demonstrate the applicability of the above-mentioned method. In the first example, the proposed approach is used to identify the kinetic parameters of a simulated reaction network, whereas in the second example it is applied to the analysis of data taken from an industrial batch reactor.

Example 4.3. Estimation of Kinetic Rate Constants

In this example, the proposed spline-smoothing approach is used to identify the rate constants of kinetic models from concentration profiles. The assumed reaction network is given by the following equation

$$\frac{dC_i}{dt} = \sum_{j=1}^{n} w_{i,j} k_j^r R_j, \tag{4.36}$$

where t is the time or a time-like variable, $w_{i,j}$ is the stoichiometric coefficient for component i in reaction j, k_j^r is the rate constant for reaction j, and the factors R_j are functions of the concentrations. Using t_1 as the initial time, (4.36) *can be integrated to yield*

$$C_{i,k} - C_{i,1} = \sum_{j=1}^{n} w_{i,j} k_j^r R_{j,k}, \tag{4.37}$$

where

$$R_{j,k} = \int_{t_1}^{t_k} R_j dt. \tag{4.38}$$

If the $C_{i,k}$ values are known and $R_{j,k}$ values can be estimated, (4.37) *becomes a system of linear equations in the rate constants k_j^r. Hence, the rate constants can be estimated by ordinary least squares method, assuming that the number of independent equations is not less than the number of rate constants. When the concentration trajectories are frequently sampled,* (4.38) *can be integrated by numerical quadrature, using data points [110]. This scheme, however, is no longer available when the number of data points is small, due to lack of precision. In this case, a spline-smoothing procedure should be followed. E.g., in the paper of Tang [112], a natural cubic spline was fitted for each R_j.*

In this example a new approach will be presented where the splines of the concentration profiles are simultaneously fitted and constraints based on the component balance equations are incorporated to ensure the consistency of the parameters of the estimated splines.

The numerical example is taken from Himmelblau, et al. [110] and Tang [112], and involves the following isothermal reactions

$$A \leftrightarrow B \leftrightarrow C \tag{4.39}$$

described by the following differential equations

$$\begin{aligned}
\frac{dC_A(t)}{dt} &= -k_1^r C_A(t) + k_2^r C_B(t) \\
\frac{dC_B(t)}{dt} &= k_1^r C_A(t) - k_2^r C_B(t) - k_3^r C_B(t) + k_4^r C_C(t) \\
\frac{dC_C(t)}{dt} &= k_3^r C_B(t) - k_4^r C_C(t).
\end{aligned} \tag{4.40}$$

The initial concentrations at $t_0 = 0$ are $C_A = 1$, $C_B = 0$ and $C_C = 0$, and the real parameters are $k_1^r = 1$, $k_2^r = 0.5$, $k_3^r = 10$, $k_4^r = 5$.

The aim of the experiment is to identify the kinetic parameters $(k_1^r, \ldots, k_4^r)$ *from the available concentration measurements. According to the quality and the quantity of the available data, 10 data sets were used to demonstrate the performances of different methods (see Table 4.3). These data sets were defined from combinations of different*

- *number of data points (the data points are taken at 12 or 31 time instants),*
- *level of normal distributed noise added to the measured data (0%, 5% or 10%),*
- *type of noise (independent or correlated).*

Table 4.3: Data sets

data sets	number of data points	relative noise level (%)	independent noise
1	31	0	–
2	31	5	yes
3	31	5	no
4	31	10	yes
5	31	10	no
6	12	0	–
7	12	5	yes
8	12	5	no
9	12	10	yes
10	12	10	no

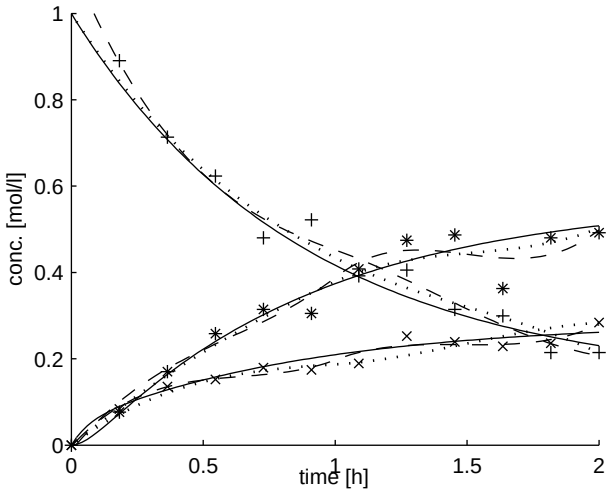


Figure 4.6: Spline approximations for 12 noisy measurements (Data Set 9) plus sign: measured conc. of A, asterisk: measured conc. of B, cross: measured conc. of C, solid line: real concentration trajectory, dashed line: spline estimation with Model 3., dotted line: spline approximation with Model 4 (proposed model)

Four different approaches were followed for the estimation of the kinetic parameters from these data sets:

Method 1. Nonlinear direct search method. *The Nelder-Mead simplex algorithm is used for the identification of the kinetic parameters. The concentration trajectories are calculated by solving the differential equation of the model* (4.40)*. This method is very effective, but it is rather slow.*

Method 2. Raw data based estimation. *In this case, the raw data points are integrated numerically to estimate the $\tilde{R}_{j,k}$ values and the estimation problem is solved by the least squares method.*

Method 3. Spline based estimation. *The standard cubic spline approximation is used, and the smoothed concentration trajectories represented by the splines are analytically integrated to estimate the $\tilde{R}_{j,k}$ values needed by the least squares method.*

Method 4. Grey-box spline based estimation. *This approach is similar to Model 3., but it utilizes the following prior-knowledge-based constraints*

$$\begin{aligned} C_A + C_B + C_C &= 1 \\ \frac{dC_A}{dt} + \frac{dC_B}{dt} + \frac{dC_C}{dt} &= 0 \\ \frac{dC_A}{dt} < 0,\ \frac{dC_B}{dt} > 0,\ \frac{dC_C}{dt} &> 0\,, \end{aligned} \tag{4.41}$$

where the equality constraints represent the component balance equations, and the inequality constraints describe the monotonic changes of the concentrations (see Figure 4.6).

In the cubic spline approximation, the choice of the knot sequence plays an important role. These parameters can be determined automatically by segmenting the time-series defined by the concentration profiles [113]; heuristically based on the visual inspection of the variables by looking for significant breaks in the concentration trajectories; or intuitively by using some a priori *knowledge about the system to be modeled (see the second example Section 4.4). In this example, three equidistant knot sequences with two, three and four knots were investigated and compared. Splines with more than four knots were not identified, because when the number of data points is 12 (Data set 6-10) too few data points would fall in the knot-intervals which would make the estimation problem to be ill-conditioned.*

The performances of the models were measured by the following normalized cost function

$$P_1 = \sqrt{\sum_{i=1}^{4} \left(\frac{k_i^r - \hat{k}_i^r}{k_i^r} \right)^2}\,, \tag{4.42}$$

where $\hat{k}_i^r$ represents the estimated kinetic parameters and k_i^r represents the real parameters.

Method 1, 3, and 4 can also be evaluated in terms of another cost function, which measures the error between the "real" and the estimated concentration trajectories

$$P_2 = 1000 \sum_{i=1}^{3} \int_0^{t_N} \big(C_i(\tau) - \hat{C}_i(\tau) \big)^2 d\tau. \tag{4.43}$$

Tables 4.4 and 4.5 show that the spline based methods (Method 3 and 4) estimate the rate coefficients more accurately than the raw-data based method (Method 2). When the constrained spline approximation is applied (Method 4), not only the estimated rate coefficients are much more accurate, but also better

Table 4.4: Estimation of kinetic parameters $(P_1)^a$

		data sets								
M[b]	#k[c]	1	2	3	4	5	6	7	8	9	10
1		0.0023	0.18	0.088	0.37	0.18	0.060	0.30	0.16	0.65	0.31
2		0.024	0.38	0.32	0.75	0.64	0.17	0.42	0.36	0.76	0.64
3	4	0.035	0.34	0.29	0.68	0.58	0.064	0.40	0.34	0.78	0.67
	3	0.12	0.33	0.26	0.62	0.46	0.15	0.41	0.36	0.78	0.64
	2	0.23	0.34	0.28	0.57	0.38	0.33	0.53	0.42	0.94	0.59
4	4	0.029	0.19	0.11	0.39	0.23	0.064	0.31	0.20	0.67	0.41
	3	0.13	0.24	0.16	0.44	0.25	0.15	0.34	0.24	0.68	0.41
	2	0.42	0.49	0.42	0.72	0.46	0.45	0.67	0.48	1.4	0.62

[a] *Means of 100 independent runs*
[b] *Model*
[c] *Number of knots for splines*

Table 4.5: Estimation of concentration trajectories $(P_2)^a$

		data sets								
M[b]	#k[c]	1	2	3	4	5	6	7	8	9	10
1		1.5e-6	0.041	0.011	0.016	0.043	0.003	0.11	0.033	0.43	0.12
3	4	0.003	0.29	0.30	1.2	1.2	0.005	0.77	0.76	3.1	3.0
	3	0.016	0.23	0.22	0.86	0.82	0.028	0.56	0.54	2.2	2.1
	2	0.036	0.18	0.18	0.62	0.61	0.065	0.42	0.38	1.5	1.3
4	4	0.004	0.16	0.075	0.59	0.27	0.005	0.43	0.19	0.16	0.70
	3	0.035	0.13	0.066	0.40	0.016	0.035	0.31	0.13	1.1	0.39
	2	0.12	0.18	0.13	0.35	0.18	0.12	0.27	0.16	0.69	0.28

[a] *Means of 100 independent runs*
[b] *Model*
[c] *Number of knots for splines*

approximation of the concentration trajectories are obtained compared to the unconstrained spline approximation (Method 3). The constrained spline estimation method (Method 4) always estimates the rate coefficients better than the simple LS method (Method 2). The nonlinear method (Method 1) offers the best estimation of rate coefficients but this method is much slower than other methods (see Table 4.6). Furthermore, the nonlinear method is based on the differential equations of the process. Hence, in contrast to the spline approximation, this method requires full knowledge of the differential equations. Therefore, if only the approximation of the concentration trajectories is the modeler's purpose, the application of the proposed spline approach is suggested. Furthermore, for the estimation of the kinetic parameters, the spline-based approach can be used to provide a good initial condition for nonlinear method as it was suggested in [112].

When the measurement noise was correlated with each other (data sets 3,5,8 and 10), usually better estimations were obtained compared to the uncorrelated case (data sets 2,4,7 and 9). This suggests that the concentration

Table 4.6: Computational effort in the identification of the models[a]

model	time (s)
1	12
2	0.0069
3	0.65
4	0.67

[a] *One typical run on AMD Duron 800-MHz computer*

trajectories with correlated noise are less conflicting to each other and to the measured data, thus the constraints can more easily handle such measurement errors compared to the totally random uncorrelated case.

In general, an increase of the number of knots improves the accuracy of the approximation. However, similarly to other estimation problems, overfitting can be occurred, so the number of the parameters (number of knots) has an optimal value. As Tables 4.4 and 4.5 show, this optimum is the function of the noise level.

□

Example 4.4. Estimation of Concentration Profiles for Industrial Batch Reactor

The second example is taken from one of the industrial partner of the Department of Process Engineering. In the studied industrial batch reactor only the concentrations of the three main components are measured (see Figure 4.7). As it will be shown in this example, the proposed grey-box spline smoothing approach can be effectively used in such typical real-world situations. Two approaches were used and compared: Standard cubic spline approximation and simultaneous spline approximation with hard- and soft-constraints.

Because the chemistry of this technology is very complex and not fully known, the reaction network is modeled by the following scheme

$$\begin{aligned} A + B &\rightarrow C \\ A &\rightarrow A^* \\ B &\rightarrow B^* \\ C &\rightarrow C^* \end{aligned} \tag{4.44}$$

where the first reaction represents the product formation, and last three reactions represent the decay of three measured components (by-product formation). Due to the fact that the technology is confidential, the components are denoted as A, B *and* C. *As it can be seen in Figure 4.7, the rate of the reactions are relatively small until the 60th minutes of operation. At about this time, a catalyst was added into the reactor that makes the product formation much faster. This* a priori *knowledge is used at the selection of the knot sequence. Several alternatives were tested with more and fewer number of intervals and different knot*

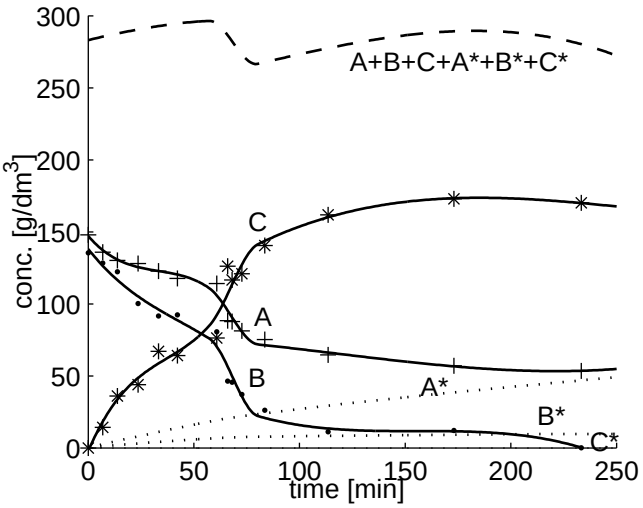


Figure 4.7: Spline approximation (second example) without constraints

sequences, and finally, based on the recipe of the technology and the analysis of the shapes of the trajectories the following knot sequence was set as [0, 57.3, 80.3, 250].

From the visual inspection of the data (see Figure 4.7, the following constraints can be defined

$$C_A(0) = C_{A0},\ C_B(0) = C_{B0},\ C_C(0) = C_{C0} \tag{4.45}$$

and

$$\frac{dC_A}{dt} \leq 0,\ \frac{dC_B}{dt} \leq 0 \tag{4.46}$$

$$\frac{dC_C}{dt} \geq 0 \quad \text{if } t \leq 114 \tag{4.47}$$

$$\frac{dC_A}{dt} + \frac{dC_B}{dt} + \frac{dC_C}{dt} \leq 0. \tag{4.48}$$

The component balance can be formulated as

$$C_0 = C_A + C_B + C_C + C_{A^*} + C_{B^*} + C_{C^*}, \tag{4.49}$$

where $C_0 = C_{A0} + C_{B0} + C_{C0}$.

Although the A^*, B^* *and* C^* *concentrations are not measured, with the use of the assumed kinetic models*

$$\begin{aligned} \frac{dC_{A^*}(t)}{dt} &= k_1^r C_A(t) \\ \frac{dC_{B^*}(t)}{dt} &= k_2^r C_B(t) \\ \frac{dC_{C^*}(t)}{dt} &= k_3^r C_C(t), \end{aligned} \tag{4.50}$$

the material balance (4.49) *can be used in practice in the following form*

$$C_0 = C_A(t) + C_B(t) + C_C(t) + k_1^r \int_0^t C_A(\tau)d\tau +$$

$$k_2^r \int_0^t C_B(\tau)d\tau + k_3^r \int_0^t C_C(\tau)d\tau. \tag{4.51}$$

(4.51) *is linear in the parameters* k_1^r, k_2^r *and* k_3^r *and linear in the* $\tilde{\theta}$ *parameters of the splines. Hence, as the concentration trajectories are represented by the* S_1, S_2 *and* S_3 *splines, in the last three terms of* (4.51) *the product of these parameters appear. Hence, this material balance represents a bilinear constraint defined on the unknown parameters. Such bilinear optimization problems are often solved by alternating optimization (see [114] for a practical application in system identification), this approach results in the following algorithm:*

Step 1. Hard-constrained spline identification. *The* $C_A(t)$, $C_B(t)$, $C_C(t)$ *concentration trajectories are represented by the* S_1, S_2, S_3 *spline functions obtained by simultaneous spline approximation with* (4.45)-(4.48) *hard-constraints.*

Step 2. Identification of the k_1, k_2, k_3 ***parameters.*** *The three kinetic parameters are identified by linear least squares algorithm from* (4.51)*:*

$$\min_{k_1^r, k_2^r, k_3^r} Q(k_1^r, k_2^r, k_3^r),$$

where

$$Q(k_1^r, k_2^r, k_3^r) = \sum_{i=1}^{\hat{N}} \left(C_0 - \left(\sum_{j=1}^{3} S_j(\hat{t}_i) + \sum_{j=1}^{3} k_j^r \int_0^{\hat{t}_i} S_j(\tau)d\tau \right) \right)^2. \tag{4.52}$$

Step 3. Soft- and Hard-constrained spline identification. *In this step, one incorporates the material balance* (4.51) *into the spline identification as a soft-constraint, so the new quadratic objective function is:*

$$\min_{\tilde{\theta}} Q(\hat{\theta}),$$

where

$$Q(\tilde{\theta}) = \sum_{j=1}^{3} \left(\frac{1}{N_j} \sum_{i=1}^{N_j} \left(x_{j,i} - S_j(t_{j,i}) \right)^2 \right)$$
$$+\lambda \frac{1}{\hat{N}} \sum_{j=1}^{\hat{N}} \left(C_0 - \left(\sum_{j=1}^{3} S_j(\hat{t}_i) + \sum_{j=1}^{3} k_j^r \int_0^{\hat{t}_i} S_j(\tau)d\tau \right) \right)^2. \tag{4.53}$$

Because $S_j(t)$ *and* $\int_0^t S_j(\tau)d\tau$ *are linear functions of the* $\tilde{\theta}$ *parameters and* $\min_{\tilde{\theta}} Q(\tilde{\theta})$ *is subject to the constraints* (4.45)-(4.48)*, the optimization problem can be solved by quadratic programming. The results of this step are the* S_1, S_2, S_3 *splines related to the* $C_A(t)$, $C_B(t)$ *and* $C_C(t)$ *concentration trajectories.*

Step 4. Iteration. *Repeat from step 2 until the estimated trajectories converge.*

Before the application of the soft-constraint, the $\hat{\mathbf{t}} = [\hat{t}_1, \ldots, \hat{t}_{\hat{N}}]$ *time instants used for the evaluation of the soft-constraints and the* λ *parameter have to be selected. In this example, the soft-constrains were evaluated in every minutes of the operation, and the* λ *parameter was set according to a rough sensitivity analysis. An increase in* λ *improved the accuracy of the mass balance. When* λ *was greater than 15, this increase did not cause further significant improvement, thus this parameter was chosen as 15 (see Figure 4.8).*

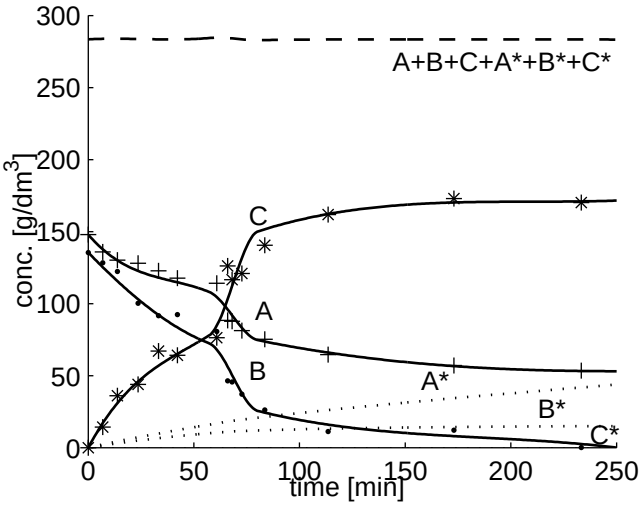


Figure 4.8: Spline approximation (second example) with constraints

Table 4.7 compares the numerical results of the two methods. The proposed constrained method fulfils the mass balance very well, in contrast to the standard spline approximation method. Certainly, the spline that was calculated by the standard method fits to the measured data better, but this difference is quite small. Hence, these numerical results support the conclusion that the proposed method is more accurate.

Table 4.7: Numerical results of spline methods in the second example

	without constraints	with constraints
mean absolute difference between estimated and measured concentrations $(\mathrm{g}/\mathrm{dm}^3)$	5.29	5.99
mean absolute error in the mass balance $(\mathrm{g}/\mathrm{dm}^3)$	6.36	0.18
absolute error in the balance at the end of the experiment $(\mathrm{g}/\mathrm{dm}^3)$	10.50	0.08

□

4.3 Conclusions

Fuzzy model identification is an effective tool for the approximation of uncertain nonlinear systems on the basis of measured data[1]. I developed a novel approach to data-driven identification of Takagi-Sugeno fuzzy models that allows to translate prior knowledge about the process (including stability, minimal or maximal static gain and settling time) into constraints on the model parameters[2]. This grey box fuzzy model approach allows the development of models also in cases where little experimental data are available. I have shown that the concept can be applied to fuzzy models with multivariate membership functions[3], and fuzzy (Hammerstein) models built on the basis of data combined with prior knowledge perform better in control than models obtained from data only[4].

In many practical situations, the involvement of laboratory and industrial experiments are expensive and time consuming and accurate measurements cannot be made. This problem results in a small number of data points that are often noisy and obtained at irregular time intervals. Hence, data smoothing and re-sampling are often required to reduce the effect of measurement noise and irregular time intervals. Typically, an interpolation method is used for this purpose, e.g. cubic spline interpolation, but the disadvantage of the common interpolation methods is that they can not utilize any a priori information. Hence, I developed a new cubic spline interpolation approach which utilizes a priori knowledge, e.g. material balance, or prior information about the measured properties. The methodology has been demonstrated through the investigation of a simulated and an industrial chemical reactor that the new method improves the accuracy of the data-driven estimation of kinetic parameters.[5]

---

[1] Abonyi J, Babuska R, Local and global identification and interpretation of parameters in Takagi-Sugeno fuzzy models, IEEE International Conference on Fuzzy System, 9th IEEE International Conference on Fuzzy Systems, Independent citations: 8

[2] Abonyi J, Babuska R, Verbruggen H B, Szeifert F, Incorporating prior knowledge in fuzzy model identification, INTERNATIONAL JOURNAL OF SYSTEMS SCIENCE 31: pp. 657-667. (2000), IF: 0.290, Independent citations: 16

[3] Abonyi J, Babuska R, Szeifert F, Fuzzy modeling with multivariate membership functions: Gray-box identification and control design, IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART B-CYBERNETICS 31: pp. 755-767. (2001), IF: 0.789, Independent citations: 27

[4] Abonyi J, Babuska R, Botto M A, Szeifert F, Nagy L, Identification and control of nonlinear systems using Fuzzy Hammerstein models, INDUSTRIAL & ENGINEERING CHEMISTRY RESEARCH 39: pp. 4302-4314. (2000), IF: 1.294 , Independent citations: 33

[5] Madar J, Abonyi J, Roubos H, Szeifert F, Incorporating prior knowledge in a cubic spline approximation-application to the identification of reaction kinetic models, INDUSTRIAL & ENGINEERING CHEMISTRY RESEARCH 42: pp. 4043-4049. (2003), IF: 1.317, Independent citations: 10

Chapter 5

Improvement of Polupation based Optimization Algorithms

5.1 Genetic Programming for System Identification

Because nonlinear dynamical models play very important role in process engineering, it is important to deal with the structure identification of these models. One of the most preferred structure identification method is Genetic Programming (GP) which is a data-driven symbolic optimization algorithm. Genetic Programming selects potential solutions from a given space of possible structures using evolutionary technique to find a minima (or maxima) of a given cost function. Although GP is an effective algorithm for the identification of model structures, it tends to generate overparameterized models when it is used for structure identification of dynamical models. Because the model transparency is very important from the aspect of practical usefulness, it is important to find a balance between accuracy and model transparency. Relatively little research has been done on this problem. The main goal of this section is to propose a method that eliminates superfluous model terms during structure identification in such a way that the model preserves its accuracy.

To improve structure identification, this session suggests the application of OLS in Genetic Programming. During the operation of GP, the algorithm generates a lot of potential solutions in the form of tree structures. These trees may have better and worse terms (subtrees) that contribute more or less to the accuracy of the model. The concept is the following: first, the trees (the individual members of the population) are decomposed to subtrees (function terms of the linear-in-parameters model) in such way that presented in Section B.4; then, the *err*'s of these function terms are calculated; finally, the less significant term(s) is/are eliminated. This "tree pruning" method is realized in every fitness evaluation before the calculation of fitness value of the tree. The main goal of this approach is to transform the trees to simpler trees that are more transparent, but whose accuracies are close to those of the original trees. Because a further goal is to preserve the original structure of trees as much as possible (because GP works with tree structures), the decomposition of the trees was based on the algorithm presented in Section B.5.

This method always guarantees that the elimination of one or more function terms of the model can be done by only "pruning" of the corresponding subtrees, so there is no need for structural rearrangement of the tree after this operation.

Example 5.1. *Let us consider a simple example that illustrates how the proposed method works. In this example the model, which must be identified, is* $y(k) = 0.8u(k-1)^2 + 1.2y(k-1) - 0.9y(k-2) - 0.2$. *The GP algorithm found the following solution with four terms after a few iteration steps:* $u(k-1)^2$, $y(k-1)$, $y(k-2)$, $u(k-1) * u(k-2)$; *Figure 5.1 illustrates the tree structure. Table 5.1 shows the calculated* err*'s for the function terms and the MSE of the linear-in-parameters model represented by this tree (the parameters were determined by LS method). Then the subtree that had the least* err *(*$F_4 = u(k-1) * u(k-2)$*) was eliminated from the tree. After that, the* err*'s and MSE values (and model parameters) were calculated again. The results show that the new model has a little higher MSE but a more adequate structure.*

Table 5.1: OLS example

	Before pruning	After pruning
$[err]_1$	0.9170	0.7902
$[err]_2$	0.0305	0.1288
$[err]_3$	0.0448	0.0733
$[err]_4$	0.0002	–
MSE	0.558	0.574

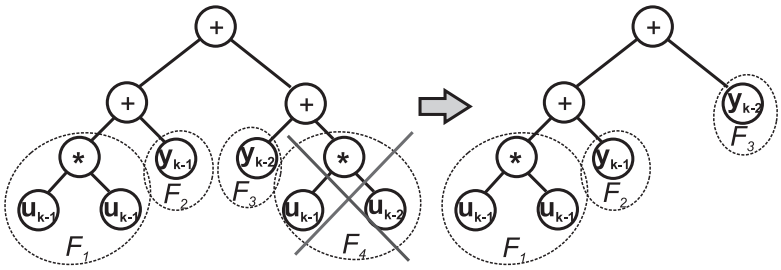


Figure 5.1: Tree pruning with OLS

□

The proposed approach has been implemented in MATLAB. The aim of the toolbox is the data-based identification of static and dynamic models. This toolbox is suitable for structure identification of dynamical input-output linear-in-parameters models, but it can also be applied for static nonlinear equation discovery. Special attention was given to the identification of dynamical models at the development of the toolbox, hence the generated model equations can be simulated to get both one- and n-step ahead predictions.

Genetic Programming has some parameters which must be adjusted. Table 5.2 shows the default parameters of the toolbox selected based on a set of trial-and-error experiments.

Table 5.2: Parameters of GP in the application examples

Population size	50
Maximum number of evaluated individuals	2500
Type of selection	roulette-wheel
Type of mutation	point-mutation
Type of crossover	one-point (2 parents)
Type of replacement	elitist
Generation gap	0.9
Probability of crossover	0.5
Probability of mutation	0.5
Probability of changing terminal – non-terminal nodes (vica versa) during mutation	0.25

Because polynomial models play an important role in process engineering, there is an option for generating polynomial models in the toolbox. If this option is selected, the set of operators is defined as $F = \{+, *\}$, and after every mutation and crossover, the algorithm validates whether the model structure is in a class of polynomial models. If necessary, the algorithm exchanges the internal nodes that are below a '$*$'-type internal node to '$*$'-type nodes. This simple trick transforms every non-polynomial model into a well-ordered polynomial model.

The OLS evaluation is inserted into the fitness evaluation step. The algorithm calculates the *err*'s of the branches of the tree before calculating of fitness of the tree. The terms (branches of the tree) that have *err*'s below a certain threshold value ($[err]_{limit}$) are eliminated from the tree. Then, the resulted individual proceeds on its way in the classical GP algorithm (fitness evaluation, selection, etc.)

Example 5.2. Continuous Polymerization Reactor

In this example the application of proposed GP-OLS method is illustrated at the identification of the model order of a continuous polymerization reactor used in Example **??**. *In this experiment, a perfect model structure does not exist, but an appropriate model order is known [88]. To estimate the model structure, 960 data points were generated by computer simulation. Four methods were compared:*

- *Method 1 generates all of the possible polynomials with degree $d = 2$. The model consists of all of these terms.*
- *Method 2 generates all of the possible polynomials with degree $d = 2$, but the model only consists of the terms which have greater error reductions ratios than 0.01.*
- *Method 3 is the polynomial GP-OLS method. The operator set is $F = \{*, +\}$. The OLS threshold value is 0.02.*

- *Method 4 is the non-polynomial GP-OLS method. The operator set is* $F = \{*, +, /, \sqrt{}\}$*. The OLS threshold value is 0.02.*

Table 5.3 shows the MSE of the obtained models for one-step and free-run predictions. Because GP is a stochastic algorithm, 10 identical experiments were performed for the third and fourth methods, and the table contains the minimum, maximum and mean results of these experiments. The input and output orders were constrained to four: $T = \{u(k-1), \cdots, u(k-4), y(k-1), \cdots, y(k-4)\}$.

Table 5.3: Results of Example II.

	Free-run MSE[a]			One-step-ahead MSE[a]		
	min	mean	max	min	mean	max
Method 1	-	∞	-	-	7.86	-
Method 2	-	26.8	-	-	30.3	-
Method 3	1.65	10.2	23.7	1.66	9.23	22.1
Method 4	0.95	7.15	20.6	0.84	6.63	17.8

[a] *MSE in* 10^{-3}

In the first method, the model consisted of 45 polynomial terms ($m = 8$ *and* $d = 2$*). This model was very accurate for the one-step ahead prediction, but it was unstable in the free-run prediction. So this model cannot be used in free-run simulation.*

In the second method, the err*'s were calculated for the 45 polynomial terms, and the terms that have very small values (below 0.01) were eliminated. After that, only three terms remained:*

$$u(k-1), y(k-1), y(k-2).$$

All of the bilinear terms were eliminated by OLS. This model is a simple linear model, which is stable in free-run, but its performance is quite weak.

The third method resulted in different models in the 10 experiments, because of stochastic nature of GP. All of resulting models were stable in free-run simulation. The most accurate model contained the the terms:

$$u(k-1)u(k-1), y(k-1), u(k-2), u(k-1)y(k-1);$$

which is the correct model order. This method found the correct model order in 6 cases out of 10 cases.

The fourth method (GP-OLS) resulted in correct model orders in 3 out of 10 cases. This method found the most accurate model, and all of the obtained models were stable in free-run simulation. Some of the obtained models were the same that was generated by the third method. Statistically, this method generated the most accurate models, but the third method was better at finding the correct model order.

□

5.2 Interactive Optimization

System identification, process optimization and controller design are often formulated as optimization problems. The key element of the formalization of optimization problems is defining a cost function. Cost function is a mathematical function which represents the objectives of the expected solution, and the goal of optimization is usually to find the minima (or the maxima) of this function. However the cost function is explicitly/mathematically not available in some cases. Sometimes, the relationship among the design variables and objectives is so complex that the cost function cannot be defined, or even there is no point in defining a quantitative function. In this kind of situation, it is very difficult to apply the common optimization methods. This session presents an interactive optimization approach that allows process engineers to effectively apply their knowledge during the optimization procedure without the explicit formalization of this knowledge.

Interactive optimization needs a special optimization algorithm. Most optimization techniques which work by improving a single solution step by step are not well-suited for this approach, because one cannot use the gradient information of the psychological space of the user. The population based optimization algorithms seem to be more plausible. The population based algorithms can be easily utilized for interactive optimization, e.g. by replacing the fitness function by a subjective evaluation. EA is especially ideal for interactive optimization, since the user can directly evaluate the fitness of the potential solutions, for example, by ranking them. This approach become known as Interactive Evolutionary Computation (IEC). In general, IEC means an EA in which the fitness evaluation is based on subjective evaluation. In most works, it means the fitness function is simply replaced with a human user who ranks the individuals [115]. For example, he/she gives marks such as "good", "acceptable", "non-acceptable". The subjective rates given by the user are transformed to fitness values (or directly applied as fitness values) for the algorithmic part of IEC.

This approach is simple, but I developed another approach for process engineering problems. In the proposed approach there is not fitness evaluation, the selection and replacement operators are replaced by human intervention , see Figure 5.2.

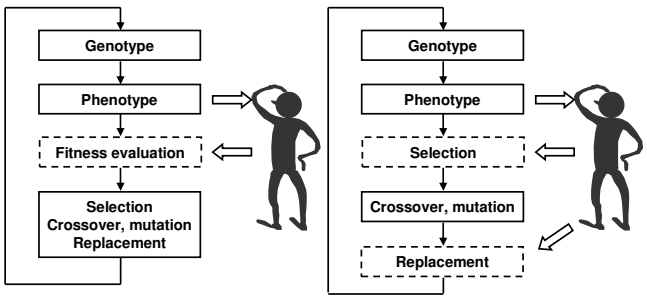


Figure 5.2: Classical IEC (left) and proposed IEC (right)

In contrast to the high number of IEC application examples, this approach has not been applied in process engineering so far. The interfacing of human ability with machine computation requires resolving some issues. This is especially relevant problem in process engineering where the user should evaluate the performances of simulated solutions obtained on the basis of different models. This section presents an IEC framework which was developed for process engineering problems in MATLAB environment. The `EAsy-IEC Toolbox` was designed to be applicable for different types of optimization problems, e.g.: system identification, controller tuning, process optimization. Thanks to the MATLAB environment, the toolbox can be used easily for various problems. In the developed toolbox, the user can analyze individuals based on the output of the target systems realized by the individuals. For example, the user can simultaneously analyze the numerical results with the plotted trajectories, profiles, etc. The number of displayed individuals is usually around seven, because it can be displayed spatially. Based on this visual inspection of the solutions and the analysis of some calculated numerical values and parameters, the user selects individuals that are used to formulate the next generation, and selects individuals that are replaced by offsprings, i.e. see Figure 5.3. The number of searching generations is limited to twenty generations due to the fatigue of human users.

In process engineering, one of the oldest optimization method is the heuristic method of trial-and-error. So the developed toolbox also allows *active human intervention* in which the user is able to modify individuals directly. This approach suits to process engineering problems more than visualised IEC or on-line knowledge embedding [115].

Because the population size is very small in IEC, the effective realization of IEC needs an EA that is able to search for a goal with a small population size within a few number of searching generations. Therefore Evolutionary Strategy was chosen, which was developed for small population size. Certainly, the application of ES for IEC involves some modification of ES. In the toolbox, a human-machine interface replaces selection and replacement operators. It should be noted that the user is able to use either the (μ, λ) or the $(\mu + \lambda)$ replacement strategy, since he/she determines which individuals can survive.

Example 5.3. MPC Controller Tuning

In this example the tuning of the parameters of a Model Predictive Control (MPC) controller is considered. The controller controls a binary distillation column which is a Multiple Input Multiple Output (MIMO) process with two manipulated inputs, reflux-flow and reboiler-flow, and two outputs, top product purity and bottom product purity. The column has two other non-manipulated input, feed rate and feed composition. When the column operates in a wide range, its characteristic are strongly nonlinear, especially towards high purity. (Certainly, the aim is to produce high-purity products at both ends of the column). In order to simulate the controlled process, this example relies on the model published by Skogestad [116]. Two disturbances were applied throughout the experiment: feed rate and feed composition disturbances.

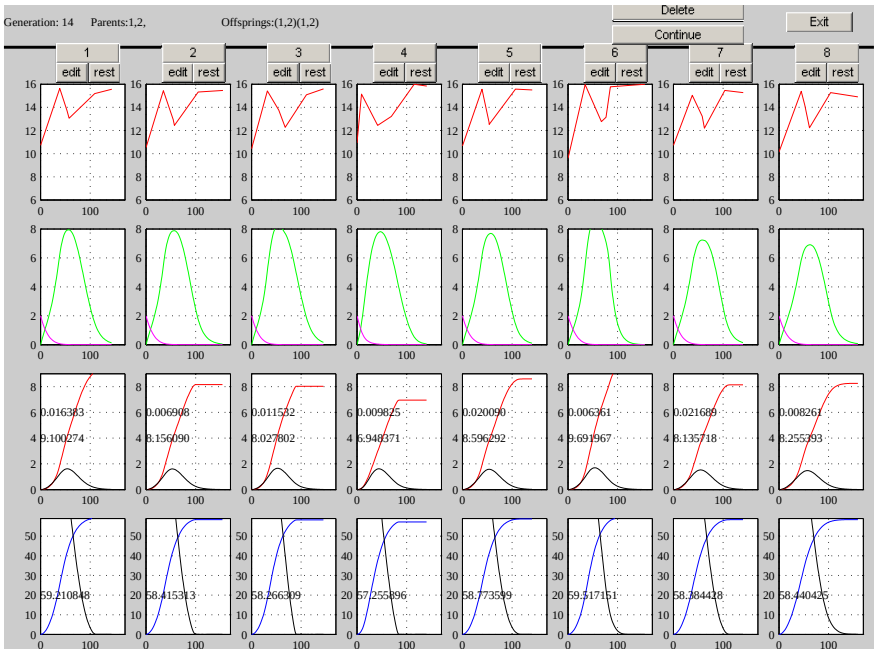


Figure 5.3: Interactive display of IEC toolbox

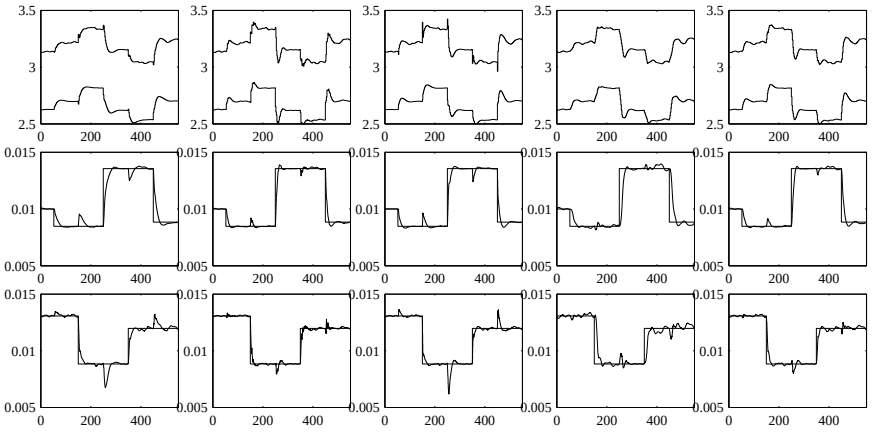


Figure 5.4: MPC controller tuning with Interactive Evolutionary Strategy
first row: manipulated inputs, second row: controlled top purity and set-point, third row: controlled bottom purity and set-point, fourth row: design variables

These disturbances were simulated by uniformly distributed white noise signal: 10% relative noise was added to the feed rate and 5% to the composition, because such disturbance levels are common under industrial conditions [116]. Because the controlled process is inherently nonlinear, nonlinear control algorithm should be utilized to achieve good control performance. Hence, according to [100], a Fuzzy Model Predictive Control was used in this example. The aim of MPC algorithm is to select a set of future control moves in control horizon in such a way to minimize a cost function:

$$\begin{aligned}\min_{\mathbf{u}(k),\ldots,\mathbf{u}(k+H_c)} & Q_1 \sum_{i=1}^{H_p} \left(w_1(k+i) - y_1(k+i|k)\right)^2 \\ & + Q_2 \sum_{i=1}^{H_p} \left(w_2(k+i) - y_2(k+i|k)\right)^2 \\ & + R_1 \sum_{i=1}^{H_c} \left(\Delta u_1(k+i)\right)^2 + R_2 \sum_{i=1}^{H_c} \left(\Delta u_2(k+i)\right)^2 \end{aligned} \tag{5.1}$$

where k is the current time instant; $\mathbf{u}(j) = [u_1(j), u_2(j)]^T$ is the manipulated input vector at j-th time instant; $\mathbf{y}(j|i) = [y_1(j|i), y_2(j|i)]^T$ is the predicted output vector for j-th time instant predicted from i-th time instant; $\mathbf{w}(j) = [w_1(j), w_2(j)]^T$ is the set-point vector; Q_1, Q_2, R_1, R_2 are weighting parameters; H_p and H_c are prediction and control horizon.

The goal is to tune the parameters of MPC controller: the H_p prediction horizon, the H_c control horizon, the weights of different terms ($Q_1, \ldots, R_2$) in (5.1)*. Tuning of a controller is a classical optimization problem in process engineering. It is usually solved by minimization of a cost function. The main problem of this method is that it is difficult to select an appropriate cost function, because there are several objectives that must be considered. In this case, there are four objectives that the controller must meet:*

- *To follow set-point changes accurately.*
- *To keep controlled variable constant at its set-point against disturbances and interaction of multiple outputs.*
- *To avoid aggressive manipulation of input variables.*
- *To avoid fast changing of output variables and big overshoot of output variables.*

Of course it is not possible to fully satisfy all of these objectives, because they are conflicting. Therefore an appropriate cost function should contain several terms and tuning parameters to balance these conflicting objectives. The selection of these tuning parameters is difficult. If the reader has already tried to obtain a cost function, he/she knows that the performance of the resulting controller does not always meet with the expectations of the designer. In this case, the interactions between outputs especially renders the control more difficult. For example, if one concentrates on control-error (e.g. the integral squared errors of outputs), the interaction causes fast changing of interacted outputs that causes aggressive manipulation of the inputs. The great advantage of IEC is that a human expert is able to observe conflicts and interactions, so he/she can handle and balance them easily. Hence, the application of interactive optimization is a promising approach to this problem. Figure 5.4 shows simulation results of a generation from IEC after 41 function evaluation. I found satisfying solutions quickly with IEC (e.g. see the second column in Figure 5.4).

The SQP optimization method was tested for this example too; where the optimization goal was to minimize a cost function. The cost function was similar to (5.1)*, it contained the squared error from the difference between system outputs and set-points and from the change of manipulated inputs. The application of SQP algorithm was difficult because it often got stuck into a local minima near the initial point. Hence, the SQP algorithm was not able to find an acceptable solution in contrast to the proposed IEC algorithm.*

□

5.3 Conclusions

The data-driven model structure identification based on genetic programming is a quite new, but more and more popular technique in the scientific literature.

I recognized that genetic programming tends to identify too complex models, especially when measurement noise is present, and most of the published works pay little attention to this problem. As a solution, I developed a new method that eliminates the negligible terms from linear-in-parameters models during the identification process based on orthogonal least squares method. I demonstrated that if the orthogonal least squares method is used, it results in more transparent models than without using it. The new method is also useful for identification of model order[1].

Process optimization problems often lead to multi-objective problems where optimization goals are non-commensurable and they are in conflict with each other. In such cases, the common approach, namely the application of a quantitative cost-function, may be very difficult or pointless. For these problems, I developed a method that handles these problems by introducing a human user into the evaluation procedure. The poposed method uses the knowledge of the experts directly in the optimization procedure. The practical usefulness of the framework was demonstrated through two application examples: tuning of a multi-input multi-output controller and optimization of a fermentation process[2]. The algorithm has been adapted to particle swarm optimization [3].

---

[1] Madar J, Abonyi J, Szeifert F, Genetic programming for the identification of nonlinear input - Output models., INDUSTRIAL & ENGINEERING CHEMISTRY RESEARCH 44: pp. 3178-3186. (2005), IF: 1.504, Independent citations: 24

[2] Madar J, Abonyi J, Szeifert F, Interactive evolutionary computation in process engineering, COMPUTERS & CHEMICAL ENGINEERING 29: pp. 1591-1597. (2005), IF: 1.501, Independent citations: 5, www.fmt.uni-pannon.hu/softcomp/EAsy/ és a www.mathworks.com

[3] Madar J, Abonyi J, Szeifert F, Interactive particle swarm optimization, In: Kwasnicka H, Paprzycki M, Proceedings 5th International Conference on Intelligent Systems Design and : Applications (ISDA 2005), Wroclaw: IEEE Computer Society, 2005. pp. 314-319, Independent citations: 10

Appendix A

Appendix: Application Examples

A.1 Process Data Warehouse

Formulated products (plastics, polymer composites) are generally produced from many ingredients, and large number of the interactions between the components and the processing conditions all have the effect on the final product quality. If these effects are detected, significant economic benefits can be realized. The major aims of monitoring plant performance are the reduction of off-specification production, the identification of important process disturbances and the early warning of process malfunctions or plant faults. Furthermore, when a reliable model is available that is able to estimate the quality of the product, it can be inverted to obtain the suitable operating conditions required for achieving the target product quality. The above considerations lead the foundation of the "Optimization of Operating Processes" project of the VIKKK Research Center at the University of Veszprem supported by the largest Hungarian polymer production company (TVK Ltd).

Problem description

TVK Ltd produces the medium (MDPE) and the high density polyethylene (HDPE) with the technology of Phillips Petroleum Co., which is divided into three separated units in aspect of information sources: A., Polymerisation Production Unit, B., Granulation Production Unit, C., Polyethylene (PE) Quality Control Laboratory. In Figure A.1 the information flow between these units are depicted. The production of the polymer powder in the Polymerisation Production Unit is the most important step of the process (see Figure A.2). The main properties of polymer products (Melt Index (MI) and density) are controlled by the reactor temperature, monomer, comonomer and chain-transfer agent concentration. An interesting problem with the process is that it requires to produce about ten product grades according to market demand. Hence, there is a clear need to minimize the time of changeover because off-specification product may be produced during transition. The difficulty of the problem comes from the fact that there are more than ten process variables to consider.

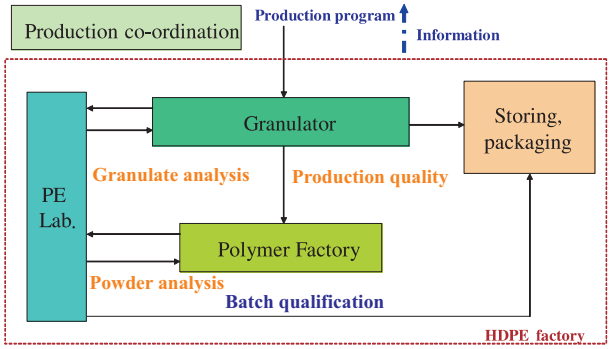


Figure A.1: The information connections between the production units.

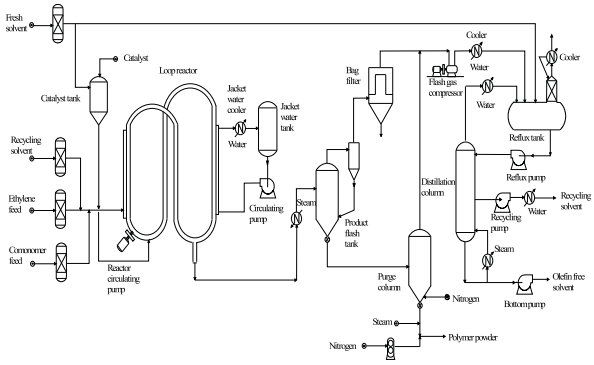


Figure A.2: Scheme of the Polymerisation Production Unit (Phillips loop reactor process).

The problem is not only arising from the wide product palette and from the frequent product change, but also the heterogeneity of the measurement data in terms of time horizons and formats:

1. A Honeywell Distributed Control System (DCS) operates in the Polymerisation Production Unit, which serves the data via the so-called Process History Database (PHD) module. This database contains the most important process variables and some technological variables calculated by the Advanced Process Control (APC) module of the DCS. In the following among these process variables the most important ones are mentioned, which are used for process modeling and monitoring purposes. Measurements are available in every 15 seconds on process variables which consist of input and output variables: $u_{k,(1,\dots,8)}$ the comonomer, the monomer, the solvent and the chain transfer agent inlet flowrate and temperature, $u_{k,9}$ polymer production rate, $u_{k,10}$ the flowrate of the catalyzator, $u_{k,(11,\dots,13)}$ cooling water flowrate, inlet and outlet temperature.

2. The devices of the Granulation Production Unit are controlled by Programmable Logical Controllers (PLCs). In this unit not all of data are stored electronic. The data are mostly logged manually in reports related to the events that happen in every one or two hours.

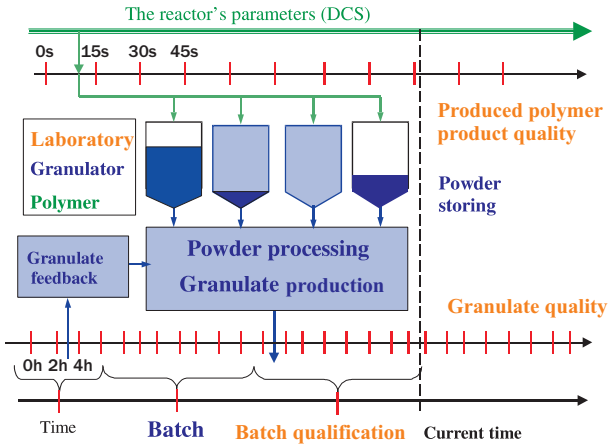


Figure A.3: Time horizons of the measured data.

3. In the PE Quality Control Laboratory the measured data are stored in reports (e.g. Polymer powder and Granulate classification report, Batch qualification report, Product change report). The product quality, $\mathbf{y}_k$, is determined by off-line laboratory analysis after drying the polymer that causes one hour time-delay. The most important quality variables are the Melt Index and the density whose sampling time intervals are between half and five hours. While the sampling and the measurement of the quality of the polymer powder and the granulate are made in every one or two hours, the time of the qualification of the batches strongly depends on the technology.

Figure A.3 shows the relation between these information sources and their sampling frequency, and the time horizons of the measured data.

Since, it would be useful to know if the product is good before testing it, the monitoring and the estimation of the state and product quality variables would help in the early detection of poor-quality product. There are other reasons why monitoring the process is advantageous. Only a few properties of the product are measured and sometimes these are not sufficient to define entirely the product quality. For example, if only rheological properties of polymer are measured (melt index), any variation in end-use application that arise due to variation of chemical structure (branching, composition, etc.) will not be captured by following only these product properties. In these cases the process data may contain more information about events with special causes that may effect the product quality [117].

The data warehouse project was implemented in three steps depicted on Figure A.4. Beside the operational database of DCS the information sources are the standard data sheets and reports which often include redundant information. Unfortunately the comparison of these reports collected from different process units proved that these separated sources of information include contradictions as well. Consequently, electronic forms have been created to avoid these problems. The following aspects were kept in mind at the design of these forms and related data tables: one data should be inserted into only one table;

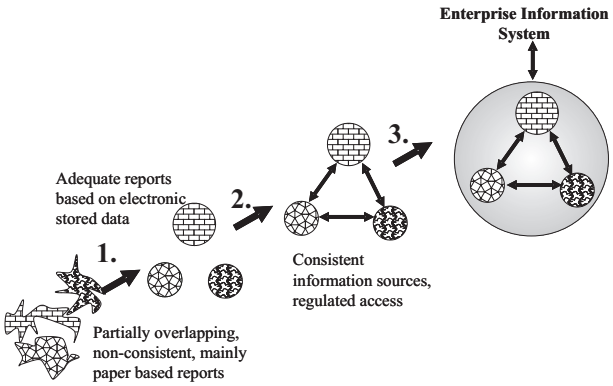


Figure A.4: Main tasks of the project.

"everybody" should be able to access the necessary information; the rights for data upload, query and change of the data should be clarified; the identification of responsible users for the data and data security should be solved. The designed database includes the measurements of the laboratory and the events which are recorded by the chargeman (and by the operators). The technological variables of the polymerisation reactors are stored via the PHD module of Honeywell system (reactor data, cleaning system etc.) as well as the features calculated by APC module: some state variables and variables of the input streams (e.g. temperature, pressure, concentrations), and other data (e.g. catalyst activation). Beside the WEB based front-end tools, applications based on MS-Excel and MS-Access and Visual Basic have been worked out. This was proven to be practical at the beginning of the project because the employees in the production unit and in the laboratory were expert in the usage of these simple tools.

Dynamic Model for Data Integration

To detect and analyze causal relationships among the process and quality variables taken from several heterogenous information sources, the laboratory measurements and the operating variables of the reactors and extruders have to be synchronized based on the model of the main process elements (e.g., pipes, silos, flash-tanks). For this purpose, based on the models of the material and information flows, MATLAB scripts were written to collect all the logged events of the whole production-line and to arrange and re-calculate the time of these events according to the "life" of the product from the reactor to the final product storing. In this subsection, the basic considerations behind of this *dynamic data integration* are presented. The general theoretical issues behind this implementation step are presented in Chapter 2.

The connection between the Polymerization Production Unit and the Granulation Unit is determined by the input flow from Polymerization Production Unit into the top of the silos and the output flow from the bottom of silos into the Granulation Unit (Figure A.5).

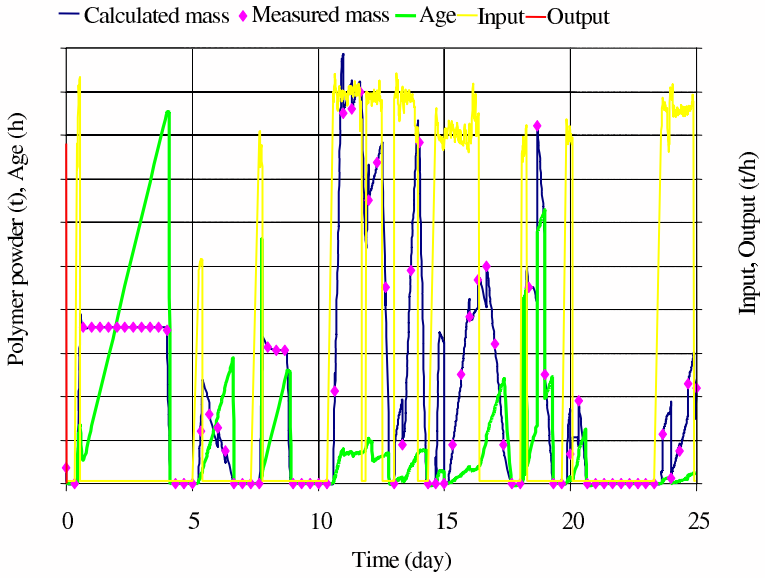


Figure A.5: Dynamic behavior of the polymer powder silos.

Since the dynamical behaviour of the silos, the integration of the information about these units is not realizable by static Structured Query Language (SQL) queries and On-Line Analytical Processing (OLAP) functions. This solution should be based on the dynamic model of the silos:

$$M_i(T) = \int_{t_0}^{T} \left(F_{in,i}(t) - F_{batch,i}(t) - F_{add,i}(t)\right) dt + M_{0,i}, \tag{A.1}$$

where $F_{in,i}(t)$ is the input mass flow that can be calculated based on the transport report and the productivity estimated by the DCS; $F_{batch,i}(t)$ and $F_{add,i}(t)$ are the mass flows of the feeding container and of the pre-mixed additive ingredients calculated by the reports which include details of extrusion process; and $M_{0,i}$ is the mass which are calculated by the previously measured levels of the silo. This simple model is the base of the calculation of the actual polymer mass, $M_i(T)$ in the ith silo.

From technological viewpoint the age of the polymer powder leaving the silo is more important than the actual mass of the polymer in the silo, because the measured polymer quality can be retrievable based on the age of the polymer. The modification of (A.1) can give the answer how old the polymer is, ($T*$ time):

$$0 = \int_{t_0}^{T^*} F_{in,i}(t)dt - \int_{t_0}^{T} \left(F_{batch,i}(t) + F_{add,i}(t)\right) dt + M_{0,i}. \tag{A.2}$$

The combination of (A.1) and (A.2) gives the age of the polymer powder leaving the silo:

$$M_i(T) = \int_{T^*}^{T} F_{in,i}(t)dt. \tag{A.3}$$

Based on this model the data warehouse is able to answer to the following questions:

1. **What is the content of the silos?** How much powder is there with a given feature in a particular silo at an arbitrary time instant? The answer of this question is very useful in the scheduling of powder processing.

2. **What kind of polymer will be granulated?** Polymer powder with what property is being processed from which silos at a given time instant. This is important to the estimation of the powder quality before the extrusion, because the operation of the extrusion could be controlled feed forward based on this estimated feature before the granulate product qualification. For this purpose, the effect of the mixing of polymer powders and the operation parameters of extrusion (e.g. temperature, power consumption) should be explored.

   The density of the mixed polymer powder is calculated by the properties of the "raw" polymers in proportion of their quantities. In the calculation of the Melt Index, it is assumed that the average molecule mass (M) can be calculated by the average molecule masses of the polymer powders in proportion with their total masses:

   $$F_1\overline{M}_1 + F_2\overline{M}_2 = (F_1 + F_2)\overline{M}, \tag{A.4}$$

   while the relationship between the average molecule mass and the melt index is the following [118]:

   $$\frac{\overline{M}_2}{\overline{M}_1} = \left(\frac{MI_2}{MI_1}\right)^{0.294}. \tag{A.5}$$

   In case of two polymer powders the melt index is calculated by the following way:

   $$\log MI = \frac{1}{0.294}\log\left(\frac{F_1 + F_2\left(\frac{MI_2}{MI_1}\right)^{0.294}}{F_1 + F_2}\right) + \log MI_1 \tag{A.6}$$

3. **What will be the quality of the end product?** It is an important question how the end-product features are correlated to the measurements of the laboratory samples used for the quality control of the process to the final batch qualification measured after the homogenization (mixing) of the product. For this purpose the mass of the batch calculated based on the extruding data sheets should be compared to its measured value. The next step of the validation is the pre-estimation of the batch features by the hourly sampled mass rate. These features are measured after the homogenization of the product which takes eight hours. Figure A.6 shows the accuracy of the estimation.

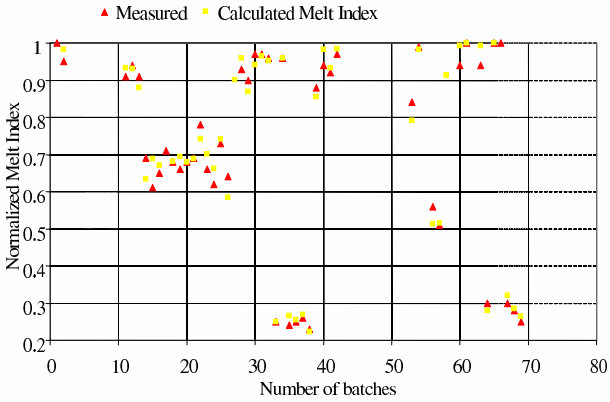


Figure A.6: Validation and estimation of the final product quality.

4. **Product retrieval:** Due to the dynamical behavior of the complex production technology, it is not trivial to assign in with period of the operation of a given process unit which final product is produced. Hence, process (silo) model shown above makes possible to calculate the time-delays represented by the flows among the distributed process units, and use these time-delays to synchronize the events of the production of a given product, that makes possible the retrieval of the details of the production process (process and product variables).

Analysis of Grade Transitions

Based on the historical data of a half-year operation, all of the productions and grade transitions have been pre-processed by the above mentioned models. As Figure A.7 shows, based on the database of the grade transitions, we have designed special plots that can be used by the operators as patterns of recent control strategies. Not only tools for the visualization of time-series have been developed but plots illustrating the safety constraints too (see Figure A.8 *(a)*). In some cases, the difficulty of the analysis of these plots comes from the fact that there are more than ten process variables to consider. As Figure A.8 *(b)* shows, the proposed OSS consists of Principal Component Analysis (PCA)-based visualization tools to solve this problem, since the two-dimensional space of the transformed variables, the plotted Hotelling T^2 and model error measures, Q are able to give insight to the process behaviour and to detect faults.

Example A.1. Application of clustering based time-series segmentation to process monitoring

The aim of this example is to show how the time-series segmentation algorithm presented in Section 3.2 is able to detect meaningful temporal shapes from multivariate historical process data. The dataset used in this example represents 160 hours of operation of the reactor and includes three product transitions around the 24, 54, and 86-th hours. The initial number of the segments was ten and the γ threshold was chosen to $\gamma = 0.4$. In the Figure 3.8 it can be seen that $q = 5$ principal components must be con-

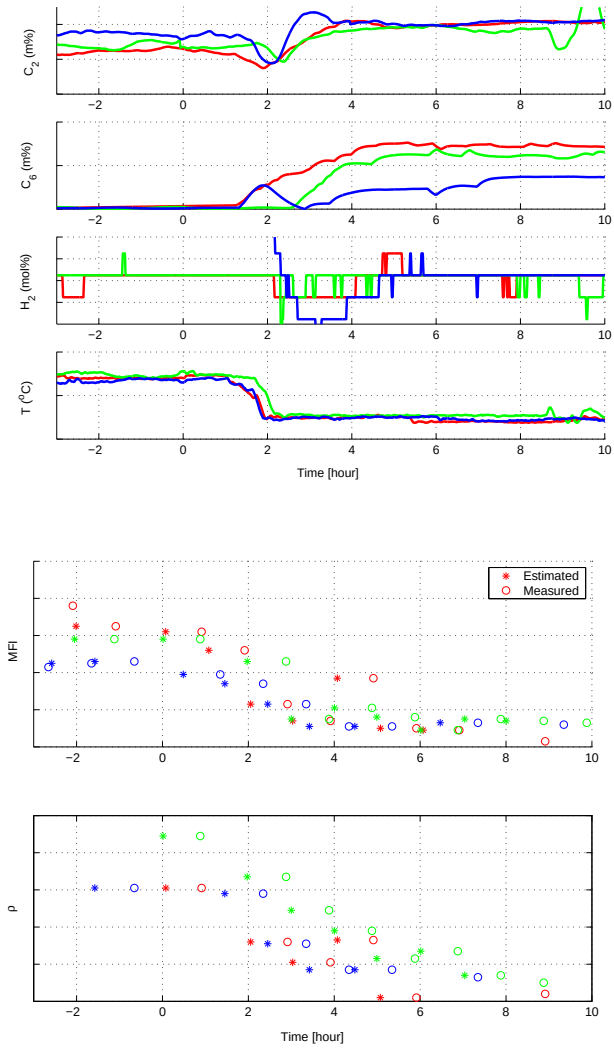
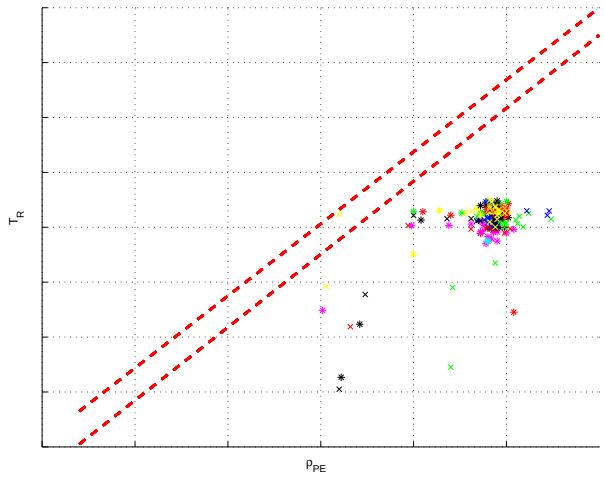
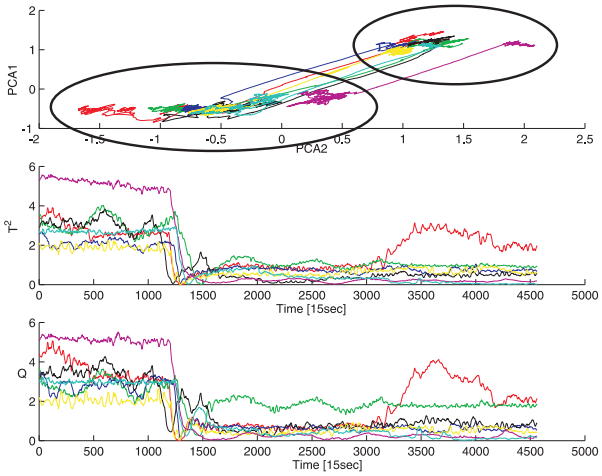


Figure A.7: Grade transition from product A to product E (upper: reactor process variables, lower: quality-related variables, lab. measurements).

sidered for 95% accuracy. As Figure A.9 shows, both the bottom-up and the clustering based algorithm are able to detect the product transitions, and all the three methods gave similar results. This reflects that the mean and the covariance of the data were not independently changed. This is also confirmed by the analysis of the compatibilities of the adjacent clusters. As it can be seen, the product transitions are represented by two independent clusters, while the third transition was not so characteristic that it would require an independent segment. This smooth transition between the third and the fourth products is also reflected by how the $p(\mathbf{z}_k|\eta_i)$ probabilities overlap between the 75-125th hour of operations. The changes of the $p(\mathbf{z}_k|\eta_i)$ probabilities around the 135th hour of operation are also informative, as the period of lower or drastically changing probabilities reflect some erroneous operation of the process. The results are similar if more than 5 principal components are taken into account.



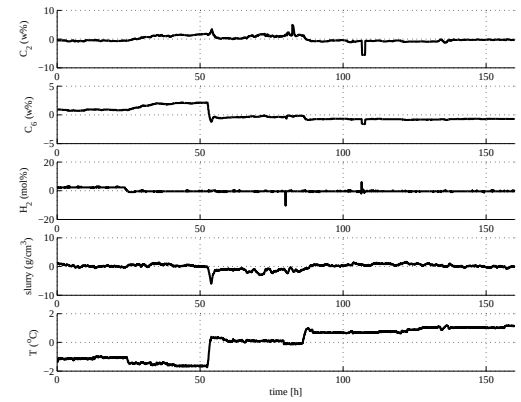
(a) Example of a safety constraint plot.



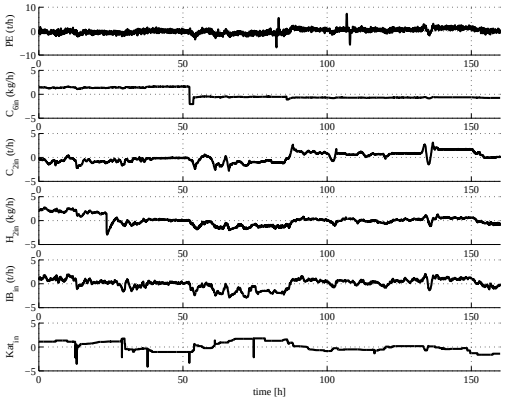
(b) PCA plot of grade-transitions from product A to product E.

Figure A.8: OSS plots based on Principal Component Analysis.

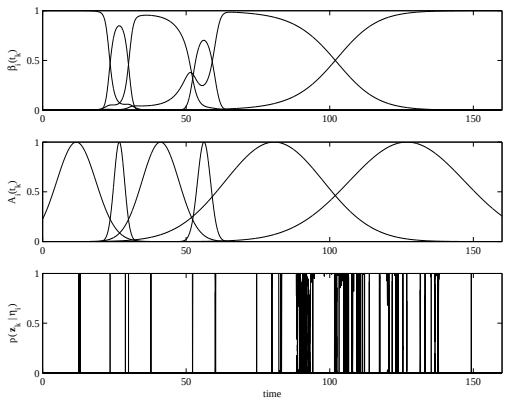
This example illustrated that the proposed tool can be applied for the segmentation of a historical database and with the application of this tool useful information can be extracted concerning the changes of the operation regimes of the process and process faults. In the current state of our project we use this tool to compare the production of different products and extract homogenous segments of operation that can be used by a Kalman-filter based state estimation algorithm for the identification of useful kinetic parameters and models which are able to predict the quality of the products [119].



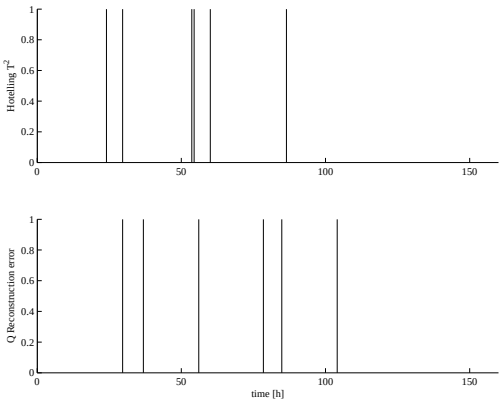
(a) States of the reactor.



(b) Input variables of the reactor.



(c) Fuzzy segmentation results, $q = 5$, $\gamma = 0.4$.



(d) Bottom-up segmentation results.

Figure A.9: Segmentation of the industrial dataset

□

A.2 Monitoring Process Transitions by State Estimation

Historical process data alone usually may not sufficient for monitoring complex processes.The current measured input-output data pairs are often not in casuality relationship because of the dead time and the dynamical behavior of the system. In practice, the state variables happen to be not measurable, or rarely measured only by off-line laboratory tests. To solve these problems, different methods can be applied that happen to force the usage of delayed measured data besides the current data, e.g. the method proposed in [120] which is based on Dynamic Principal Component Analysis. The main idea of this section is to apply nonlinear state-estimation algorithm to detect changes in the estimated state-variables and the correlation of their modelling error.

Covariance based Similarity Measure

Time-series segmentation is often used to extract *internally homogeneous segments* from a given time-series. Usually, the cost function describing the internal homogeneity of the individual segments is defined based on the distances between the actual values of the time-series and the values given by a simple univariate function fitted to the data of each segment.

Due to the hidden nature of the process the measured variables are correlated. In some cases the hidden process, so the correlation among the variables, vary in time. This phenomena can occur at process transitions or when there is a significant process fault, etc. The segmentation of only one measured variable is not able to detect such changes. Hence, the segmentation algorithm should be based on multivariate statistical tools as it was seen in Chapter 2.

Covariance matrices, $\mathbf{P}_k$, describe the relationship between the variables around the kth data point and they can also be used to calculate the cost function based on a covariance matrix similarity measure:

$$cost(S_i(a_i, b_i)) = \frac{1}{b_i - a_i + 1} \sum_{k=a_i}^{b_i} S_{cov}(\mathbf{P}_k, \mathbf{P}_{S_i}) \tag{A.7}$$

where $\mathbf{P}_{S_i}$ is the covariance matrix of the ith segment with the borders a_i and b_i which can be calculated by the averaging of the matrices $\mathbf{P}_k | a_i \leq k \leq b_i$, and S_{cov} is the PCA similarity factor introduced by Krzanowski [121, 71], which can be seen as a measure of the similarity between the two covariance matrices.

The similarity of the found segments can be displayed as a dendrogram. A dendrogram is a tree-shaped map of the similarities that shows the merging of segments into clusters at various stages of the analysis. The interpretation of the results is intuitive, which is the major reason of these methods used to illustrate the results of hierarchical clustering (see Figure A.14).

Covariance of the Monitored Variables

In the previous subsection it has been shown that the covariance of the monitored process variables can be used to measure the homogeneity of the segments of multivariate time-series. The main problem of the application of this approach is how we can estimate covariance matrices that contain useful information about the operation of the monitored process.

The most straightforward approach is the recursive estimation of the $\mathbf{P}_k$ covariances:

$$\mathbf{P}_k = \frac{1}{\alpha_{j,k}} \left[\mathbf{P}_{k-1} - \frac{\mathbf{P}_{k-1} \widetilde{\mathbf{x}}_k \widetilde{\mathbf{x}}_k^T \mathbf{P}_{k-1}}{\alpha_{j,k} + \widetilde{\mathbf{x}}_k^T \mathbf{P}_{k-1} \widetilde{\mathbf{x}}_k} \right] \tag{A.8}$$

where $\mathbf{P}_k$ is a matrix proportional to the covariance matrix, and α_j is a scalar forgetting factor of the jth rule adaptation.

This tool can be directly used to analyze the measured input-output data, $\widetilde{\mathbf{x}}_k = [\mathbf{u}^T, \mathbf{y}]^T$, which approach is considered as the basis of the first algorithm proposed in this section (**Algorithm 1**). Historical input-output process data alone may be not sufficient for the monitoring of complex processes. Hence, the main idea of this section is to apply nonlinear state-estimation algorithm to detect changes in the estimated state-variables (**Algorithm 2**) and the correlation of their modelling error (**Algorithm 3**).

The proposed algorithms have been developed for the general nonlinear model of a dynamical system:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k), \;\; \mathbf{y}_k \;\; = \;\; \mathbf{g}(\mathbf{x}_k, \mathbf{w}_k) \tag{A.9}$$

where $\mathbf{v}_k$ and $\mathbf{w}_k$ are noise variables assumed to be independent of the current and past states, $\mathbf{v}_k \sim \mathcal{N}(\overline{\mathbf{v}}_k, \mathbf{Q}_k)$, $\mathbf{w}_k \sim \mathcal{N}(\overline{\mathbf{w}}_k, \mathbf{R}_k)$.

The developed algorithm is based on the results of standard state-estimation algorithms, i.e. the estimated state-variables,

$$\hat{\mathbf{x}}_k = \overline{\mathbf{x}}_k + \mathbf{K}_k[\mathbf{y}_k - \overline{\mathbf{y}}_k] \tag{A.10}$$

and their *a posteriori* covariance matrix,

$$\hat{\mathbf{P}}_k = \mathbf{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T] \tag{A.11}$$

In these expressions

$$\overline{\mathbf{x}}_k = \mathbf{E}[\mathbf{x}_k | \mathbf{Y}^{k-1}],$$
$$\overline{\mathbf{y}}_k = \mathbf{E}[\mathbf{y}_k | \mathbf{Y}^{k-1}],$$

where $\mathbf{Y}^{k-1}$ is a matrix containing the past measurements, and $\mathbf{K}_k$ is the Kalman gain:

$$\mathbf{K}_k = \mathbf{P}_{xy,k} \mathbf{P}_{y,k}^{-1},$$

where

$$\begin{aligned} \mathbf{P}_{xy,k} &= \mathbf{E}[(\mathbf{x}_k - \overline{\mathbf{x}}_k)(\mathbf{y}_k - \overline{\mathbf{y}}_k)^T | \mathbf{Y}^{k-1}], \\ \mathbf{P}_{y,k} &= \mathbf{E}[(\mathbf{y}_k - \overline{\mathbf{y}}_k)(\mathbf{y}_k - \overline{\mathbf{y}}_k)^T | \mathbf{Y}^{k-1}]. \end{aligned} \tag{A.12}$$

By selecting the update of the estimated variables and their covariance so that the covariance for the estimation error is minimized, we can obtain the following update-rule of the covariance matrix

$$\hat{\mathbf{P}}_k = \overline{\mathbf{P}}_k - \mathbf{K}_k \mathbf{P}_{y,k} \mathbf{K}_k^T \,, \tag{A.13}$$

where

$$\overline{\mathbf{P}}_k = \mathbf{E}[(\mathbf{x}_k - \overline{\mathbf{x}}_k)(\mathbf{x}_k - \overline{\mathbf{x}}_k)^T | \mathbf{Y}^{k-1}] \,. \tag{A.14}$$

As the various expectations used in these equations in general are intractable, some kind of approximation is commonly used. The Extended Kalman Filter (EKF) is based on Taylor linearization of the state transition and output equations. Although the developed algorithm can be applied to any state-estimation algorithms, the effectiveness of the selected filter has an effect on the results of the segmentation. The utilized DD2 filter is based on approximations obtained with a multivariable extension of Stirling's interpolation formula. This filter is simple to implement as no derivatives of the model equations are needed, yet it provides excellent accuracy [122].

Based on the result of this nonlinear state estimation two different algorithms can be defined. **Algorithm 2** is based on the direct analysis of the estimated state variables, $\widetilde{\mathbf{x}} = \hat{\mathbf{x}}$ in (A.8), while **Algorithm 3**, which is the main contribution of this section, uses the *a posteriori* covariance matrices, $\hat{\mathbf{P}}_k$, given by the nonlinear state estimation algorithm ($\mathbf{P}_k = \hat{\mathbf{P}}_k$ in (A.8)).

The proposed process monitoring tool has been implemented independently from the DCS; the database of the historical process data is stored by a MySQL SQL-server. Most of the measurements are available in every 15 seconds on process variables which consist of input and output variables: the comonomer hexene, the monomer ethylene, the solvent isobutene and the chain transfer agent hydrogen inlet flowrates and temperatures ($u_{1,\dots,4} = F^{in}_{C_6,C_2,C_4,H_2}$ and $u_{5,\dots,8} = T^{in}_{C_6,C_2,C_4,H_2}$), the flowrate of the catalyst ($u_9 = F^{in}_{cat}$), and the flowrate, the inlet and the outlet temperatures of the cooling water ($u_{10,\dots,12} = F^{in}_w, T^{in}_w, T^{out}_w$).

The prototype of the proposed process monitoring tool has been implemented in MATLAB with the use of the Database and Kalman filter Toolboxes.

The Model of the Process

The model used in the state-estimation algorithm contains the mass, components and energy balance equations to estimate the mass of the fluid and the formulated polymer in the reactor, the concentrations of the main components (ethylene, hexene, hydrogen and catalyst) and the reactor temperature. Hence, the state-variables of this detailed first-principles model are the mass of the fluid and the polymer in the reactor ($x_1 = G_F$ and $x_2 = G_{PE}$), the chain transfer agent concentration ($x_3 = c_{H_2}$), monomer, comonomer and catalyst concentration in the loop reactor ($x_4 = c_{C_2}$, $x_5 = c_{C_6}$ and $x_6 = c_{cat}$), and reactor temperature ($x_7 = T_R$). Since there are some unknown parameters related to the reaction rates of the different catalysts applied to produce the different products, there are additional state-variables: the reaction rate coefficients $x_8 = k_{C_2}$, $x_9 = k_{C_6}$, $x_{10} = k_{H_2}$.

With the use of these state variables the main model equations are formulated as follows:

$$\frac{dG_F}{dt} = \sum_j F_j^{in} - F_F^{out} - \sum_i k_i c_i G_F c_{cat} G_{PE} \tag{A.15}$$

$$\frac{dG_{PE}}{dt} = \sum_i k_i c_i G_F c_{cat} G_{PE} - F_{PE}^{out} \tag{A.16}$$

$$\frac{dc_i}{dt} = \frac{1}{G_F}\left(F_i^{in} - F_F^{out} c_i - k_i c_i G_F c_{cat} G_{PE} - c_i \frac{dG_F}{dt}\right) \tag{A.17}$$

$$\frac{dc_{cat}}{dt} = \frac{1}{G_{PE}}\left(F_{cat}^{in} - F_{PE}^{out} c_{cat} - c_{cat} \frac{dG_{PE}}{dt}\right) \tag{A.18}$$

$$\frac{dT_R}{dt} = \frac{1}{G_F c_p^F + G_{PE} c_p^{PE} + G_{reactor} c_p^{reactor}}\left(\sum_j F_j^{in} c_p^j (T_j^{in} - T_R) + \right.$$
$$\left. + \sum_i k_i c_i G_F c_{cat} G_{PE} \Delta H_i - Q_{cooling} + Q_{stirring}\right) \tag{A.19}$$

Notation: $i = C_2, C_6, H_2$, $j = C_4, C_2, C_6, H_2$, $Q_{cooling} = F_w^{in} c_p^w (T_w^{out} - T_w^{in})$, and $G_{(.)}$ means mass, $F_{(.)}$ means mass rate, $c_p^{(.)}$ means the specific heat of the $(.)$ component, and ΔH_i represents the heat of the ith reaction.

For the feedback to the filter, measurements are available on the chain transfer agent, monomer and comonomer concentration ($y_{1,2,3} = x_{3,4,5}$), reactor temperature ($y_4 = x_7$) and the density of the slurry in the reactor ($y_5 = \rho_{slurry}$, which is related to x_1 and x_2). The concentration measurements are available only in every 8 minutes.

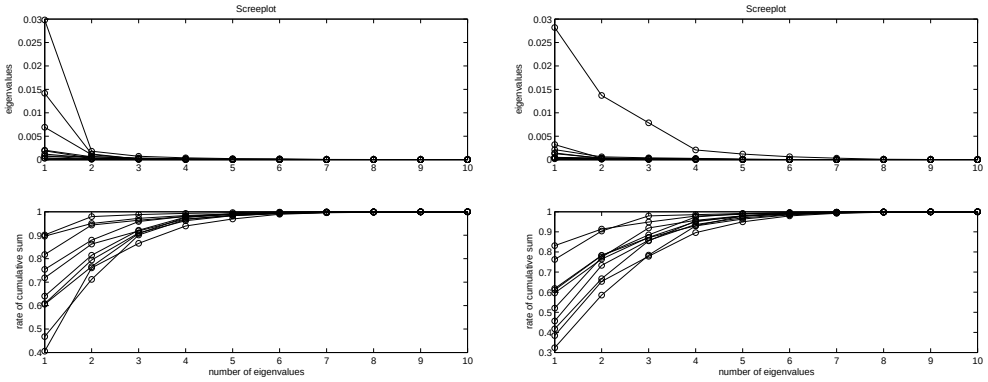
The dimensionless state variables are obtained by the normalizing of the variables,

$$x^n = \frac{x - x_{min}}{x_{int}},$$

where x_{min} is a minimal value and x_{int} is the interval of the variable (based on a priori knowledge, e.g. the operators' experiences if available). The values of the input and state variables have not been depicted in the figures presented in the next sections because they are secret so not publishable.

Parameters of the Segmentation Algorithms

The results studied in the next sections have been obtained by setting the initial process noise covariance matrix to $\mathbf{Q} = diag(10^{-4})$, the measurement noise covariance matrix to $\mathbf{R} = diag(10^{-8})$, and the initial state covariance matrix to $\mathbf{P}_0 = diag(10^{-8})$. The values of these parameters heavily depends on the analyzed dataset. That is why the proper normalization method has an influence on the results. However, the parameters above can be used to estimate the state variables not only the datasets presented in the next sections, but also other datasets that contain data from production of other products in different operation conditions but in the same reactor and produced by the same type



(a) *(b)*

Figure A.10: Screeplot for determining the proper number of principal components in case of datasets presented in (a) Example A.2 and (b) Example A.3, respectively.

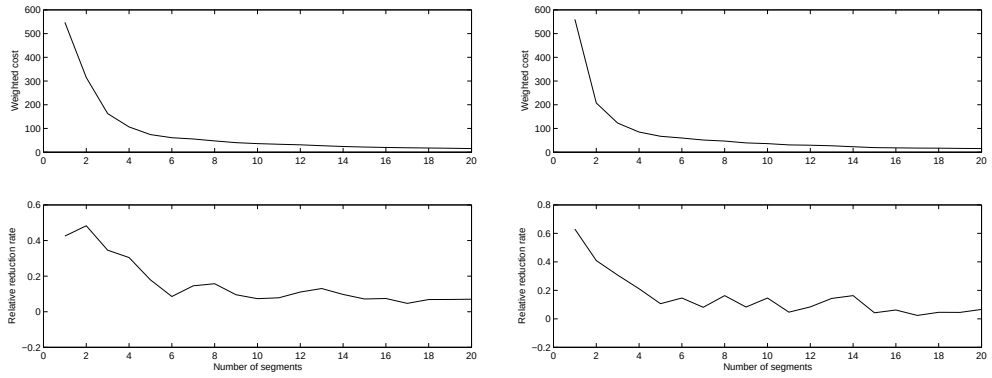
of catalyst. In these cases the state estimation algorithm was robust enough related to the parameters above, they can be varied in the range of two orders of magnitude around the values above.

For the segmentation algorithm some parameters have to be chosen in advance, one of them is *the number of principal components*. This can be done by the analysis of the eigenvalues of the covariance matrices of some initial segments. This method was used in Section 3.2. The datasets shown in Figure A.12 and in Figure A.13 were initially partitioned into ten segments. As Figure A.10 illustrates, the cumulative rate of the sum of the eigenvalues shows that five PCs are sufficient to approximate the distribution of the data with 97% accuracy in both cases.

Another important parameter is *the number of segments*. Unlike the segmentation method presented in Section 3.2, the number of segments should be defined before the segmentation because the hierarchical clustering applied in this section is not able to determine this value. One of the applicable methods is presented by Vasko et al in [49]. This method is based on permutation test so as to determine whether the increase of the model accuracy with the increase of the number of segments is due to the underlying structure of the data or due to the noise. In this section the simplified version of this method has been used. It is based on the relative reduction of the modelling error (see (3.42) and (A.7)):

$$RR(c|T) = \frac{cost(S_T^{c-1}) - cost(S_T^c)}{cost(S_T^{c-1})} \tag{A.20}$$

where $RR(c|T)$ is the relative reduction of error when c segments are used instead of $c-1$ segments.



(a) *(b)*

Figure A.11: Determining the number of segments by **Algorithm 3** in case of datasets presented in (a) Example A.2 and (b) Example A.3, respectively.

As it can be seen in Figure A.11, significant reductions are not achieved by using more than 5 or 6 segments in case of both datasets. Similar figures can be obtained by **Algorithm 2**.

Example A.2. Monitoring of process transitions*]*

In this study a set of historical process data covered 100 hours period of operation has been analyzed. These datasets include at least three segments because of a product transition around the 45*th hour (see Figure A.12). Based on the relative reduction of error in Figure A.11 (a), the algorithm searched for five segments (*$c = 5$*).*

*The results depicted in Figure A.12 show that the most reasonable segmentation has been obtained based on the covariance matrices of state estimation algorithm (**Algorithm 3**). The segmentation obtained based on the estimated state variables is similar: the boundaries of the segment that contains the transition around the* 45*th hour are nearly the same, and the other segments contain parts of the analyzed dataset with similar properties. Contrary to these nice results, when only the measured input-output data were used for the segmentation the algorithm was not able to detect even the process transition.*

*It has to be noted that **Algorithm 3** can be found more reasonable than **Algorithm 2**, because one additional parameter has to be chosen in the last case: the forgetting factor,* α *in the recursive estimation of the covariance matrices in* (A.8)*. The result obtained by **Algorithm 2** is very sensitive to its choice. The* $\alpha = 0.95$ *seemed to be a good trade-off between robustness and flexibility.*

□

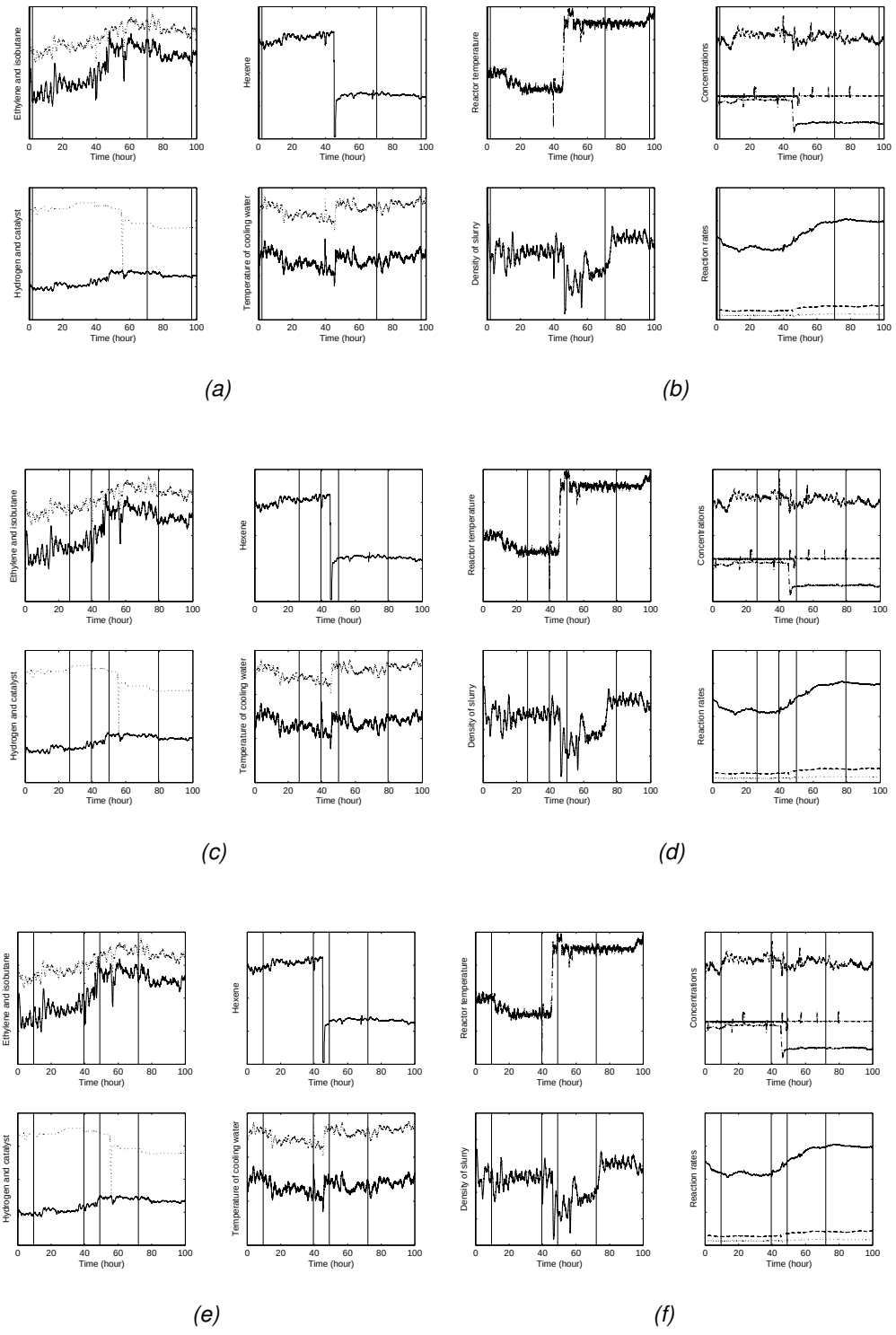


Figure A.12: a., b.: Segmentation based **Algorithm 1**; c., d.: Segmentation based on **Algorithm 2**,; e., f.: Segmentation based on **Algorithm 3**; a., c., e.: Input variables: $F_{C_2}^{in}, F_{C_4}^{in}, F_{C_6}^{in}, F_{H_2}^{in}, F_{cat}^{in}, T_w^{in}, T_w^{out}$; b., d., f.: Process outputs and states: $T_R, c_{C_2}, c_{C_4}, c_{C_6}, \rho_{slurry}, k_{C_2}, k_{C_6}, k_{H_2}$

Example A.3. Detection of changes in the catalyst productivity

Beside the analysis of the process transitions, the time-series of "stable" operations have also been segmented to detect interesting patterns of relatively homogeneous data. For this purpose ***Algorithm 3*** *was chosen from the methods presented above, because it gives good results in case of product changes. One of these results can be seen in Figure A.13, which shows a 120-hour long production period without any product changes. Based on the relative reduction of error in Figure A.11 (b), the number of segments was chosen to be equal to six ($c = 6$).*

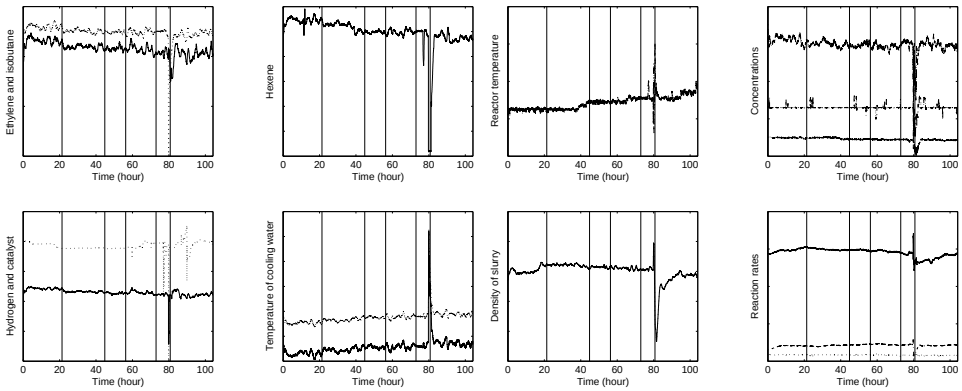


Figure A.13: Segmentation based on the error covariance matrices.

The homogeneity of a historical process data set can be characterized by the similarity of the segments that can be illustrated as a dendrogram (see Figure A.14).

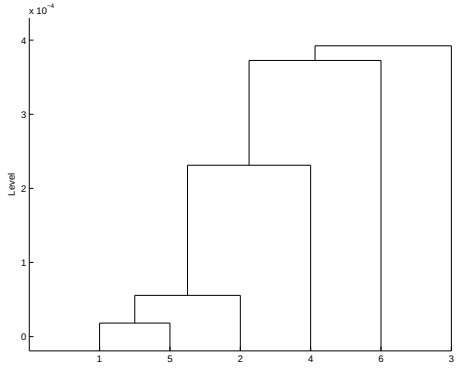


Figure A.14: Similarity of the found segments.

This dendrogram and the border of the segments give a chance to analyze and to understand the hidden processes of complex systems. E.g. in this example these results confirm that the quality of the catalyst has an important influence in productivity. During the 20, 47, 75, 90th hours of the presented period of operation changes between the catalyst feeder bins happened. The segmentation algorithm based on the estimated state variables was able to detect these changes that had an effect to the catalysis productivity, but when only the input-output variables were used segments without any useful information were detected.

*It has to be noted that the borders of the segments given by **Algorithm 2** and **Algorithm 3** are similar also in this case, but the dendrograms are different. This is because that the segments without product transition are much more similar to each other than in case of the time-series which contains a product transition. So it is a more difficult problem to differentiate segments of operations related to the minor changes of the technology, like the changes of the catalyst productivity. This phenomena can also be seen in the dendrogram: the values that belong to the axis of ordinates are smaller with one or two order(s) of magnitude in case of a time-series without product transition. In case of product transition not only the borders of the segments are similar but also the shape of the dendrograms are nearly the same. This shows that both algorithms are applicable for similar purposes.*

□

This section presented the synergistic combination of state-estimation and advanced statistical tools for the analysis of multivariate historical process data. The key idea of the presented segmentation algorithm is to detect changes in the correlation among the state-variables based on their *a posteriori* covariance matrices estimated by a state-estimation algorithm. PCA similarity factor can be used to analyze these covariance matrices. Although the developed algorithm can be applied to any state-estimation algorithms, the performance of the filter has huge effect on the segmentation. The applied DD2 filter has been proven to be accurate, and it was straightforward to include a varying number of parameters in the state vector for simultaneous state and parameter estimation, which was really useful for the analysis of the reaction kinetic parameters during process transitions. The application example showed the benefits of the incorporation of state estimation tools into segmentation algorithms.

A.3 Semi-mechanistic Models for Product Quality Estimation

Process monitoring based on multivariate statistical analysis, neural networks and advanced state-estimation tools has recently been investigated by a number of researchers, and widely applied in polymer industry. This is not surprising. Formulated products (plastics, polymer composites) are generally produced from many ingredients, and large number of the interactions between the components and the processing conditions all have the effect on the final product quality. If these effects are detected, significant economic benefits can be realized. The major aims of monitoring plant performance are the reduction of off-specification production, the identification of important process disturbances and the early warning of process malfunctions or plant faults. Furthermore, when a reliable model is available that is able **to estimate the quality of the product**, it can be inverted to obtain the suitable operating conditions required for achieving the target product quality. The aim of the this section is to present how neural-networks can be used as a soft sensor, and how the neural-network part of the developed semi-mechanistic model can be identified based on a spline-smoothing approach.

Advanced process control and monitoring algorithms are based on state variables which are not always measurable or they are measured off-line. Hence, for the effective application of these tools there is a need for state estimation algorithms that are based on the model of the monitored and/or controlled process. In the presence of additive white Gaussian noise Kalman filter provides optimal (in the sense of maximum likelihood) estimates of the states of a linear dynamical system. For nonlinear processes Extended Kalman Filtering (EKF) should be used. The model of EKF can be a *first-principle model* formulated by a set of nonlinear differential equations or *black-box model*, e.g. a neural network (NN).

Sometimes it is advantageous to combine these modeling approaches. E.g. Psichogios et. al. [123] applied so-called hybrid models that combines a first-principle model with a NN model which serves as an estimator of unmeasured process parameters that are difficult to model from first-principles. Since this seminar thesis, many industrial applications of these semi-mechanistic models have been reported, and it has been proofed that this kind of models has better properties than stand-alone NN applications, e.g. in the pyrolysis of ethane [124], in industrial polymerization [125], and or bioprocess optimization [126]. The aim of this section is the examination of the applicability of semi-mechanistic models in industrial environment, namely how this model structure can be identified and applied for state estimation.

Problem description

The proposed approach is applied to the modeling and monitoring of a medium- and high-density polyethylene plant. The polymerization unit is controlled by a Honeywell Distributed Control System. Measurements are available *in every*

15 seconds on process variables which consist of input and output variables: $u_{k,(1,\ldots,8)}$ the comonomer, the monomer, the solvent and the chain transfer agent inlet flowrate and temperature, $u_{k,9}$ polymer production rate, $u_{k,10}$ the flowrate of the catalyzator, $u_{k,(11,\ldots,13)}$ cooling water flowrate, inlet and outlet temperature. The product quality $\mathbf{y}_k$ is determined by off-line laboratory analysis after drying the polymer that causes approximately *one hour time-delay*. The interval between the product samples is *between half and five hours*. Since, it would be useful to know if the product is good before testing it, the monitoring of the process would help in the early detection of poor-quality product. This section focuses on the melt index prediction. MI depends on the state variables which describe the behavior of the dynamic system, so for the development of a soft-sensor it is necessary to estimate these variables. There are additional state variables except these ones which must be identified in this example: the concentration of the comonomer in the instantly formulated polyethylene is x_{11} (it is needed for the MI prediction), and the melt index is also can be seen as a state variable x_{12}.

Semi-mechanistic model of the polymerization unit

Generally, models used in the state estimation of process systems are formulated by macroscopic balance equations, for instance, mass or energy balances. In general, not all of the terms in these equation are exactly or even partially known. In semi-mechanistic modeling black-box models are used to represent the otherwise difficult-to-obtain parts of the model. It is a very difficult question to choose the architecture of the applied black-box model in advance, and there is no general procedure to do that. There are various viewpoints that should be taken into account, e.g. the problem type, intended use, problem dimensionality, available amount and quality of data, memory restrictions, learning methods, experience of the user and availability of tools [127]. Because there is no a priori information about what type of black-box models should be used in case of the problem described above, feedforward neural network was chosen. From a didactical point of view, it could be better to choose normalized radial basis function networks (NRBF) because under some conditions they are equivalent to fuzzy models [127]. However, feedforward neural network with global, e.g. sigmoid type nonlinear activation function has the advantage that it could have better extrapolation capability than the NRBF because the later has only local activation function. (However, in general black-box models should not be used for extrapolation but in case of semi-mechanistic models, during the training it could be that it is needed.)

Usually, in the modeling phase it turns out which parts of the first principles model are easier and which are more laborious to obtain and often we can get the following hybrid model structure:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k, \mathbf{f}_{NN}(\mathbf{x}_k, \mathbf{u}_k), \theta)\,, \tag{A.21}$$

$$\mathbf{y}_k = g(\mathbf{x}_k, \mathbf{w}_k)\,, \tag{A.22}$$

where $\mathbf{x}_k, \mathbf{y}_k$ and $\mathbf{u}_k$ represents the states, the outputs and the inputs of the

system at kth time instant, $\mathbf{v}_k$ and $\mathbf{w}_k$ are noise variables.

The $\mathbf{f}_{NN} = [f_{NN,1}, \ldots, f_{NN,n}]^T$ represents the black-box elements of the model (neural networks) and θ the parameter set of $\mathbf{f}_{NN}$ represented by feed-forward multi-input single-output neural networks with one hidden layer and one output neuron:

$$f_{NN,i}(\mathbf{z}, \theta) = \mathbf{w}_2 \tanh(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1) + b_2 \,, \tag{A.23}$$

where nn represents the number of hidden neurons, $\mathbf{z} = [z_1, \ldots, z_{ni}]^T$ is the input of network $(ni \times 1)$, $\mathbf{W}_1$ is the weight of hidden layer $(nn \times ni)$, $\mathbf{b}_1$ is the bias of hidden layer $(nn \times 1)$, $\mathbf{w}_2$ is the weight of output layer $(1 \times nn)$, b_2 is the bias of output layer (1×1), so the θ denotes the set of parameters: $\theta = \{\mathbf{W}_1, \mathbf{w}_2, \mathbf{b}_1, b_2\}$.

The melt index of the instantly produced polyethylene is mainly depend on the current ethylene concentration in the loop reactor (x_4), the reactor temperature (x_7) and the concentration of the hexene in the instantly formulated polyethylene (x_{11}). These three variables and the other state variables can be calculated by a nonlinear state estimation algorithm based on the first-principle model of the system (see *FP1* in Figure A.15). The *BB1* box contains a black-box in which a neural network calculates the melt index of the instantaneously produced polyethylene ($\mathbf{f}_{NN}$). Since the produced polyethylene which leaves the reactor is the mixture of the previously produced products, the evolving MI can be calculated by the first-principle model of the mixing (see *FP2* in Figure A.15):

$$\frac{dx_{12}^{\xi}}{dt} = \frac{1}{x_2} \left(R \; \mathbf{f}_{NN}^{\xi}(x_4, x_7, x_{11}) - F_{PE}^{out} \; x_{12}^{\xi} - x_{12}^{\xi} \frac{dx_2}{dt} \right) \tag{A.24}$$

where $R = (x_8 x_4 x_1 x_6 x_2 + x_9 x_5 x_1 x_6 x_2 + x_{10} x_3 x_1 x_6 x_2)$ represents the instantaneously produced mass of the polyethylene, F_{PE}^{out} is the polymer mass leaving the reactor, and $\xi = -0.294$ is an empirical coefficient.

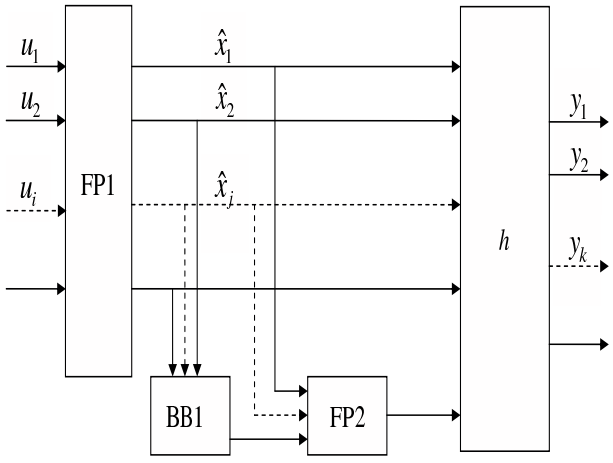


Figure A.15: The semi-mechanistic model of the system.

Spline-smoothing based identification of neural networks

To train the NN parts of the previously presented semi-mechanistic process model, pairs of input/output data should be used to determine the θ parameter set (weights of the NN) in such way that the sum of the squared deviations,

$$\mathbf{V}_N = \frac{1}{2N}\sum_{k=1}^{N}\left(y_k - \widetilde{y}_k\right)^2 \tag{A.25}$$

between the predicted output of network and the corresponding training data becomes minimal. The usual way to minimize $\mathbf{V}_N$ is to use gradient procedures, like Gauss-Newton algorithm. Weights in the ith step of this iterative process are changed in the direction of gradient.

$$\theta^{i+1} = \theta^i - \mu\mathbf{R}_i^{-1}\mathbf{V}'_N \tag{A.26}$$

where

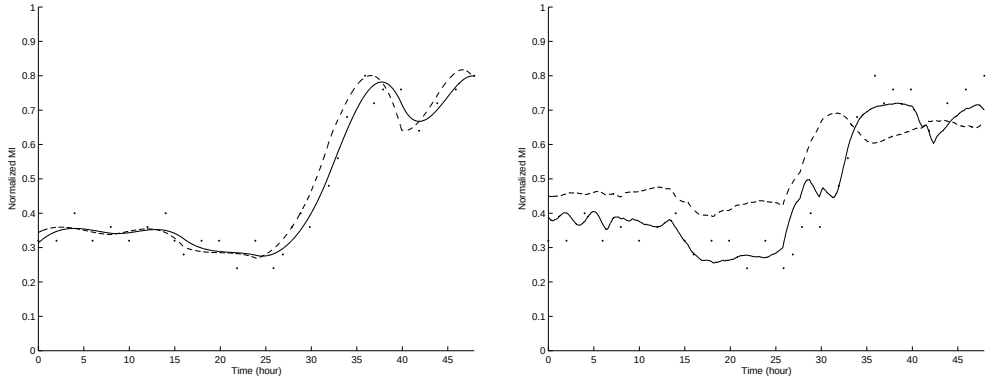
$$\begin{gathered}\mathbf{R}_i = \tfrac{1}{N}\textstyle\sum_{k=1}^{N}\mathbf{j}_{k,\theta}\mathbf{j}_{k,\theta}^T,\ \mathbf{V}'_N = -\tfrac{1}{N}\textstyle\sum_{k=1}^{N}\left(\mathbf{y}_k - \widetilde{\mathbf{y}}_k\right)\mathbf{j}_{k,\theta},\\ \mathbf{j}_{k,\theta} = \tfrac{\partial y_{k,\theta}}{\partial\theta} = \tfrac{\partial g_k}{\partial\mathbf{x}}\tfrac{\partial\mathbf{x}_{k,\theta}}{\partial\theta}.\end{gathered} \tag{A.27}$$

The key problem of the application of this approach is the determination of $\partial\mathbf{x}/\partial\theta$, because in semi-mechanistic models the NN's output does not appear explicitly in the above expression as it is part of the differential equation system. In this case NN can be trained by the integration of the sensitivity equations, using a Nonlinear Programming Technique, using Extended Kalman Filter for state and parameter estimation, or by using a spline-smoothing approach [126].

In this thesis the Hermite spine-smoothing method (see e.g. [128]) has been applied to interpolate between the measured data (shown in Figure A.16) and estimate the corresponding derivatives in the rearranged (A.24) to obtain the desired outputs of the neural network:

$$\mathbf{f}_{NN}(x_4, x_7, x_{11}) = \left(\frac{1}{R}\left(x_2\frac{dx_{12}^{\xi}}{dt} + F_{PE}^{out}\ x_{12}^{\xi} + x_{12}^{\xi}\frac{dx_2}{dt}\right)\right)^{1/\xi} \tag{A.28}$$

In the applied cubic splines, which are piecewise third-order polynomials, the polynomials are defined such that their values and first derivatives are continuous at so-called knots where the individual polynomials are interconnected. When such splines are identified, a continuous function is fitted to the available measured data, $\mathbf{x} = [x_1, \ldots, x_n]^T$ given at time instants $\mathbf{t} = [t_1, \ldots, t_n]^T$ (see Figure A.16).



(a) *(b)*

Figure A.16: (a) Spline-interpolation of MI (solid line) and current MI (dashed line), (b) Comparison the NN (solid line) and linear model (dashed line)

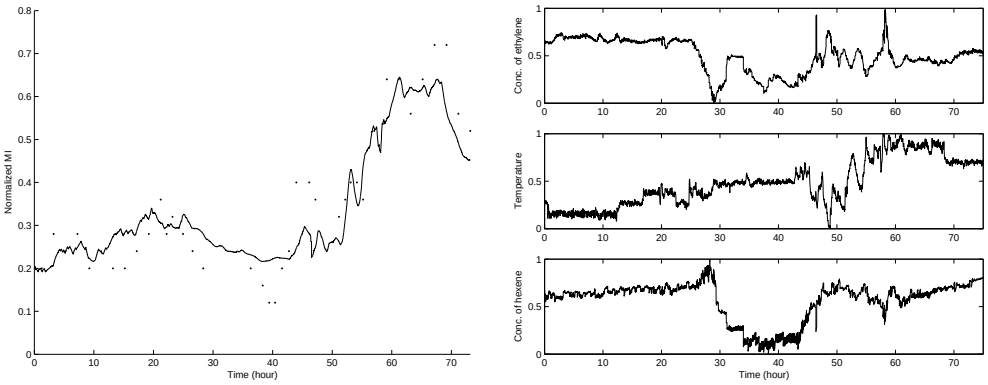
Example A.4. Application of semi-mechanistic model in real-time state-estimation

For the identification of the neural network four data sets that include the same product transition have been used. Each data set contains 80 hours of operation in which the product change starts around the 50th hour. Among the four data sets three were used for the training of the semi-mechanistic model and the other one for the validation. This ratio fits to the proposed ratio in [129], and it could improve the accuracy of the NN. The Levenberg-Marquardt algorithm was used for training the NN. The number of the hidden neurons, $nn = 4$, was estimated by applying the four cross-validation method. The results were compared with the results of a linear model identified based on the same datasets. The average validation mean square error (MSE) is equal to 0.0037 in case of NN with 4 hidden neurons, but this value is much worse, 0.0204 in case of the linear model. It can be determined that the linear model gives acceptable, but worse results than the NN in case of 'normal' operating conditions, i.e. without product changes. However, it cannot be handle and predict accurately product changes and unusual operations like changes in the catalyst activity because of changes from one catalyst tank to another etc.

The identified hybrid model was used in nonlinear state estimation. The Extended Kalman Filter is based on Taylor linearization of the state transition and output model equations. Instead of this solution, a more advanced state-estimation tool, the DD1 filter, has been used that is based on approximations of the model equations with a multivariable extension of Stirling's interpolation formula. This filter is simple to implement as no derivatives of the model equations are needed, yet it provides excellent accuracy [18]. For the feedback of the filter the $\mathbf{y}_k$ outputs of the system were chosen variables that are measured connected to the reactor and the product. Measurements are available on $y_{k,1}$ chain transfer agent, $y_{k,2}$ monomer and $y_{k,3}$ comonomer concentration in every 8 minutes, $y_{k,4}$ reactor temperature and $y_{k,5}$ density of the slurry in the reactor (which is connected with the mass of the fluid and the polymer in the reactor) in every 15 seconds. Measurements of $y_{k,6}$ melt index was mentioned above. As Figure **??** *shows, the*

resulted soft-sensor gives an excellent performance.

For the identification of the neural network four data set that include the same product transition has been used. Each data set contains 80 hours of operation in which the product change starts around the 50th hour. Among the four data sets three were used for the training of the semi-mechanistic model and the other one for the validation. The Levenberg-Marquardt algorithm was used for training the NN. The number of the hidden neurons, $nn = 4$, was estimated by applying the four cross-validation method. The results were compared with the results of a linear model identified based on the same datasets. The difference between the simple estimation performances of the linear and neural models can be seen on the right side of Figure A.16.



(a) *(b)*

Figure A.17: (a) Estimated MI given by the DD1 algorithm, (b) Estimated state variables used by the $\mathbf{f}_{NN}$ model.

The identified hybrid model was used in nonlinear state estimation based on the so-called DD1 filter. This filter is simple to implement as no derivatives of the model equations are needed, yet it provides excellent accuracy [122] (see also in Section A.2). For the feedback of the filter the $\mathbf{y}_k$ outputs of the system were chosen variables that are measured connected to the reactor and the product. Measurements are available on $y_{k,1}$ chain transfer agent, $y_{k,2}$ monomer and $y_{k,3}$ comonomer concentration in every 8 minutes, $y_{k,4}$ reactor temperature and $y_{k,5}$ density of the slurry in the reactor (which is connected with the mass of the fluid and the polymer in the reactor) in every 15 seconds. In this example, according to our purpose another variable has to be chosen as output: $y_{k,6}$ melt index because its measured value is available but not as frequent as other state variables' (see Section A.3 for more details). As Figure A.17 shows, the resulted soft-sensor gives an excellent performance.

□

Appendix B

Appendix: Theoretical Background

B.1 Introduction to Fuzzy Clustering

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. Since clusters can formally be seen as subsets of the data set, one possible classification of clustering methods can be according to whether the subsets are fuzzy or crisp (hard). Hard clustering methods are based on classical set theory, and require that an object either does or does not belong to a cluster. Hard clustering in a data set $\mathbf{X}$ means partitioning the data into a specified number of mutually exclusive subsets of $\mathbf{X}$. The number of subsets (clusters) is denoted by c. Fuzzy clustering methods allow objects to belong to several clusters simultaneously, with different degrees of membership. The data set $\mathbf{X}$ is thus partitioned into c fuzzy subsets. In many real situations, fuzzy clustering is more natural than hard clustering, as objects on the boundaries between several classes are not forced to fully belong to one of the classes, but rather are assigned membership degrees between 0 and 1 indicating their partial memberships. The discrete nature of hard partitioning also causes analytical and algorithmic intractability of algorithms based on analytic functionals, since these functionals are not differentiable.

The objective of clustering is to partition the data set $\mathbf{X}$ into c clusters. For the time being, assume that c is known, based on prior knowledge, for instance. Fuzzy and possibilistic partitions can be seen as a generalization of hard partition.

A $c \times N$ matrix $\mathbf{U} = [\mu_{i,k}]$ represents the fuzzy partitions, where $\mu_{i,k}$ denotes the degree of the membership of the $\mathbf{x}_k$-th observation belongs to the $1 \leq i \leq c$-th cluster, so the i-th row of $\mathbf{U}$ contains values of the *membership function* of the i-th fuzzy subset of $\mathbf{X}$. The matrix $\mathbf{U}$ is called the fuzzy partition matrix. Conditions for a fuzzy partition matrix are given by:

$$\mu_{i,k} \in [0,1],\ 1 \leq i \leq c,\ 1 \leq k \leq N, \tag{B.1}$$

$$\sum_{i=1}^{c} \mu_{i,k} = 1,\ 1 \leq k \leq N, \tag{B.2}$$

$$0 < \sum_{k=1}^{N} \mu_{i,k} < N,\ 1 \leq i \leq c. \tag{B.3}$$

Fuzzy partitioning space *Let* $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ *be a finite set and let* $2 \leq c < N$ *be an integer. The fuzzy partitioning space for* $\mathbf{X}$ *is the set*

$$M_{fc} = \{\mathbf{U} \in \mathbb{R}^{c \times N} | \mu_{i,k} \in [0,1], \forall i,k; \sum_{i=1}^{c} \mu_{i,k} = 1, \forall k; 0 < \sum_{k=1}^{N} \mu_{i,k} < N, \forall i\}. \tag{B.4}$$

(B.2) constrains the sum of each column to 1, and thus the total membership of each $\mathbf{x}_k$ in $\mathbf{X}$ equals one. The distribution of memberships among the c fuzzy subsets is not constrained.

A large family of fuzzy clustering algorithms is based on minimization of the *fuzzy c-means* objective function formulated as:

$$J(\mathbf{X}; U, V) = \sum_{i=1}^{c} \sum_{k=1}^{N} (\mu_{i,k})^m \|\mathbf{x}_k - \mathbf{v}_i\|_{\mathbf{A}}^2 \tag{B.5}$$

where $\mathbf{U} = [\mu_{i,k}]$ is a fuzzy partition matrix of $\mathbf{X}$,

$$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c], \ \mathbf{v}_i \in \mathbb{R}^n \tag{B.6}$$

is a matrix of *cluster prototypes* (centers), which have to be determined,

$$D_{i,k\mathbf{A}}^2 = \|\mathbf{x}_k - \mathbf{v}_i\|_{\mathbf{A}}^2 = (\mathbf{x}_k - \mathbf{v}_i)^T \mathbf{A} (\mathbf{x}_k - \mathbf{v}_i) \tag{B.7}$$

is a squared inner-product distance norm where $\mathbf{A}$ is the distance measure, and $m \in \langle 1, \infty)$ is a weighting exponent which determines the fuzziness of the resulting clusters. The measure of dissimilarity in (B.5) is the squared distance between each data point $\mathbf{x}_k$ and the cluster prototype $\mathbf{v}_i$. This distance is weighted by the power of the membership degree of that point $(\mu_{i,k})^m$. The value of the cost function (B.5) is a measure of the total weighted within-group squared error incurred by the representation of the c clusters defined by their prototypes $\mathbf{v}_i$. Statistically, (B.5) can be seen as a measure of the total variance of $\mathbf{x}_k$ from $\mathbf{v}_i$. The minimization of the c-means functional (B.5) represents a nonlinear optimization problem that can be solved by using a variety of available methods, ranging from grouped coordinate minimization, over simulated annealing to genetic algorithms. The most popular method, however, is a simple Picard iteration through the first-order conditions for stationary points of (B.5), known as the fuzzy c-means (FCM) algorithm. The stationary points of the objective function (B.5) can be found by adjoining the constraint (B.2) to J by means of Lagrange multipliers:

$$\overline{J}(\mathbf{X}; U, V, \lambda) = \sum_{i=1}^{c} \sum_{k=1}^{N} (\mu_{i,k})^m D_{i,k\mathbf{A}}^2 + \sum_{k=1}^{N} \lambda_k \left(\sum_{i=1}^{c} \mu_{i,k} - 1 \right), \tag{B.8}$$

and by setting the gradients of $\overline{J}$ with respect to $\mathbf{U}, V$ and λ to zero. If $D_{i,k\mathbf{A}}^2 > 0, \forall i,k$ and $m > 1$, then $(\mathbf{U}, V) \in M_{fc} \times \mathbb{R}^{n \times c}$ may minimize (B.5) only if

$$\mu_{i,k} = \frac{1}{\sum_{j=1}^{c} \left(D_{i,k\mathbf{A}} / D_{j,k\mathbf{A}} \right)^{2/(m-1)}}, \quad 1 \leq i \leq c, \ 1 \leq k \leq N, \tag{B.9}$$

and

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} \mu_{i,k}^m \mathbf{x}_k}{\sum_{k=1}^{N} \mu_{i,k}^m}, \quad 1 \leq i \leq c. \tag{B.10}$$

This solution also satisfies the remaining constraints (B.1) and (B.3). Note that equation (B.10) gives $\mathbf{v}_i$ as the weighted mean of the data items that belong to a cluster, where the weights are the membership degrees. That is why the algorithm is called "c-means". One can see that the FCM algorithm is a simple iteration through (B.9) and (B.10) (see Algorithm B.1.1).

A common limitation of clustering algorithms based on a fixed distance norm is that such a norm induces a fixed topological structure on $\mathbb{R}^n$ and forces the objective function to prefer clusters of that shape even if they are not present. Generally, different matrices A_i are required for the different clusters, but there is no guideline as to how to choose them a priori. The norm-inducing matrix A can be adapted by using estimates of the data covariance, and can be used to estimate the statistical dependence of the data in each cluster. The Gustafson–Kessel algorithm (GK) and the fuzzy maximum likelihood estimation algorithm (Gath–Geva algorithm (GG)) are based on an adaptive distance measure.

The fuzzy maximum likelihood estimates clustering algorithm employs a distance norm based on the fuzzy maximum likelihood estimates, proposed by Bezdek and Dunn [70]:

$$D_{i,k}(\mathbf{x}_k, \mathbf{v}_i) = \frac{(2\pi)^{\left(\frac{n}{2}\right)} \sqrt{\det(\mathbf{F}_i)}}{\alpha_i} \exp\left(\frac{1}{2} \left(\mathbf{x}_k - \mathbf{v}_i\right)^T \mathbf{F}_i^{-1} \left(\mathbf{x}_k - \mathbf{v}_i\right)\right) \tag{B.11}$$

Note that, contrary to the GK algorithm, this distance norm involves an exponential term and thus decreases faster than the inner-product norm. $\mathbf{F}_i$ denotes the fuzzy covariance matrix of the i-the cluster. The α_i is the prior probability of selecting cluster i, given by: $\alpha_i = \frac{1}{N} \sum_{k=1}^{N} \mu_{i,k}$. The membership degrees $\mu_{i,k}$ are interpreted as the posterior probabilities of selecting the i-th cluster given the data point $\mathbf{x}_k$. Gath and Geva reported that the fuzzy maximum likelihood estimates clustering algorithm is able to detect clusters of varying shapes, sizes and densities. This is because the cluster covariance matrix is used in conjunction with an "exponential" distance, and the clusters are not constrained in volume. However, this algorithm is less robust in the sense that it needs a good initialization, since due to the exponential distance norm, it converges to a near local optimum. The minimum of (B.5) is sought by the alternating optimization (AO) method (Gath–Geva clustering algorithm) given in Algorithm B.1.2:

Algorithm B.1.1 (Fuzzy c-Means).

Given the data set $\mathbf{X}$*, choose the number of clusters* $1 < c < N$*, the weighting exponent* $m > 1$*, the termination tolerance* $\epsilon > 0$ *and the norm-inducing matrix* $\mathbf{A}$*. Initialize the partition matrix randomly, such that* $\mathbf{U}^{(0)} \in M_{fc}$*.*

Repeat *for* $l = 1, 2, \ldots$

Step 1 *Compute the cluster prototypes (means):*

$$\mathbf{v}_i^{(l)} = \frac{\sum_{k=1}^{N} (\mu_{i,k}^{(l-1)})^m \mathbf{x}_k}{\sum_{k=1}^{N} (\mu_{i,k}^{(l-1)})^m}, \; 1 \leq i \leq c. \tag{B.12}$$

Step 2 *Compute the distances:*

$$D_{i,k\mathbf{A}}^2 = (\mathbf{x}_k - \mathbf{v}_i)^T \mathbf{A} (\mathbf{x}_k - \mathbf{v}_i), \quad 1 \leq i \leq c, \quad 1 \leq k \leq N. \tag{B.13}$$

Step 3 *Update the partition matrix:*

$$\mu_{i,k}^{(l)} = \frac{1}{\sum_{j=1}^{c} \left(D_{i,k\mathbf{A}} / D_{j,k\mathbf{A}} \right)^{2/(m-1)}}. \tag{B.14}$$

until $||\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}|| < \epsilon$.

Algorithm B.1.2 (Gath–Geva Algorithm).

Given a set of data $\mathbf{X}$ *specify* c*, choose a weighting exponent* $m > 1$ *and a termination tolerance* $\epsilon > 0$*. Initialize the partition matrix such that* (B.1), (B.2) *and* (B.3) *holds.*

Repeat *for* $l = 1, 2, \ldots$

Step 1 *Calculate the cluster centers:* $\mathbf{v}_i^{(l)} = \frac{\sum_{k=1}^{N} (\mu_{i,k}^{(l-1)})^m \mathbf{x}_k}{\sum_{k=1}^{N} (\mu_{i,k}^{(l-1)})^m}, \; 1 \leq i \leq c$

Step 2 *Compute the distance measure* $D_{i,k}^2$*.*

The distance to the prototype is calculated based the fuzzy covariance matrices of the cluster

$$\mathbf{F}_i^{(l)} = \frac{\sum_{k=1}^{N} (\mu_{i,k}^{(l-1)})^m \left(\mathbf{x}_k - \mathbf{v}_i^{(l)} \right) \left(\mathbf{x}_k - \mathbf{v}_i^{(l)} \right)^T}{\sum_{k=1}^{N} (\mu_{i,k}^{(l-1)})^m}, \; 1 \leq i \leq c \tag{B.15}$$

The distance function is chosen as

$$D_{i,k}^2(\mathbf{x}_k, \mathbf{v}_i) = \frac{(2\pi)^{\left(\frac{n}{2}\right)} \sqrt{\det(\mathbf{F}_i)}}{\alpha_i} \exp\left(\frac{1}{2} \left(\mathbf{x}_k - \mathbf{v}_i^{(l)} \right)^T \mathbf{F}_i^{-1} \left(\mathbf{x}_k - \mathbf{v}_i^{(l)} \right) \right) \tag{B.16}$$

with the a priori probability $\alpha_i = \frac{1}{N} \sum_{k=1}^{N} \mu_{i,k}$

Step 3 *Update the partition matrix*

$$\mu_{i,k}^{(l)} = \frac{1}{\sum_{j=1}^{c} \left(D_{i,k}(\mathbf{x}_k, \mathbf{v}_i) / D_{j,k}(\mathbf{x}_k, \mathbf{v}_j) \right)^{2/(m-1)}}, \quad 1 \leq i \leq c, \; 1 \leq k \leq N \, . \tag{B.17}$$

until $||\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}|| < \epsilon$.

Example B.1 (Demonstration for Gath–Geva algorithm)**.** *In Figure B.1 the effect of the different cluster size can be seen as the Gath-GevaG algorithm can cluster the data perfectly.*

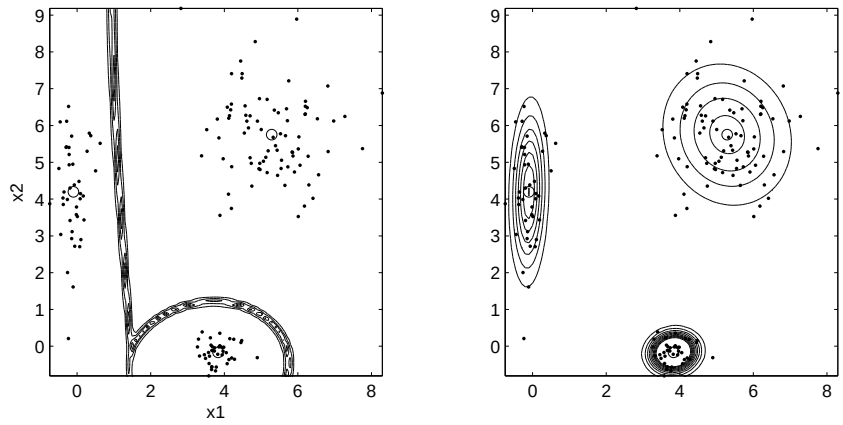


Figure B.1: The results of the Gath–Geva algorithm by the synthetic data set.

In Figure B.2 the results obtained by the clustering of the well-known motorcycle dataset is shown where two different initializations are compared. On the left part of the membership values and the distances obtained by random initialization are shown, while on the remaining two subplots the results of the clustering initialized by fuzzy c-means are shown. As the difference between these two plots shows, the GG algorithm is very sensitive to initialization, and it can adapt the distance norm to the underlying distribution of the data which is reflected in the different sizes of the clusters.

The motorcycle data set can be considered a function with additional noise. Therefore the human analyzer finds better the results shown on the second two diagrams of Figure B.2. However, the value of the fuzzy objective function (B.5) *is bigger in the second case than in the first case.*

□

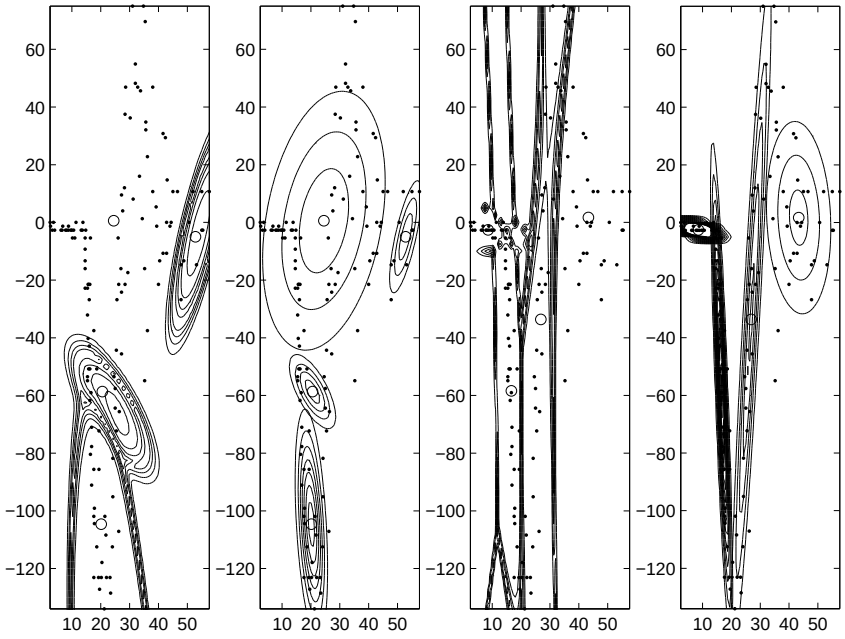


Figure B.2: The results of the Gath–Geva algorithm by the motorcycle data set.

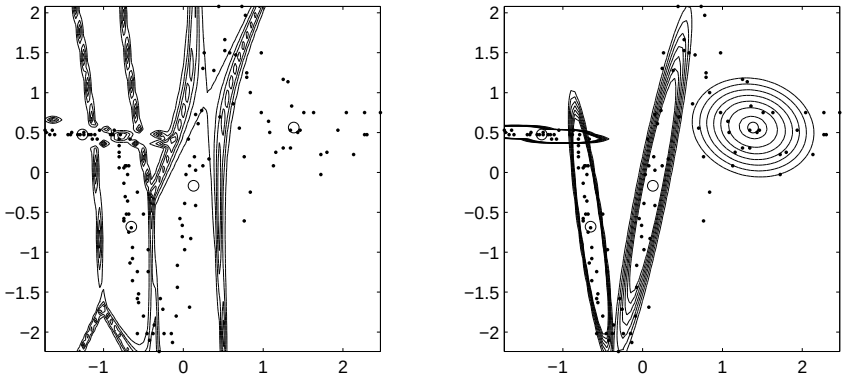


Figure B.3: The results of the Gath–Geva algorithm by the motorcycle data set with normalization.

B.2 Introduction to Fuzzy Modeling

For many real world applications a great deal of information is provided by human experts, who do not reason in terms of mathematics but instead describe the system verbally through vague or imprecise statements like,

If The *Temperature* is *Big* **then** The *Pressure* is *High* (B.18)

Because so much human knowledge and expertise come in terms of verbal rules, one of the sound engineering approaches is to try to integrate such linguistic information into the modeling process. A convenient and common approach of doing this is to use fuzzy logic concepts to cast the verbal knowledge into a conventional mathematics representation (model structure), which subsequently can be fine-tuned using input-output data.

A fuzzy model is a computation framework based on the concepts of fuzzy sets, fuzzy if-then rules, and fuzzy reasoning. This section will present detailed information about particular fuzzy models which are used in this thesis. It will not attempt to provide a broad survey of the field. For such a survey the reader is referred to "An Introduction to Fuzzy Control" by Driankov, Hellendoorn, and Reinfrank [130] or "Fuzzy Control" by K.M. Passino and S. Yurkovic [131], or "A course in Fuzzy Systems and Control" by L.X. Wang [132].

In fuzzy set theory, a precise representation of imprecise knowledge is not enforced since strict limits of a set are not required to be defined, instead a *membership function* is defined. A membership function describes the relationship between a variable and the degree of membership of the fuzzy set that correspond to particular values of that variable. This degree of membership is usually defined in terms of a number between 0 and 1, inclusive, where 0 implies total absence of membership, 1 implies complete membership, and any value in between implies partial membership of the fuzzy set. This may be written as follows: $A(x) \in [0,1] \quad for \quad x \in U$ where $A(\cdot)$ is the membership function and U is the *universe of discourse* which defines the total range of interest over which the variable x should be defined. For example, to define membership of the fuzzy set, *hot*, a function which rises from 0 to 1 over the range $15°C$ to $25°C$ may be used, i.e.

$$A(x) = \begin{cases} 0 & x < 15°C \\ \frac{x-15}{10} & 15 \geq x \geq 25°C \\ 1 & x > 25°C \end{cases}$$

While seeming imprecise to a human being, fuzzy sets are mathematically precise in that they can be fully represented by exact numbers. They can therefore be seen as a method of tieing together human and machine knowledge representations. The basic configuration of a fuzzy model is shown in Figure B.4. As it is depicted, the fuzzy model involves the following components [133]:

- *Data preprocessing.* The physical values of the input of the fuzzy system may differ significantly in magnitude. By mapping these to proper normalized (but interpretable) domains via scaling, one can instead work with signals roughly are of the same magnitude, which is desirable from an estimation point of view.

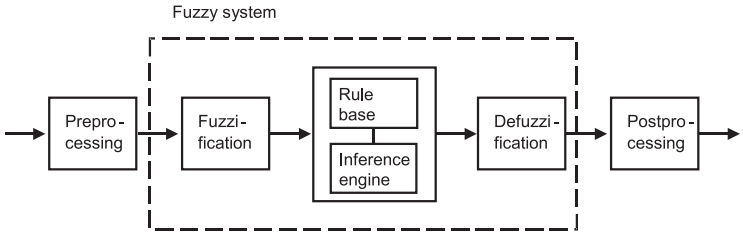


Figure B.4: Structure of a fuzzy system.

- *Fuzzification.* Fuzzification maps the crisp values of the preprocessed input of the model into suitable fuzzy sets represented by *membership functions* (MF). As the antecedent and consequent fuzzy sets take on linguistic meanings such as "high temperature" they are called linguistic labels of the sets of linguistic variables.

- *Rule base.* The rule base is the cornerstone of the fuzzy model. The expert knowledge, which is assumed to be given as a number of if-then rules, is stored in a fuzzy rule base. In rule-based fuzzy systems, the relationship between variables are represented by means of If-Then rules of the following general form:

  If antecedent proposition **then** consequent proposition (B.19)

  This thesis deals with this *Takagi-Sugeno (TS) fuzzy models* where the consequent is a crisp function of the input variables, $f_j(\mathbf{x})$, rather than a fuzzy proposition [28].

  $$R_j: \ \textbf{If}\ x_1 \text{ is } A_{1,j} \ \textbf{and}\ \ldots\ \textbf{and}\ x_n \text{ is } A_{n,j} \ \textbf{then}\ y = f_j(\mathbf{x}) \tag{B.20}$$

- *Inference engine.*

  The inference mechanism or inference engine is the computational method which calculates the degree to which each rule *fires* for a given fuzzified input pattern by considering the rule and label sets. A rule is said to fire when the conditions upon which it depends occur. Since these conditions are defined by fuzzy sets which have degrees of membership, a rule will have a degree of firing or *firing strength*, β_j. The firing strength is determined by the mechanism which is used to implement the *and* in the expression (B.20); in this book the product of the degrees of membership will be used, that is:

  $$\beta_j = \prod_{i=1}^{n} A_{i,j} \tag{B.21}$$

  where $A_{i,j}$ defines the membership function on input i is used in rule j. Again, there are different methods for implementing each of the logical operators and the reader is referred to [130] for details on these.

- *Defuzzification.* A defuzzifier compiles the information provided by each of the rules and makes a decision from this basis. In linguistic fuzzy models the defuzzification converts the resulted fuzzy sets defined by the inference engine to the output of the model to a standard crisp signal. The method which is used in this book is the method commonly called the centre-of-gravity or *centroid* method. In case of TS fuzzy models it is described by the following equation:

$$y = \frac{\sum_{j=1}^{N_r} \beta_j f_j(\mathbf{x})}{\sum_{j=1}^{N_r} \beta_j} \tag{B.22}$$

  It can be seen that the centroid method of defuzzification takes a weighted sum of the designated consequences of the rules according to the firing strengths of the rules. There are numerous other types of defuzzifiers such as centre-of-sums, first-of-maxima, and middle-of-maxima [130].

- *Postprocessing.* The preprocessing step gives the output of the fuzzy system based on the crisp signal obtained after defuzzification. This often means the scaling of the output.

This thesis mainly deals with a Takagi-Sugeno (TS) fuzzy model proposed by Takagi, Sugeno, and Kang [28, 29] to develop a systematic approach for generating fuzzy rules from a given input-output data set. In the following the structure of this model and the related modeling paradigms will be presented.

The TS model is a combination of a logical and a mathematical model. This model is also formed by logical rules; that consists of a fuzzy antecedent and a mathematical function as consequent part. The antecedents of fuzzy rules partition the input space into a number of fuzzy regions, while the consequent functions describe the system behavior within a given region:

$$\begin{gathered} R_j: \textbf{ If } z_1 \text{ is } A_{1,j} \textbf{ and } \ldots \textbf{ and } z_n \text{ is } A_{n,j} \textbf{ then} \\ y = f_j\left(q_1, \ldots, q_m\right) \end{gathered} \tag{B.23}$$

where $\mathbf{z} = [z_1, \ldots, z_n]^T$ is the n-dimensional vector of the antecedent variables, $\mathbf{z} \in \mathbf{x}$, $\mathbf{q} = [q_1, \ldots, q_m]^T$ is the m-dimensional vector of the consequent variables $\mathbf{q} \in \mathbf{x}$, where $\mathbf{x}$ denotes the set of all inputs of the $y = f(\mathbf{x})$ model. $A_{i,j}(z_i)$ denotes the antecedent fuzzy set for the i-th input. The antecedents of fuzzy rules partition the input space into a number of fuzzy regions, while the $f_j(\mathbf{q})$ consequent functions describe the system behavior within a given region.

The spirit of fuzzy inference systems resembles that of 'divide and conquer' concept – the antecedent of fuzzy rules partition the input-space into a number of local fuzzy regions, while the consequents describe the behavior within a given region via various constituents [30].

This is based on the fact that while it may be difficult to find a model to describe the unknown system globally, it is often possible to construct local linear models around selected operating points. The modeling framework that is based on combining local models valid in predefined operating regions is called

operating regime-based modeling [98]. In this framework, the model is generally given by:

$$\hat{y} = \sum_{i=1}^{c} \phi_i(\mathbf{x}) \left(\mathbf{a}_i^T \mathbf{x} + b_i \right) \tag{B.24}$$

where $\phi_i(\mathbf{x})$ is the validity function for the ith operating regime and $\theta_i = [\mathbf{a}_i^T b_i]^T$ is the parameter vector of the corresponding local linear model. The operating regimes can also be represented by fuzzy sets in which case the Takagi–Sugeno fuzzy model is obtained [28]:

$$R_i: \ \textbf{If } \mathbf{x} \text{ is } \mathbf{A}_i(\mathbf{x}) \textbf{ then } \hat{y} = \mathbf{a}_i^T \mathbf{x} + b_i, \ [w_i] \qquad i = 1, \ldots, c\,. \tag{B.25}$$

Here, $\mathbf{A}_i(\mathbf{x})$ is a multivariable membership function, $\mathbf{a}_i$ and b_i are parameters of the local linear model, and $w_i \in [0, 1]$ is the weight of the rule. The value of w_i is usually chosen by the designer of the fuzzy system to represent the belief in the accuracy of the i-th rule. When such knowledge is not available $w_i = 1, \forall\, i$ is used.

The antecedent proposition "$\mathbf{x}$ is $\mathbf{A}_i(\mathbf{x})$" can be expressed as a logical combination of propositions with univariate fuzzy sets defined for the individual components of $\mathbf{x}$, usually in the following conjunctive form:

$$R_i: \ \textbf{If } x_1 \text{ is } A_{i,1}(x_1) \textbf{ and } \ldots \textbf{ and } x_n \text{ is } A_{i,n}(x_n) \textbf{ then } \hat{y} = \mathbf{a}_i^T \mathbf{x} + b_i, \ [w_i]\,. \tag{B.26}$$

The *degree of fulfillment* of the rule is then calculated as the product of the individual membership degrees and the rule's weight:

$$\beta_i(\mathbf{x}) = w_i \mathbf{A}_i(\mathbf{x}) = w_i \prod_{j=1}^{n} A_{i,j}(x_j)\,. \tag{B.27}$$

The rules are aggregated by using the fuzzy-mean formula

$$\hat{y} = \frac{\sum_{i=1}^{c} \beta_i(\mathbf{x}) \left(\mathbf{a}_i^T \mathbf{x} + b_i \right)}{\sum_{i=1}^{c} \beta_i(\mathbf{x})}\,. \tag{B.28}$$

B.3 Fuzzy Model Structures for Classification

Classical Bayes Classifier

The identification of a classifier system means the construction of a model that predicts the class $y_k = \{c_1, \ldots, c_C\}$ to which pattern $\mathbf{x}_k = [x_{1,k}, \ldots, x_{n,k}]^T$ should be assigned. The classic approach for this problem with C classes is based on Bayes' rule. The probability of making an error when classifying an example $\mathbf{x}$ is minimized by Bayes' decision rule of assigning it to the class with the largest *a posteriori* probability:

$$\mathbf{x} \text{ is assigned to } c_i \Longleftrightarrow p(c_i|\mathbf{x}) \geq p(c_j|\mathbf{x})\, \forall j \neq i \tag{B.29}$$

The *a posteriori* probability of each class given a pattern $\mathbf{x}$ can be calculated based on the $p(\mathbf{x}|c_i)$ class conditional distribution, which models the density of the data belonging to the class c_i, and the $P(c_i)$ class prior, which represents the probability that an arbitrary example out of data belongs to class c_i

$$p(c_i|\mathbf{x}) = \frac{p(\mathbf{x}|c_i)P(c_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|c_i)P(c_i)}{\sum_{j=1}^{C} p(\mathbf{x}|c_j)P(c_j)} \tag{B.30}$$

As (B.29) can be rewritten using the numerator of (B.30)

$$\mathbf{x} \text{ is assigned to } c_i \Longleftrightarrow p(\mathbf{x}|c_i)P(c_i) \geq p(\mathbf{x}|c_j)P(c_j)\, \forall j \neq i\,, \tag{B.31}$$

we would have an optimal classifier if we would perfectly estimate the class priors and the class conditional densities.

In practice one needs to find approximate estimates of these quantities on a finite set of training data $\{\mathbf{x}_k, y_k\}$, $k = 1, \ldots, N$. Priors $P(c_i)$ are often estimated on the basis of the training set as the proportion of samples of class c_i or using prior knowledge. The $p(\mathbf{x}|c_i)$ class conditional densities can be modeled with non-parametric methods like histograms, nearest-neighbors or parametric methods such as mixture models.

A special case of Bayes classifiers is the quadratic classifier, where the $p(\mathbf{x}|c_i)$ distribution generated by the class c_i is represented by a Gaussian function

$$p(\mathbf{x}|c_i) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\mathbf{F}_i)}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{v}_i)^T \mathbf{F}_i^{-1}(\mathbf{x} - \mathbf{v}_i)\right) \tag{B.32}$$

where $\mathbf{v}_i = [v_{1,i}, \ldots, v_{n,i}]^T$ denotes the center of the i-th multivariate Gaussian and $\mathbf{F}_i$ stands for a covariance matrix of the data of the class c_i. In this case, the (B.31) classification rule can be reformulated based on a distance measure. The sample $\mathbf{x}_k$ is classified to the class that minimizes the $d^2(\mathbf{x}_k, \mathbf{v}_i)$ distance, where the distance measure is inversely proportional to the probability of the data:

$$d^2(\mathbf{x}_k, \mathbf{v}_i) = \left(\frac{P(c_i)}{(2\pi)^{n/2}\sqrt{\det(\mathbf{F}_i)}} \exp\left(-\frac{1}{2}(\mathbf{x}_k - \mathbf{v}_i)^T \mathbf{F}_i^{-1}(\mathbf{x}_k - \mathbf{v}_i)\right)\right)^{-1} \tag{B.33}$$

Classical Fuzzy Classifier

The classical fuzzy rule-based classifier consists of fuzzy rules each one describing one of the C classes. The rule antecedent defines the operating region of the rule in the n-dimensional feature space and the rule consequent is a crisp (non-fuzzy) class label from the $\{c_1, \ldots, c_C\}$ label set:

$$r_i: \quad \textbf{If } x_1 \text{ is } A_{i,1}(x_{1,k}) \textbf{ and } \ldots x_n \text{ is } A_{i,n}(x_{n,k}) \textbf{ then } \hat{y} = c_i, \ \ [w_i] \tag{B.34}$$

where $A_{i,1}, \ldots, A_{i,n}$ are the antecedent fuzzy sets and w_i is a certainty factor that represents the desired impact of the rule. The value of w_i is usually chosen by the designer of the fuzzy system according to his or her belief in the accuracy of the rule. When such knowledge is not available, w_i is fixed to value 1 for any i.

The **and** connective is modeled by the product operator allowing for interaction between the propositions in the antecedent. Hence, the degree of activation of the ith rule is calculated as:

$$\beta_i(\mathbf{x}_k) = w_i \prod_{j=1}^{n} A_{i,j}(x_{j,k}) \tag{B.35}$$

The output of the classical fuzzy classifier is determined by the *winner takes all* strategy, i.e. the output is the class related to the consequent of the rule that gets the highest degree of activation:

$$\hat{y}_k = c_{i^*} \ , \ i^* = \underset{1 \le i \le C}{\arg\max} \ \beta_i(\mathbf{x}_k) \tag{B.36}$$

To represent the $A_{i,j}(x_{j,k})$ fuzzy set, we use Gaussian membership functions

$$A_{i,j}(x_{j,k}) = \exp\left(-\frac{1}{2}\frac{(x_{j,k} - v_{i,j})^2}{\sigma_{i,j}^2}\right) \tag{B.37}$$

where $v_{i,j}$ represents the center and $\sigma_{i,j}^2$ stands for the variance of the Gaussian function. The use of Gaussian membership function allows for the compact formulation of (B.35):

$$\beta_i(\mathbf{x}_k) = w_i \mathbf{A}_i(\mathbf{x}_k) = w_i \exp\left(-\frac{1}{2}(\mathbf{x}_k - \mathbf{v}_i)^T \mathbf{F}_i^{-1}(\mathbf{x}_k - \mathbf{v}_i)\right) \tag{B.38}$$

where $\mathbf{v}_i = [v_{1,i}, \ldots, v_{n,i}]^T$ denotes the center of the i-th multivariate Gaussian and $\mathbf{F}_i$ stands for a diagonal matrix that contains the $\sigma_{i,j}^2$ variances.

The fuzzy classifier defined by the previous equations is in fact a quadratic Bayes classifier when $\mathbf{F}_i$ in (B.32) contains only diagonal elements (variances).

In this case, the $\mathbf{A}_i(\mathbf{x})$ membership functions and the w_i certainty factors can be calculated from the parameters of the Bayes classifier following equations (B.32) and (B.38) as

$$\mathbf{A}_i(\mathbf{x}) = p(\mathbf{x}|c_i)(2\pi)^{n/2}\sqrt{\det(\mathbf{F}_i)}, \ \ w_i = \frac{P(c_i)}{(2\pi)^{n/2}\sqrt{\det(\mathbf{F}_i)}}\,. \tag{B.39}$$

Bayes Classifier based on Mixture of Density Models

One of the possible extensions of the classical quadratic Bayes classifier is to use mixture of models for estimating the class-conditional densities. The usage of mixture models in Bayes classifiers is not so widespread [39]. In these solutions each conditional density is modeled by a separate mixture of models. A possible criticism of such Bayes classifiers is that in a sense they are modeling too much: for each class many aspects of the data are modeled which may or may not play a role in discriminating between the classes.

In this section a new approach is presented. The $p(c_i|\mathbf{x})$ posteriori densities are modeled by $R > C$ mixture of models (clusters)

$$p(c_i|\mathbf{x}) = \sum_{l=1}^{R} p(r_l|\mathbf{x})P(c_i|r_l) \tag{B.40}$$

where $p(r_l|\mathbf{x})$ represents the *a posteriori* probability of $\mathbf{x}$ has been generated by the r_l-th local model and $P(c_i|r_l)$ denotes the *prior* probability of this model represents the class c_i.

Similarly to (B.30) $p(r_l|\mathbf{x})$ can be written as

$$p(r_i|\mathbf{x}) = \frac{p(\mathbf{x}|r_i)P(r_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|r_i)P(r_i)}{\sum_{j=1}^{R} p(\mathbf{x}|r_j)P(r_j)} \tag{B.41}$$

By using this mixture of density models the posteriori class probability can be expressed following equations (B.30), (B.40) and (B.41) as

$$p(c_i|\mathbf{x}) = \frac{p(\mathbf{x}|c_i)P(c_i)}{p(\mathbf{x})} = \sum_{l=1}^{R} \frac{p(\mathbf{x}|r_i)P(r_i)}{\sum\limits_{j=1}^{R} p(\mathbf{x}|r_j)P(r_j)} P(c_i|r_l) = \frac{\sum\limits_{l=1}^{R} p(\mathbf{x}|r_i)P(r_i)P(c_i|r_l)}{p(\mathbf{x})} \tag{B.42}$$

The Bayes decision rule can be thus formulated similarly to (B.31) as

$$\begin{gathered} \mathbf{x} \text{ is assigned to } c_i \iff \\ \sum_{l=1}^{R} p(\mathbf{x}|r_l)P(r_l)P(c_i|r_l) \geq \sum_{l=1}^{R} p(\mathbf{x}|r_l)P(r_l)P(c_j|r_l)\, \forall j \neq i \end{gathered} \tag{B.43}$$

where the $p(\mathbf{x}|r_l)$ distribution is represented by Gaussians similarly to (B.32).

Extended Fuzzy Classifier

A new fuzzy model that is able to represent Bayes classifier defined by (B.43) can be obtained. The idea is to define the consequent of the fuzzy rule as the probabilities of the given rule represents the $c_1, \ldots, c_C$ classes:

$$\begin{aligned} r_i: \quad & \textbf{If } x_1 \text{ is } A_{i,1}(x_{1,k}) \textbf{ and } \ldots x_n \text{ is } A_{i,n}(x_{n,k}) \textbf{ then} \\ & \hat{y}_k = c_1 \text{ with } P(c_1|r_i) \ \ldots, \hat{y}_k = c_C \text{ with } P(c_C|r_i) \ [w_i] \end{aligned} \tag{B.44}$$

Similarly to Takagi-Sugeno fuzzy models [28], the rules of the fuzzy model are aggregated using the normalized fuzzy mean formula and the output of the classifier is determined by the label of the class that has the highest activation:

$$\hat{y}_k = c_{i^*} \ , \ i^* = \arg\max_{1 \le i \le C} \frac{\sum_{l=1}^{R} \beta_l(\mathbf{x}_k) P(c_i|r_l)}{\sum_{i=1}^{R} \beta_l(\mathbf{x}_k)} \tag{B.45}$$

where $\beta_l(\mathbf{x}_k)$ has the meaning expressed by (B.35).

As the previous equation can be rewritten using only its numerator, the obtained expression is identical to the Gaussian mixtures of Bayes classifiers (B.43) when similarly to (B.39) the parameters of the fuzzy model are calculated as

$$\mathbf{A}_i(\mathbf{x}) = p(\mathbf{x}|r_i)(2\pi \det(\mathbf{F}_i))^{n/2}, \ w_i = \frac{P(r_i)}{(2\pi \det(\mathbf{F}_i))^{n/2}} \, , \tag{B.46}$$

The main advantage of the previously presented classifier is that the fuzzy model can consist of more rules than classes and every rule can describe more than one class. Hence, as a given class will be described by a set of rules, it should not be a compact geometrical object (hyper-ellipsoid).

Fuzzy Decision Tree for Classification

Using not only crisp but also fuzzy predicates, decision trees can be used to model vague decisions. The basic idea of fuzzy decision trees is to combine example based learning in decision trees with approximative reasoning of fuzzy logic [134]. This hybridization integrates the advantages of both methodologies compact knowledge representation of decision trees with the ability of fuzzy systems to process uncertain and imprecise information. Viewing fuzzy decision trees as a compressed representation of a (fuzzy) rule set, enables us to use decision trees not only for classification, but also for approximation of continuous output functions.

An example of how a fuzzy decision tree can be used for the compressed representation of a fuzzy rule base is given in Figure B.5, where the rule defined by the dashed path of the tree is the following:

If x_3 is $large$ **and** x_2 is $medium$ **and** x_1 is $small$ **and** x_5 is $medium$ **then** C_1 (B.47)

ID3 and its fuzzy variant (FID) assume discrete and fuzzy domains with small cardinalities. This is a great advantage as it increases comprehensibility of the induced knowledge, but may require an *a priori* partitioning of the numerical attributes (see the bottom of Figure B.5 for the illustration of such fuzzy partitioning). Since this partitioning has significant effect to the performance of the generated model, recently some research has been done in the area of domain partitioning while constructing a symbolic decision tree. For example, Dynamic-ID3 [135] clusters multivalued ordered domains, and Assistant [136] produces binary trees by clustering domain values (limited to domains of small

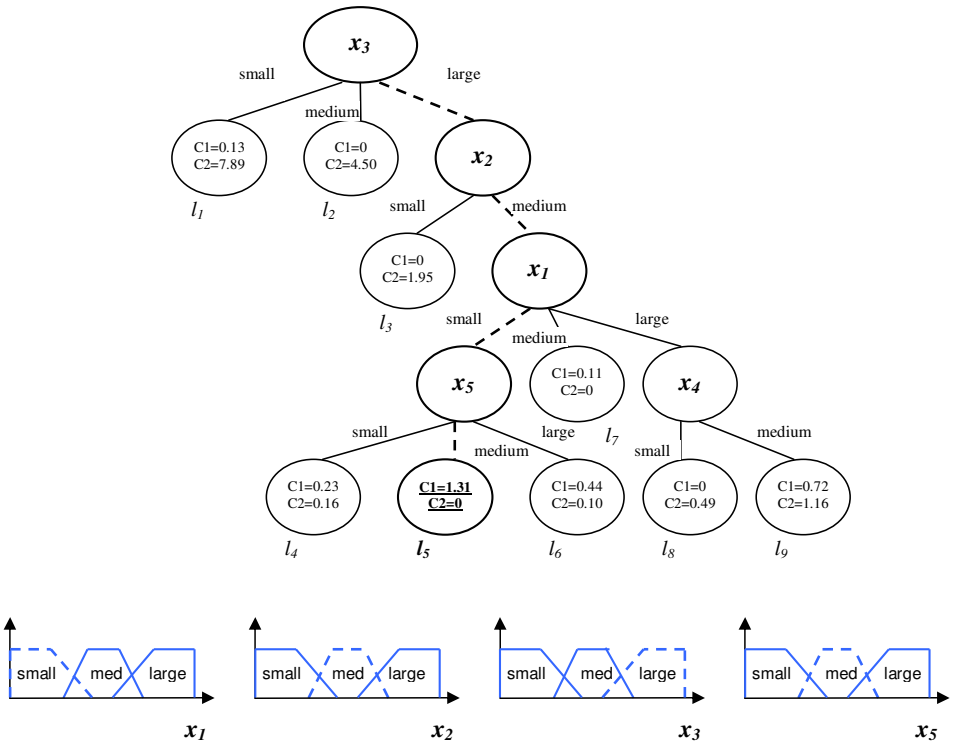


Figure B.5: Example of a fuzzy decision tree and a fuzzy partitioning

cardinality). However, most research has concentrated on *a priori* partitioning techniques [137].

An example for a fuzzy decision tree is given in Figure B.5. As can be seen in this figure each internal node is associated with a decision function (represented by a fuzzy membership function) to indicate which nodes to visit next. Each terminal node represents the output of a given input that leads to this node. In classification problems each terminal node contains the conditional probabilities $P(c_1|r_i), \ldots, P(c_C|r_i)$ of the predicted classes.

As a result of the increasing complexity and dimensionality of classification problems, it becomes necessary to deal with structural issues of the identification of classifier systems. Important aspects are the selection of the relevant features and the determination effective initial partition of the input domain [138]. Moreover, when the classifier is identified as part of an expert system, the linguistic interpretability is also an important aspect which must be taken into account.

B.4 Population Based Optimization

Evolutionary Algorithm

Evolutionary Algorithm (EA) [139, 140, 141, 142] is a widely used population based iterative optimization technique that mimics the process of natural selection. EA works with a *population* of *individuals*, where every individual within the population represents a particular solution. Every individual has a *chromosome* that encodes the decision variables of the represented solution. Because a chromosome can contain a mixture of variable formats (numbers, symbols, and other structural parameters), EA can simultaneously optimize diverse types of variables. Every individual has a *fitness value* that expresses how good the solution is at solving the problem. Better solutions are assigned higher values of fitness than worse performing solutions. The key of EA is that the fitness also determines how successful the individual will be at propagating its genes (its code) to subsequent generations.

The population is evolved over generations to produce better solutions to the problem. The evolution is performed using a set of stochastic operators which manipulate the genetic code used to represent the potential solutions. Evolutionary algorithm includes operators that select individuals for reproduction, produce new individuals based on those selected, and determine the composition of the population at the subsequent generation. Algorithm B.4.1 outlines a typical EA. The individuals are randomly initialized, then evolved from generation to generation by repeated applications of evaluation, selection, mutation, recombination and replacement.

Algorithm B.4.1. *A typical evolutionary algorithm*

```
procedure EA; {
   Initialize population;
   Evaluate all individuals;
   while (not terminate) do {
      Select individuals;
      Create offsprings from selected individuals
       using Recombination and Mutation;
      Evaluate offspring;
      Replace some old individuals by offsprings;
   }
}
```

In the selection step, the algorithm selects the parents of the subsequent generation. The population is subjected to "environmental pressure". It means that the higher fitness the individual has, the higher probability it is selected. The most important selection methods are Tournament Selection, Fitness Ranking Selection and Fitness Proportional Selection. After the selection of the parents, the new individuals of the subsequent generation (also called offsprings) are created by recombination and mutation.

- The recombination (also called crossover) operator exchanges information between two selected individuals to create one or two new offsprings.
- The mutation operator makes small, random changes to the chromosome of the selected individual.

The final step in the evolutionary procedure is the replacement, when new individuals are inserted into the population, and old individuals are deleted. Once the new generation has been constructed, the whole procedure is repeated until termination criterions satisfy.

Evolutionary Algorithm searches directly and represents potentially much greater efficiency than a totally random or enumerative search [143]. The main benefit of EA is that it can be applied to a wide range of problems without significant modification. However, it should be noted that EA has several implementations: Evolutionary Programming (EP), Evolutionary Strategy (ES), Genetic Algorithm (GA) and Genetic Programming (GP). The main difference among them is that GA uses bit-string representation, ES uses real-valued representation, and GP uses symbolic representation. In the case of common process engineering optimization problems, the decision variables are usually real-valued, hence ES will be presented in the following subsection.

Evolutionary Strategy

Evolution Strategy was developed by Rechenberg [143], with selection, mutation, and a population of size one. Schwefel introduced recombination and populations with more than one individual, and provided a nice comparison of ES with more traditional optimization techniques [144]. The main elements of the ES algorithm are the followings:

Evolutionary Strategy typically searches in continuous space, i.e. it uses real-valued representation. Search points in ES are n-dimensional vectors $\mathbf{x} \in R^n$ of object variables. To allow for a better adaptation to the objective functions's topology, the object variables are accompanied by a set of so-called strategy parameters. The function of strategy variables is to control the mutation operator separately for each individual. So an ES-individual $\mathbf{a}_j = (\mathbf{x}_j, \sigma_j)^T$ consists of two components, the object variables $\mathbf{x}_j = [x_{j,1}, \ldots, x_{j,n}]^T$ and the strategy variables $\sigma_j = [\sigma_{j,1}, \ldots, \sigma_{j,n}]^T$.

The mutation operator adds $z_{j,i}$ normal distributed random numbers to the objective variables:

$$x_{j,i} = x_{j,i} + z_{j,i}, \tag{B.48}$$

where $z_{j,i} = N(0, \sigma_{j,i})$ is a normal distributed random number with $\sigma_{j,i}$ standard deviation. Thus the σ_j strategy variables control the step size of standard deviations in the mutation for j-th individual. Before the object variables are changed by mutation operator, σ_j standard deviations are mutated using a multiplicative normally distributed process, that is

$$\sigma_{j,i}^{(t)} = \sigma_{j,i}^{(t-1)} \exp(\tau' N(0,1) + \tau N_i(0,1)), \tag{B.49}$$

with $\exp(\tau' N(0,1))$ as a global factor which allows an overall change of the mutability and $exp(\tau N_i(0,1))$ allowing for individual changes of the mean step sizes $\sigma_{j,i}$. The τ' and τ parameters can be interpreted in the sense of global learning rates. Schwefel suggests to set them as [145]:

$$\tau' = \frac{1}{\sqrt{2n}},\ \tau = \frac{1}{\sqrt{2\sqrt{n}}}. \tag{B.50}$$

Recombination in ES can be either sexual, where only two parents are involved in the creation of an offspring, or global, where up to the whole population contributes to a new offspring. Traditional recombination operators are discrete recombination, intermediate recombination, and geometric recombination, all existing in a sexual and global form. When F and M denote two randomly selected individuals from the μ parent population, the following operators can be defined:

$$x_i' = \begin{cases} x_{F,i} & \text{no recombination} \\ x_{F,i} \text{ or } x_{M,i} & \text{discrete} \\ (x_{F,i} + x_{M,i})/2 & \text{intermediate} \\ \sum_{k=1}^{\mu} x_{K,i}/\mu & \text{global avarage} \end{cases} \tag{B.51}$$

$$\sigma_i' = \begin{cases} \sigma_{F,i} & \text{no recombination} \\ \sigma_{F,i} \text{ or } x_{M,i} & \text{discrete} \\ (\sigma_{F,i} + \sigma_{M,i})/2 & \text{intermediate} \\ \sqrt{(\sigma_{F,i}\sigma_{M,i})} & \text{geometric} \\ \sum_{k=1}^{\mu} \sigma_{K,i}/\mu & \text{global avarage} \end{cases}. \tag{B.52}$$

At first, a certain number of individuals are selected from the current generation to be parent. The number of parents is usually denoted as μ. Then the algorithm composes a certain number of parent-pairs from the set of parents. The number of parent-pairs is usually denoted as λ. The parent pairs are selected uniformly randomly. After that, the algorithm generates λ number of offsprings using recombination and then mutation. Finally, the algorithm arranges the subsequent generation. Evolutionary Strategy has two commonly-used replacement strategies: the $(\mu + \lambda)$ and the (μ, λ) strategy. The $(\mu + \lambda)$ strategy inserts the parents to the subsequent generation (it is elitist), while the (μ, λ) strategy does not conserve the parents. So when the $(\mu + \lambda)$ strategy is used, the size of population is $\mu + \lambda$, while when the (μ, λ) strategy is used, the size of population is λ.

Genetic Programming

Genetic Programming is a symbolic optimization technique, developed by Koza [146]. It is an evolutionary computation technique (like e.g. Genetic Algorithm, Evolutionary Strategy) based on the so-called "tree representation". This representation is extremely flexible, because trees can represent computer programs, mathematical equations or complete models of process systems. This scheme has been already used for circuit design in electronics, algorithm development for quantum computers, and it is suitable for generating model structures: e.g.

identification of kinetic orders [147], steady-state models [148], and differential equations [149]. It should be noted that there are several variants of Genetic Programming, e.g. Gene Expression Programming [150], the [151] provides a good general review of algorithm of GP.

In contrast to common optimization methods, in which potential solutions are represented as continuous numbers (usually a vector of real numbers), the symbolic optimization algorithms represent the potential solutions by structured ordering of several symbols. One of the most popular method for representing structures is the binary tree.; e.g. see Figure B.6.

A population member in GP is a hierarchically structured tree consisting of functions and terminals. These functions and terminals are selected from a set of functions (operators) and a set of terminals. For example, the set of operators F can contain the basic arithmetic operations: $F = \{+, -, *, /\}$; however, it may also include other mathematical functions, Boolean operators, conditional operators or Automatically Defined Functions (ADFs). ADFs [152] are sub-trees which are used as functions in the main tree, and they are varied in the same manner as the main trees. It is especially worth using of ADF if the problem is regularity-rich, because GP with ADF may solve these problems in a hierarchical way (e.g. chip design). In this work, only arithmetic operations and mathematical functions were used. The set of terminals T contains the arguments for the functions. For example, $T = \{x, y\}$ with x and y being two independent variables.

In general, GP creates nonlinear models in addition to linear-in-parameters models. In order to avoid nonlinear-in-parameters models, the parameters must be removed from the set of terminals, i.e. T contains only variables: $T = \{x_1(k), \cdots, x_m(k)\}$, where $x_i(k)$ denotes the i-th regressor variable. Hence, a population member represents only the F_i nonlinear functions (**??**). The parameters are assigned to the model after 'extraction' of the F_i function terms from the tree, and they are determined using the LS method (**??**).

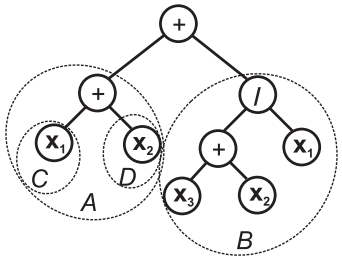


Figure B.6: Decomposition of a tree to function terms
$x_1 + x_2 + (x_3 + x_2)/x_1$

A very simple method was applied for the decomposition of the tree into function terms. The subtrees, which represent the F_i function terms, were determined by decomposing the tree starting from the root and reaching to non-linear nodes (nodes are not '+' or '-').

For example, let us see Figure B.6. The root node is a '+' operator, so it is possible to decompose the tree to two subtrees: 'A' and 'B' subtrees. The root node of 'A' subtree is again a linear operator, so we decompose it to 'C' and 'D' subtrees. The root node of 'B' is a nonlinear node ('/') so we do not decompose it. The root nodes of 'C' and 'D' are nonlinear too, so finally we get three subtrees: 'B', 'C' and 'D'. After that we can assign the parameters to the function terms represented by these subtrees, so the resulted linear-in-parameters model is: $y = p_0 + p_1(x_3 + x_2)/x_1 + p_2x_1 + p_3x_3$. Certainly, one may use other decomposition methods (which may lead different results, e.g. in this case $y = p_0 + p_1x_3/x_1 + p_2x_2/x_1 + p_3x_1 + p_4x_3$).

Besides, GP can be used to identify polynomial models. To achieve this goal, one has to restrict the set of operators and introduce some syntactic rules. For example, if the set of operators is defined as $F = \{+, *\}$, and there is a syntactic rule that exchanges the internal nodes that are below a '$*$'-type internal node to '$*$'-type nodes, the algorithm will generate only polynomial models.

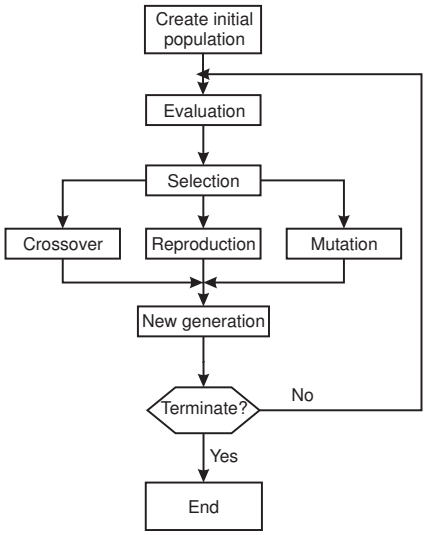


Figure B.7: Algorithm of GP

Genetic Programming is an Evolutionary Algorithm, see Section B.4. In every iteration, the algorithm evaluates the individuals (potential solutions), selects individuals for reproduction, generates new individuals by mutation, crossover and direct reproduction, and finally creates the new generation. Figure B.7 shows the scheme of the algorithm. The initial step is the creation of an *initial* population. This usually means generating individuals randomly to achieve high diversity. The first step is *fitness evaluation*, i.e. calculation of fitness values of individuals. Usually, the fitness value is calculated on the basis of a cost function. After that, in the *selection* step, the algorithm selects the parents of the next generation. In this chapter, the roulette-wheel selection strategy was used. In the roulette-wheel selection, every individual has a probability to be selected as parent, and this probability is proportional to fitness value:

$$p_i = \frac{f_i}{\sum f_i} \tag{B.53}$$

When an individual is selected for reproduction, three operators can be applied: *direct reproduction*, *mutation* and *crossover* (recombination). The probability of mutation is p_m, the probability of crossover is p_c, and the probability of direct reproduction is $1 - p_m - p_c$. The *direct reproduction* puts the selected individual into the new generation without any change. In *mutation* (see Figure B.8), a random change is performed on the selected tree structure. It means that the operator selects a node randomly and then replaces it with another element randomly. If an internal element (an operator) is changed to a leaf element (an argument) or vice-versa, the structure of tree changes, too. In *crossover*, two individuals are selected, and their tree structure are divided at a randomly selected crossover point, and the resulting sub-trees are exchanged to form two new individuals. There are two types of crossover, one-point and two-point crossover. In one-point crossover, the same crossover point selected for the two parent-trees, in two-point crossover, the two parent-trees are divided at different points (see Figure B.8).

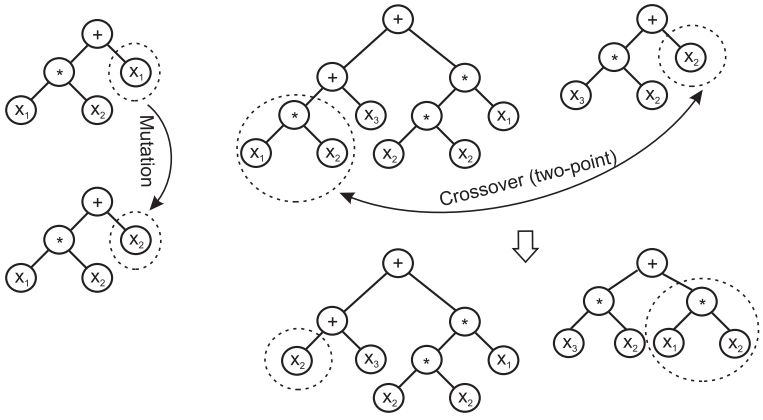


Figure B.8: Mutation and crossover operations

Before new individuals are inserted to the population, it is necessary to 'kill' the old individuals. In this chapter, the elitist *replacement* strategy was used in order to keep the best solutions. A 'generation gap' P_{gap} parameter determines how many individuals can survive. For example, $P_{gap} = 0.9$ means that 90% of population are 'killed' (replaced by new offsprings) and the best 10% survive.

The fitness function has two aspects. On the one hand, it reflects the goodness of a potential solution from the viewpoint of the cost function. On the other hand, it reflects a selection probability to determine which individuals form the next generation. Usually, the fitness function is based on the mean square error (MSE) between estimated and measured output values:

$$\chi^2 = \frac{1}{N} \sum_{k=1}^{N} \left(y(k) - \sum_{i=1}^{M} p_i F_i \left(\mathbf{x}(k) \right) \right)^2 , \tag{B.54}$$

where N is the number of data points used for the identification of the model. Instead of MSE, in symbolic optimization often the correlation coefficient between the desired and the estimated (model) output are used [153].

A good model is not only accurate but also simple, transparent, and interpretable. In addition, a complex, overparametrized model decreases the general estimation performance of the model. Hence there is a need for such a fitness function that ensures a tradeoff between the complexity and model accuracy. [148] suggests using a penalty term in the fitness function:

$$fit = \frac{r}{1 + \exp\left(a_1(L - a_2)\right)}, \tag{B.55}$$

where fit is the fitness value, r is the correlation coefficient between the desired and the estimated (model) output, L is the size of the tree (number of nodes), and a_1 and a_2 are parameters of the penalty function.

In practice, the measured data contains noise, so a model that gives a good prediction performance on the training data may be overparameterized and may contain unnecessary, complex terms. The penalty function (**??**) handles this difficulty because it decreases the fitness values of trees that have large number of terms. However, parameters of this penalty term are not easy to determine and the penalty function does not provide efficient solution for this difficulty. An efficient solution may be the elimination of complex and unnecessary terms from the model. For linear-in-parameters models, it can be done by the OLS algorithm. In the following section, this method will be presented.

B.5 Identification of Linear-in-Parameters Models

Linear-in-parameters Models

Data-driven identification of model structure cannot be separated from the identification of model parameters, because it is not possible to determine how good and accurate a given model structure is without the parameters. So to evaluate potential model structures, one has to identify the parameters for these models. Unfortunately, if a model is nonlinear, the identification of its parameters requires nonlinear optimization algorithm. In most cases data-driven model structure identification leads to very difficult and ill-conditioned nonlinear optimization problems with numerical difficulties and high sensitivity to noise. Hence, even if one finds a good model structure, this model structure may turn out to be useless due to the model parameter identification step. For example, it is possible that the modeler chooses a poor model structure instead of a good one because he or she is not able to identify the parameters of the 'better' model.

The very first step of model structure identification is the selection a model family that contains the set of candidate model structures. Consequently, it is worth to select such a model family which does not suffer from the above described difficulties. For this purpose this chapter proposes the application of linear-in-parameters models. Linear-in-parameters models are quite widespread in process engineering, e.g. let us consider the following well-known model classes:

- NAARX Nonlinear Additive AutoRegressive models with eXogenous inputs models are defined as [154]

$$\hat{y}(k) = \sum_{i=1}^{n_a} f_i(y(k-i)) + \sum_{j=1}^{n_b} g_j(u(k-j)) + e(k) \tag{B.56}$$

  where the functions f_i and g_i are scalar nonlinearities. As can be seen, this model does not permit 'cross terms' involving products of input and output values at different times.

- Volterra models are defined as multiple convolution sums

$$\begin{aligned} \hat{y}(k) = y_0 \quad &+ \quad \sum_{i=1}^{n_b} b_i u(k-i) \\ &+ \quad \sum_{i=1}^{n_b}\sum_{j=1}^{n_b} b_{ij} u(k-i)u(k-j) + \cdots + e(k)\,. \end{aligned} \tag{B.57}$$

- Polynomial ARMA models are superior to Volterra series models in the sense that the number of parameters needed to approximate a system is generally less than with polynomial models [155] because of the use of previous output values.

$$\hat{y}(k) = y_0 \quad + \quad \sum_{i=1}^{n_a} a_{1,i} y(k-i) + \sum_{i=1}^{n_b} b_{1,i} u(k-i)$$

$$+ \sum_{i=1}^{n_a} \sum_{j=1}^{i} a_{1,ij} y(k-i) y(k-j)$$
$$+ \sum_{i=1}^{n_b} \sum_{j=1}^{i} b_{2,ij} u(k-i) u(k-j) + \ldots + e(k)\,. \quad \text{(B.58)}$$

Generally, linear-in-parameters models are formulated as

$$\hat{y}(k) = \sum_{i=1}^{M} p_i F_i\left(\mathbf{x}(k)\right), \quad \text{(B.59)}$$

where $F_1, \ldots, F_M$ are nonlinear functions (they do not contain parameters), $p_1, \ldots, p_M$ are model parameters, $\hat{y}(k)$ is the model output at k-th time instant. $\mathbf{x}(k)$ is the regressor-vector at the k-th time instant and it consists of u input, y output and e error values:

$$\mathbf{x}(k) = (u(k-n_d-1), \cdots, u(k-n_d-n_u), y(k-n_d-1), \cdots, \\ y(k-n_d-n_y), e(k-n_d-1), \cdots, e(k-n_d-n_e)), \quad \text{(B.60)}$$

where n_d is the dead-time, n_u, n_y and n_e are the input-, output- and error-orders. Structure identification of linear-in-parameters models includes two types of problems:

- Identification of model order, namely finding appropriate M, n_d, n_u, n_y and n_e values (integer values).
- Identification of $F_1, \ldots, F_M$ nonlinear model equations (symbolic optimization).

This chapter will deal with these types of identification problems in the followings.

The great advantage of linear-in-parameters models is that the linear Least Squares (LS) method can be used for parameter identification, which is much less computationally demanding than nonlinear optimization algorithms. The LS method minimizes the square error between measured output and calculated output, i.e. minimizes the

$$\chi^2 = \sum_{k=1}^{N} \left(y(k) - \sum_{i=1}^{M} p_i F_i\left(\mathbf{x}(k)\right) \right)^2 \quad \text{(B.61)}$$

cost function, where N is the number of data-points and M is the number of regressors. The optimal $\mathbf{p} = [p_1, \ldots, p_M]$ parameter vector, where χ^2 is minimal, can be calculated by

$$\mathbf{p} = \left(\mathbf{F}^T \mathbf{F}\right)^{-1} \mathbf{F}^T \mathbf{y}, \quad \text{(B.62)}$$

where $\mathbf{y} = [y(1), \ldots, y(N)]$ is the measured output vector, and the $\mathbf{F}$ regression matrix is:

$$\mathbf{F} = \begin{pmatrix} F_1(\mathbf{x}(1)) & \ldots & F_M(\mathbf{x}(1)) \\ \vdots & \ddots & \vdots \\ F_1(\mathbf{x}(N)) & \ldots & F_M(\mathbf{x}(N)) \end{pmatrix}. \quad \text{(B.63)}$$

Orthogonal Least Squares Method for Linear-In-Parameter Models

The problem of model structure identification for linear-in-parameters models is to find the model order and the proper set of nonlinear F_i functions of (B.59). To attack this problem, two approaches can be distinguished:

- The first approach generates all of the possible model structures and then selects the best.
- The second approach transforms the problem into an optimization problem and solves it by an optimization algorithm.

The bottleneck of the first approach is that there are a vast number of possible structures; hence, it is impossible to evaluate all of them in practice. Even if the set of possible structures is restricted only to polynomial models

$$\hat{y}(k) = p_0 + \sum_{i_1=1}^{m} p_{i_1} x_{i_1}(k) + \sum_{i_1=1}^{m} \sum_{i_2=i_1}^{m} p_{i_1 i_2} x_{i_1}(k) x_{i_2}(k) + \cdots + \sum_{i_1=1}^{m} \cdots \sum_{i_d=i_{d-1}}^{m} p_{i_1 \cdots i_d} \prod_{j=1}^{m} x_{i_j}(k), \tag{B.64}$$

the number of possible terms could be very large. If the number of regressors is m and the maximum polynomial degree is d, the number of parameters (number of polynomial terms) is

$$n_p = \frac{(d+m)!}{d! \cdot m!}. \tag{B.65}$$

E.g. if $m = 8$ and $d = 4$ then $n_p = 495$. If the model does not only consist of polynomial terms, the number of possible terms multiplies. In the case of a reasonable number of regressors, the model terms can be sorted based on their *err*'s, and the best terms (which have the biggest values) can be selected to constitute the model.

The second approach transforms the structure selection problem into an optimization problem, in which the search space consists of the possible model structures. This method uses a search algorithm that looks for an optimal structure. The advantage of this method is that it does not need to evaluate every possible model structure. This chapter suggests the application of Genetic Programming to this task.

The Orthogonal Least Squares (OLS) method [156, 157] is an effective algorithm to determine which terms are significant in a linear-in-parameters model. The OLS method introduces *error reduction ratio*, which provides the decrease in the variance of output by a given term. The compact matrix form of linear-in-parameters models (B.59) is the following:

$$\mathbf{y} = \mathbf{F} \cdot \mathbf{p} + \mathbf{e}, \tag{B.66}$$

where $\mathbf{F}$ is the regression matrix (B.63), $\mathbf{p}$ is the parameter vector, $\mathbf{e}$ is the error vector. The OLS method transforms columns of $\mathbf{F}$ matrix into a set of orthogonal basis vectors in order to inspect the individual contributions of each term.

The OLS method assumes that the $\mathbf{F}$ regression matrix can be orthogonally decomposed as $\mathbf{F} = \mathbf{WA}$, where $\mathbf{A}$ is an $M \times M$ upper triangular matrix (it means $A_{i,j} = 0$ if $i > j$) and $\mathbf{W}$ is an $N \times M$ matrix with orthogonal columns in the sense that $\mathbf{W}^T\mathbf{W} = \mathbf{D}$ diagonal matrix. (N is the length of $\mathbf{y}$ vector, M is the number of regressors.) The orthogonal decomposition can be done by standard mathematical packages such as MATLAB. Then one can calculate the OLS auxiliary parameter vector as

$$\mathbf{g} = \mathbf{D}^{-1}\mathbf{W}^T\mathbf{y}, \tag{B.67}$$

The output variance $(\mathbf{y}^T\mathbf{y})/N$ can be explained as

$$\mathbf{y}^T\mathbf{y} = \sum_{i=1}^{M} g_i^2 w_i^T w_i + \mathbf{e}^T\mathbf{e}, \tag{B.68}$$

thus the *err* (error reduction ratio) of F_i can be expressed as

$$[err]^i = \frac{g_i^2 w_i^T w_i}{\mathbf{y}^T\mathbf{y}}. \tag{B.69}$$

This ratio offers a simple mean for ordering the terms, so it can be easily used to select the significant model terms.

Bibliography

[1] J.H. Krieger. Process simulation seen as pivotal in corporate information flow. *Chemical & Engineering News*, 1995.

[2] B. Bayer W. Marquardt, L. von Wedel. Perspectives on lifecycle modeling. *A.I.Ch.E Symposium Series*, 96(323):192–214, 2000.

[3] J. Bausa and G. Dünnebier. Life cycle modelling in the chemical industries: Is there any reuse of models in automation and control? In *16th European Symposium on Computer Aided Process Engineering*, 2006.

[4] J. Valdes and A. Barton. Virtual reality spaces: Visual data mining with a hybrid computational intelligence tool. Technical report, NRC/ERB-1137 (NRC 48501), 2006.

[5] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[6] J.B. Tenenbaum, V. Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[7] Y.Z. Lu. *Industrial Intelligent Control: Fundamentals and Application*. John Wiley and Sons, 1996.

[8] S-H. Huang, J.-X. Qian, and H-H. Shao. Human-machine cooperative control for ethylene production. *Artificial Intelligence in Engineering*, 9:203–209, 1995.

[9] S. Lane, E.B. Martin, R. Kooijmans, and A.J. Morris. Performance monitoring of a multi-product semi-batch process. *Journal of Process Control*, 11:1–11, 2001.

[10] C. Lindheim and K.M. Lien. Operator support systems for new kinds of process operation work. *Computers and Chemical Engineering,*, 21:113–118, 1997.

[11] G. Capocaccia. Intellution production is the heart of manufacturing ebusiness, ihistorian. *Distributed Control Systems 7th Meeting, Miskolc, Hungary*, 2001.

[12] U. Seidl. Simatic pcs 7: Efficient integration for tomorrow's dcs applications. *Distributed Control Systems 5th Meeting, Miskolc, Hungary*, 1999.

[13] S. Füle. Integration of distributed - and enterprise control systems. *Distributed Control Systems 5th Meeting, Miskolc, Hungary*, 1999.

[14] I. Ajtonyi and A. Ballagi. Integration of dcs in the complex producing system with wonderware factorysuite 2000 mmi software package. *Distributed Control Systems 7th Meeting, Miskolc, Hungary*, 2001.

[15] A. Mjaavatten and B.A. Foss. A modular system for estimation and diagnosis. *Computers and Chemical Engineering,*, 21:1203–1218, 1997.

[16] W.H. Inmon. *Building the Data Warehouse*. John Wiley and Sons Inc., 3 edition, 2002.

[17] C. Ballard, D. Herreman, D. Schau, R. Bell, E. Kim, and A. Valencic. Data modeling techniques for data warehousing. *International Technical Support Organization, IBM, http://www.redbooks.ibm.com*, page 25, 1998.

[18] M. Norgaard, N. Poulsen, and O. Ravn. New developments in state estimation for nonlinear systems. *Automatica*, 36:1627–1638, 2000.

[19] D.C. Psichogios and L.H. Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1498–1511, 1992.

[20] H.-J. Zander, R. Dittmeyer, and J. Wagenhuber. Dynamic modeling of chemical reaction systems with neural networks and hybrid models. *Chemical Engineering Technology*, 7:571–574, 1999.

[21] C.A.O. Nascimento, R. Giudici, and N. Scherbakoff. Modeling of industrial nylon-6,6 polymerization process in a twin-screw extruder reactor. ii. neural networks and hybrid models. *Journal of Applied Polymer Science*, 723:905–912, 1999.

[22] M. Dors I. Havlik A. Lubbert J. Schubert, R. Simutis. Bioprocess optimization and control: Application of hybrid modeling. *Journal of Biotechnology*, 35:51–68, 1994.

[23] S. Lakshminarayanan, H. Fujii, B. Grosman, E. Dassau, and D.R. Lewin. New product design via analysis of historical databases. *Computers and Chemical Engineering*, 24:671–676, 2000.

[24] J.F. MacGregor and T. Kourti. Statistical process control of multivariate processes. *Control Eng. Practice*, 3(3):403–414, 1995.

[25] X.Z. Wang. *Data Mining and Knowledge Discovery for Process Monitoring and Control*. Springer, 1999.

[26] Y. Moteki and Y. Arai. Operation planning and quality design of a polymer process,. *IFAC DYCORD*, pages 159–165, 1986.

[27] H. Hellendoorn and D. Driankov, editors. *Fuzzy Model Identification: Selected Approaches*. Springer, Berlin, Germany, 1997.

[28] T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1):116–132, 1985.

[29] M. Sugeno and G.T. Kang. Fuzzy modelling and control of multilayer incinerator. *Fuzzy Sets and Systems*, 18:329–346, 1986.

[30] J.-S.R. Jang, C.-T. Sun, and E. Mizutani. *Neuro–Fuzzy and Soft Computing; a Computational Approach to Learning and Machine Intelligence*. Prentice–Hall, Upper Sadle River, 1997.

[31] Y. Jin. Fuzzy modeling of high–dimensional systems. *IEEE Transactions on Fuzzy Systems*, 8:212–221, 2000.

[32] J.A. Roubos and M. Setnes. Compact fuzzy models through complexity reduction and evolutionary optimization. In *Proc. of IEEE international conference on fuzzy systems*, pages 762–767, San Antonio, USA, 2000.

[33] M. Sugeno and T. Yasukawa. A fuzzy–logic–based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems*, 1(1):7–31, 1993.

[34] R. Babuška and H.B. Verbruggen. Constructing fuzzy models by product space clustering. In H. Hellendoorn and D. Driankov, editors, *Fuzzy Model Identification: Selected Approaches*, pages 53–90. Springer, Berlin, Germany, 1997.

[35] R. Babuška. *Fuzzy Modeling for Control*. Kluwer Academic Publishers, Boston, 1998.

[36] T.A. Johansen and R. Babuska. On multi-objective identification of takagi-sugeno fuzzy model parameters. In *Preprints 15th IFAC Word Congress*, Barcelona, Spain, 2002.

[37] E. Kim, , S. Kim, and M. Park. A transformed input–domain approach to fuzzy modeling. *IEEE Transactions on Fuzzy Systems*, 6:596–604, 1998.

[38] I. Gath and A.B. Geva. Ieee transactions on pattern analysis and machine intelligence. *Knowledge-Based Systems*, 7:773–781, 1989.

[39] N. Kambhatala. *Local Models and Gaussian Mixture Models for Statistical Data Processing*. Ph.D. Thesis, Oregon Gradual Institute of Science and Technology, 1996.

[40] J. Abonyi and R. Babuška. Local and global identification and interpretation of parameters in Takagi–Sugeno fuzzy models. In *Proceedings IEEE International Conference on Fuzzy Systems*, San Antonio, USA, May 2000.

[41] T.A. Johansen, R. Shorten, and R. Murray-Smith. On the interpretation and identification of Takagi–Sugeno fuzzy models. *IEEE Transactions on Fuzzy Systems*, 8:297–313, 2000.

[42] N.R. Draper and H. Smith. *Applied Regression Analysis, 3rd Edition*. John Wiley and Sons, Chichester, 1994.

[43] N. Gershenfeld, B. Schoner, and E. Metois. Cluster-weighted modelling for time-series analysis. *Nature*, 397:329–332, 1999.

[44] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis – Methods for Classification, Data Analysis and Image Recognition*. John Wiley and Sons, 1999.

[45] J.-S.R. Jang. Input selection for ANFIS learning. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, volume 2, pages 1493–1499, New York, USA, 1996.

[46] J.S.R. Jang and C.T. Sun. Neuro–fuzzy modelling and control. *Proceedings of the IEEE*, 83:378–406, 1995.

[47] J.A. Roubos, M. Setnes, and J. Abonyi. Learning fuzzy classification rules from data. In R. John and R. Birkenhead, editors, *Developments in Soft Computing*. Springer, Physica Verlag, 2001.

[48] J. Abonyi, B. Feil, S. Nemeth, and P. Arva. Modified Gath–Geva clustering for fuzzy segmentation of multivariate time-series. *Fuzzy Sets and Systems – Fuzzy Sets in Knowledge Discovery*, 149(1):39–56, 2005.

[49] K. Vasko and H.T.T. Toivonen. Estimating the number of segments in time series data using permutation tests. *IEEE International Conference on Data Mining*, pages 466–473, 2002.

[50] M. Last, Y. Klein, and A. Kandel. Knowledge discovery in time series databases. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(1):160–169, 2000.

[51] S. Kivikunnas. Overview of process trend analysis methods and applications. *ERUDIT Workshop on Applications in Pulp and Paper Industry*, page CD ROM, 1998.

[52] Y. Yamashita. Supervised learning for the analysis of the process operational data. *Computers and Chemical Engineering*, 24:471–474, 2000.

[53] J.Zhang, E.B. Martin, and A.J. Morris. Process monitoring using nonlinear statistical techniques. *Chemical Engineering Journal*, 67:181–189, 1997.

[54] K. Chakrabarti and S. Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. *Proceedings of the 26th VLDB Conference Cairo Egypt*, page P089, 2000.

[55] M.E. Tipping and C.M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[56] Cs. Vincze P. Arva J. Abonyi, S. Nemeth. Process analysis and product quality estimation by self-organizing maps with an application to polyethylene production. *Computers in Industry*, 52(3):221–234, 2003.

[57] J.F. Baldwin, T.P. Martin, and J.M. Rossiter. Time series modelling and prediction using fuzzy trend information. *Proceedings of 5^{th} International Conference on Soft Computing and Information Intelligent Systems*, pages 499–502, 1998.

[58] A.B. Geva. Hierarchical-fuzzy clustering of temporal-patterns and its application for time-series prediction. *Pattern Recognition Letters*, 20:1519–1532, 1999.

[59] J.C. Wong, K. McDonald, and A. Palazoglu. Classification of process trends based on fuzzified symbolic representation and hidden markov models. *Journal of Process Control*, 8:395–408, 1998.

[60] U. Kaymak and R. Babuska. Compatible cluster merging for fuzzy modeling. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 897–904. Yokohama, Japan, 1995.

[61] W.J. Krzanowsky. Between group comparison of principal components. *J. Amer. Stat. Assoc.*, pages 703–707, 1979.

[62] T.W. Liao. Clustering of time series data - a survey. *Pattern Recognition*, 2005. In Press.

[63] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H. T. Toivonen. Time-series segmentation for context recognition in mobile devices. *IEEE International Conference on Data Mining (ICDM'01), San Jose, California*, pages 203–210, 2001.

[64] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. *IEEE International Conference on Data Mining*, page http://citeseer.nj.nec.com/keogh01online.html, 2001.

[65] G. Stephanopoulos and C. Han. Intelligent systems in process engineering: A review. *Comput. Chem. Engng.*, 20:743–791, 1996.

[66] E. Keogh and M. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. *4th Int. Conf. on KDD.*, pages 239–243, 1998.

[67] J. Abonyi, F. Szeifert, and R. Babuska. Modified Gath-Geva fuzzy clustering for identification of Takagi-Sugeno fuzzy models. *IEEE Systems, Man and Cybernetics, Part B.*, pages 612–621, 2002.

[68] R. Babuška, P. J. van der Veen, and U. Kaymak. Improved covariance estimation for Gustafson-Kessel clustering. *IEEE International Conference on Fuzzy Systems*, pages 1081–1085, 2002.

[69] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[70] J.C. Bezdek and J.C. Dunn. Optimal fuzzy partitions: A heutristic for estimating the parameters in a mixture of normal distributions. *IEEE Transactions on Computers*, pages 835–838, 1975.

[71] A. Singhal and D.E. Seborg. Matching patterns from historical data using PCA and distance similarity factors. *Proceedings of the American Control Conference*, pages 1759–1764, 2001.

[72] P. Marcelino, P. Nunes, P. Lima, and M. I. Ribeiro. Improving object localization through sensor fusion applied to soccer robots. *Actas do Encontro Cientifico do Robotica*, 2003.

[73] P.M. Kelly. An algorithm for merging hyperellipsoidal clusters. *Technical Report LA-UR-94-3306, Los Alamos National Laboratory, Los Alamos, NM*, 1994.

[74] H. Ishibuchi, T. Nakashima, and T. Murata. Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. *IEEE Transaction on Systems, Man, and Cybernetics: Part B*, 29:601–618, 1999.

[75] M. Setnes and R. Babuška. Fuzzy relational classifier trained by fuzzy clustering. *IEEE Trans. SMC–B*, 29:619–625, 1999.

[76] J. Abonyi and F. Szeifert. Supervised fuzzy clustering for the identification of fuzzy classifiers. *Pattern Recognition Letters*, 24(14):2195–2207, 2003.

[77] A. Biem, Katagiri S., McDermott E., and Juang BH. An application of discriminative feature extraction to filter-bank-based speech recognition. *IEEE Transactions On Speech And Audio Processing*, 9(2):96–110, 2001.

[78] A.F.R. Rahman and M.C. Fairhurst. Multi-prototype classification: improved modelling of the variability of handwritten data using statistical clustering algorithms. *Electron. Lett.*, 33(14):1208—1209, 1997.

[79] A.L. Corcoran and S. Sen. Using real-valued genetic algorithms to evolve rule sets for classification. In *IEEE-CEC*, pages 120–124, Orlando, USA, 1994.

[80] H. Akaike. A new look at the statistical model identification. *IEEE Trans. Autom. Control*, 19:716–723, 1974.

[81] G. Liang, D. Wilkes, and J. Cadzow. Arma model order estimation based on the eigenvalues of the covariance matrix. *IEEE Trans. Signal Process*, 41(10):3003–3009, 1993.

[82] Aguirre LA and Billings SA. Improved structure selection for nonlinear models based on term clustering. *International Journal of Control*, 62(3):569–587, September 1995.

[83] Aguirre L.A. and Mendes E.M.A.M. Global nonlinear polynomial models: Structure, term clusters and fixed points. *International Journal of Bifurcation and Chaos*, 6(2):279–294, Februar 1996.

[84] E. M. A. M. Mendes and S. A. Billings. An alternative solution to the model structure selection problem. *IEEE Trans. Syst. Man Cybernetics, Part A: Syst. Humans*, 31(6):597–608, 2001.

[85] Korenberg M., Billings S.A., Liu Y.P., and McIlroy P.J. Orthogonal parameter-estimation algorithm for nonlinear stochastic-systems. *International Journal of Control*, 48(1):193–210, 1988.

[86] R. K. Pearson. Selecting nonlinear model structures for computer control. *Journal of Process Control*, 13(1):1–26, 2003.

[87] J.D. Bomberger and D.E. Seborg. Determination of model order for NARX models directly from input–output data. *Journal of Process Control*, 8:459–468, Oct–Dec 1998.

[88] C. Rhodes and M. Morari. Determining the model order of nonlinear input/output systems. *AIChE Journal*, 44:151–163, 1998.

[89] M.B. Kennen, R. Brown, and H.D.I. Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review*, A:3403–3411, 1992.

[90] B. Feil, J. Abonyi, and F. Szeifert. Model order selection of nonlinear input-output models – a clustering based approach. *Journal of Process Control*, 14(6):593–602, 2004.

[91] R. Babuška and H.B. Verbruggen. New approach to constructing fuzzy relational models from data. In *Proceedings Third European Congress on Intelligent Techniques and Soft Computing EUFIT'95*, pages 583–587, Aachen, Germany, August 1995.

[92] D.E. Gustafson and W.C. Kessel. Fuzzy clustering with fuzzy covariance matrix. In *Proceedings of the IEEE CDC*, pages 761–766, 1979.

[93] J.A. Cadzow and O.M. Solomon. Algebraic approach to system identification. *IEEE Trans. on Acoust. Speach, Signal Processing*, 34(3):492–496, 1988.

[94] F.J. Doyle, B.A. Ogunnaike, and R. K. Pearson. Nonlinear model-based control using second-order volterra models. *Automatica*, 31:697–714, 1995.

[95] Letellier C. and Aguirre L.A. Investigating nonlinear dynamics from time series: The influence of symmetries and the choice of observables. *Chaos*, 12(3):549–558, September 2002.

[96] R. Murray-Smith and T.A. Johansen. *Multiple Model Appraoches to Modelling and Control*. Taylor and Francis, London, 1997.

[97] T.A. Johansen. Identification of non–linear systems using empirical data and a priori knowledge – an optimisation approach. *Automatica*, 32:337–356, 1996.

[98] R. Murray-Smith and T.A. Johansen, editors. *Multiple Model Approaches to Nonlinear Modeling and Control*. Taylor & Francis, London, UK, 1997.

[99] L.E. Scales. *Introduction to Non–linear Optimization*. Computer Science Series, Macmillan, 1985.

[100] J. Abonyi. *Fuzzy Model Identification for Control*. Birkhauser Boston, 2003.

[101] M. Karny, A. Haluskova, and P. Nedoma. Recursive approximation by ARX model: A tool for grey–box modelling. *Int. J. Adaptive Control and Signal Processing*, 9:525–546, 1995.

[102] W.D. Timmons, H.J. Chizeck, and P.G. Katona. Parameter–constrained adaptive control. *Ind. Eng. Chem. Res.*, 36:4894–4905, 1997.

[103] H.J.A.F Tulleken. Gray–box modelling and identification using physical knowledge and Bayesian techniques. *Automatica*, 29:285–308, 1993.

[104] J.H. Ahlberg, N.E. Nilson, and J.L. Walsh. *The Theory of Splines and Their Application*. Academic Press: New York, 1967.

[105] E.V. Shikin and E.V. Plis. *Handbook on Splines for the User*. CRC Press: Boca Raton, FL, 1995.

[106] K. Kollar-Hunek, M. Lang-Lazi, N. Herrmann, D. Miklos, and I. Kovács. Application of special numerical approximation in thermodynamics. *Hungarian Journal of Industrial Chemistry*, 26:269–274, 1998.

[107] J. Horiuchi, M. Kishimoto, M. Kamasawa, and H. Miyakawa. Data base simulation of batch and fed-batch cultures for amylase production using a culture data base and a statistical procedure. *Journal of Fermentation and Bioengineering*, 76(4):326–332, 1993.

[108] G. Maria and O. Muntean. Model reduction and kinetic parameters identification for the methanol conversion to olefins. *Chemical Engineering Science*, 42(6):1451–1460, 1987.

[109] D. Schaich, R. Becker, and R. King. Qualitative modeling for automatic identification of mathematical models of chemical reaction systems. *Control Engineering Practice*, 9:1373–1381, 2001.

[110] D. M. Himmelblau, C. R. Jones, and K. B. Sichoff. Determination of rate constants for kinetics models. *I&EC Fundamentals*, 6(4):539–543, 1967.

[111] *MATLAB Optimization Toolbox*. MathWorks Inc.: Natick, MA, 2002.

[112] Y.P. Tang. On the estimation of rate constants for complex kinetics models. *Ind. Eng. Chem. Funadam.*, 10(2):321–322, 1971.

[113] C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman and Hall, 1996.

[114] J. Abonyi, R. Babuska, M.A. Botto, F. Szeifert, and L. Nagy. Identification and control of nonlinear systems using fuzzy hammerstein models. *Industrial and Engineering Chemistry Research*, 39:4302–4314, 2000.

[115] H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *IEEE*, 89(9):1275–1296, 2001.

[116] S. Skogestad. Dynamics and control of distilation columns. *Chem. Eng. Res. Des. (Trans IChemE)*, 75:539–562, 1997.

[117] C.M. Jeackle and J.F. MacGregor. Product design through multivariate statistical analysis of process data. *AIChE, American Institute of Chemical Engineering Journal*, 44(5):1105–1118, 1998.

[118] N.P Cheremisinoff. *Encyclopedia of Engineering Materials Part A: Polymer Science and Technology*, volume 1. Marcel Dekker Inc., New York US, 1989.

[119] J. Abonyi, P. Arva, S. Nemeth, Cs. Vincze, B. Bodolai, Zs.Horvath, G. Nagy, and M. Nemeth. Operator support system for multi product processes - application to polyethylene production. In *European Symposium on Computer Aided Process Engineering*, pages 347–352. Lappeenranta, Finland, 2003.

[120] R. Srinivasan, C. Wang, W.K. Ho, and K.W. Lim. Dynamic principal component analysis based methodology for clustering process states in agile chemical plants. *Ind. Eng. Chem. Res.*, 43:2123–2139, 2004.

[121] W.J. Krzanowski. Between-groups comparison of principal components. *Journal of the American Statistical Society*, 74:703–707, 1979.

[122] M. Norgaard, N.K. Poulsen, and O. Ravn. New developments in state estimation for nonlinear systems. *Automatica*, 36(11):1627–1638, 2000.

[123] D.C. Psichogios and L.H. Ungar. A hybrid neural network – first principles approach to process modeling. *AIChE J.*, 38:1499–1511, 1992.

[124] H.J. Zander, R. Dittmeyer, and J. Wagenhuber. Dynamic modeling of chemical reaction systems with neural networks and hybrid models. *Chemical Engineering Technology*, 21(7):571–574, 1999.

[125] C.A.O. Nascimento, R. Giudici, and N. Scherbakoff. Modeling of industrial nylon-6,6 polymerization process in a twin-screw extruder reactor. *Journal of Applied Polymer Science*, 723:905–912, 1999.

[126] J. Schubert, R. Simutis, M. Dors, and I. Havlik ab A. Lubbert. Bioprocess optimization and control: Application of hybrid modelling. *Journal of Biotechnology*, 35:51–68, 1994.

[127] O. Nelles. *Nonlinear System Identification*. Springer, Berlin, Germany, 2001.

[128] J. Horiuchi, M. Kishimoto, M. Kamasawa, and H. Miyakawa. Data base simulation of batch and fed-batch cultures for α-amylase production using a culture data base and a statistical procedure. *Journal of Fermentation and Bioengineering*, 76(4):326–332, 1993.

[129] R.J. Schalkoff. *Artificial Neural Networks*. McGraw-Hill, New York, New York, 1997.

[130] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An Introduction to Fuzzy Control*. Springer-Verlag, Heidelberg, Germany, 1993.

[131] K.M. Passino and S. Yurkovic. *Fuzzy Control*. Addison-Wesley, New York, USA, 1998.

[132] L.X. Wang. *A course in Fuzzy Systems and Control*. Prentice Hall, New York, USA, 1997.

[133] R.R. Yager and D.P. Filev. *Essentials of Fuzzy Modeling and Control*. John Wiley, New York, 1994.

[134] C.Z. Janikow. Fuzzy decision trees: Issues and methods. *IEEE Trans. SMC-B*, 28:1–14, 1998.

[135] R. Gallion, D.C.St. Clair, C. Sabharwahl, and W.E. Bond. Dynamic id3: A symbolic learning algorithm for many-valued attribute domains. In *in Proc. 1993 Symp.Applied Computing.*, pages 14–20, New York, ACM Press, 1993.

[136] I. Konenko, I. Bratko, and E. Roskar. *Experiments in automatic learning of medical diagnostic rules*. Tech. Rep., J. Stefan Inst. Yugoslavia, 1994.

[137] M. Lebowitz. Categorizing numeric information for generalization. *Cognitive Science*, 9:285–308, 1985.

[138] K.J. Cios, W. Pedrycz, and R.W. Swiniarski. *Data Mining Methods for Knowledge Discovery*. Kluwer Academic Press, Boston, 1998.

[139] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[140] K.S. Tang, K.F. Man, S. Kwong, and Q. He. Genetic algorihms and their applications. *IEEE Signal Processing Magazine*, pages 22–37, november 1996.

[141] W. Banzhaf, P. Nordin, R.E. Keller, and F.D. Francone. *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Morgan-Kaufmann, 1997.

[142] D. Dasgupta and Z. Michalewicz. *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag, London, 1997.

[143] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Stuttgart: Frommann-Holzboog, 1973.

[144] H.P. Schwefel. Kybernetische evolution als strategie der experimentellen forschung in der stromungstechnik. 1965.

[145] H.P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, 1995.

[146] J.R. Koza. *Genetic Programming: On the programming of Computers by Means of Natural Evolution*. MIT Press, Cambridge, 1992.

[147] H. Cao, J. Yu., L. Kang, and Y. Chen. The kinetic evolutionary modeling of complex systems of chemical reactions. *Computers and Chem. Eng.*, 23:143–151, 1999.

[148] B. McKay, M. Willis, and G. Barton. Steady-state modelling of chemical process systems using genetic programming. *Computers and Chem. Eng.*, 21(9):981–996, 1997.

[149] E. Sakamoto and H. Iba. Inferring a system of differential equations for a gene regulatory network by using genetic programming. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 720–726, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 2001. IEEE Press.

[150] C. Ferreira. Gene expression programming: a new adaptive algorithm for solving problems. *Complex Syst*, 13(2):87–129, 2001.

[151] S. Sette and L. Boullart. Genetic programming: principles and application. *Engineering Application of Artificial Intelligence*, 14:727–2736, 2001.

[152] J.R. Koza. *Genetic programming II: automatic discovery of reusable programs*. MIT Press, 1994.

[153] M.C. South. *The application of genetic algorithms to rule finding in data analysis*. PhD. thesis, Dept. of Chemical and Process Eng., The Univeristy of Newcastle upon Tyne, UK, 1994.

[154] R.K. Pearson and B.A. Ogunnaike. Nonlinear process identification. In M.A. Henson and D.E. Seborg, editors, *Nonlinear Process Control*, pages 11–109. Prentice–Hall, Englewood Cliffs, NJ, 1997.

[155] E. Hernandez and Y. Arkun. Control of nonlinear systems using polynomial ARMA models. *AICHE Journal*, 39(3):446–460, 1993.

[156] S.A. Billings, M. Korenberg, and S. Chen. Identification of nonlinear output-affine systems using an orthogonal least-squares algorithm. *International Journal of Systems Science*, 19(8):1559–1568, 1988.

[157] 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.