A GENERALIZED RULE ANTECEDENT STRUCTURE FOR TSK TYPE OF DYNAMIC MODELS: STRUCTURE IDENTIFICATION AND PARAMETER ESTIMATION

By

MING SU

Bachelor of Science
East China University of Science and Technology
Shanghai, People's Republic of China
2000

Master of Science
East China University of Science and Technology
Shanghai, People's Republic of China
2003

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2009

A GENERALIZED RULE ANTECEDENT STRUCTURE FOR TSK TYPE OF DYNAMIC MODELS: STRUCTURE IDENTIFICATION AND PARAMETER ESTIMATION

Dissertation Approved:

Dr. R. Russell Rhinehart

Dissertation Adviser

Dr. Karen A. High

Dr. James R. Whiteley

Dr. Martin T. Hagan

Dr. Gary G. Yen

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my research advisor, Dr. R. Russell Rhinehart for his guidance, instruction, knowledge sharing and encouragement over years. He is ready all the time to discuss research problems and give advice. The conversations with Dr. Rhinehart have always been inspirational and joyful. His knowledge, analytical skills, creative thinking and sharp questions broaden my view on many problems. His advising is not limited to research problems. He taught me how to ask questions and sharpened my conversation and writing skills. He guided me how to approach a problem from scratch and organize time effectively to do research.

I would also like to thank my committee members, Dr. James R. Whiteley, Dr. Karen A. High, Dr. Gary G. Yen, and Dr. Martin T. Hagan for their insightful comments and suggestions as well as warming support and encouragement. I thank Dr. Yen for the time and effort that he spent on helping me preparing academic papers. I thank Dr. Hagan for his help and time to guide me studying Neural Networks and System Identification,

I want to thank Dr. A. H. Johannes from Chemical Engineering for his help over years to improve my presentation and teaching skills. I want to acknowledge Mr. Nittin Sharma, Mr. Pedro de Lima, Ms. Preetica Kumar, and Mr. Garov Aurora for their help and contribution to this project.

I wish to extend my thanks to the Edward E. and Helen Turner Bartlett Foundation for financial support.

Last, but not least I am thankful to my wife, Yumiao for her unconditional patience and support, and her absurd confidence in me. I am also thankful to my 2-year old daughter, Serena, who has always been a comfort to me when I feel frustrated. My special appreciation goes to my parents, Wenxin Su and Minghua Du for their understanding, caring and love.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

Efforts to describe chemical processes exist in various forms. Preferentially, based on idealized and simplified understanding of the underlying mechanism, first-principles models are developed. Many of these models have been standardized in commercial software such as ChemCAD for education or AspenPlus for prototyping process design. However, hardly can an idealized first-principles model find its application in practice; because, often, some artificial factors (like tray efficiency in a distillation column) have to be introduced to augment an ideal model to improve modeling accuracy via tuning against experiment data. Moreover, first-principles models are expensive to develop. It takes time for researchers to acquire sufficient knowledge for describing a new process mathematically and comprehensively. An ultimate goal of first-principles modeling is to understand the fundamental physics. However, in practice, partial or empirical understanding is often sufficient for certain practical applications. For instance, a modestly accurate input-output dynamic model makes controller design possible.

Contrasting to first-principles modeling, another effort is black-box modeling by system identification. Black-box modeling tends to overlook details in mechanism, but focuses on input-output behavior of a process. For instance, the input-output description via first-order-plus-time-delay models is often adequate for process control engineers to tune PID controllers. There are many choices for model structures including Finite Impulse Response, Autoregressive with exogenous inputs, Output Error, Autoregressive and Moving Average with exogenous inputs, and Box-Jenkins. For each structure, the simplest one is a linear model. Surprisingly, many chemical processes can be quite well described using linear models due to the fact that most chemical processes are operated around a steady state operating point. The linear model could be interpreted as a local

linearization of the truly nonlinear chemical process.

Despite the fact that linear models have been successfully used in many chemical processes, efforts have been devoted to describe nonlinear dynamical chemical processes in a more compact or unified approach. It is also expected that nonlinear modeling can provide more accurate description. If a nonlinear model is desired, users have options to represent a nonlinear function mapping. These options include but are not limited to polynomial models, piecewise models, basis function models, network models, and fuzzy models.

Interestingly, there is also experienced-based knowledge existing for chemical processes. These rules are familiar to us in various forms including process operating instructions and manuals, handbooks and rules of thumb. Some rules are derived from prior knowledge, which could be either understanding of fundamentals or experts' experience. For instance, our knowledge regarding distillation behavior might produce two following rules expressing steady state relations:

IF *Reflux (R) is Fast* ***THEN*** *Overhead Purity (x_d) is High*

IF *Reflux (R) is Slow* ***THEN*** *Overhead Purity (x_d) is Low*

where linguistic terms 'Fast' and 'Slow' are used to specify Reflux (R) while 'High' and 'Low' are used to specify x_d.

Knowledge expressed in logical rules is easy to understand but often difficult to use. Linguistic terms such as Fast, Slow, High, and Low are often not clearly defined. Moreover, human knowledge might be incomplete or outdated.

In this work, one focus is to describe the input-output behavior of a nonlinear dynamic process. We choose TSK (Takagi-Sugeno-Kang) (Sugeno & Kang, 1986; Takagi & Sugeno, 1985) fuzzy models to approximate nonlinearity. The choice is motivated to take advantage of simplicity, interpretability, modularity and flexibility in a fuzzy model.

The concept of a fuzzy set was introduced by Zadeh (Zadeh, 1965) to express degrees of membership of elements to sets, which could be viewed as a generalization of the classical

notion of set defined on a two-value (0 and 1 or Ture and False) membership value. Subsequently, fuzzy logic is invented to handle the reasoning based on fuzzy sets. There are many ways to define fuzzy logic. An interesting application of fuzzy logic in engineering fields (fuzzy logic in broad sense) is fuzzy modeling, which uses fuzzy models to represent a nonlinear function. A fundamental proof, which permits the belief in fuzzy modeling shows that a fuzzy model is a universal approximator (Kosko, 1994). It simply means that fuzzy models can theoretically approximate almost any nonlinear function. Although a fuzzy model is not the only universal approximator, it is preferable over other modeling approaches because of its simplicity, interpretability, modularity and flexibility.

One aspect of simplicity could be the modeling simplicity. One merit in fuzzy modeling is to allow users to translate their intuition and knowledge into a qualitative model description at first, by a fuzzy model, and leave quantitative description to a later tuning phase. For instance, an experienced operator can quickly provide a model with several rules to describe a distillation column as shown above, then, subsequently the break points defining linguistic categories can be fine tuned.

Because fuzzy models are strongly connected to human knowledge, they are often accredited interpretability. The use of linguistic terms seems to be an ‘obvious’ reason. For sure, the involvement of linguistic terms makes a fuzzy model appear friendly to users. More fundamentally, the interpretability is due to the fact that a fuzzy model is expressed in IF…THEN structure, which matches the reasoning procedure for humans and makes a fuzzy model appear ”intelligent”.

Another important aspect of interpretability is knowledge transparence, which is due to the modularity in a fuzzy model. Fuzzy models are made of rules. Regardless how ‘big’ a fuzzy model is, each rule in the fuzzy model is relatively simple. A fuzzy model as a whole with thousands of rules looks by no means interpretable no matter how many linguistic terms are used. However, the modularity in a fuzzy model allows users to look at a fuzzy model in a different way by shifting focus onto individual rules. In each rule, knowledge on local behavior of a nonlinear process becomes clear, and interpretability is possible.

Modularity is also aligned with the concept of divide-and-conquer in dealing with complex problems. In fuzzy model identification, modularity could be exploited to convert the identification of a fuzzy model to a number of smaller and simpler identification problems, each of which focuses on a rule. In applications, for instance, designing a fuzzy model based controller, modularity is used to translate the controller design for a fuzzy model into a number of smaller and simpler controller design problems.

Modularity also leads to flexibility in fuzzy models. A fuzzy model can be viewed as an interface rather than a model. It serves as a common gateway to connect different types of models and allow communication among them. As shown below is a possibility to let a fuzzy model to incorporate different types of models

IF *x is High* ***THEN*** *use a first-principles model*

IF *x is Low* ***THEN*** *use a Neural Network model*

IF *x is Medium* ***THEN*** *y is High*

The flexibility and modularity also simplifies the model management maintenance. In addition to adapting model parameters to compensate model-plant mismatch, fuzzy models also allow insertion and deleting operations on rules to incorporate newly discovered events and eliminate obsolete behavior.

Different from most black-box modeling approaches, in our view, fuzzy models explicitly separate nonlinear components in a model from its linear components. This work will exploit this feature to simplify the model structure.

However, the applications of fuzzy models are limited by their insufficiency to handle high-dimension problems due to a well known problem, the curse of dimensionality. With this restriction, fuzzy models can hardly have any significant practical impact. Even for a single-input-single-output (SISO) dynamic process, fuzzy models will be embarrassed if the SISO process has high dynamic orders. Many successful academic examples of using fuzzy

models are demonstrated on dynamic processes with low dynamic orders, often not exceeding four.

In this work, fuzzy models, particularly TSK type of fuzzy models are chosen to describe nonlinear dynamics due to the potential benefits mentioned above. The TSK model used in this work is featured with a generalized rule structure, which is proposed to overcome its insufficiency in dealing with high dimensional problem. The new structure has different dimensions in rule antecedent and consequent. Usually, in this work, the antecedent dimension is lower than consequent and contains only 'nonlinear variables', which tends to directly handle the curse of dimensionality by having fewer variables included in antecedents. Additionally, the new structure replaces the combinatorial antecedent structure by a more flexible one, where an extra degree of freedom is introduced to 'rotate' the coverage of a rule. The new addition reduces the number of rules needed in a TSK model by improving the covering efficiency of each rule. With the generalized rule antecedent structure, the TSK model in this work is referred to as GTSK (generalized TSK).

The structure of a GTSK model includes the overall model dimension and antecedent dimension. In this work, since the primary modeling target is nonlinear dynamic processes, the determination of the overall dimension of a GTSK model is translated to discover the dynamic orders from measured input-output data. The antecedent dimension of a GTSK model is determined by finding nonlinear components in a GTSK model.

Parameter estimation of the GTSK model is automated heuristically by recognizing rules from an iteratively partitioned space. Following the heuristic procedure is the fine tuning of the fuzzy model parameters by solving a nonlinear optimization problem with matrix inequality constraints.

This work tends to provide a unified and systematic procedure to obtain a GTSK model with new rule structure from input-output data for a nonlinear dynamic process. The procedure is demonstrated on several theoretical benchmark problems, which are drawn from published research works and are used primarily for illustrating ideas, comparing methods and verifying results. The procedure is also tested on a distillation column simulator, which

has been successfully used in past research work (Ou, 2001). Additionally, the procedure is tested on a pilot-scale chemical process, two-phase flow, which exhibit nonlinear dynamics, time delay, and measurement noise.

Innovations of this work are design of a new rule antecedent structure, which has a reduced antecedent dimension and a more flexible antecedent structure, design of a systematic approach to determine dynamic orders and detect nonlinear variables, and design of a heuristic procedure that iteratively partition an antecedent to generate regions, within which a linear relation is valid.

CHAPTER II

LITERATURE SURVEY

2.1 Literature Survey for Dynamic Order Determination

TSK type of fuzzy models is used in this work to describe a nonlinear dynamic process. Several potential benefits that users might expect from a fuzzy model have been listed in the Introduction. The modeling procedure proposed in this work is capable of dealing with multiple-input-multiple-output (MIMO) processes. However, the majority of technical elaboration will be based on single-input-single-output (SISO) models as described in Equation (2.1) for the simplicity of presentation. The extension to MIMO models will be addressed accordingly.

$$y(t) = f\left(y(t-1),\mathrm{L}\ , y(t-ny), u(t-d),\mathrm{L}\ , u(t-nu-d)\right) + e(t) \tag{2.1}$$

Equation (2.1) is a nonlinear autoregressive with exogenous input (NARX) model. The term NARX is chosen to be consistent with its linear counterpart, ARX models. The terminology is however not unique in the literature. In (Seborg & Henson, 1996), the structure in Equation (2.1) is named as a nonlinear autoregressive and moving average model (NARMA). In this work, ARX structure is chosen for its simplicity. More importantly, function arguments (lagged *y* and *u*) in Equation (2.1) include only input and output measurements. Some operations and treatment on raw data in this work are currently limited to model structures that have only measurable function arguments.

More complex structures could be used to describe nonlinear dynamics if necessary. A nonlinear NARMAX model is described in (Johansen & Foss, 1993). Its structure information is retrieved from its linear counterpart, ARMAX. As commented in (Nelles, 2001), more advanced structures are often not worth their additional complexities in describing nonlinear dynamics. On the other hand, NARX models as simpler models should often be tried first for any unknown structure nonlinear dynamic processes.The btained NARX models could be the basis for further structure variation or complication. In (Fischer, Nelles & Isermann, 1998), an NARX is first identified then converted to a nonlinear output error model (NOE) by some regressor replacements (for instance, $y(t\text{-}1)$ is replaced by its prediction $\hat{y}(t\text{-}1)$) followed by model parameter retuning.

Additionally, we assume, in this work, that *ny*, *nu* and *d* in Equation (2.1) are time invariants. The additional simplification may be against the nature of some realistic processes. For instance, a time-varying delay is often encountered in chemical processes, where a transportation delay strongly relates to a flow rate that is time varying in nature. On the other hand, a constant delay is often a good enough approximation in practice, especially in a relatively steady working condition.

The first step in system identification is to determine orders of the model. For the SISO model in Equation (2.1), the problem is then to discover *ny*, *nu* and *d*.

In terms of dynamic order determination, there are well-developed methods for linear systems. For dynamical linear systems, a preliminary analysis using autocorrelation and partial autocorrelation (Box, Jenkins & Reinsel, 1994) is able to identify dynamic orders. The result is often a set of candidate orders to be tried and validated further against data. Dynamic order determination can also be translated to problems regarding regressor analysis. Regressor analysis does not result in the dynamic orders and time delay directly. However, it would be a trivial practice to draw, *ny*, *nu* and *d* from the result of regressor analysis. One method is subset selection (Miller, 1990), which has different versions including forward selection, backward elimination, cycling replacement and exhaustive search methods. Among them, only exhaustive search, the most expensive one, is guaranteed to be able to find a global optimal solution, the best set

of regressors. Other methods are heuristically motivated aiming at a suboptimal solution with improvement in searching speed or efficiency.

Analysis of variance (ANOVA) as a tool to find the influential experimental factors can also be used to find influential regressors (Lind & Liung, 2008). ANOVA method suffers from the curse of dimensionality and the evaluation of interacting influence among factors requires a combinatorial amount of trials. In addition, a conventional ANOVA procedure takes finite levels of experimental factors rather than continuous ('infinite' levels) values. Extra computation is required to prepare the raw data for ANOVA analysis (for instance by clustering).

For nonlinear dynamical models, even for NARX models, there is not a general method such as the autocorrelation or partial autocorrelation method available for dynamic order determination. Rigorous analysis based on nonlinear correlation is possible if the nonlinear structure of f is known or presumed (Haber & Unbehauen, 1990). There are a variety of choices of predefined nonlinear structures such as bilinear, Wiener, Hammerstein models or their combinations. Another approach aims at a general target and does not depend on a predefined nonlinear structure. The geometric method (Molina, Sampson, Fitzgerald & Niranjan, 1996) is proposed to determine the embedded dimension in deterministic nonlinear autoregressive nonlinear systems. Following the same concept, its extension to dealing with deterministic ARX by including inputs is proposed in (Rhodes & Morari, 1995) based on False Nearest Neighbor. Both methods are more intuitively motivated rather than rigorously derived, and can be roughly argued based upon the first-order Taylor expansion. Another method also based on the first-order Taylor expansion argument is Lipschitz Quotient (He & Asada, 1993) aiming at deterministic NARX dynamic processes.

The difficulty in determining the order in Equation (2.1) is the unknown nonlinear function, *f*. Even if *f* is known to be nonlinear, the richness of nonlinearity would keep users from exhausting all possible nonlinear forms, making it difficult to find *ny*, *nu* and *d*. If the nonlinearity is known, it is possible to transform a nonlinear problem into a linear problem. If the nonlinearity is unknown, users could resort to any one of 'big'

models such as neural network or any other one being proved to be a universal approximator. These complex structures are able to capture almost any nonlinearity given enough flexibility. Without nonlinearity being a problem, users can then experiment and compare different sets of orders in these 'big' models. The drawback of using 'big' models is high computational burden. Additionally, as we will present later, experimentation of dynamic orders in 'big' models is not suitable for another objective in this work, nonlinear component detection. In our work, a unified approach is proposed for both dynamic order determination and nonlinear component detection.

2.2 Literature Survey for Fuzzy Model Structure

There are several different types of fuzzy models. One of them is the Mamdani fuzzy model (Mamdani, 1974). For the nonlinear dynamic process in Equation (2.1), Mamdani fuzzy models might be defined by rules as below

$$\begin{aligned} &\textbf{IF } \left(y(t-1) \text{ is } A_1^r \textbf{ AND } \mathrm{L} \textbf{ AND } u(t-nu-d) \text{ is } A_{ny+nu+1}^r \right) \\ &\textbf{THEN } \quad \mathrm{y}(\mathrm{t}) \textbf{ is } C^r \end{aligned} \tag{2.2}$$

where, the expression $y(t-1)$ is A_1^r **AND** L **AND** $u(t-nu-d)$ is $A_{ny+nu+1}^r$ is the antecedent of the rule. The expression $y(t)$ **is** C^r is the consequent of the rule. The variables $y(t\text{-}1)$, …, $y(t\text{-}ny)$, $u(t\text{-}d)$, …, $u(t\text{-}nu\text{-}d)$ are antecedent variables and A_1^r is the fuzzy subset for $y(t\text{-}1)$ in the rule. Notations of fuzzy subsets for other variables should be clear in context. A Mamdani fuzzy model has the perhaps the simplest consequent models.

An extension of Mamdani fuzzy models is Takagi-Sugeno-Kang (TSK) fuzzy model (Sugeno & Kang, 1986; Takagi & Sugeno, 1985). The generalization goes to rule consequent. For the nonlinear dynamic process in Equation (2.1), a rule in a TSK fuzzy model could be defined by

$$\begin{aligned} &\textbf{IF } \left(y(t-1) \text{is } A_1^r \textbf{ AND } \mathrm{L} \textbf{ AND } u(t-nu-d) \text{ is } A_{ny+nu+1}^r \right) \\ &\textbf{THEN } \quad \mathbf{A}^r\left(z^{-1}\right) y(t) = k^r + \mathbf{B}^r\left(z^{-1}\right) u(t-d) + e^r(t) \\ &\mathbf{A}^r\left(z^{-1}\right) = 1 + a_1^r z^{-1} + \mathrm{L} + a_{ny}^r z^{-ny} \\ &\mathbf{B}^r\left(z^{-1}\right) = b_0^r + b_1^r z^{-1} + \mathrm{L} + b_{nu}^r z^{-nu} \end{aligned} \tag{2.3}$$

where, consequent model is $\mathbf{A}^r\left(z^{-1}\right)y(t)=k^r+\mathbf{B}^r\left(z^{-1}\right)u(t-d)+e^r(t)$ with dynamic orders *ny* and *nu*, pure time delay *d* and a constant k^r. *z* is the backshift operator. The local model could be interpreted as a linearization of the nonlinear dynamic process in Equation (2.1). The linearization explains the inclusion of the constant term k^r. As mentioned in (Leith & Leithead, 1999; Shorten, Smith, Bjorgan & Gollee, 1999), the linearization could be interpreted as conducted around either a steady state or transitional working point. Including of the later is commented to be able to improve modeling performance for transient behavior (Smith & Johansen, 1997).

Mamdani and TSK represent two major types of fuzzy models and are different in consequents. In fact, a TSK fuzzy model could be further generalized by replacing its linear consequent models with other types of models. In (Mastorocostas & Theocharis, 2002), a new type of fuzzy model is proposed with neural network consequent models. Hierarchical fuzzy models (Lee, Chung & Yu, 2003; Liu & Li, 2005; Zeng & Keane, 2005) are often mentioned in the literature and could also be considered as a particular type of generalization by having fuzzy models as local models.

Interestingly, fuzzy models could also be compared with models originated from other disciplines. It is shown in (Andersen, Lotfi & Westphal, 1998; Roger & Sum, 1993) that a TSK fuzzy model with Gaussian membership functions and product operator for AND logic conjunction is functionally equivalent to a normalized radial basis network under certain restrictions. In (Smith & Johansen, 1997), a TSK fuzzy model is addressed in a broader perspective as a multi-model network.

The above mentioned fuzzy models represent one direction of generalization of fuzzy model structure by making consequent models more complex. Interestingly, not much effort is devoted to generalize the antecedent structure in a fuzzy model. The maneuverability in antecedents lies mainly in the choices of different types of membership functions including triangular, trapezoidal and Gaussian, etc., as well as different configurations for a particular type of membership functions.

Another degree of freedom in designing antecedents is via using different logic operators. For instance, the AND conjunction in the antecedent expression in Equation (2.2) or (2.3) could be quantitatively evaluated using either product or minimum operator. In addition to these two, there are in fact many other choices for the evaluation of AND conjunction, which is defined by a variety of T-norms as a result of research on symbolic fuzzy logic (Lee & Zhu, 1995).

2.3 Literature Survey for Fuzzy Model Identification

Identifying a fuzzy model generally involves two objectives, structure identification and parameter estimation. The structure identification selects variables for antecedent and consequent, determining number of fuzzy subsets for each variable, and estimating number of rules in a fuzzy model. Parameter estimation determines values of model parameters.

As shown in a TSK rule in Equation (2.3), model parameters include parameters defining all fuzzy subsets (membership functions) in the antecedent and those defining consequent models. There are many different approaches for fuzzy model identification. They vary for different types of fuzzy models to be identified or based on different assumptions. Very often in practice, the structure identification and parameter value estimation are coupled. For instance, the number of rules is related to the number of variables in the antecedent as well as the number of fuzzy subsets for each antecedent variable. Meanwhile, an addition or deletion of a fuzzy subset to a variable is expected to affect of the distribution of other fuzzy subsets, which in turn results in retuning of membership functions for optimal result. Inevitably, any variation in antecedent parameter values should be accompanied by corresponding change in consequent model coefficients.

2.3.1 Variable Selection

Variable selection determines the variables for rule antecedent and consequent. Very often, it is implicitly assumed for simplicity that all rules in a fuzzy model share the same set of antecedent and consequent variables. It is therefore equivalent in practice to

define the problem as antecedent and consequent variable selection for a fuzzy model. Variable selection is not conducted separately but often accompanied by parameter estimation/retuning. A common explicit procedure is to try different sets of selections with evaluation of their corresponding model accuracy and complexity, and find the best. In (Pomares, Rojas, González & Prieto, 2002), the variable selection is conducted iteratively in a constructive approach to build a fuzzy model. In each iteration, a fuzzy model is augmented by either changing the number of fuzzy subsets of already selected variables or adding a variable in antecedent. The better one is kept. Similar to the approach widely used in classification tree identification, the antecedent variable selection is implicitly conducted in (Nelles & Isermann, 1996). In each step, the augmentation of the existing fuzzy model is tried by adding a new rule for each candidate variable. The best rule is then kept. In the end, antecedent variable selection is automatically achieved by discarding variables from the antecedent, which are never selected. The variable selection becomes more complicated for a dynamic process as described in Equation (2.1) since each variable is associated with an unknown dynamic order. The variable selection problem should then be extended to determine the dynamic order for each variable. The extension could be simply achieved by including more lagged terms, which, however, largely increases the problem dimension and makes many methods designated for low dimension problems become difficult.

2.3.2 Fuzzy Model Identification

There are several different ways to categorize methods in fuzzy model identification. Some identification methods are based on heuristic criterion for linguistic interpretability and knowledge transparency. On the other hand, many other identification methods tend to find a more accurate fuzzy model by minimizing a quantitative performance index.

The approaches to extract fuzzy rules heuristically are mainly inspired by two procedures. The Pittsburgh approach focuses on rule set evolution while the Michigan approach evolves individual rules independently. Both Pittsburgh and Michigan approaches use genetic algorithms for optimization, which is consistent with the main

theme being heuristic. More importantly, it is due to the fact that heuristic criteria are unable to provide explicit searching directions expressed by gradients or Hessians. The research on this field focus primarily on inventing new heuristics by digging deep how human process linguistic information, or devise more efficient searching or combinatorial optimization techniques (Cordon, Herrera, Gomide, Hoffmann & Magdalena, 2001).

Different from those heuristically inspired approaches, a modeling error driven approach estimate parameter values of a fuzzy model by optimizing the performance index, for instance, sum of squared error. In this approach, one could take either a 'global' procedure to tune all parameters (antecedent, consequent parameters) simultaneously or a 'local' procedure starting from individual rules and combine them to be a fuzzy model. The 'global' procedure requires a good initial guess to avoid trivial solutions or poor local minimal. In (Dickerson & Kosko, 1996), an initial fuzzy model is generated by recognizing piece-wise patches along a SISO nonlinear function to be approximated. Then a steepest descent optimizer is followed. Heuristics based on clustering are also used to recognize the prototype rules (Dickerson & Kosko, 1996; Vernieuwe, Baets & Verhoest, 2006; Wang & Yang, 2009). In (Nelles, 2001) rules are progressively generated by conducting an equal division in a dimension in each step. It is also possible to over-parameterize a fuzzy model and let a simplification procedure (Yen & Wang, 1999) to merge redundant rules or eliminate invalid rules.

It is worthy pointing out that there is a procedure that tends to obtain a fuzzy model representation of a known nonlinear process by mathematical equivalence (Kawamoto, 1992). This approach has nothing to do with above mentioned fuzzy model identification from data. The main purpose of this procedure is to represent a nonlinear model by a fuzzy model and exploit the structure features in the fuzzy model to design controller, and investigate stability for the original nonlinear model.

Additionally, heuristic-based stochastic procedures exist to gain both model structures and parameter values simultaneously (Du & Zhang, 2008; Guenounou, Belmehdi & Dahhou, 2009; Lin, 2008; Lin & Xu, 2006), which however require even more computation resources.

CHAPTER III

A GENERALIZED RULE ANTECEDENT STRUCTURE

In this chapter, a generalized rule antecedent structure is proposed. The new rule antecedent uses only nonlinear variables. Additionally, one more degree of freedom is introduced to design antecedents to cover an antecedent space more efficiently. The following elaboration focuses on a single-input-single-output (SISO) model. The extension to multiple-input-multiple-output MIMO models is provided at the end.

3.1 Model Complexity

Equation (3.1) represents a SISO dynamic process with dynamic orders *ny*, *nu*, pure time delay *d*, and an additive noise *e* (*t*)

$$y(t) = f\left(y(t-1),\mathrm{L}\ , y(t-ny), u(t-d),\mathrm{L}\ , u(t-nu-d)\right) + e(t) \tag{3.1}$$

where *y* is the process response and *u* is the input. The nonlinear function, *f* could be approximated by a TSK model in Equation (2.3) and reproduced as below for simple reference

$$\begin{aligned}
&\textbf{IF}\ \left(y(t-1)\ \text{is}\ A_1^r\ \textbf{AND}\ \mathrm{L}\ \ \textbf{AND}\ u(t-nu-d)\ \text{is}\ A_{ny+nu+1}^r\ \right)\\
&\textbf{THEN}\ \ \ \mathbf{A}^r\left(z^{-1}\right) y(t) = k^r + \mathbf{B}^r\left(z^{-1}\right) u(t-d) + e^r(t)\\
&\mathbf{A}^r\left(z^{-1}\right) = 1 + a_1^r z^{-1} + \mathrm{L}\ + a_{ny}^r z^{-ny}\\
&\mathbf{B}^r\left(z^{-1}\right) = b_0^r + b_1^r z^{-1} + \mathrm{L}\ + b_{nu}^r z^{-nu}
\end{aligned} \tag{3.2}$$

Complexity of a TSK model could simply be regarded as the number of rules. For the TSK model in Equation (3.2), given that each variable has 5 fuzzy subsets (could be linguistically labeled as Low, Medium-Low, Medium, Medium-High, High), there would be $5^{ny+nu+1}$ possible rules to be considered. The problem dimension (ny+nu+1 in this case) is an obvious cause for the complexity. Moreover, the number of rules also depends on the number of fuzzy subsets for each variable. The illustrated number, 5 is quite conservative in practice. Simply put, the TSK model described in Equation (3.2) has difficulty to deal with high dimension problems or it is subject to "the curse of dimensionality".

In the following, a generalized rule antecedent structure is proposed to design an efficient GTSK model by using fewer rules. The new rule antecedent only uses nonlinear variables, which separates the antecedent dimension from the problem dimension. The complexity of a GTSK model is only related to the antecedent dimension. It is then possible to apply a GTSK model to a high dimension problem so long as its antecedent dimension is acceptable.

Additionally, the proposed rule antecedents are expressed as ellipsoids covering the underlying local regions and feature one more degree of freedom in design. The extra flexibility makes spatial coverage more efficient and simplifies a fuzzy model in terms of number of rules.

3.2 Antecedent Dimension

The direct approach to reduce the number of rules is to control the problem dimension, which is unfortunately determined by the nature of the problem but not by users. However, dimension reduction in the antecedent is still possible by excluding variables that appear linearly.

To illustrate dimension reduction, consider the following nonlinear dynamic model with three regressors, [$y(t\text{-}1)$ $y(t\text{-}2)$ $u(t\text{-}1)$]

$$y(t) = y(t-1)\left[y(t-2) + 2.5\right] + y^2(t-1)u(t-1) \tag{3.3}$$

Using the rule structure in Equation (3.2), the rule antecedent could then be expressed as ($y(t\text{-}1)$ is A_1 **AND** $y(t\text{-}2)$ is A_2 **AND** $u(t\text{-}1)$ is A_3)). The antecedent dimension is 3, which is same as the problem dimension. Assuming that each variable has 5 fuzzy sets, the combinatorial construction will then generate 5^3=125 possible rules.

The dynamic model in Equation (3.3) can be represented in a linear format using time-varying parameters.

$$y(t) = a_1(t)\,y(t-1) + a_2(t)\,y(t-2) + b_0(t)\,u(t-1) \tag{3.4}$$

with $a_1(t)$ = 2.5, $a_2(t) = y(t\text{-}1)$ and $b_0(t) = y(t\text{-}1)^2$ where, model parameters a_2 and b_0 are not only time-varying but functions of the regressor, $y(t\text{-}1)$. It indicates that the model can be expressed linearly in all variables except $y(t\text{-}1)$. The coefficient values in Equation (3.4) are independent of $y(t\text{-}2)$ and $u(t\text{-}1)$. Equivalently, the regressor, $y(t\text{-}1)$, is the only regressor that changes the otherwise linear model coefficient values. Therefore, $y(t\text{-}1)$ should be the only one included in the antecedent. The simplified rule is defined by

$$\begin{aligned} &\textbf{IF } y(t-1) \text{ is } A^r \textbf{ THEN} \\ &y(t) = k^r + a_1^r y(t-1) + a_2^r y(t-2) + b_1^r u(t-1) \end{aligned} \tag{3.5}$$

where the antecedent dimension is reduced to 1. The possible number of rules is reduced from 125 to 5. In Equation (3.5), y(t-1) is then an antecedent variable and collected in a vector $\mathbf{c}(t)$. Regressors in the consequent including $y(t\text{-}1)$, $y(t\text{-}2)$ and $u(t\text{-}1)$ are collected in vector $\mathbf{x}(t)$.

The concept to include only nonlinear variables in antecedents have been explicitly mentioned in (Shorten, Smith, Bjorgan & Gollee, 1999) or implicitly applied in (Nelles & Isermann, 1996; Tanaka & Wang, 2001), where fuzzy models are used to describe known nonlinear dynamic processes. However, the above mentioned dimension reduction can only be made practically applicable if it is able to find antecedent variables from data. The detection of antecedent variables will be addressed in Chapter 4.

3.3 Antecedent Structure

3.3.1 A Generalized Antecedent Structure

As mentioned above, the number of rules is related to the problem dimension by $5^{ny+nu+1}$. In Section 3.2, it is illustrated that it is possible to use a number for the exponent less than *ny*+*nu*+1. However, the exponential relation between the number of rules and the dimension (antecedent dimension) is still preserved. The underlying cause for the exponential connection is the combinatorial antecedent structure expressed in the TSK rule in Equation (3.2), using **AND** conjunction to connect antecedent variables. For example, given a two dimensional antecedent (c_1 is A_1 and c_2 is A_2), if Gaussian membership functions are assumed and the product operator is used for the **AND** conjunction, the antecedent is then evaluated by the truth of antecedent (*TA*)

$$TA = \exp\left(-\left(\frac{c_1 - o_1}{\sigma_1} \right)^2 - \left(\frac{c_2 - o_2}{\sigma_2} \right)^2 \right) \tag{3.6}$$

where *TA* is an ellipsoid centering at (o_1,o_2) with width by σ_1 and σ_2. A contour plot of *TA* is shown below

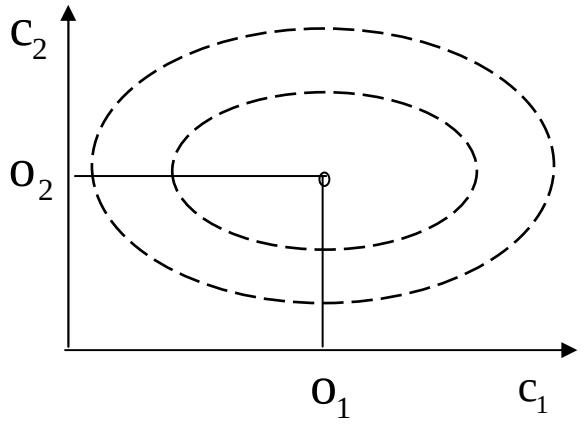


Figure 3.1. The ellipsoid contour of *TA*

In Figure 3.1, the highest value of *TA* =1 is reached at the centroid. The further out is the contour, the smaller the *TA*. The value of *TA* can be interpreted as the

belongingness of a data point to a local region.

A fuzzy model has several rules. Given a two-dimensional antecedent with equal number of fuzzy sets for each antecedent variable, a typical combinatorial antecedent space partition and representation by horizontal and vertical ellipsoids is shown in Figure 3.2(a)

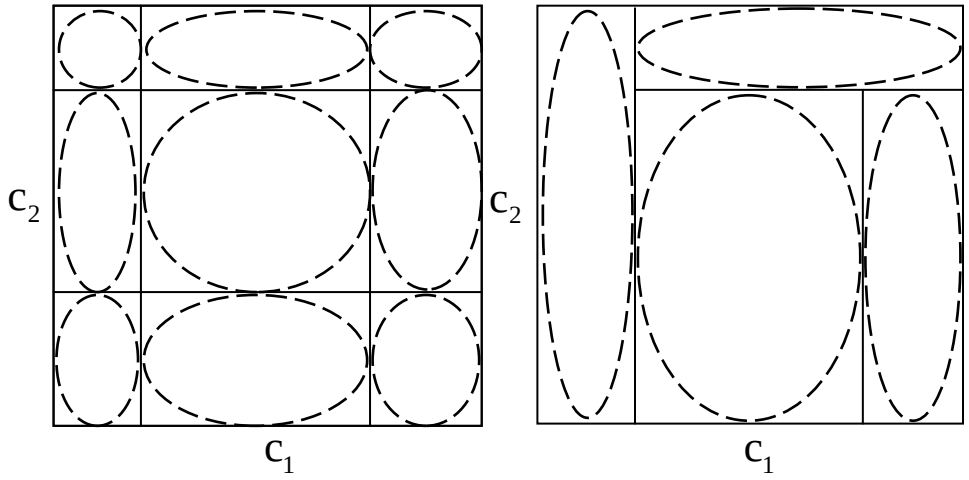


Figure 3.2. Antecedent space partition and representation

where, 9 rules result from the exhaustive combinations of 3 fuzzy sets for each antecedent variable. Users might resort to the techniques in (Yen & Wang, 1999) to reduce the redundancy in consequent models and have a more compact fuzzy model. The number of rules can be reduced by merging some regions that exhibit similar local behavior. Figure 3.2(b) shows a possible simplified partition after merging some regions. The partition in Figure 3.2(b) will also become inefficient as shown in Figure 3.3, where neither a horizontal nor a vertical ellipsoid provides an efficient representation of the underlying local region represented by either the rotated “space” of correlated variables or irregular polygons.

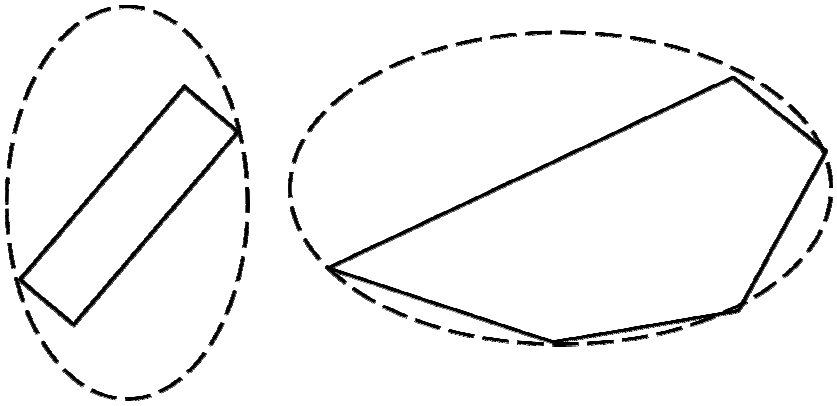


Figure 3.3. A rotated local region covered by a horizontal or vertical ellipsoid

One possible solution for covering the space is to use many smaller ellipsoids as shown in Figure 3.4, which however might result in a lot of rules.

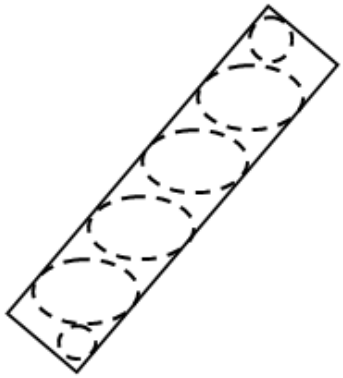


Figure 3.4. A rotated local region covered by many small ellipsoids

Another solution is to rotate the ellipsoid as shown in Figure 3.5.

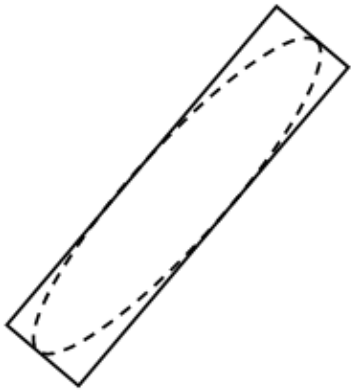


Figure 3.5. A rotated local region covered by a rotated ellipsoid

The rotated ellipsoid proposed here with the stretching and contraction is flexible enough to match many geometric shapes. In order to address the rotation mathematically, the parameters σ in Equation(3.6) are replaced by a symmetric positive definite matrix **P**, which is termed as the shape matrix in this work and redefines the truth of antecedent by

$$TA = \exp\left(-(\mathbf{c}-\mathbf{o})^T \mathbf{P}(\mathbf{c}-\mathbf{o})\right) \tag{3.7}$$

where **o** is a vector with dimension of *nc* and represents the centroid, and the dimension for the shape matrix **P** is *nc* by *nc*. The flexibility in representing antecedent subspaces is at cost of additional *nc*(*nc*-1)/2 new parameters in the shape matrix in Equation (3.7). This approach could be interpreted as a transition from a TSK fuzzy model with many simple rules to a GTSK fuzzy model with fewer complex rules. Clearly, the simplicity and complexity in this context refers to that in rule antecedents.

3.3.2 Interpretation of the Proposed Structure

Interested readers could follow the following method to convert the new antecedent structure in Equation (3.7) to a conventional antecedent in Equation (3.2) with new defined variables. Since the treatment in Section 3.3.2 is not essential, readers might also choose to skip it.

The conversion is aided via representing the shape matrix in Equation (3.7) by its spectral decomposition.

$$\mathbf{P} = \sum_{i=1}^{nc} \lambda_i \mathbf{z}_i \mathbf{z}_i^T \tag{3.8}$$

where λ_i is an eigenvalue and $\mathbf{z}_i$ is the corresponding eigenvector. Substituting **P** by its spectral decomposition, Equation (3.7) then becomes

$$
\begin{aligned}
TA &= \exp\left(-\left(\mathbf{c}-\mathbf{o}\right)^T \sum_{i=1}^{nc} \lambda_i \mathbf{z}_i \mathbf{z}_i^T \left(\mathbf{c}-\mathbf{o}\right) \right) \\
&= \prod_{i=1}^{nc} \exp\left(-\lambda_i \left(\mathbf{c}-\mathbf{o}\right)^T \mathbf{z}_i \mathbf{z}_i^T \left(\mathbf{c}-\mathbf{o}\right) \right) \\
&= \prod_{i=1}^{nc} \exp\left(-\lambda_i \left(\sum_{j=1}^{nc} \left(c_j - o_j\right) z_{i,j} \right)^2 \right)
\end{aligned}
\tag{3.9}
$$

Then *TA* could be converted to the conventional form

$$
TA = \prod_{i=1}^{nc} \exp\left(-\left(\frac{v_i - q_i}{\sigma_i} \right)^2 \right) \tag{3.10}
$$

with the new introduced antecedent variable v_{i}, centroid, q_{i} and σ_{i} are defined by

$$
\begin{aligned}
v_i &= \sum_{j=1}^{nc} c_j z_{i,j} \\
q_i &= \sum_{j=1}^{nc} o_j z_{i,j} \\
\sigma_i &= \lambda_i^{-1/2}
\end{aligned}
\tag{3.11}
$$

The rule antecedent could then be represented in the conventional format using **AND** conjunction as

$$
v_1 \text{ is } A_1\left(q_1, \sigma_1\right) \textbf{ANDL } \textbf{AND } v_{nc} \text{ is } A_{nc}\left(q_{nc}, \sigma_{nc}\right) \tag{3.12}
$$

where $A_1(q_1, \sigma_1)$ denotes a Gaussian membership function with the centroid, q_1 and the width specified by σ_1.

The above mentioned interpretation might be useful to convert an existing GTSK model with the generalized antecedent structure to a conventional TSK fuzzy model to regain the interpretability offered in antecedents using **AND** conjunction. It seems also that the antecedent structure generalization is to extend a conventional TSK fuzzy model architecturally by introducing an extra layer to linearly combined raw variables to form

antecedent variables.

However, the above interpretation might not be helpful in estimating model parameters in general. For instance, there are $nc(nc+1)/2$ variables required to specify the shape matrix. However, there are $nc(nc+1)$ parameters expressed in Equation (3.11); $z_{i,j}$ (i=1,…,*nc*; j=1,…,*nc*) and λ_i (i=1,…,*nc*). One might need to add additional constraints to eliminate the extra $nc(nc+1)/2$ degrees of freedom. For instance, eigenvectors are orthogonal to each other and eigenvectors have unit length.

3.4 SISO Models

In a GTSK model, model parameters include both antecedent and consequent parameters. Antecedent parameters specify active regions for each rule while consequent parameters describe local models. For simplicity of presentation, a vector $\mathbf{x}(t)$ is defined as below to collect the input arguments in Equation (3.1)

$$\mathbf{x}(t) = \begin{bmatrix} 1 & y(t-1) & \text{L} & y(t-ny) & u(t-d) & \text{L} & u(t-d-nu) \end{bmatrix}^T \tag{3.13}$$

where the dimension of $\mathbf{x}(t)$ is $nx+1$ with $nx = ny+nu+1$.

If a GTSK model is used to approximate the nonlinear function f in Equation (3.1), the fuzzy model is then defined as below using the proposed antecedent structure

$$\begin{aligned} &\textbf{IF } \left(\mathbf{c}(t) \text{ is in } R^1\left(\mathbf{o}^1, \mathbf{P}^1\right)\right) \textbf{THEN } \hat{y}^1(t) = \boldsymbol{\theta}^1 \mathbf{x}(t) \\ &\text{M} \\ &\textbf{IF } \left(\mathbf{c}(t) \text{ is in } R^M\left(\mathbf{o}^M, \mathbf{P}^M\right)\right) \textbf{THEN } \hat{y}^M(t) = \boldsymbol{\theta}^M \mathbf{x}(t) \end{aligned} \tag{3.14}$$

where, superscript 1 indicates the first rule in a GTSK model. The antecedent representation using **AND** conjunction in Equation (3.2) is replaced by the statement $\mathbf{c}(t)$ is in R^1 $(\mathbf{o}^1,\mathbf{P}^1)$. The expression of R^1 $(\mathbf{o}^1,\mathbf{P}^1)$ could be interpreted to represent an ellipsoidal active region for the first rule. The number of rules, *M*, is assumed known. $\mathbf{c}(t)$ containing *nc* antecedent variables is defined as below and obtained as nonlinear components in Chapter 4 for nonlinear dynamic processes.

$$\mathbf{c}(t) = \begin{bmatrix} c_1(t) & \mathrm{L} & c_{nc}(t) \end{bmatrix}^T \tag{3.15}$$

3.4.1 Model Parameters

Figure 3.6 illustrates the model parameters to be estimated for a GTSK model in a two-dimension antecedent structure.

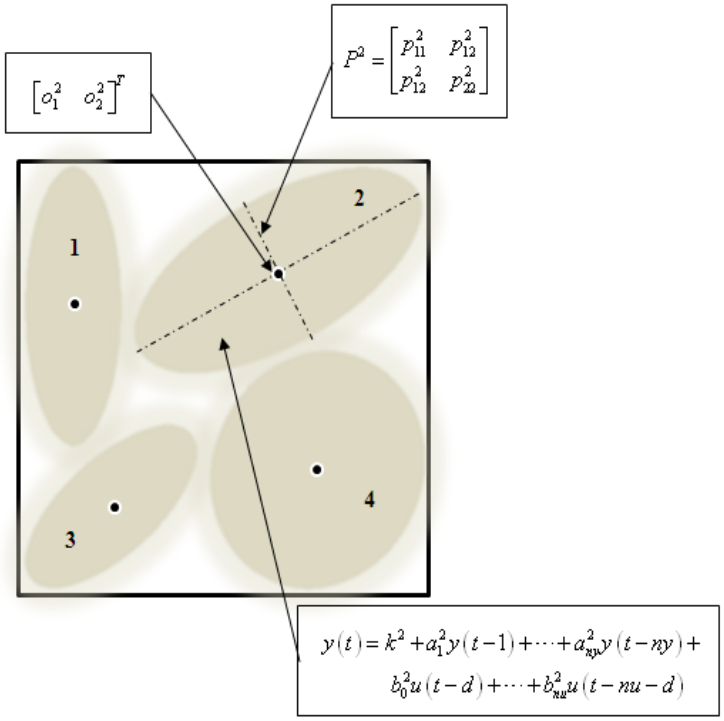


Figure 3.6. Model parameters for a 4-rule GTSK model

R^i represents the active region for the i^{th} rule. Its location and shape are specified by antecedent parameters; a centroid vector, $\mathbf{o}^i \in \mathrm{R}^{nc}$ and a positive definite shape matrix, $\mathbf{P}^i \in \mathrm{R}^{nc \times nc}$. They are respectively defined by

$$\mathbf{o}^i = \begin{bmatrix} o_1^i & \mathrm{L} & o_{nc}^i \end{bmatrix}^T \tag{3.16}$$

$$\mathbf{P}^i = \begin{bmatrix} p_1^i & p_2^i & p_4^i & \mathrm{L} \\ p_2^i & p_3^i & p_5^i & \mathrm{L} \\ p_4^i & p_5^i & p_6^i & \mathrm{L} \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} \end{bmatrix} \tag{3.17}$$

where the symmetric matrix $\mathbf{P}^i$ is specified by a vector $\mathbf{p}^i \in \mathrm{R}^{nc\times(nc+1)/2}$

$$\mathbf{p}^i = \begin{bmatrix} p_1^i & p_2^i & p_3^i & \mathrm{L} & p_{nc(nc+1)/2}^i \end{bmatrix} \tag{3.18}$$

The $\mathbf{P}^i$ matrix can be expressed as a weighted sum of symmetric basis matrices in order to facilitate the computation later on

$$\mathbf{P}^i = \sum_{j=1}^{nc(nc+1)/2} p_j^i \mathbf{B}_j \tag{3.19}$$

where symmetric basis matrices, $\mathbf{B}_j$, are defined in the following manner

$$\mathbf{B}_1 = \begin{bmatrix} 1 & 0 & \mathrm{L} & 0 \\ 0 & 0 & \mathrm{L} & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ 0 & 0 & \mathrm{L} & 0 \end{bmatrix}, \qquad \mathbf{B}_2 = \begin{bmatrix} 0 & 1 & \mathrm{L} & 0 \\ 1 & 0 & \mathrm{L} & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ 0 & 0 & \mathrm{L} & 0 \end{bmatrix}$$

$$\mathbf{B}_3 = \begin{bmatrix} 0 & 0 & \mathrm{L} & 0 \\ 0 & 1 & \mathrm{L} & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ 0 & 0 & \mathrm{L} & 0 \end{bmatrix}, \ldots, \mathbf{B}_{nc(nc+1)/2} = \begin{bmatrix} 0 & 0 & \mathrm{L} & 0 \\ 0 & 0 & \mathrm{L} & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ 0 & 0 & \mathrm{L} & 1 \end{bmatrix} \tag{3.20}$$

The local model parameters (consequent parameters) are included in vector $\boldsymbol{\theta}^i \in \mathrm{R}^{n\times+1}$ defined by

$$\boldsymbol{\theta}^i = \begin{bmatrix} \theta_0^i & \theta_1^i & \mathrm{L} & \theta_{nx}^i \end{bmatrix} \tag{3.21}$$

3.4.2 Model Computation

The computation of the model in Equation (3.14) is defined by

$$\hat{y}(t) = \sum_{i=1}^{M} w^i(t)\,\hat{y}^i(t) \tag{3.22}$$

where $\hat{y}^i(t)$ is output from the local model in Rule i and weighted by $w^i(t)$. Weights $w^i(t)$ are defined as the normalized truth of the antecedent (*TA*)

$$w^i(t) = \frac{TA^i(t)}{\sum_{i=1}^{M} TA^i(t)} \tag{3.23}$$

with *TA* evaluated by Equation (3.7)

3.5 Extension to MIMO Models

Dealing with MIMO models becomes simple in this work. As below, a MIMO model is shown a collection of several MISO models. Interested readers might follow Section 3.5 to see how a MIMO model is equivalent to multiple MISO or come back later to revisit the subject when dealing with a MIMO case.

For a MIMO process, a general description of its one-step predictor is defined by

$$\hat{\mathbf{y}}(t) = \mathbf{f}\left(\mathbf{y}(t-1), \mathrm{L}\ , \mathbf{y}(t-ny), \mathbf{u}(t-d), \mathrm{L}\ , \mathbf{u}(t-d-nu)\right) \tag{3.24}$$

where, the MIMO model has *n* outputs and *m* inputs. The output and input vectors, $\mathbf{y}(t)$ and $\mathbf{u}(t)$ are defined by

$$\begin{aligned} \mathbf{y}(t) &= \left[y_1(t) \quad \mathrm{L} \quad y_n(t) \right]^T \\ \mathbf{u}(t) &= \left[u_1(t) \quad \mathrm{L} \quad u_m(t) \right]^T \end{aligned} \tag{3.25}$$

The above model structure implicitly assumes that the dynamic orders in all y_i (i=1,…,*n*) and u_j(j=1,…,*m*) for each output $y_k(t)$ are *ny* and *nu* respectively. A universal time delay is also assumed between each pair of u_j and y_k. The universal order assumption is in general not true in practice. However, a MIMO GTSK in discrete time model could be modified to have such a universal-order structure by adding zeros if necessary. A regressor vector $\mathbf{x}(t)$ is defined to collect all input arguments in Equation (3.24)

$$\mathbf{x}(t) = \begin{bmatrix} 1 & \mathbf{y}^T(t-1) & \text{L} & \mathbf{y}^T(t-ny) & \mathbf{u}^T(t-d) & \text{L} & \mathbf{u}^T(t-d-nu) \end{bmatrix}^T \quad (3.26)$$

where the dimension of $\mathbf{x}(t)$ is $nx+1$ with $nx = n \times ny + (m+1) \times nu$. The model is then defined as below

$$\begin{aligned} &\textbf{IF } \left(\mathbf{c}(t) \text{ is in } R^1\left(\mathbf{o}^1, \mathbf{P}^1\right) \right) \textbf{THEN } \hat{\mathbf{y}}^1(t) = \boldsymbol{\theta}^1 \mathbf{x}(t) \\ &\text{M} \\ &\textbf{IF } \left(\mathbf{c}(t) \text{ is in } R^M\left(\mathbf{o}^M, \mathbf{P}^M\right) \right) \textbf{THEN } \hat{\mathbf{y}}^M(t) = \boldsymbol{\theta}^M \mathbf{x}(t) \end{aligned} \quad (3.27)$$

The model in Equation (3.27) is almost identical to that in Equation (3.14). $\mathbf{o}^i$ and $\mathbf{P}^i$ have the same meaning. Antecedent variables are included in vector $\mathbf{c}(t)$, which is also a subset of $\mathbf{x}(t)$. The only difference is that local models in Equation (3.28) are multiple-output. The vector $\hat{\mathbf{y}}^i$ collects the n output predictions by the local model in the i^{th} rule

$$\hat{\mathbf{y}}^i = \begin{bmatrix} \hat{y}_1^i & \text{L} & \hat{y}_n^i \end{bmatrix}^T \quad (3.28)$$

Consequently, local model parameters are organized in a matrix $\boldsymbol{\theta}^i \in \text{R}^{n \times (nx+1)}$ defined by

$$\boldsymbol{\theta}^i = \begin{bmatrix} \theta_{1,0}^i & \theta_{1,1}^i & \text{L} & \theta_{1,nx}^i \\ \text{M} & \text{M} & \text{O} & \text{M} \\ \theta_{n,0}^i & \theta_{n,1}^i & \text{L} & \theta_{n,nx}^i \end{bmatrix} \quad (3.29)$$

Each row of $\boldsymbol{\theta}^i$ corresponds to an output and every column of $\boldsymbol{\theta}^i$ is related to a regressor. It is possible to decompose $\boldsymbol{\theta}^i$ in terms of columns or rows as below

$$\boldsymbol{\theta}^i = \begin{bmatrix} \boldsymbol{\theta}_0^i & \text{L} & \boldsymbol{\theta}_{nx}^i \end{bmatrix}; \qquad \boldsymbol{\theta}^i = \begin{bmatrix} {}_1\boldsymbol{\theta}^i \\ \text{M} \\ {}_n\boldsymbol{\theta}^i \end{bmatrix} \quad (3.30)$$

Where $\boldsymbol{\theta}_j^i$ (j=0,…,nx) represents the j^{th} column in matrix $\boldsymbol{\theta}^i$ and rows ${}_k\boldsymbol{\theta}^i$ represents the k^{th} row in $\boldsymbol{\theta}^i$ (k=1,…,n)

The computation in Equation (3.22) is then extended as below to deal with a multiple-output model.

$$\begin{bmatrix} \hat{y}_1(t) \\ \text{M} \\ \hat{y}_n(t) \end{bmatrix} = \sum_{i=1}^{M} w^i(t) \begin{bmatrix} {}_1\boldsymbol{\theta}^i \\ \text{M} \\ {}_n\boldsymbol{\theta}^i \end{bmatrix} \mathbf{x}(t) \tag{3.31}$$

Equation (3.31) could be viewed as a collection of n single-output models. For instance, the computation for the k^{th} output is

$$\hat{y}_k(t) = \sum_{i=1}^{M} w^i(t)\, {}_k\boldsymbol{\theta}^i \mathbf{x}(t) \tag{3.32}$$

where ${}_k\boldsymbol{\theta}^i$ defined in Equation (3.30) is the k^{th} row of matrix $\boldsymbol{\theta}^i$. It then is possible to define a single output GTSK model for the k^{th} output only by

$$\begin{aligned} &\textbf{IF } \left(\mathbf{c}(t) \text{ is in } R^1\left(\mathbf{o}^1, \mathbf{P}^1\right)\right) \textbf{THEN } \hat{y}_k^1(t) = {}_k\boldsymbol{\theta}^1 \mathbf{x}(t) \\ &\text{L} \\ &\textbf{IF } \left(\mathbf{c}(t) \text{ is in } R^M\left(\mathbf{o}^M, \mathbf{P}^M\right)\right) \textbf{THEN } \hat{y}_k^M(t) = {}_k\boldsymbol{\theta}^M \mathbf{x}(t) \end{aligned} \tag{3.33}$$

Comparing the single-output model for output y_{k} with that in Equation (3.14), equivalence is established by equating ${}_{\text{k}}\boldsymbol{\theta}^{\text{i}}$ in Equation (3.33) to $\boldsymbol{\theta}^{\text{i}}$ in Equation (3.14). However, two models are different. Model in Equation (3.14) is SISO while that in Equation (3.33) is MISO. The $\mathbf{x}(t)$ in Equation (3.33) actually collects the lagged multiple inputs and lagged multiple outputs. Fortunately, the difference in contents in $\mathbf{x}(t)$ has no impact on evaluation of the first and second order derivatives to be presented later. The computation of gradients and Hessian matrices for a SISO GTSK model can be extended directly to each MISO element in a MIMO GTSK model.

A matrix ${}_{\text{k}}\boldsymbol{\theta}$ is defined to collect all local model parameters for the k^{th} output.

$${}_k\boldsymbol{\theta} = \begin{bmatrix} {}_k\boldsymbol{\theta}^1 \\ \text{M} \\ {}_k\boldsymbol{\theta}^r \end{bmatrix} \tag{3.34}$$

The above decomposition can facilitate estimation of model parameters in terms of evaluation of derivatives if a decomposable performance index is used. Simply, the centroids and shape matrices have global influence on a GTSK model. Their influence on all outputs should be accumulated. To the contrary, the consequent parameters, ${}_k\boldsymbol{\theta}$ have only local influence on its corresponding output y_k. It then could be expected that the interactions between ${}_k\boldsymbol{\theta}$ and ${}_l\boldsymbol{\theta}$ ($k \neq l$) is zero. The representation of a MIMO GTSK model by several single-output GTSK models will be exploited in Chapter 5 to derive the first and second order derivatives of an objective function with respect to model parameters.

CHAPTER IV

DYNAMIC ORDER DETERMINATION AND NONLINEAR COMPONENT DETECTION

Determination of dynamic orders (*ny*, *nu* and *d* in Equation (3.1)) is the first step in system identification. Order determination is in general difficult for nonlinear system identification due to the interaction of model structure (unknown orders) and unknown nonlinearity. If the attenuation of unknown nonlinearity is possible, different model structures could then be fairly compared. Guided by this concept, the work in this chapter uses a recursive estimation to reduce the effect of the underlying nonlinearity on parameter variation, and proposes a sequential nearest neighbor rearrangement to enhance the reduction. The "best" dynamic order will minimize a final prediction error with the consideration of the locality of the model parameters. In addition to determining dynamic orders, the sequential nearest neighbor rearrangement is also extended to detect nonlinear components, which are regressors responsible for parameter variation if a nonlinear dynamic model is converted to a liner time-varying dynamic model. The result from Chapter 4 could be viewed as the preliminary analysis for building a GTSK model to be presented in Chapter 5. The dynamic order determination defines the overall dimension of a model. The nonlinear component detection selects antecedent variables for the model.

4.1 Dynamic Order Determination

The dynamic orders *ny*, *nu* and delay *d* are described in Equation (3.1). The difficulty in discovering the dynamic orders for a nonlinear dynamic model is caused by the unknown nonlinearities. Even if *f* is known to be nonlinear, the richness of nonlinearity would keep users from exhausting all possible nonlinear forms, making it difficult to find *ny*, *nu* and *d*. If the unknown nonlinearity is not a problem or at least not as severe as it was, it is possible to devise a procedure for dynamic order analysis for a NARX. The objective of the following methodology is to detangle the nonlinearity and dynamic orders, which makes it possible to define model orders. The methodology simply involves two stages of works. The first is to attenuate the unknown nonlinearity. The second is search for dynamic orders.

4.1.1 Nonlinearity Representation

Nonlinearity could be explicitly or implicitly expressed. It is possible to transform a nonlinear dynamic model into a linear one if the nonlinear function is known. For instance, the following nonlinear dynamic model

$$y(t) = 0.4y(t-1)^3 + u(t-1)^3 + e(t) \tag{4.1}$$

could be redefined as a linear dynamic model by static transformation $z(t) = y(t\text{-}1)^3$, $v(t) = u(t\text{-}1)^3$

Unknown nonlinearity could be addressed by using structure-rich models such as neural network models, basis function models and fuzzy systems. These models are all universal approximators and able to capture almost any nonlinearity given enough flexibility. If a neural network model is used, one then could use the following procedure to find proper dynamic orders. A neural network is tried for different sets of *ny*, *nu* and *d* and the best set is then reported. Due to the application of a neural network, the nonlinearity is presumably addressed. The only affecting factors for modeling performance are *ny*, *nu* and *d*. It then is possible to find the set with the best performance. This approach is very general and could be applied to any scenarios, any nonlinear

dynamic models by any universal approximators. The drawback is the computational burden in terms of training 'big' models and efforts put to select appropriate network architecture (number of layers, nodes in each layer in neural networks; number of fuzzy subsets, number of rules in a fuzzy system).

If simple models are preferred such as linear models, nonlinearity could be addressed by adaptation. Model parameters are recursively updated to track the model parameter variation caused by nonlinearity. Linear models with parameter adaptation require much less computation compared to 'big' models. The following example shows how convert a nonlinear dynamic process to a linear format. The example uses a NARX model defined by

$$y(t) = \frac{y(t-1)}{1+y(t-1)^3} + u(t-1)^3 + e(t) \tag{4.2}$$

which could be represented in a linear format

$$y(t) = a_1(t)\,y(t-1) + b_0(t)\,u(t-1) + e(t) \tag{4.3}$$

where $a_1(t)$ and $b_0(t)$ are time-varying model parameters and are defined in Equation (4.4) as functions of $y(t\text{-}1)$ and $u(t\text{-}1)$ to establish one-to-one correspondence between Equation (4.3) match Equation (4.2)

$$a_1(t) = \frac{1}{1+y(t-1)^2}, \qquad b_0(t) = u(t-1)^2 \tag{4.4}$$

In general, the nonlinear dynamic model in Equation (3.1) could be expressed in the following linear format

$$\begin{aligned} y(t) = {} & a_1(t)\,y(t-1) + \mathrm{L} + a_{ny}(t)\,y(t-ny) + \\ & b_0(t)\,u(t-d) + \mathrm{L} + b_{nu}(t)\,u(t-nu-d) + e(t) \end{aligned} \tag{4.5}$$

The linear format could be established from a known nonlinear dynamical model by one-to-one correspondence as shown in Equation (4.4). However, the linear format is not always unique and one could have options. For instance, given a NARX model defined in Equation (4.6)

$$y(t)=\frac{y(t-1)y(t-2)\left(y(t-1)+2.5\right)}{1+y(t-1)^2+y(t-2)^2}+u(t-1)+e(t) \tag{4.6}$$

It is possible to define a linear format

$$y(t)=a_1(t)y(t-1)+b_0(t)u(t-1)+e(t) \tag{4.7}$$

with

$$a_1(t)=\frac{y(t-2)\left(y(t-1)+2.5\right)}{1+y(t-1)^2+y(t-2)^2}, \qquad b_0(t)=1$$

another possibility is defined in Equation (4.8) with a different set of time-varying model parameters

$$y(t)=a_1(t)y(t-1)+a_2(t)y(t-2)+b_0(t)u(t-1)+e(t) \tag{4.8}$$

with

$$a_1(t)=\frac{2.5y(t-2)}{1+y(t-1)^2+y(t-2)^2}, \qquad a_2(t)=\frac{y(t-1)^2}{1+y(t-1)^2+y(t-2)^2}, \qquad b_0(t)=1$$

In general, it is rather difficult (maybe impossible) to extract the exact parameter functions as defined in Equation (4.4) from data only. There are few exceptions such as the one mentioned in (Young, 1993), where $a_1(t)$ and $b_0(t)$ are known to be linear functions of $y(t\text{-}1)$ and $u(t\text{-}1)$.

The nonlinear dynamic model in Equation (3.1) could also be approximately expressed by the following time-varying model

$$\begin{aligned} y(t) \approx k(t) + a_1(t)\, y(t-1) + \mathrm{L} \; + a_{ny}(t)\, y(t-ny) + \\ b_0(t)\, u(t-d) + \mathrm{L} \; + b_{nu}(t)\, u(t-nu-d) + e(t) \end{aligned} \tag{4.9}$$

The approximation is due to the first-order Taylor expansion of Equation (3.1) with following definitions

$$\begin{aligned} k(t) = y(t_0) - \frac{\partial f}{\partial y(t-1)}\Big|_{t_0} y(t_0-1) - \mathrm{L} \; - \frac{\partial f}{\partial y(t-ny)}\Big|_{t_0} y(t_0-ny) - \\ \frac{\partial f}{\partial u(t-d)}\Big|_{t_0} u(t_0-1) - \mathrm{L} \; - \frac{\partial f}{\partial y(t-ny)}\Big|_{t_0} u(t_0-nu-d) \end{aligned}$$

$$\begin{aligned} a_1(t) &= \frac{\partial f}{\partial y(t-1)}\Big|_{t_0} \\ &\mathrm{M} \\ a_{ny}(t) &= \frac{\partial f}{\partial y(t-ny)}\Big|_{t_0} \end{aligned} \tag{4.10}$$

$$\begin{aligned} b_0(t) &= \frac{\partial f}{\partial u(t-d)}\Big|_{t_0} \\ &\mathrm{M} \\ b_{nu}(t) &= \frac{\partial f}{\partial u(t-nu-d)}\Big|_{t_0} \end{aligned}$$

where, t_0 represents the reference point that the Taylor expansion is based on.

The representation of Equation (3.1) by Equation (4.5) or (4.9) are different, although both share the same notations for time-varying model parameters $a(t)$ and $b(t)$. Equation (4.5) is due to the one-to-one correspondence to Equation (3.1), while Equation (4.9) is based on one-to-one correspondence to the first-order Taylor expansion of Equation (3.1). The only difference is the additional time-varying intercept term $k(t)$ in

Equation (4.9) and the following presented order determination procedure is applicable to both structures.

4.1.2 Recursive Estimation for Time Varying Parameters

Equation (4.5) or (4.9) could be represented in a more compact format

$$y(t) = \mathbf{x}^T(t)\boldsymbol{\theta}(t) + e(t) \tag{4.11}$$

with

$$\boldsymbol{\theta}(t) = \begin{bmatrix} k(t) & a_1(t) & \text{L} & a_{ny}(t) & b_0(t) & \text{L} & b_{nu}(t) \end{bmatrix}^T$$

$$\mathbf{x}(t) = \begin{bmatrix} 1 & y(t-1) & \text{L} & y(t-ny) & u(t-d) & \text{L} & u(t-nu-d) \end{bmatrix}^T$$

where, the constant regressor will be dropped if format in Equation (4.5) is used. The output prediction is then defined using the estimates of time-varying parameters

$$\hat{y}(t) = \mathbf{x}^T(t)\hat{\boldsymbol{\theta}}(t) \tag{4.12}$$

There are several different ways to estimate $\boldsymbol{\theta}(t)$. Recursive estimation attempts to estimate local model parameters instantaneously. Another approach uses stochastic models to describe parameter variation if the statistics regarding parameter variation is assumed known. Among them, the simplest one is a random walk model. A Kalman filter is then used to estimate the time-varying parameter values as the states in the stochastic model. The second approach will not be investigated in this work since we assume the lack of knowledge on the statistics of parameter variation.

Recursive estimation for parameter values, $\boldsymbol{\theta}(t)$, is based on a time-varying weighted quadratic performance as below

$$J(t) = \sum_{\tau=1}^{t} \left(y(\tau) - \hat{y}(\tau) \right)^2 w(\tau, t) \tag{4.13}$$

where $w(\tau,t)$ is a weighting function. Commonly used weighting functions include rectangular window weighting and exponential weighting (Ljung & Soderstrom, 1986). In this work, the exponential weighting is used and described by,

$$w(\tau,t)=\alpha^{t-\tau},\ \tau=0,1,\mathrm{L}\ ,N \tag{4.14}$$

where the variable, α, a scalar between 0 and 1, is termed as forgetting factor. Figure 4.1 illustrates a particular exponential weighting with $\alpha = 0.95$.

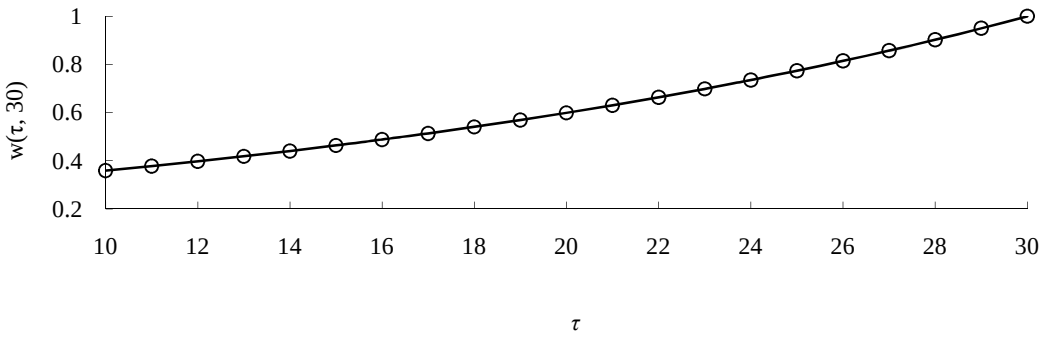


Figure 4.1. Exponential weighting with α = 0.95

Using exponential weighting, the following equations (Young, 1984) are used to update model parameters from $\hat{\boldsymbol{\theta}}(N-1)$ to $\hat{\boldsymbol{\theta}}(N)$

$$\begin{aligned}
&\hat{y}(t)=\mathbf{x}^T(t)\hat{\boldsymbol{\theta}}(t-1)\\
&\mathbf{H}(t)=\mathbf{P}(t-1)\mathbf{x}(t)\left(\mathbf{x}^T(t)\mathbf{P}(t-1)\mathbf{x}(t)+\alpha\right)^{-1}\\
&\hat{\boldsymbol{\theta}}(t)=\hat{\boldsymbol{\theta}}(t-1)-\mathbf{H}(t)\left(\hat{y}(t)-y(t)\right)\\
&\mathbf{P}(t)=\frac{1}{\alpha}\left(\mathbf{P}(t-1)-\mathbf{H}(t)\mathbf{x}^T(t)\mathbf{P}(t-1)\right)
\end{aligned} \tag{4.15}$$

The forgetting factor, determines the influence of data in the past to the current estimation. The suggested range for α is between 0.9 and 0.99 (Young, 1984). In practice, trials for α might be needed for a balanced performance for nonlinearity adaptation speed and parameter estimation precision.

4.1.3 Sequential Nearest Neighbor Rearrangement

In the recursive estimation with exponential weighting, the tuning variable is the forgetting factor. When adjusting the forgetting factor, one should be aware of its conflicting affects on parameter estimates. The forgetting factor relates to the rate of variation. A smaller forgetting factor is expected for faster parameter variation. On the other hand, the precision of parameter estimates is determined by the size of data included in an "effective" window. The length of the window is also a function of the forgetting factor. Smaller is the tuning factor (shorter window), fewer data are included for estimation. In turn, the variance in estimates is high. Therefore, a larger forgetting factor should be preferred for higher estimation precision. However, a larger forgetting factor is only a good choice for slow parameter variation. The above argument verifies the suggested range for forgetting factor over 0.90, where the precaution is also mentioned for using exponential recursive estimation for slow variation at best (Young, 1984).

As a result of the conflicting influence of forgetting factor on parameter estimates, dynamical nonlinear processes being dealt are expected to have slow parameter variation. Unfortunately, the nonlinearity is inherited in the data and determined by the nature of the process to be investigated. There is nothing one can possibly do to alter the nature of the process given only access to test it and generate input-output data. However, the nonlinearity is in fact not really the difficulty that we are aiming at but the source of difficulty, the parameter variation. The nonlinearity is believed to be the cause of parameter variation. It is desired to get around the inherited and inaccessible nonlinear nature of a process to change the parameter variation directly. If it is possible, the improvement of the recursive estimation becomes probable. As proposed below is an approach to manipulate raw data in time sequence to create an artificial sequence of data with slowed parameter variation. The following elaboration starts by defining parameter variation explicitly

$$\begin{aligned} \Delta a_i(t) &= a_i(t) - a_i(t-1) \qquad i = 1,\mathrm{L}\ , ny \\ \Delta b_i(t) &= b_i(t) - b_i(t-1) \qquad j = 0,\mathrm{L}\ , nu \end{aligned} \tag{4.16}$$

with the definition, the following vector collecting variations for all parameters is defined

$$\begin{aligned}\Delta\boldsymbol{\theta}(t) &= \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1) \\ &= \left[\Delta a_1(t) \quad \text{L} \quad \Delta a_{ny}(t) \quad \Delta b_0(t) \quad \text{L} \quad \Delta b_{nu}(t)\right]^T\end{aligned} \tag{4.17}$$

The parameter variation at t could then be quantified by $\left\|\Delta\boldsymbol{\theta}(t)\right\|$, where norm is not specified. The parameter variation (*pv*) for the entire data set is

$$pv = \sum_{t=2}^{N}\left\|\Delta\boldsymbol{\theta}(t)\right\| \tag{4.18}$$

If it is possible to minimize *pv*, it is then expected that resultant data set would be more suitable for a recursive estimation. The optimal solution would be a permutation of a sequence of number (1, …, *N*). Find the right permutation is like to solve a travelling salesman problem to find the shortest path traveling through all cities and visiting each city only once. The optimization problem is *NP*-complete. In this work, a suboptimal solution is pursued rather than the exact optimal solution. The suboptimal solution is the result of a greedy procedure (Cormen, Leiserson, Rivest & Stein, 2001), where minimization of *pv* is decomposed into *N*-1 simpler minimization problems.

$$\begin{aligned}&\min\left(\left\|\Delta\boldsymbol{\theta}(2)\right\| + \left\|\Delta\boldsymbol{\theta}(3)\right\| + \text{L} \ + \left\|\Delta\boldsymbol{\theta}(N)\right\|\right) \\ &\qquad \le \min\left\|\Delta\boldsymbol{\theta}(2)\right\| + \min\left\|\Delta\boldsymbol{\theta}(3)\right\| + \text{L} \ + \min\left\|\Delta\boldsymbol{\theta}(N)\right\|\end{aligned} \tag{4.19}$$

where *N*-1 minimization problems are slightly dependent to each with dependence in every two consecutive tasks.

The greedy procedure is then conduced as below. Assuming $\boldsymbol{\theta}(1)$ is known, then $\boldsymbol{\theta}(2)$ is searched for the problem of min $\|\boldsymbol{\theta}(1)-\boldsymbol{\theta}(2)\|$, which in turn determines $\boldsymbol{\theta}(2)$. Subsequently, $\|\boldsymbol{\theta}(2)-\boldsymbol{\theta}(3)\|$ is minimized and $\boldsymbol{\theta}(3)$ is determined. The procedure stops when $\boldsymbol{\theta}(N)$ is determined. Two fundamental steps are involved in this procedure, determination of $\boldsymbol{\theta}(1)$ and solving the problem of min $\|\boldsymbol{\theta}(k-1)-\boldsymbol{\theta}(k)\|$ to determine $\boldsymbol{\theta}(k)$.

With known $\boldsymbol{\theta}$(k-1), the problem of min $\|\boldsymbol{\theta}$(k-1)-$\boldsymbol{\theta}$(k)$\|$ is fully expanded as below

$$\begin{aligned}
&\min\left\|\boldsymbol{\theta}(k-1)-\boldsymbol{\theta}(k)\right\| \\
&\quad=\min\left\|\left[\Delta a_1(k),\mathrm{L}\ ,\Delta a_{ny}(k),\Delta b_0(k),\mathrm{L}\ ,\Delta b_{nu}(k)\right]^T\right\| \\
&\quad\le\sum_{i=1}^{ny}\min\left\|\Delta a_i(k)\right\|+\sum_{j=0}^{nu}\min\left\|\Delta b_j(k)\right\|
\end{aligned} \tag{4.20}$$

The bound is due to the triangular inequality. The minimization of $\|\boldsymbol{\theta}$(k-1)-$\boldsymbol{\theta}$(k)$\|$ is then translated to minimize *ny*+*nu*+1 smaller objectives simultaneously. Given a time-varying model, the parameters a_i(k) and b_j(k) can be expressed as functions of all states

$$a_i(k)=a_i\begin{pmatrix}y(t-1),\mathrm{L}\ ,y(t-ny),\\ u(t-d),\mathrm{L}\ ,u(t-nu-d)\end{pmatrix} \tag{4.21}$$

The expression for $b_j(t)$ is similar. Note that the indices are different in both sides of Equation (4.21), which simply means that the k^{th} sample in the optimal result is the t^{th} sample in time order. If a_i(k-1) is known and its correspondence sample in time order is τ.

$$a_i(k-1)=a_i\begin{pmatrix}y(\tau-1),\mathrm{L}\ ,y(\tau-ny),\\ u(\tau-d),\mathrm{L}\ ,u(\tau-nu-d)\end{pmatrix}$$

The exact functional form of a_i is unknown. If its continuity and differentiability are assumed and its high order derivatives are assumed to be negligible, the difference between a_i(k-1) and a_i(k) could be approximated b

$$\begin{aligned}
a_i(k-1)-a_i(k)\approx&\sum_{i=1}^{ny}\frac{\partial a_i}{\partial y(t-i)}\Big|_{t_0}\left(y(\tau-i)-y(t-i)\right)\\
&+\sum_{j=0}^{nu}\frac{\partial a_i}{\partial u(t-j-d)}\Big|_{t_0}\left(u(\tau-j-d)-u(t-j-d)\right)
\end{aligned} \tag{4.22}$$

If the first order derivative is bounded by a constant Ga_i, the minimization of $\|a_i(k\text{-}1) - a_i(k)\|$ could be approached by

$$\min \left\| a_i(k-1) - a_i(k) \right\| \le \min_{t \neq \tau} G a_i \left\| \mathbf{x}(\tau) - \mathbf{x}(t) \right\| \tag{4.23}$$

where, **x** is defined in Equation (4.11) and t becomes the decision variable. Since the functional form is uniformly assumed for all parameter functions, the solution of Problem (4.23) will simultaneously minimize the all upper bounds. In this work, the Euclidean norm is used and described by

$$\left\| \mathbf{x}(\tau) - \mathbf{x}(t) \right\|_2 = \sqrt{\sum_{i=1}^{ny} \left(y(\tau - i) - y(t-i) \right)^2 + \sum_{i=0}^{nu} \left(u(\tau - d - i) - u(t - d - i) \right)^2} \tag{4.24}$$

A nearest neighbor will define the solution for Problem (4.20). The solving procedure is then termed as Sequential Nearest Neighbor Rearrangement (SNNR). The resultant regressor and output are labeled as $\mathbf{x}_{snnr}$ and y_{snnr} . The rearrangement starts letting $\mathbf{x}_{snnr}(1) = \mathbf{x}(1)$ and $y_{snnr}(1) = y(1)$. If the nearest neighbor of $\mathbf{x}_{snnr}(1)$ is found to be $\mathbf{x}(t)$, $\mathbf{x}(t)$ and $y(t)$ is then added to the rearranged data set by letting $\mathbf{x}_{snnr}(2) = \mathbf{x}(t)$ and $y_{snnr}(2) = y(t)$. Then the nearest neighbor of $\mathbf{x}_{snnr}(2)$ is found and added to the rearranged data set. The procedure continues until the $\mathbf{x}_{snnr}(N)$ is found.

By conducting the SNNR, the raw data in time-sequence is reorganized in spatial-order. The treatment is expected to reduce the parameter variation, which enables the choice of a larger forgetting factor, α which in turn improves the parameter estimates. The results of the SNNR procedure are the basis for the analysis in the following section for dynamic order determination.

However, first is a demonstration of the impact of the SNNR procedure on parameter variation as well as recursive estimation. The demonstration is based on the deterministic nonlinear dynamic model in Equation (4.25)

$$y(t) = \frac{y(t-1)}{1 + y(t-1)^2} + u(t-1)^3 \tag{4.25}$$

Figure 4.2 shows the first 1000 out of 5000 samples generated from the deterministic model when $u(t)$ is driven by a "skyline" function.

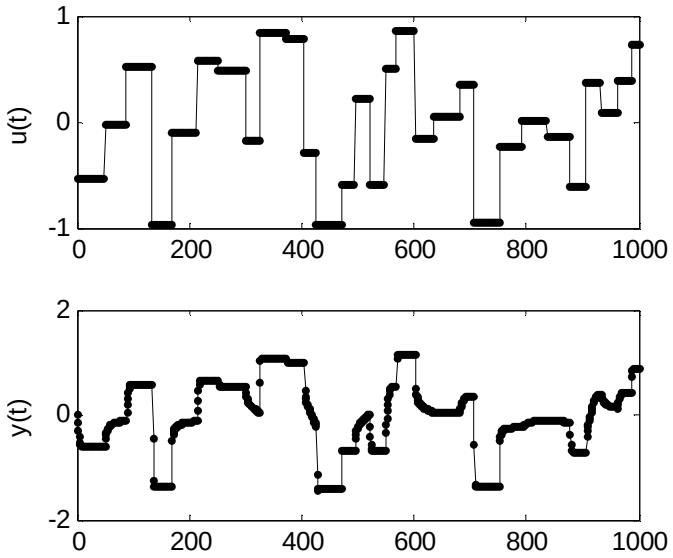


Figure 4.2. Data generated from the model in Equaiton (4.25)

The time-varying model parameters $a_1(t)$ and $b_0(t)$ are defined in Equation (4.4) and their variation over time is shown in Figure 4.3.

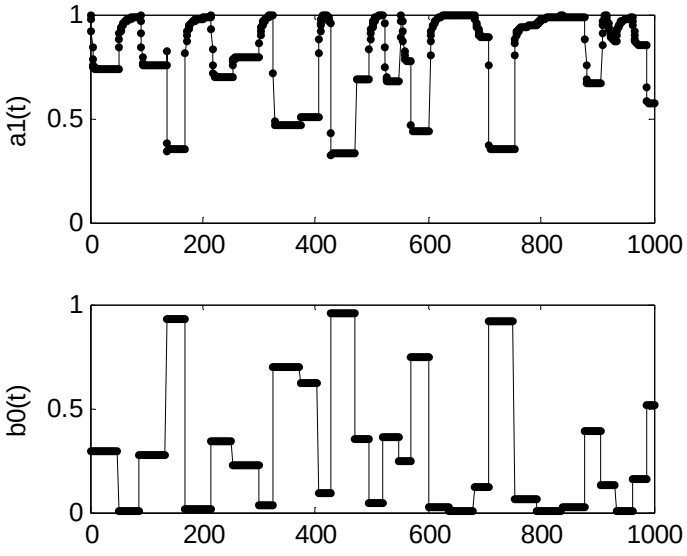


Figure 4.3. Time varying parameters $a_1(t)$ and $b_0(t)$ in Equation (4.4)

The parameter variation (*pv*) defined Equation (4.18) is then evaluted using the

Euclidean norm, $pv = \sum_{t=2}^{5000} \sqrt{\left(a_1(t) - a_1(t-1)\right)^2 + \left(b_0(t) - b_0(t-1)\right)^2}$. The obtained pv is 99.25. The mean squared error (MSE) resulted from a recursive estimation on the time-sequenced data is 0.0044.

The SNNR operation is illustrated on a segment of data with 10 samples. The raw data in time sequence is shown in Table 4.1 indexed by t.

Table 4.1. A segment of 10 data samples in time sequence

t	1	2	3	4	5	6	7	8	9	10
$y(t\text{-}1)$	0	0.2488	-0.8683	0.7200	-0.3775	-1.1465	-0.2815	-0.1014	-0.8542	0.1648
$u(t\text{-}1)$	0	0.2076	0.7200	0.7603	0.3617	0.8668	-0.0913	-0.3199	0.7120	-0.2645

The SNNR rearranged data is shown in Table 4.2 and indexed by k. The index t in Table 4.2 tracks the rearrangement and relates the k^{th} data sample in Table 4.2 to its original position in Table 4.1. Two regressors in Table 4.2 are denoted by y_1 and u_1 rather than the time-lagged notations in the original time sequence data set.

Table 4.2. SNNR rearranged data for the time-sequence data in Table 4.1

k	1	2	3	4	5	6	7	8	9	10
t	1	7	8	10	2	5	9	3	4	6
y_1	0	-0.2815	-0.1014	0.1648	0.2488	-0.3775	-0.8542	-0.8683	-0.9385	-1.1465
u_1	0	-0.0913	-0.3199	-0.2645	0.2076	0.3617	0.7120	0.7200	0.7603	0.8668

Figure 4.4 shows the first 1000 samples of SNNR rearranged data for the time-sequenced data in Figure 4.2. It is observed that the abrupt transition between adjaent levels in Figure 4.2 for both $u(t)$ and $y(t)$ is replaced by a smooth transition in both u_1 and y_1 in Figure 4.4.

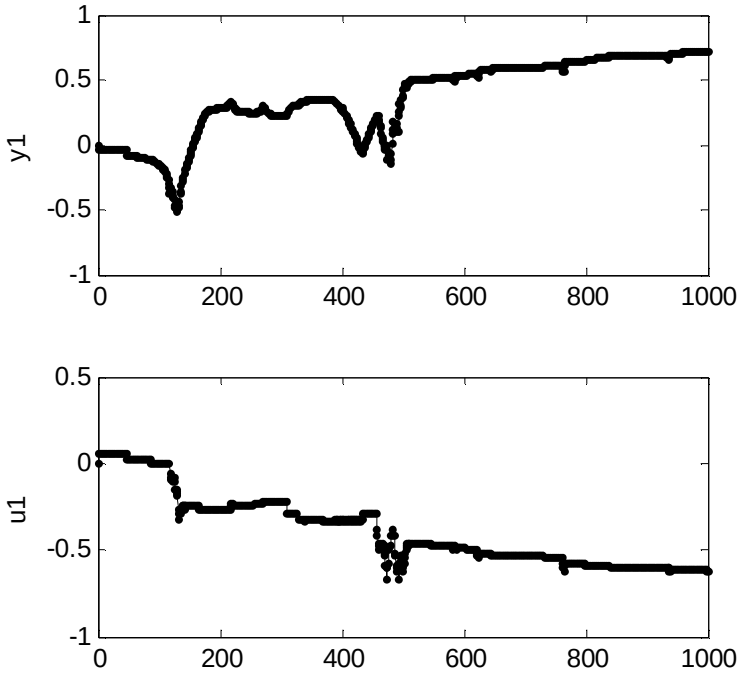


Figure 4.4. SNNR Rearranged regressors from the time-sequence data in Figure 4.1

For the rearranged data, the varying parameters are redefined in terms of u_1 and y_1

$$a_1(k) = \left(1 + y_1(k)^2\right)^{-1} \qquad b_0(k) = u_1(k)^2$$

The variation of $a_1(k)$ and $b_0(k)$ is shown in Figure 4.5, which results in a parameter variation of 32.03, only about a third of that in the time-sequence data. The mean squared error (MSE) resulted from a recursive estimation on the rearranged data is 0.0022, which is half of that in the time-sequence data.

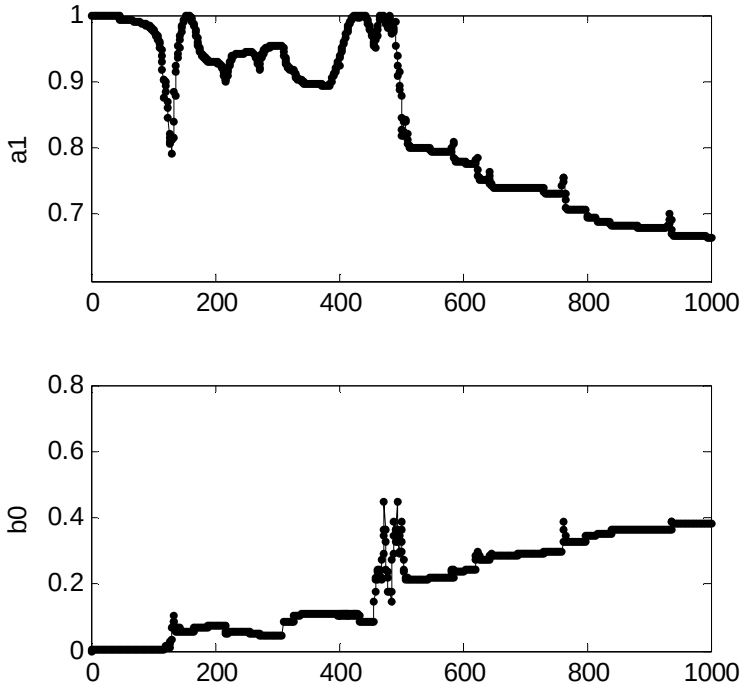


Figure 4.5. Varying parameters for the SNNR rearranged data.

The same test and comparison is conducted on 6 deterministic models, their stochastic versions are defined in Equations (4.40~4.45). The results are summarized in Table 4.3.

Table 4.3. MSE for a recursive estimation

	Time Sequence	SNNR
Model 1	0.0148	0.0112
Model 2	0.0486	0.0084
Model 3	0.0044	0.0025
Model 4	0.0022	0.0017
Model 5	0.0064	0.0034
Model 6	7.38e-8	3.57e-5

As observed in Table 4.3, SNNR is able to reduce the MSE in the Models 1~5.

Increase of MSE is however observed in the Model 6 test, where the tested model is linear. Therefore, the increase of MSE might signal the ineffectiveness of SNNR treatment and imply that the model is linear. Using this feature, one might use the SNNR to tell if a given process is linear or nonlinear.

4.1.4 Model Comparison Criterion

The methodology for determination of dynamic orders could be trying different sets of *ny*, *nu* and *d* and find the best values. Given a set of *ny*, *nu* and *d*, regressors are determined first on the original time-sequenced data, $\mathbf{x}(t)$. A SNNR is then conducted on $\mathbf{x}(t)$ and $y(t)$ producing $\mathbf{x}_{snnr}(t)$ and $y_{snnr}(t)$, to which an exponential weighting recursive estimation will be applied. The quality of the hypothesized *ny*, *nu* and *d* will then be evaluated by a criterion considering both fitting and generalization performance. In this work, the evaluation is based upon a modified final prediction error (FPE) criterion. The original FPE (Ljung, 1999) is defined for a linear model with *N* samples by

$$FPE = \frac{N + np}{N - np} \frac{1}{N} \sum_{t=1}^{N} \varepsilon^2 \left(\mathrm{t}, \hat{\boldsymbol{\theta}}_N \right) \tag{4.26}$$

Equation (4.26) can be interpreted as a weighted mean squared error where the weighting is determined by *N*, the size of data set as well as the model complexity, *np*, the number of parameters. The *FPE* criterion results from the performance index

$$V_N = \sum_{t=1}^{N} \varepsilon^2 \left(\mathrm{t}, \hat{\boldsymbol{\theta}}_N \right) \tag{4.27}$$

In application to exponentially-weighted recursive estimation, the definition of FPE is modified according to the exponentially weighted performance index

$$V_k = \sum_{\mathrm{t}=1}^{k} \alpha^{k-\mathrm{t}} \varepsilon^2 \left(\mathrm{t}, \hat{\boldsymbol{\theta}}_k \right) \tag{4.28}$$

where V_k is varying, and progressively includes more data. The weighting factor, α^{k-t} would become very small for long-past data sets, making the remote error

inconsequential in estimating $\boldsymbol{\theta}_k$. A critical number L is hence introduced to decompose V_k as below

$$\begin{aligned} V_k &= \sum_{\mathrm{t}=1}^{k-L} \alpha^{k-\mathrm{t}} \varepsilon^2\left(\mathrm{t}, \hat{\boldsymbol{\theta}}_k\right) + \sum_{\mathrm{t}=k-L+1}^{k} \alpha^{k-\mathrm{t}} \varepsilon^2\left(\mathrm{t}, \hat{\boldsymbol{\theta}}_k\right) \\ &\approx \sum_{\mathrm{t}=k-L+1}^{k} \alpha^{k-\mathrm{t}} \varepsilon^2\left(\mathrm{t}, \hat{\boldsymbol{\theta}}_k\right) \end{aligned} \tag{4.29}$$

where, V_k is approximated by its recent portion. By this approximation, the number of data involved in V_k is a constant, L. Subsequently, the FPE based on V_k is redefined

$$FPE(k) = \frac{L+np}{L-np}\frac{1}{L}\sum_{\mathrm{t}=k-L+1}^{k} \alpha^{k-\mathrm{t}} \varepsilon^2\left(\mathrm{t}, \hat{\boldsymbol{\theta}}_k\right) \tag{4.30}$$

where, the implicit constraints on t by k-L+1≥1 and k≤N bound k between L and N. The average of FPE(k) over all k is then defined

$$\begin{aligned} \overline{FPE} &= \frac{1}{N-L+1}\sum_{k=L}^{N} FPE(k) \\ &= \frac{L}{N-L+1}\frac{L+np}{L-np}\sum_{k=L}^{N}\sum_{\mathrm{t}=k-L+1}^{k} \alpha^{k-\mathrm{t}} \varepsilon^2\left(\mathrm{t}, \hat{\boldsymbol{\theta}}_k\right) \end{aligned} \tag{4.31}$$

where, the double sum is decomposed into three parts after being switched

$$\begin{aligned} \sum_{k=L}^{N}\sum_{\mathrm{t}=k-L+1}^{k} \alpha^{k-\mathrm{t}} \varepsilon^2\left(\mathrm{t}, \hat{\boldsymbol{\theta}}_k\right) &= \sum_{\mathrm{t}=1}^{L-1}\sum_{k=L}^{\mathrm{t}+L-1} \alpha^{k-\mathrm{t}} \varepsilon^2\left(\mathrm{t}, \hat{\boldsymbol{\theta}}_k\right) \\ &+ \sum_{\mathrm{t}=L}^{N-L+1}\sum_{k=\mathrm{t}}^{\mathrm{t}+L-1} \alpha^{k-\mathrm{t}} \varepsilon^2\left(\mathrm{t}, \hat{\boldsymbol{\theta}}_k\right) + \sum_{\mathrm{t}=N-L+2}^{N}\sum_{k=\mathrm{t}}^{N} \alpha^{k-\mathrm{t}} \varepsilon^2\left(\mathrm{t}, \hat{\boldsymbol{\theta}}_k\right) \end{aligned} \tag{4.32}$$

The recursive estimation works well if parameter variation within a local range is assumed to be small

$$\hat{\boldsymbol{\theta}}_{\mathrm{t}+L-1} \approx \hat{\boldsymbol{\theta}}_{\mathrm{t}+L-2} \approx \mathrm{L} \approx \hat{\boldsymbol{\theta}}_{\mathrm{t}} \tag{4.33}$$

which in turn results in the following approximation

$$\varepsilon^2\left(\mathrm{t},\hat{\boldsymbol{\theta}}_{\mathrm{t}+L-1}\right)\approx\varepsilon^2\left(\mathrm{t},\hat{\boldsymbol{\theta}}_{\mathrm{t}+L-2}\right)\approx\mathrm{L}\ \approx\varepsilon^2\left(\mathrm{t},\hat{\boldsymbol{\theta}}_{\mathrm{t}}\right) \tag{4.34}$$

The double sum is then simplified to

$$\begin{aligned}\sum_{k=L}^{N}\sum_{\mathrm{t}=k-L+1}^{k}\alpha^{k-\mathrm{t}}\varepsilon^2\left(\mathrm{t}\right)=&\sum_{\mathrm{t}=1}^{L-1}\varepsilon^2\left(\mathrm{t},\hat{\boldsymbol{\theta}}_t\right)\sum_{k=L}^{\mathrm{t}+L-1}\alpha^{k-\mathrm{t}}\\&+\sum_{\mathrm{t}=L}^{N-L+1}\varepsilon^2\left(\mathrm{t},\hat{\boldsymbol{\theta}}_t\right)\sum_{k=\mathrm{t}}^{\mathrm{t}+L-1}\alpha^{k-\mathrm{t}}+\sum_{\mathrm{t}=N-L+2}^{N}\varepsilon^2\left(\mathrm{t},\hat{\boldsymbol{\theta}}_t\right)\sum_{k=\mathrm{t}}^{N}\alpha^{k-\mathrm{t}}\end{aligned} \tag{4.35}$$

If N is large, the second part dominates, which results in a further simplified average FPE as

$$\overline{FPE}\approx\frac{L\sum_{k=0}^{L-1}\alpha^k}{N-L+1}\frac{L+np}{L-np}\sum_{\mathrm{t}=L}^{N-L+1}\varepsilon^2\left(\mathrm{t},\hat{\boldsymbol{\theta}}_t\right) \tag{4.36}$$

The average FPE in Equation (4.36) is similar to the original one in Equation (4.26), and has the same interpretation as a weighted prediction error, except that the weighting is different. Once L is chosen, the first term on the right-hand side of Equation (4.36) is a constant. Then Equations (4.26) and (4.36) are similar, with L the data window length, replacing N, the total number of data. A simplified FPE in Equation (4.37) is used in this work and will continue to be denoted as FPE

$$FPE=\frac{L+np}{L-np}\sum_{t=L}^{N-L+1}\varepsilon^2\left(t,\hat{\boldsymbol{\theta}}_t\right) \tag{4.37}$$

The value of L is related to the decomposition by Equation (4.29) and determined by considering α^{L} small enough to be negligible. In this work, L is determined as below

$$L=\frac{4}{1-\alpha} \tag{4.38}$$

where $(1-\alpha)^{-1}$ is termed as *memory time-constant* (Ljung, 1999). As shown in Figure 4.6 , the specification of L in Equation (4.29) will ignore the past data with weights less than

0.02. Additionally, the number $\alpha^{4/(1-\alpha)}$ remains relatively constant between 0.016 and 0.018 if α is over 0.9, which is a common choice for a forgetting factor.

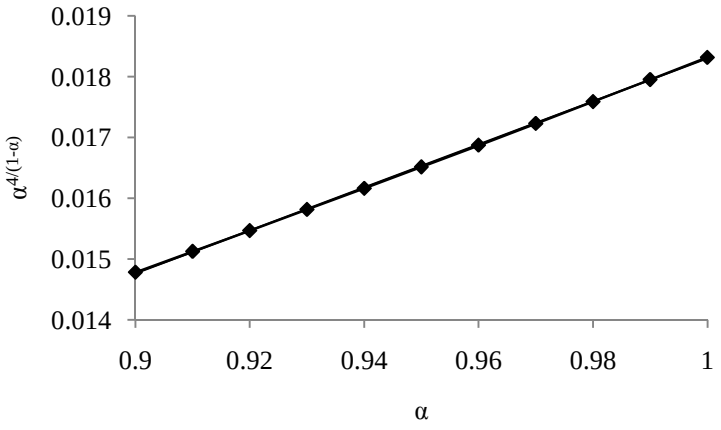


Figure 4.6. α vs. $\alpha^{4/(1-\alpha)}$ (the weight for the most remote data)

4.1.5 Regressor Selection Procedure

Given several sets of *ny*, *nu* and *d*, their FPEs are evaluated. The set with the minimum FPE on SNNR data is reported including the determined orders. The determination procedure could be conducted in an exhaustive approach for all possible combinations of different *ny*, *nu* and *d* given pre-defined *max_ny*, *max_nu* and *max_d* for possible maximum *ny*, *nu* and *d*. The pseudo-code for the exhaustive search is shown in Figure 4.7.

Exhaustive order selection

```
for ny = 1 to max_ny
        for nu = 0 to max_nu
                for d = 1 to max_d
                        Compute FPE(ny,nu,d)
                        Keep the minmum FPE
                end loop
        end loop
end loop
return the ny,nu and d with minimum FPE
```

Figure 4.7. Exhaustive dynamic order search

One concern with the exhaustive search is the computational burden. The pay off of the expensive exhaustive search is optimality of the final solution. Suboptimal search techniques are available in a subset selection for linear regression. Subset selection methods include forward selection, backward elimination, cycling replacement as well as heuristic combinatorial search (Miller, 1990). For linear regression problems, one could fully exploit the superposition feature in a linear model to simplify a search. It explains that subset selection method is always accompanied by orthogonalization. An orthogonalization procedure removes the redundant components of two regressors and eliminates the candidate regressors that are highly correlated with selected regressors.

In nonlinear systems with unknown nonlinearity, orthogonalization is not possible. However, it does not mean that the subset selection is inapplicable. In this work, a forward selection procedure combing the above mentioned recursive estimation on spatially ordered data is proposed to find important regressors. The procedure starts with users' input *max_ny*, *max_nu* and *max_d*. Then, a number of candidate regressors are generated and denoted as $[x_1\ x_2\ x_3\ \ldots\ x_m\ x^{random}]$. x_{random} is a random regressor that presumably contains no meaningful information to predict output. At first, m+1 FPEs are computed for $(y,[x_1])$, $(y, [x_2])$, … , $(y, [x_m])$, $(y, [x_{random}])$, where y is the output and x_i in bracket is the regressor in consideration. The regressor with the minimum FPE is selected. If x_2, for instance, is the first selected regressor, there will be other m FPEs to be evaluated for $(y, [x_2, x_1])$, $(y, [x_2, x_3])$, …, $(y, [x_2, x_m])$, $(y, [x_2, x_{random}])$. Each bracket contains a combination of x_2 (first selected) with the rest. The regressor combination with the minimum FPE is then kept. The selection continues until the minimum FPE increases or the x_{random} is selected. The injection of a random regressor is mentioned in (Miller, 1990) as a stopping criterion. The selection of x_{random} signifies that the rest of candidates are less influential on $y(t)$ than a presumably irrelevant one.

The selected regressors might define values of *ny*, *nu* and *d* if selected regressors are consecutive due to implicit constraint on the model structure in Equation (3.1), which requires consecutive regressors. For instance, a set of regressors [$y(t$-1), $y(t$-2), $u(t$-1), $u(t$-2)] defines ny=2, nu=1, and, d=1. Absences, however, could exist in selected regressors such as [$y(t$-1) $y(t$-4) $u(t$-1) $u(t$-3)], which does not correspond a set of *ny*, *nu* and *d*.

It seems unlikely in most situations that $y(t\text{-}2)$, $y(t\text{-}3)$ and $u(t\text{-}2)$ should not be included. However, if there are strong correlations or recycle phenomena, those missing variables may be redundant, and the particular selection may not be unique. Another realization of excitement and noise, might select another subset from the correlated variables. The inclusion of redundant variables increases the model complexity. However, for database management simplicity, in this work, if the situation with absence occurs, a further comparison is executed on different order values. For the illustrated example, an exhaustive comparison is conducted on possible values of ny=1, 2, 3 or 4 combined the possible values of nu=0, 1, or 2, with d = 1. However, the extra computation would be unnecessary if the constraint on having consecutive regressors is dropped.

4.2 Nonlinear Component Detection

There is an implicit assumption made on the above SNNR operation. The time-varying parameters are functions of all regressors. The assumption is valid for the dynamic model in Equation (4.2), where parameters are functions of two regressors, $u(t\text{-}1)$ and $y(t\text{-}1)$. The model in Equation (4.6) has regressors $y(t\text{-}1)$, $y(t\text{-}2)$ and $u(t\text{-}1)$. However the parameters a_1 and a_2 are functions of only $y(t\text{-}1)$ and $y(t\text{-}2)$. The regressor $u(t\text{-}1)$ has no impact on parameter variation. It is then expected that the SNNR on [$y(t\text{-}1)$ $y(t\text{-}2)$] might reduce more parameter variation than operating SNNR on [$y(t\text{-}1)$ $y(t\text{-}2)$ $u(t\text{-}1)$]. The further reduction in parameter variation should be revealed by a smaller MSE resulted from a recursive estimation.

An extension of the SNNR-based order determination procedure is the used to detect the regressors that are affecting the output nonlinearly. The detected regressors are termed as nonlinear components and to be used as antecedent variables in Chapter 5. The purpose of conducting SNNR is to reduce parameter variation so that the recursive estimation is able to capture the variation better, which in turn, results in a smaller MSE. The SNNR mentioned above rearranges data based on all the regressors in order to compare different sets of ny, nu and d. However, it is possible that only a subset of regressors is affecting time-varying parameters. The subset is denoted by $[c_1,...,c_{nc}]$. It is a subset of selected regressors denoted by $[x_1,\ldots,x_{nx}]$. The regressors not included in

$[c_1,...,c_{nc}]$ have no affect on parameter variation. It is then expected that a SNNR on $[c_1,...,c_{nc}]$ only would be able to reduce more parameter variation and produce an smaller MSE. There are totally 2^{nx-1} subsets in $[x_1,\ldots,x_{nx}]$ excluding the empty one. Each subset from $[x_1,\ldots,x_{nx}]$ is considered as a candidate set of nonlinear components, $[c_1,...,c_{nc}]$, on which the SNNR is conducted and a corresponding MSE is computed. The subset with minimum MSE is reported to contain the nonlinear components.

4.3 Extension to MIMO Processes

Extending the above technique to a MIMO(m,n) (m inputs and n outputs) process is straight forward. The SISO model in Equation (3.1) is expanded as below for the k^{th} output by including more regressors.

$$y_k(t) = f_k\begin{pmatrix} y_k(t-1),\text{L}\ y_k(n-ny_{kk}) \\ y_1\left(t-d_{k1}^y\right),\text{L}\ , y_1\left(t-ny_{k1}-d_{k1}^y\right), \\ \text{L} \\ y_n\left(t-d_{kn}^y\right),\text{L}\ , y_1\left(t-ny_{kn}-d_{kn}^y\right) \\ u_1\left(t-d_{k1}^u\right),\text{L}\ , u_1\left(t-nu_{k1}-d_{k1}^u\right) \\ \text{L} \\ u_m\left(t-d_{km}^u\right),\text{L}\ , u_m\left(t-nu_{km}-d_{km}^u\right) \end{pmatrix} + e_k(t) \tag{4.39}$$

where dynamic orders include ny_{k1}, .., ny_{kn} and nu_{k1}, …, nu_{km}, and delay $d_{k1}^y,\text{L}\ ,d_{kn}^y$ between y_k and other outputs as well as delay $d_{k1}^u,\text{L}\ ,d_{km}^u$ between y_k and all inputs. All of these numbers are to be determined using the above method for the single output case. The nonlinear components for y_k are then selected after orders are determined.

4.4 Simulations and Discussions

4.4.1 Testing Models and Processes

The proposed order determination and nonlinear component detection method are tested on data generated by several nonlinear dynamic models, an experimental unit and a distillation column simulator. The first five models are nonlinear autoregressive with exogenous inputs models (NARX). They are different in terms of nonlinear interactions

between inputs and outputs. Model 1 has nonlinearity only in the lagged input, $u(t\text{-}1)$. Model 2 is nonlinear in lagged output only. Model 3 is nonlinear in both lagged input and output, $u(t\text{-}1)$ and $y(t\text{-}1)$. Model 4 is also nonlinear in both lagged input and output but have more regressors included than Model 3. Like Model 1, Model 5 is another model with nonlinearity in the lagged input, $u(t\text{-}1)$. The nonlinear function with respect to $u(t\text{-}1)$ is, however, different in both models. Model 6 is a linear ARX model used only once to demonstrate the impact of SNNR on recursive estimation with result in Table 4.3.

The input signals used in the first five models are generated by a skyline function and bounded between -1 and 1. The shortest and longest durations are 20 and 50 samples respectively. Output signals are initialized as zeros. The noise $e(t)$ is subject to a normal distribution, $N(0,\sigma^2)$. The value of σ is different in each model and specified such that $e(t)$ has a small magnitude compared to outputs. As below, a portion of input-output data for the first five models is illustrated along with model equations. A total of 5000 samples are generated and used in order determination and nonlinear component detection.

Model 1 (Narendra & Parthasarathy, 1990)

$$y(t) = 0.3y(t-1) + 0.6y(t-2) + 0.6\sin\left(\pi u(t-1)\right) + 0.3\sin\left(3\pi u(t-1)\right) + 0.1\sin\left(5\pi u(t-1)\right) + e(t) \tag{4.40}$$

where $e(t) \sim N(0,0.5^2)$

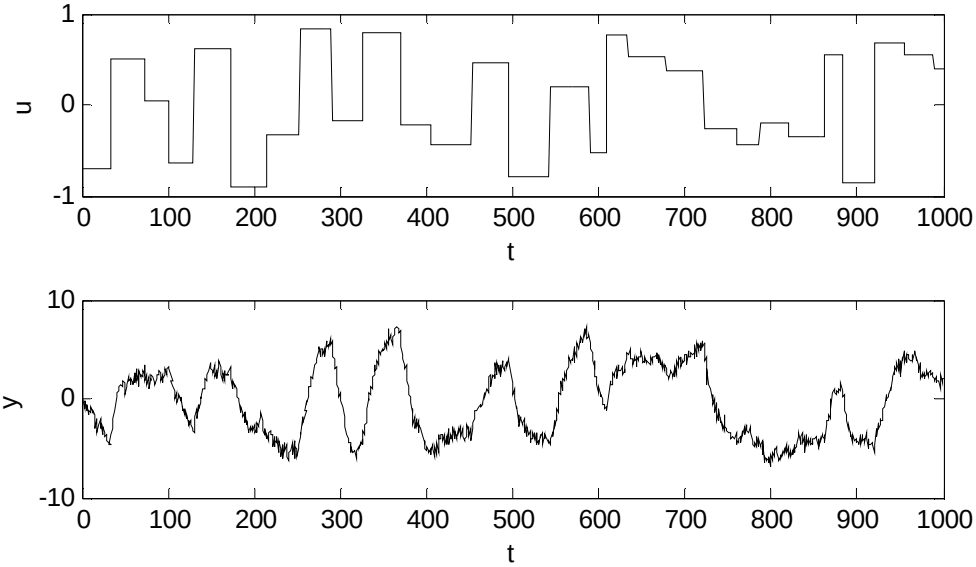


Figure 4.8. Input-output data generated for Model 1 in Equation (4.40)

Model 2 (Narendra & Parthasarathy, 1990)

$$y(t)=\frac{y(t-1)y(t-2)\left(y(t-1)+2.5\right)}{1+y(t-1)^2+y(t-2)^2}+u(t-1)+e(t) \tag{4.41}$$

where $e(t)\sim N(0,0.5^2)$

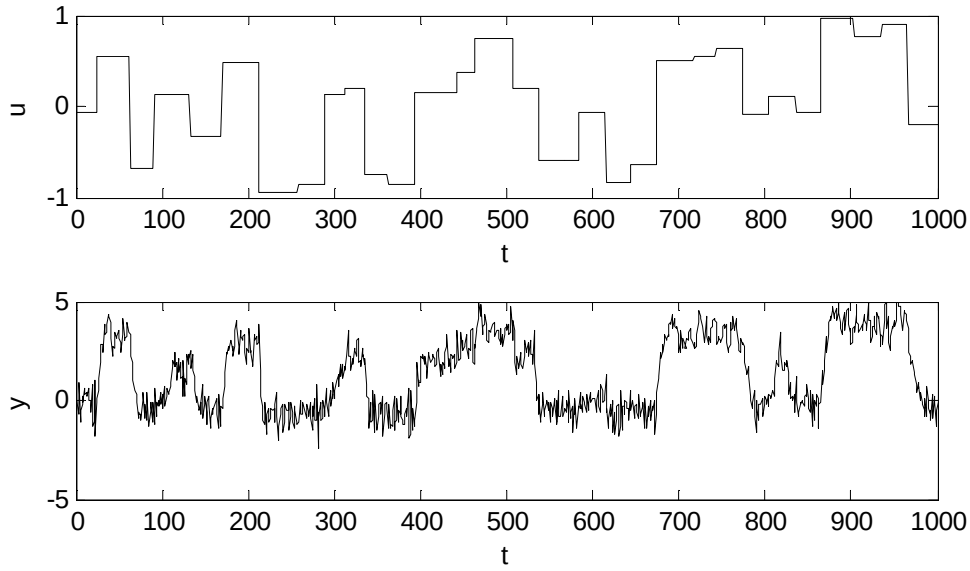


Figure 4.9. Input-output data generated for Model 2 in Equation (4.41)

Model 3 (Narendra & Parthasarathy, 1990)

$$y(t)=\frac{y(t-1)}{1+y(t-1)^2}+u(t-1)^3+e(t) \tag{4.42}$$

where $e(t)\sim N(0,0.5^2)$

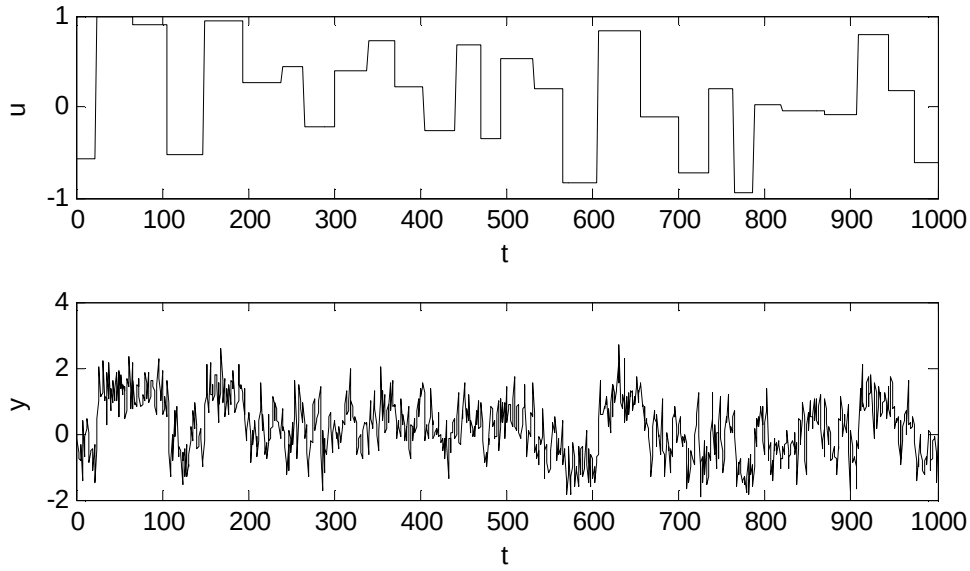


Figure 4.10. Input-output data generated for Model 3 in Equation (4.42)

Model 4 (Narendra & Parthasarathy, 1990)

$$y(t)=\frac{y(t-1)y(t-2)y(t-3)u(t-2)\big(y(t-3)-1\big)+u(t-1)}{1+y(t-3)^2+y(t-2)^2}+e(t) \tag{4.43}$$

where $e(t)\sim N(0,0.05^2)$

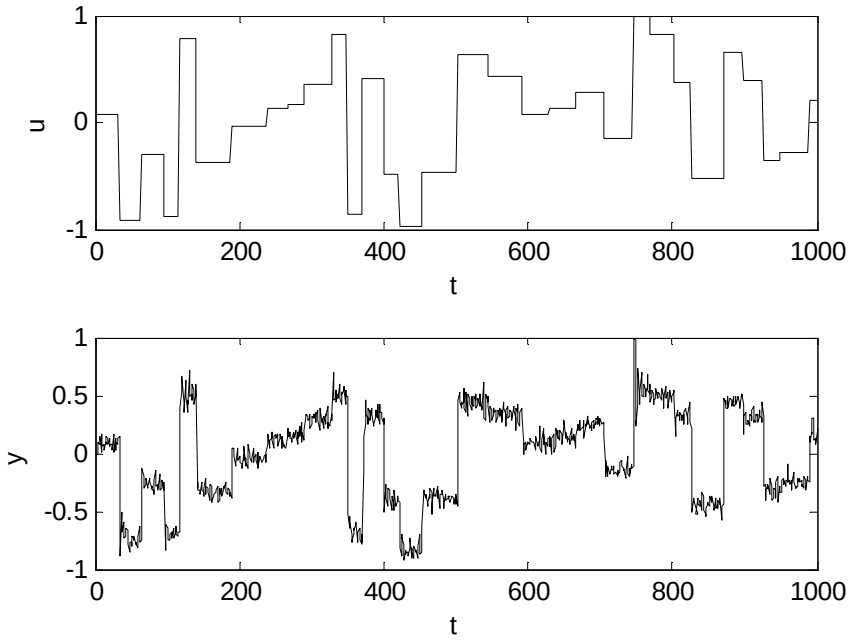


Figure 4.11. Input-output data generated for Model 4 in Equation (4.43)

Model 5 (Narendra & Parthasarathy, 1990)

$$y(t)=0.8y(t-1)+\big(u(t-1)-0.8\big)u(t-1)\big(u(t-1)+0.5\big)+e(t) \tag{4.44}$$

where $e(t)\sim N(0,0.1^2)$

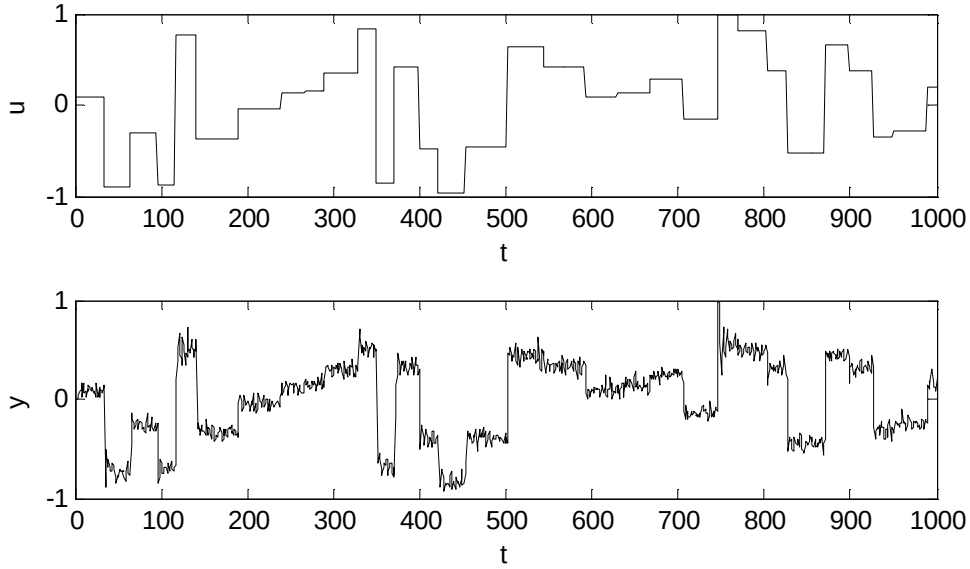


Figure 4.12. Input-output data generated for Model 5 in Equation (4.44)

Model 6:

$$y(t) = 0.8y(t-1) + 0.6y(t-2) + 0.4u(t-1) \tag{4.45}$$

Models 7 and 8 are two deterministic nonlinear dynamic models.

Model 7

$$y(t) = 0.8y(t-1) + u(t-1)^2 \tag{4.46}$$

Model 8

$$y(t) = 0.8y(t-1) + \cos(\pi u(t-1)) \tag{4.47}$$

Different from Models 1-5, Model 7 has a quadratic term $u(t\text{-}1)$, where $u(t)$ is also generated by a "skyline" function between -1 and 1. The effect of $u(t)$ on $y(t)$ would be missed in average. As below, Equation (4.48) is the linear time-varying model for Model 7 with $a_1(t)$=0.8 and $b_0(t)$=$u(t\text{-}1)$. In average, the effect of $u(t\text{-}1)$ in Equation (4.48) is reflected by E($b_0(t)$). In this case, E($b_0(t)$) is 0 since $u(t)$ is a random signal between -1 and 1.

$$y(t) = a_1(t)\,y(t-1) + b_0(t)\,u(t-1) \tag{4.48}$$

Therefore, the regressor $u(t\text{-}1)$ would be missed if a recursive estimation is conducted in time sequence, where $b_0(t)$ is a random number between -1 and 1 in time sequence The recursively estimated $b_0(t)$ would be wandering around zero. However, the proposed SNNR is able to reveal the impact of $u(t\text{-}1)$ on model output. By rearrangement, the randomness in $u(t\text{-}1)$ is eliminated. Consequently, the varying parameter, $b_{0,}$ is no longer a random variable but gradually increases from -1 to 1. A recursive estimation on the rearranged data is then able to reflect the impact of $u(t\text{-}1)$ on $y(t)$. Model 8 has a quadratic-like term cos($\pi u(t\text{-}1)$) and will be used to test the proposed order determination technique.

Model 9 in Equation (4.49) is used to demonstrate the non-uniqueness of obtained result as discussed in Section 4.1.1. By observing Equation (4.49), the nonlinear component could be either $y(t\text{-}1)$ or $y(t\text{-}2)$. A detailed test will reveal the observation.

Model 9

$$y(t) = 0.2\,y(t-1)\,y(t-2) + u(t-1) \tag{4.49}$$

Models 10 and 12 are deterministic nonlinear autoregressive (NAR) models in (Molina, Sampson, Fitzgerald & Niranjan, 1996) and used for method comparison. Models 11 and 13 are derived from Models 10 and 12 with noise added to the output and used to compare the influence of noise on different methods. The noise $e(t)$ in Models 11 and 13 has a small magnitude compared to output signals and is subject to $N(0,\sigma^2)$, where σ^2 is set to about one thousandth of the average magnitude of output signal in the corresponding deterministic models.

Model 10 (Molina, Sampson, Fitzgerald & Niranjan, 1996)

$$y(t) = 4\,y(t-1)\big(1 - y(t-1)\big) \tag{4.50}$$

Model 11:

$$\begin{aligned} y_o(t) &= 4\,y_o(t-1)\big(1 - y_o(t-1)\big) \\ y(t) &= y(t)_o + e(t) \end{aligned} \tag{4.51}$$

where $e(t) \sim N(0, 0.0225^2)$

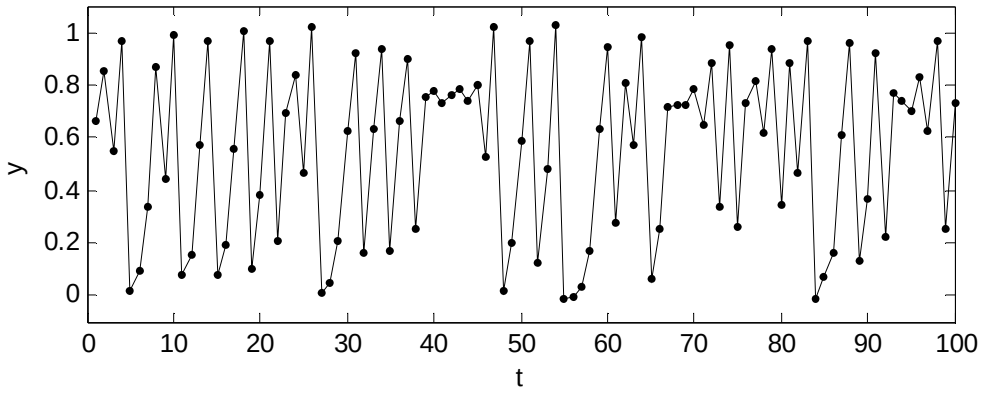


Figure 4.13. Data generated for Model 10 in Equation (4.50)

Model 12 (Molina, Sampson, Fitzgerald & Niranjan, 1996)

$$y(t) = 1 - 1.4y(t-1)^2 + 0.3y(t-2) \tag{4.52}$$

Model 13:

$$\begin{aligned} y_o(t) &= 1 - 1.4y_o(t-1)^2 + 0.3y_o(t-2) \\ y(t) &= y_o(t) + e(t) \end{aligned} \tag{4.53}$$

where $e(t) \sim N(0, 0.0272^2)$

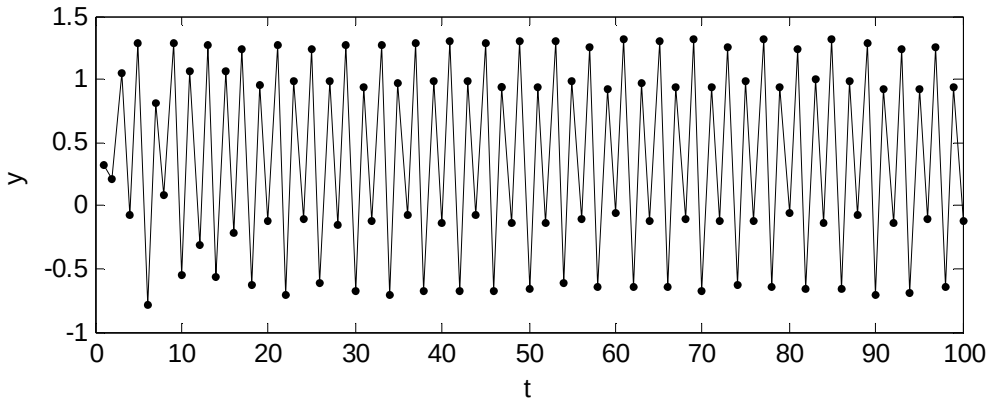


Figure 4.14. Data generated for Model 12 in Equation (4.52)

Model 14: Two-phase flow process

Figure 4.15 shows an experiment setup of a two-phase flow process in the unit operation lab in the School of Chemical Engineering at Oklahoma State University. This unit is managed by a laboratory scale distributed control system, Camile. The schematic diagram of the process is shown in Figure 4.16. Both bottom and top pressures of the vertical pipe are measured. There are two air flow supplies labeled as 'Small air' and 'Large air' in Figure 4.16. Air from the two pipes merges and flows to a T, whose outlet end is connected to the bottom of the vertical pipe. The other inlet end of the T is connected to the water pipe labeled as 'water' in Figure 4.16.

In this work, this unit is used to study the dynamics between mixed air & water and the pressure drop across the vertical pipe. Experiment is conducted in an open loop and only the air valve opening ('Large air' pipe) is manually changed. The 'Small air' pipe is closed. The 'water' pipe is controlled at 20 lbmol/hr. The measurements of the water flowrate in the 'water' pipe are shown in Figure 4.17.

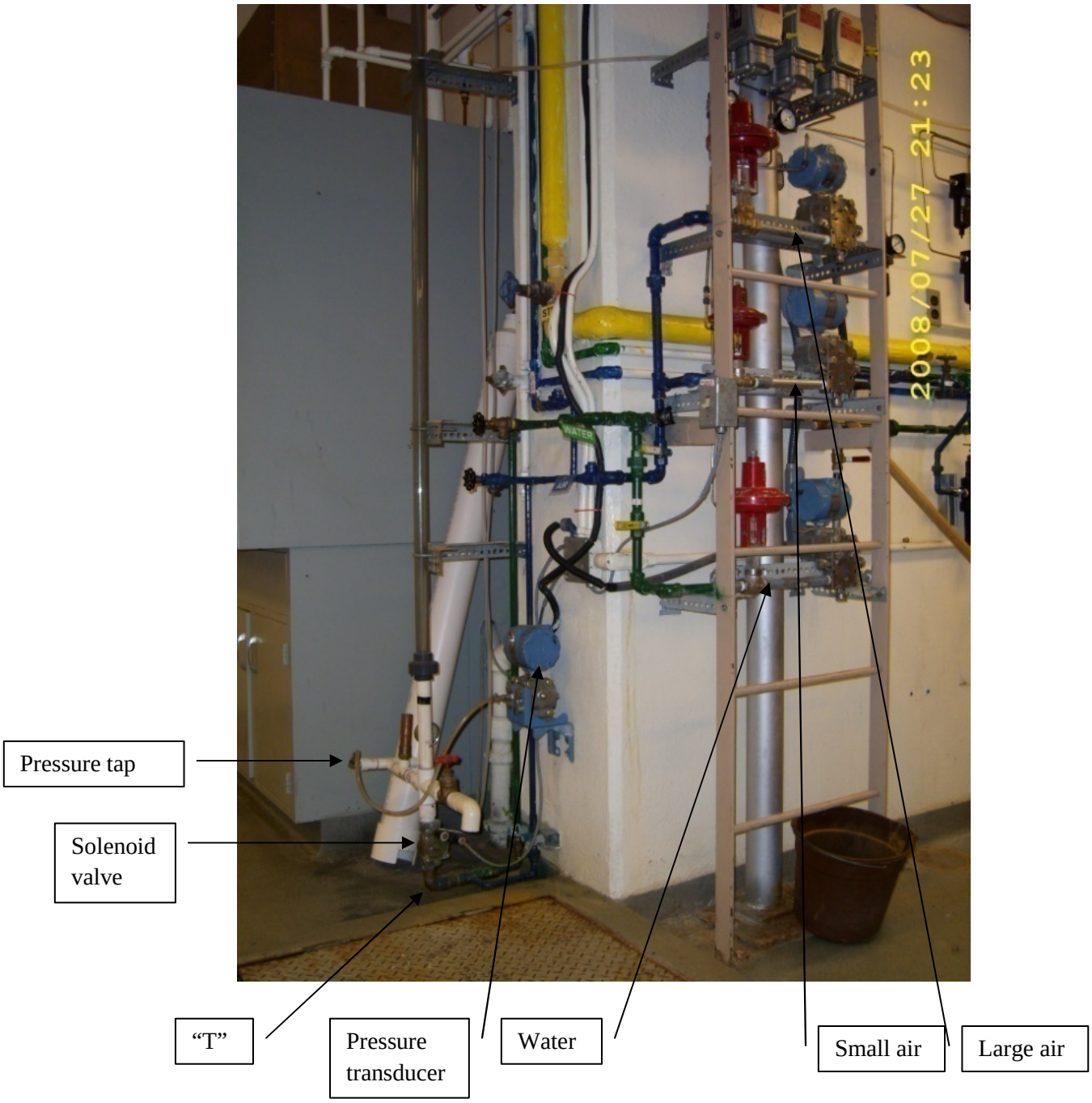


Figure 4.15. The two-phase flow experiment setup

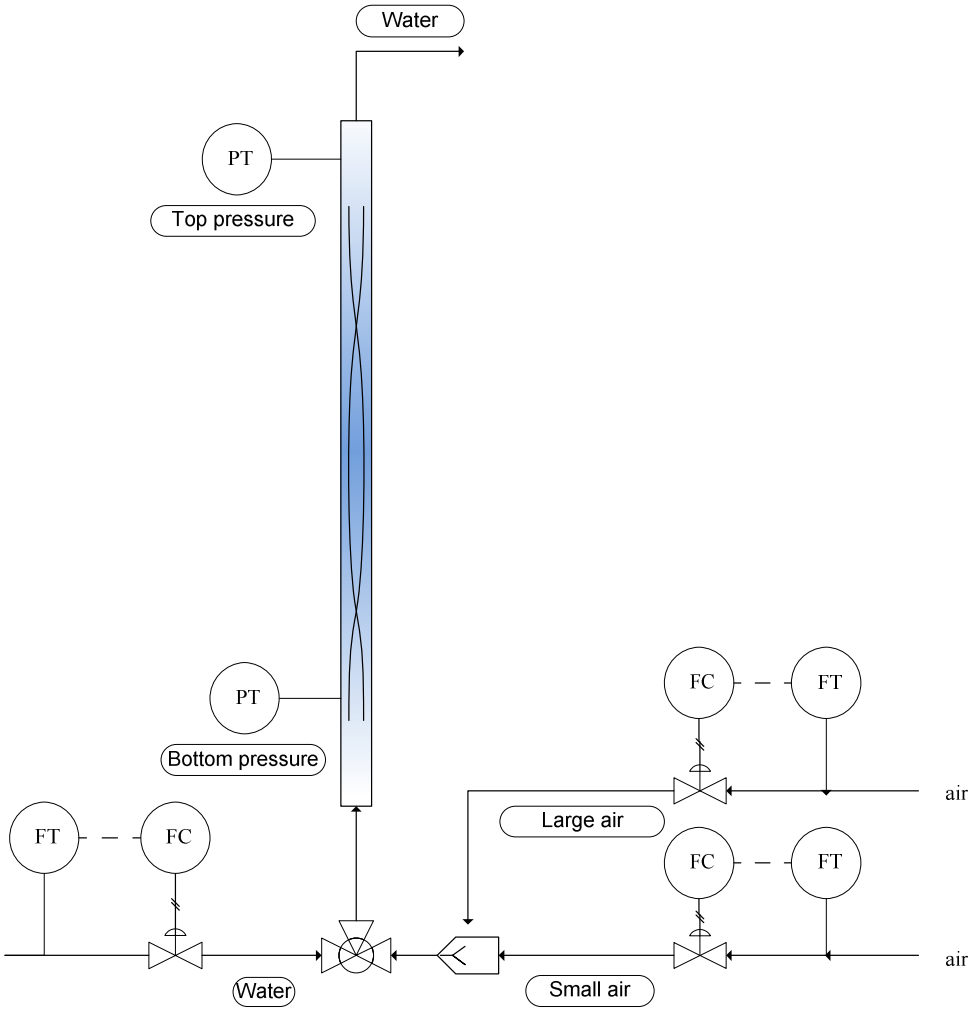


Figure 4.16. The schematic diagram for the two phase flow experiment

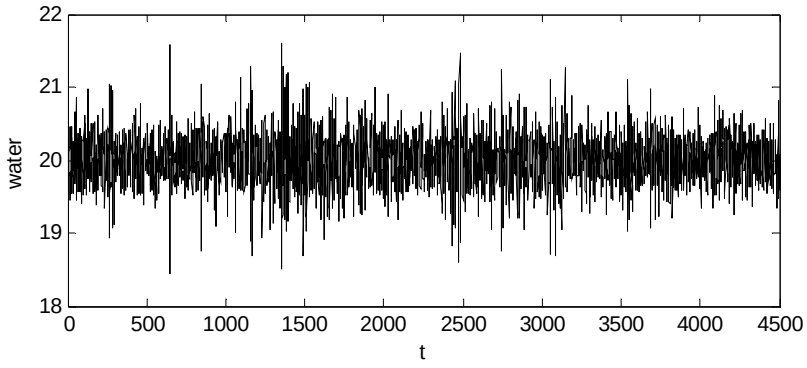
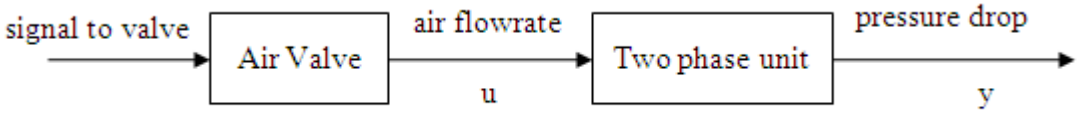
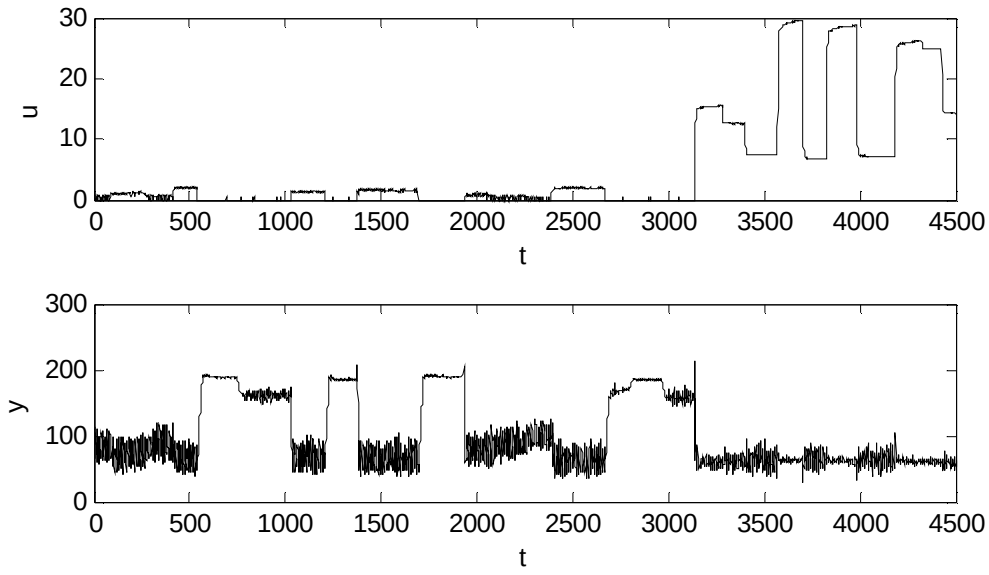


Figure 4.17. Water flowrate measurements with set point at 20 lbmol/hr

The process could be defined differently by taking signals from different channels. Figure 4.18(a) shows a possible choice. The input, *u* is chosen to be the measurement of the air flowrate. The output, *y* is the measuremtn of pressure drop, the difference between top and bottom pressure shown in Figure 4.16. A portion of 4500 measurements are displayed in Figure 4.18(b). There are totally 8830 measurements are recorded. Although the control interval was 0.1 second, the sampling rate for this data was chosen as 0.5 second.



(a)

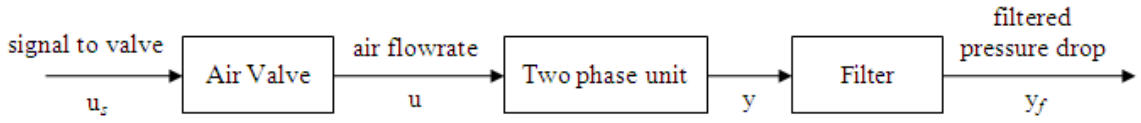


(b)

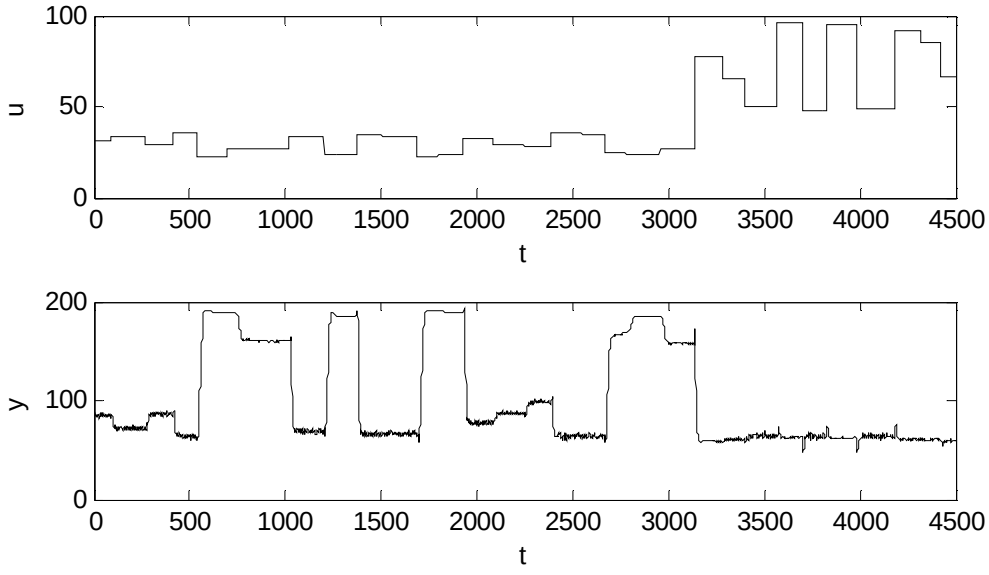
Figure 4.18. A choice of input and output channels; input, *u* is the measurement of air flowrate and output, *y* is the pressure drop measurement. a) The flowchart; b) The corresponding input and output data

As observed in Figure 4.18(b), the pressure drop measurement at low values is

noisier than at high values of y. A first-order filter is added in the data acquisition and control devise (a Camile 2000 unit) to suppress some noise in y for observation convenience. With the filter included, Figure 4.19(a) shows another possible process definition. The input, denoted as u_s is the command signal for the air valve opening, which as shown precedes the air flowrate measurement. The output becomes the filtered pressure drop measurement and denoted as y_f. The data is shown in Figure 4.19(b).



(a)



(b)

Figure 4.19. A choice of input and output channels; input, u_s is the signal to the air valve opening and output, y_f is the filtered pressure drop measurement. a) The flowchart; b) The corresponding input and output data

Model 15: Binary distillation column

Model 15 is a methanol-water binary distillation column simulator (Ou & Rhinehart, 2002) modified to have 20 trays. The distillation column simulator is a MIMO process. Two inputs are reflux flowrate (gmol/hr), u_1, and reboiler heating percentage (TY%), u_2. The sample interval is 30 seconds. The reflux flowrate varies between 50 and 90 (gmol/hr) and heating percentage is between 40% and 55%. The duration time for each step change randomly varies between 0.05 and 1 hour. The first 1000 samples of inputs are illustrated in Figure 4.20.

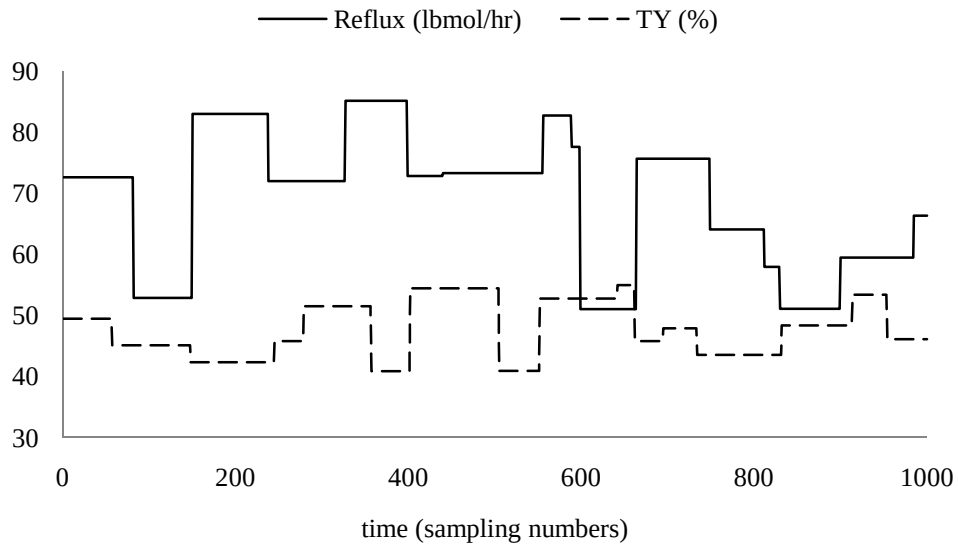


Figure 4.20. Reflux flowrate (solid line) and reboiler heat rate (dash line) Inputs to the distillation column

Two outputs, y_1 and y_2, are the overhead and bottom concentrations of methanol x_D and x_B, in mole faction. The first 1000 output samples are shown in Figure 4.21.

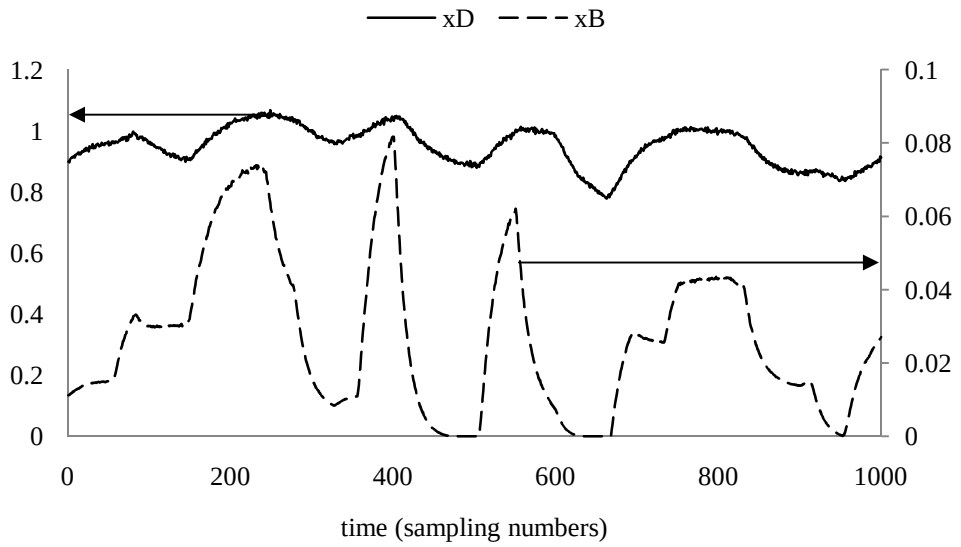


Figure 4.21. The x_D (solid line, left scale) and x_B (dash line, right scale) in distillation column experiments

4.4.2 Testing on Dynamic Order Determination

An example is presented at first to demonstrate the details in order determination. The example is based on the deterministic version of Model 2. The first 1000 data samples are shown in Figure 4.22 and a total 5000 data are generated and used for the order determination.

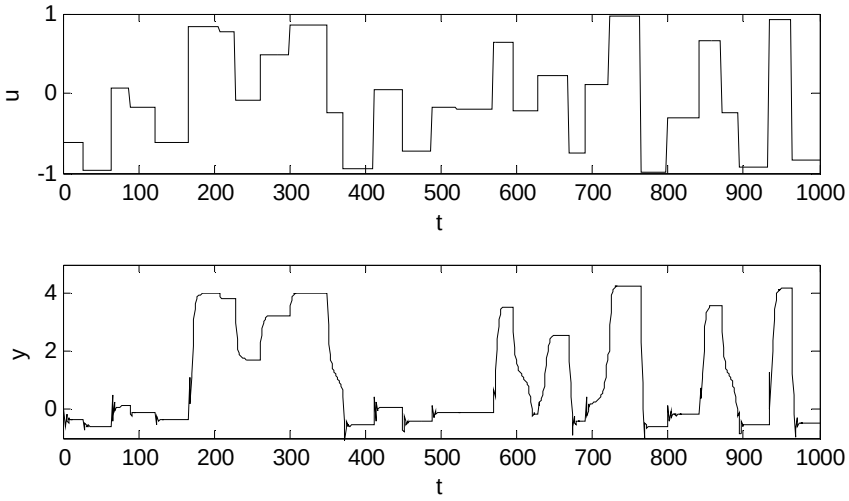


Figure 4.22. Data generated for the determinist version of model in Equation (4.42)

In using forward selection to select important regressors, the maximum *ny*, *nu* and

maximum *d* are set to 5, 4 and 1 respectively. The selection procedure is collected in

Table 4.4. Regressor forward selection for data in Figure 4.22

Step	*y*(*t*-1)	*y*(*t*-2)	*y*(*t*-3)	*y*(*t*-4)	*y*(*t*-5)	*u*(*t*-1)	*u*(*t*-2)	*u*(*t*-3)	*u*(*t*-4)	*u*(*t*-5)	random
1	**0.0516**	0.1248	0.2360	0.3572	0.4907	0.4246	0.3697	0.3243	0.3323	0.3661	3.348
2		0.0408	0.0390	0.0400	0.0429	**0.0131**	0.0371	0.0411	0.0510	0.0534	0.0551
3		**0.0045**	0.0060	0.0082	0.0146		2.1E+56	0.0117	0.0089	0.0114	0.0193
4			0.0057	0.0073	0.0055		**0.**0043	0.0093	0.**0035**	0.0059	0.0098
5			0.0068	0.**0042**	0.0062		0.0062	1.9E+10		0.0043	0.0105

In Table 4.4, there are 11 regressors including 10 time-lagged regressors and a random regressor, the last one. At the first run, all 11 regressors are tried one by one. Their corresponding FPEs are recorded in the first row. Among them, the one with the smallest FPE at 0.0516 is chosen, and the related regressor is *y*(*t*-1). In the next step, the selected regressor, *y*(*t*-1) is combined with the rest of 10 regressors. The results of 10 trials are in row 2, where the minimum FPE is due to *u*(*t*-1) at 0.0131. The blank for *y*(*t*-1) in row 2 only indicates that *y*(*t*-1) has been included. Continuing on this procedure, we then need have both *y*(*t*-1) and *u*(*t*-1) included and try their combinations with rest of 9 candidate regressors. The next minimum FPE is 0.0045 for *y*(*t*-2). Then *y*(*t*-2) is included. The next discovery is *u*(*t*-4) with FPE at 0.0035. At the fifth step, the minimum FPE is 0.042, which is however greater than the previous minimum FPE of 0.035. The increase in FPE signals to terminate the forward selection.

The above forward selection selects the four regressors [*y*(*t*-1) *y*(*t*-2) *u*(*t*-1) *u*(*t*-4)]. In theory, one could create an arbitrary model including these regressors. In practice, it is however unlikely to exclude *u*(*t*-2) and *u*(*t*-3) while having *u*(*t*-4) is included. In addition, the objective of this work is to determine dynamic orders, *ny* and *nu*. In order to include *u*(*t*-4), *nu* and *d* should be set to 3 and 1 respectively. This configuration however contains additional regressors *u*(*t*-2) and *u*(*t*-3), which are however rejected by the forward selection. In this work, a minor exhaustive search is conducted to compare different values for several values for *nu*, 0, 1, 2 and 3 with fixed *ny* at 2 and *d* at 1. The result is collected in Table 4.5, where the best value for nu is 0 with the minimum FPE of

0.0045. Therefore, the determined regressors are [$y(t\text{-}1)$ $y(t\text{-}2)$ $u(t\text{-}1)$] with ny=2, nu=0, d=1.

Table 4.5. Exhaustive search on *nu* with ny=2, d=1 for data in Figure 4.22

nu	0	1	2	3
FPE	**0.0045**	0.0275	0.0066	0.0378

The above order determination procedure by a forward selection followed by a minor exhaustive search uses SNNR rearranged data in recursive estimation. To reveal the impact of SNNR on order determination, the forward selection procedure is repeated for the same data set without using SNNR. The details of selection are collected in Table 4.6. The selected regressors are $y(t\text{-}1)$, $u(t\text{-}1)$ and $u(t\text{-}2)$. No minor exhaustive search is needed. Compared the model definition in Equation (4.2), the result misses $y(t\text{-}1)$ while find $u(t\text{-}2)$ that is not presented in the deterministic model. We simply state that the result include two 'mistakes'.

Table 4.6. Regressor forward selection for data in Figure 4.22 using time-sequence data

Step	$y(t\text{-}1)$	$y(t\text{-}2)$	$y(t\text{-}3)$	$y(t\text{-}4)$	$y(t\text{-}5)$	$u(t\text{-}1)$	$u(t\text{-}2)$	$u(t\text{-}3)$	$u(t\text{-}4)$	$u(t\text{-}5)$	random
1	**0.0526**	0.1260	0.2367	0.3611	0.5010	1.6116	1.5373	1.4841	1.4906	1.5271	5.7726
2		0.0556	0.0521	0.0533	0.0530	**0.0462**	0.0557	0.0634	0.0619	0.0588	0.0555
3		0.0515	0.0508	0.0506	0.0502		**0.0301**	0.0501	0.0493	0.0493	0.0484
4		0.0352	0.0325	0.0322	0.0317			0.0402	0.0401	0.0372	**0.0316**

The order determination method is also applied to other deterministic models, deterministic versions of models in Equations (4.40~4.44). The results are summarized in Table 4.7, which also include the results using original time-sequence data for comparison.

Table 4.7. Regressors determined for deterministic versions of Models 1-5

Model	Time Sequence	SNNR	Truth
1	$y(t\text{-}1)y(t\text{-}2)y(t\text{-}3)$	$y(t\text{-}1)y(t\text{-}2)y(t\text{-}3)u(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}2)u(t\text{-}1)$
2	$y(t\text{-}1)u(t\text{-}1)u(t\text{-}2)$	$y(t\text{-}1)\ y(t\text{-}2)\ u(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}2)u(t\text{-}1)$
3	$y(t\text{-}1)\ u(t\text{-}1)$	$y(t\text{-}1)u(t\text{-}1)$	$y(t\text{-}1)\ u(t\text{-}1)$
4	$u(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}2)y(t\text{-}3)u(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}2)y(t\text{-}3)u(t\text{-}1)u(t\text{-}2)$
5	$y(t\text{-}1)y(t\text{-}2)$	$y(t\text{-}1)u(t\text{-}1)$	$y(t\text{-}1)u(t\text{-}1)$

As observed in Table 4.7, both approaches are tied in the Model 3 test. In the Model 1 test, the 'Time Sequence' misses $u(t\text{-}1)$ but adds $y(t\text{-}3)$, making 2 mistakes, while the 'SNNR' adds $y(t\text{-}3)$, making 1 mistake. In the Model 2 test, the 'Time Sequence' misses $y(t\text{-}2)$ but adds $u(t\text{-}2)$, making 2 mistakes. The 'SNNR' makes 1 mistakes in the Model 4 test while 'Time Sequence' makes 4 mistakes by finding only $u(t\text{-}1)$. In the Model 5 test, 'Time Sequence' adds $y(t\text{-}2)$ but misses $u(t\text{-}1)$. For the first 5 tests, the 'SNNR has 2 mistakes while the 'Time Sequence' makes 10 mistakes. Illustrated by this comparison, neither approach is perfect, but the 'SNNR' outperforms the 'Time Sequence' in terms of number of mistakes made.

Tables 4.8 collects the comparison results using time-sequence and SNNR rearranged data for stochastic models in Equations (4.40~4.44) with example data shown in Figures 4.8~4.12.

Table 4.8. Regressors determined for Models 1-5

Model	Time Sequence	SNNR	Truth
1	$y(t\text{-}1)y(t\text{-}2)u(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}2)u(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}2)u(t\text{-}1)$
2	$y(t\text{-}1)y(t\text{-}2)y(t\text{-}3)u(t\text{-}1)$	$y(t\text{-}1)\ y(t\text{-}2)\ u(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}2)u(t\text{-}1)$
3	$y(t\text{-}1)\ u(t\text{-}1)$	$y(t\text{-}1)u(t\text{-}1)$	$y(t\text{-}1)\ u(t\text{-}1)$
4	$u(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}2)y(t\text{-}3)u(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}2)y(t\text{-}3)u(t\text{-}1)u(t\text{-}2)$
5	$y(t\text{-}1)u(t\text{-}1)$	$y(t\text{-}1)u(t\text{-}1)$	$y(t\text{-}1)u(t\text{-}1)$

Observed in Table 4.8, the 'SNNR' performs better with 1 mistake while 'Time

Sequence' makes 5 mistakes. It is also observed that only the result for the Model 1 is different in both Tables 4.7 and 4.8 for the 'SNNR' while the results for Models 1, 2, 3 and 5 are different in both tables for the 'Time Sequence'. It seems that the result due to 'SNNR' is less influenced by the additional noise than the 'Time Sequence'. It might be difficult to draw a general conclusion on the observation. Intuitively, the noise term will affect how model parameters vary, which in turn affects the performance of recursive estimation. Consequently, the order determination results, which are based on recursive estimation, should also be affected. On the other hand, the additional parameter variation after the noise being injected could be attenuated by the 'SNNR', which reduces the influence of noise on parameter variation then subsequently on order determination.

The details of regressor selection for Models 7-8 are shown in Tables 4.9 and 4.10, where an extra regressor $y(t\text{-}2)$ is found for each. It implies that the regressor $y(t\text{-}2)$ has influence on $y(t)$. In Equations (4.46) and (4.47), although $y(t)$ is not directly related to $y(t\text{-}2)$, the regressor $y(t\text{-}2)$ is able to affect $y(t)$ via $y(t\text{-}1)$. More importantly, Tables 4.9 and 4.10 show that the regressor $u(t\text{-}1)$ is found for both models.

Table 4.9. Regressor selection for Model 7

	$y(t\text{-}1)$	$y(t\text{-}1)u(t\text{-}1)$	$y(t\text{-}1)u(t\text{-}1)y(t\text{-}2)$	$y(t\text{-}1)u(t\text{-}1)y(t\text{-}2)y(t\text{-}4)$
FPE	0.0137	0.0028	0.0025	0.0032 (Stop)

Table 4.10. Regressor selection for Model 8

	$y(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}2)$	$y(t\text{-}1)y(t\text{-}2)u(t\text{-}1)$	$y(t\text{-}1)y(t\text{-}1)u(t\text{-}1)y(t\text{-}4)$
FPE	0.0787	0.0293	0.0188	0.0222 (Stop)

The proposed order determination is also compared to the geometric method .(Molina, Sampson, Fitzgerald & Niranjan, 1996) The testing is conducted on Models 10-13, and results are summarized in Table 4.11. As observed, the geometric method is able to extract correct orders for deterministic nonlinear AR models while performs poorly with the presence of additive noise. The geometric method makes a total of four mistakes for both Models 11 and 13. The proposed order determination makes

one mistake for Model 12.

Table 4.11. Regressors determined for Models 10-13

Model	SNNR	Geometric	Truth
10	$y(t\text{-}1)$	$y(t\text{-}1)$	$y(t\text{-}1)$
11	$y(t\text{-}1)$	$y(t\text{-}1)\ y(t\text{-}2)\ y(t\text{-}3)$	$y(t\text{-}1)$
12	$y(t\text{-}1)\ y(t\text{-}2)\ y(t\text{-}3)$	$y(t\text{-}1)\ y(t\text{-}2)$	$y(t\text{-}1)y(t\text{-}2)$
13	$y(t\text{-}1)\ y(t\text{-}2)$	$y(t\text{-}1)\ y(t\text{-}2)y(t\text{-}3)y(t\text{-}4)$	$y(t\text{-}1)y(t\text{-}2)$

The dynamic order determination is applied to the two-phase flow process with two possible input-output selections in Figures 4.18 and 4.19. The results are collected in Table 4.12.

Table 4.12. Regressors determined for the two-phase flow process

Input	Output	
Air flowrate	Pressure drop	$y(t\text{-}1)\ y(t\text{-}2)\ u(t\text{-}1)$
Air valve opening signal	Filtered pressure drop	$y_f(t\text{-}1)\ y_f(t\text{-}2)\ y_f(t\text{-}3)\ u_s(t\text{-}4)$

For the input and output defined in Figure 4.18, the recognized regressors are [$y(t\text{-}1)\ y(t\text{-}2)\ u(t\text{-}1)$]. The regressors determined for the input and ouput defined in Figure 4.19 include [$y_f(t\text{-}1)\ y_f(t\text{-}2)\ y_f(t\text{-}3)\ u_s(t\text{-}4)$]. Unlike the previous examples, it is not possible to justify the obtained results by 'true' dynamic orders for the two phase flow process, which are unknown. However, the difference expressed in results can be justified by our empirical knowledge regarding the process. The dynamic order in y_f in Figure 4.19 is one order higher than that in the output, y in Figure 4.18. The extra order in y_f is due to the first order filtering operation applied to the output, y. In two input channels, difference is in delay, which is consistent with the physical process. The signal u_s 'command to the valve' precedes the signal, u, air flowrate measurement. There are few steps between u_s and u. The signal, u_s is generated manually and recorded. It then is converted to a 3~15 psi pneumatic signal. The variation in the pneumatic signal changes the pressure on the diaphragm, which then pushes or releases the stem connected to the valve plug. The air

flowrate is then altered and measured. The measurement is u. The delay difference of 3 between u and u_s should be considered as an average difference over the entire data set. The exact difference might be different sample by sample.

Applying the proposed order determination to Model 15, the result obtained using the procedure extended in Section 4.3 and is summarized in Table 4.13 for both outputs.

Table 4.13. Results of order determination for the distillation column

Output y_1, distillate (x_D)		Output y_2, bottoms (x_B)	
FPE	Forward Selection	FPE	Forward Selection
2.11e-4	$y_1(t\text{-}1)$	7.02e-7	$y_2(t\text{-}1)$
3.83e-5	$y_1(t\text{-}2)$	1.52e-7	$y_2(t\text{-}4)$
3.55e-5	$u_1(t\text{-}3)$	1.17e-7	$u_2(t\text{-}1)$
3.52e-5	$y_1(t\text{-}3)$	1.13e-7	$u_1(t\text{-}3)$
3.43e-5	$y_2(t\text{-}3)$	1.13e-7	$u_2(t\text{-}3)$
3.62e-5	$y_2(t\text{-}4)$		

The selected regressors for $y_1(t)$ are $[y_1(t\text{-}1)\ y_1(t\text{-}2)\ y_1(t\text{-}3)\ y_2(t\text{-}3)\ u_1(t\text{-}3)]$ and the selected regressors for $y_2(t)$ are $[y_2(t\text{-}1)\ y_2(t\text{-}4)\ u_1(t\text{-}3)\ u_2(t\text{-}1)]$. For the output y_2, the value of ny needs to be 4 if $y_2(t\text{-}4)$ is included. It would also include both $y_2(t\text{-}2)$ and $y_2(t\text{-}3)$. Therefore, a minor exhaustive search is needed to compare several different values of ny and the result is summarized in Table 4.14.

Table 4.14. Exhaustive search on ny for y_2

ny	1	2	3	4
FPE	1.25E-07	1.17E-07	1.23E-07	1.28E-07

Then the regressors determined for output y_2 is $[y_2(t\text{-}1)\ y_2(t\text{-}2)\ u_1(t\text{-}3)\ u_2(t\text{-}1)]$.

4.4.3 Testing on Nonlinear Component Detection

The nonlinear component detection will be based on the results in the above order determination. The implementation detail of nonlinear component detection is given for the Model 1 in Equation (4.40) with selected regressors [$y(t\text{-}1)$ $y(t\text{-}2)$ $u(t\text{-}1)$]. The result is recorded in Table 4.15, wherein the numbers for the row "Subsets" represent the combination of the $1^{st}(y(t\text{-}1))$, $2^{nd}(y(t\text{-}2))$, and $3^{rd}(u(t\text{-}1))$ regressors.

Table 4.15. Exhaustive search for nonlinear components for Model 1

Subsets	1	2	3	1&2	1&3	2&3	1&2&3
MSE	0.3328	0.3366	0.2747	0.3373	0.2987	0.3006	0.3195

In the first trial, the entry for Subset "1", the SNNR procedure is conducted based on $y(t\text{-}1)$. The resultant data is then used in a recursive estimation that results in a MSE of 0.3328. The trial continues until all combinations of regressors are exhausted. The minimum MSE is 0.2747, which corresponds to the third regressor, $u(t\text{-}1)$. According to the result, the time-varying model could be described as below using the detected nonlinear component $u(t\text{-}1)$.

$$y(t) = a_1(u(t-1))\,y(t-1) + a_1(u(t-1))\,y(t-2) + b_0(u(t-1))\,u(t-1) + e(t)$$

The results of nonlinear component detection for the first five models in Equations (4.40~4.44) are summarized in Table 4.16.

Table 4.16. Results for nonlinear component detection for Models 1~5

Model	Detected nonlinear components
1	$u(t\text{-}1)$
2	$y(t\text{-}1)$, $y(t\text{-}2)$
3	$y(t\text{-}1)$, $u(t\text{-}1)$
4	$y(t\text{-}2)$
5	$u(t\text{-}1)$

Observed from Table 4.16, it seems that the difference between detected and desired nonlinear components is clear in the Model 4 test. It seems that in Equation (4.43), every regressor is nonlinear. However, only $y(t\text{-}2)$ is reported to be a nonlinear component while others are perhaps ignored. However, the result should not be interpreted that only $y(t\text{-}2)$ is nonlinearly expressed in the Model. Since we are only reporting the minimum MSE as shown in Table 4.16, the results include only the 'dominant' nonlinear components.

Table 4.17 shows the details for nonlinear component detection for Model 4. The last row in Table 4.17 is the MSE on the raw data without SNNR operation. The first observation is that the minimum MSE is due to the regressor $y(t\text{-}2)$, which is the reported nonlinear component in Table 4.16. On the other hand, it is observed in the last row that the MSE without SNNR is the maximum, which implies that every regressor has impact on parameter variation. It then indicates that every regressor is nonlinearly expressed in the model. However, the regressor, $y(t\text{-}2)$ seems to dominate others in this test.

Table 4.17. Exhaustive search for nonlinear components for Model 4

Regressors	MSE		Regressors	MSE
$y(t\text{-}1)$	0.004021		$y(t\text{-}2)u(t\text{-}1)$	0.003668
$y(t\text{-}2)$	0.003241		$y(t\text{-}3)u(t\text{-}1)$	0.00343
$y(t\text{-}3)$	0.003579		$y(t\text{-}1)y(t\text{-}2)y(t\text{-}3)$	0.00349
$u(t\text{-}1)$	0.005035		$y(t\text{-}1)y(t\text{-}2)u(t\text{-}1)$	0.003711
$y(t\text{-}1)y(t\text{-}2)$	0.003353		$y(t\text{-}1)y(t\text{-}3)u(t\text{-}1)$	0.003571
$y(t\text{-}1)y(t\text{-}3)$	0.003312		$y(t\text{-}2)y(t\text{-}3)u(t\text{-}1)$	0.003453
$y(t\text{-}1)u(t\text{-}1)$	0.004086		$y(t\text{-}1)y(t\text{-}2)y(t\text{-}3)u(t\text{-}1)$	0.003509
$y(t\text{-}2)y(t\text{-}3)$	0.003365		No SNNR	0.005578

The details of nonlinear component detection for Model 9 with regressors $1^{st}(y(t\text{-}1))$, $2^{nd}(y(t\text{-}2))$, and $3^{rd}(u(t\text{-}1))$ are collected in Table 4.18.

Table 4.18. Details of nonlinear components detection for Model 9

Subsets	1	2	3	1&2	1&3	2&3	1&2&3
MSE (10^{-3})	0.0144	0.0143	0.6442	0.1266	0.2186	0.2007	0.3159

In Table 4.18, the minimum MSE, 0.0143 corresponds the regressor $y(t\text{-}2)$. Interestingly, the corresponding MSE for regressor $y(t\text{-}1)$ is 0.0144 and very close to that due to $y(t\text{-}2)$. It would be fair to conclude that both regressors are equally good, which is consistent with model structure in Equation (4.49)

The nonlinear component detection results for the two-phase process are collected in Table 4.19 for two different choices of input and output channels. The results are reasonable and both include lagged input and output signals.

Table 4.19. Nonlinear components detected for the two phase flow process

Input	Output	Nonlinear components
Air flowrate	Pressure drop	$y(t\text{-}2)$ $u(t\text{-}1)$
Air valve opening signal	Filtered pressure drop	$y_f(t\text{-}1)$ $y_f(t\text{-}2)$ $u_s(t\text{-}4)$

The nonlinear component results for the distillation column test are listed in Table 4.20. For each output, competing choices are listed in terms of dimension and error. For y_1, the minimum MSE = 3.153e-5 is to have $[y_1(t\text{-}2)\ y_2(t\text{-}3)]$ as nonlinear components. The next minimum MSE is 3.155e-5 that has only one nonlinear component, $y_2(t\text{-}3)$. The second choice features a low dimension while the first one has a lower MSE. For y_2, two competing choices are listed. The minimum MSE = 9.10e-8 corresponds to the selection of $[y_2(t\text{-}1)\ u_2(t\text{-}1)]$ as nonlinear components. The next minimum MSE is 1.06e-7, which has only one nonlinear component $[u_2(t\text{-}1)]$ included. All listed choices for nonlinear components will be further investigated and tried in creating GTSK models.

Table 4.20. Choices of nonlinear components for the distillation column

Output y_1, distillate (x_D)		Output y_2, bottoms (x_B)	
MSE		MSE	
3.15e-5	$[y_1(t\text{-}2)\ y_2(t\text{-}3)]$	9.10e-8	$[y_2(t\text{-}1)\ u_2(t\text{-}1)]$
3.16e-5	$y_2(t\text{-}3)$	1.06e-7	$[u_2(t\text{-}1)]$

CHAPTER V

PARAMETER ESTIMATION FOR GTSK MODELS

In this chapter, a two-stage approach is described to estimate model parameter values for the GTSK model described in Chapter 3 with selected antecedent and consequent variables in Chapter 4. Model parameters include both antecedent parameters, centorid (**o**), shape matrix (**P**) and coefficients for local linear relations (**θ**) for each rule. A brief summary of all parameters could be found in Equation (3.14). In Chapter 5, a constrained optimization problem with matrix inequalities is defined to estimate model parameter values, which are initialized by a proposed heuristic approach. The following elaboration focuses on a SISO model. The extension to MIMO models will be provided at the end of each section if necessary.

5.1 Parameter Estimation by Newton's Method

5.1.1 A Constrained Optimization Problem

Estimation of model parameter values is generally treated in an optimization scheme by minimizing a performance index defined over a data set. The entire data set is collectively denoted by [**y C X**] as output, antecedent and consequent variables. In detail the denotation is described by

$$[\mathbf{y}\,|\,\mathbf{C}\,|\,\mathbf{X}] \equiv \begin{bmatrix} y(1) & c_1(1) & \text{L} & c_{nc}(1) & x_0(1) & \text{L} & x_{nx}(1) \\ \text{M} & \text{M} & \text{O} & \text{M} & \text{M} & \text{O} & \text{M} \\ y(N) & c_1(N) & & c_{nc}(N) & x_0(N) & & x_{nx}(N) \end{bmatrix} \tag{5.1}$$

where x_0 is the constant regressor in Equation (3.13). Variables, $x_1 \sim x_{\text{nx}}$ are regressors due to the determined dynamic orders *ny, nu* and pure delay, *d* in Chapter 4. Variables, $c_1 \sim c_{\text{nc}}$ are antecedent variables as the nonlinear components determined also in Chapter 4.

The following optimization problem is then defined given the number of rules, M is known.

$$\begin{aligned} &\underset{\mathbf{o}^i,\mathbf{P}^i,\theta^i}{\text{minimize}}\ J=\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)^2 \\ &\text{subject to} \\ &\qquad \mathbf{P}^i>\mathbf{0}, \qquad i=1,\mathrm{L}\ ,M \end{aligned} \tag{5.2}$$

where, the computation of $\hat{y}$ is described in Equation (3.22). Inequality constraints signify that all shape matrices $\mathbf{P}^i$ are positive definite. The following matrix function with respect to $\mathbf{P}^i$ is used to convert the constrained optimization to an unconstrained one (Boyd, Balakrishnan, Feron & Ghaoui, 1994)

$$\phi\left(\mathbf{P}^i\right)=\begin{cases}\log\det\left(\left(\mathbf{P}^i\right)^{-1}\right) & \mathbf{P}^i>\mathbf{0} \\ \infty & \text{else.}\end{cases} \tag{5.3}$$

Then, the augmented objective performance index is defined by

$$J_{\text{aug}}(s)=sJ+\sum_{i=1}^{M}\phi\left(\mathbf{P}^i\right) \tag{5.4}$$

where the scalar s is used to adjust the relative importance of J with respect to the sum of $\phi\left(\mathbf{P}^i\right)$. The treatment of matrix inequality is borrowed from the interior-point method to solve a convex linear matrix inequality optimization problem (Boyd & Vandenberghe, 2004), although the optimization problem in Equation (5.2) is not convex.

The first and second-order derivatives of $J_{\text{aug}}(s)$ to model parameters consist of those from the performance index and the penalty function. The derivatives due to the penalty function is described as below aided by the parameterization of $\mathbf{P}^i$ in Equation (3.19)

$$\frac{\partial \phi\left(\mathbf{P}^i\right)}{\partial p_j^i} = -\mathrm{Tr}\left(\left(\mathbf{P}^i\right)^{-1}\mathbf{B}_j\right) \qquad (5.5)$$
$$i = 1,\mathrm{L}\ ,M; \qquad j = 1,\mathrm{L}\ ,nc\left(nc+1\right)/2$$

$$\frac{\partial \phi\left(\mathbf{P}^i\right)}{\partial p_j^i \partial p_b^i} = \mathrm{Tr}\left(\left(\mathbf{P}^i\right)^{-1}\mathbf{B}_j\left(\mathbf{P}^i\right)^{-1}\mathbf{B}_b\right) \qquad (5.6)$$
$$i = 1,\mathrm{L}\ ,M; \qquad j,b = 1,\mathrm{L}\ ,nc\left(nc+1\right)/2$$

where **Tr** is the trace of a matrix. Clearly, $\phi\left(\mathbf{P}^i\right)$ is independent of centroid, $\mathbf{o}^i$ and local model parameters, $\boldsymbol{\theta}^i$.

The first-order derivatives of *J* to model parameters are described by

$$\frac{\partial J}{\partial \boldsymbol{\theta}^i} = -2\sum_{t=1}^{N}\left(y\left(t\right)-\hat{y}\left(t\right)\right)w^i\left(t\right)\mathbf{x}\left(t\right), \quad i = 1,\mathrm{L}\ ,M \qquad (5.7)$$

$$\frac{\partial J}{\partial \mathbf{o}^i} = -4\sum_{t=1}^{N}\left(y\left(t\right)-\hat{y}\left(t\right)\right)\left(\hat{y}^i\left(t\right)-\hat{y}\left(t\right)\right)w^i\left(t\right)\mathbf{P}^i\left(\mathbf{c}\left(t\right)-\mathbf{o}^i\right), \qquad (5.8)$$
$$i = 1,\mathrm{L}\ ,M$$

$$\frac{\partial J}{\partial p_j^i} = 2\sum_{t=1}^{N}\left(y\left(t\right)-\hat{y}\left(t\right)\right)\left(\hat{y}^i\left(t\right)-\hat{y}\left(t\right)\right)w^i\left(t\right)\left(\mathbf{c}\left(t\right)-\mathbf{o}^i\right)^T\mathbf{B}_j\left(\mathbf{c}\left(t\right)-\mathbf{o}^i\right) \qquad (5.9)$$
$$i = 1,\mathrm{L}\ ,M; \qquad j = 1,\mathrm{L}\ ,nc\left(nc+1\right)/2$$

The gradient vector is then described by

$$\mathbf{g} = \begin{bmatrix} \mathbf{g}^1 \\ \mathrm{M} \\ \mathbf{g}^M \end{bmatrix}, \qquad \text{with} \quad \mathbf{g}^i = \begin{bmatrix} \dfrac{\partial J_{\mathrm{aug}}}{\partial \mathbf{o}^i} & \dfrac{\partial J_{\mathrm{aug}}}{\partial \mathbf{P}^i} & \dfrac{\partial J_{\mathrm{aug}}}{\partial \boldsymbol{\theta}^i} \end{bmatrix} \qquad (5.10)$$

where $\dfrac{\partial J_{\mathrm{aug}}}{\partial \mathbf{o}^i} = s\dfrac{\partial J}{\partial \mathbf{o}^i}$; $\dfrac{\partial J_{\mathrm{aug}}}{\partial p_j^i} = s\dfrac{\partial J}{\partial p_j^i} - \mathrm{Tr}\left(\left(\mathbf{P}^i\right)^{-1}\mathbf{B}_j\right)$; $\dfrac{\partial J_{\mathrm{aug}}}{\partial \boldsymbol{\theta}^i} = s\dfrac{\partial J}{\partial \boldsymbol{\theta}^i}$

The Hessian matrix is then defined by

$$\mathbf{H}=\begin{bmatrix} \frac{\partial^2 J_{\text{aug}}}{\partial \mathbf{o}^2} & \frac{\partial^2 J_{\text{aug}}}{\partial \mathbf{o}\partial \mathbf{P}} & \frac{\partial^2 J_{\text{aug}}}{\partial \mathbf{o}\partial \boldsymbol{\theta}} \\ \frac{\partial^2 J_{\text{aug}}}{\partial \mathbf{P}\partial \mathbf{o}} & \frac{\partial^2 J_{\text{aug}}}{\partial \mathbf{P}^2} & \frac{\partial^2 J_{\text{aug}}}{\partial \mathbf{P}\partial \boldsymbol{\theta}} \\ \frac{\partial^2 J_{\text{aug}}}{\partial \boldsymbol{\theta}\partial \mathbf{o}} & \frac{\partial^2 J_{\text{aug}}}{\partial \boldsymbol{\theta}\partial \mathbf{P}} & \frac{\partial^2 J_{\text{aug}}}{\partial^2 \boldsymbol{\theta}} \end{bmatrix} \tag{5.11}$$

The element-wise calculation of the second-order derivatives of J to model parameters are described by

$$\begin{aligned} \frac{\partial^2 J}{\partial p_j^i \partial p_b^a} &= -2\sum_{t=1}^{N}\frac{\partial \hat{y}(t)}{\partial p_b^a}\left(\hat{y}^i(t)-\hat{y}(t)\right)w^i(t)\left(\mathbf{c}(t)-\mathbf{o}^i\right)^T\mathbf{B}_j\left(\mathbf{c}(t)-\mathbf{o}^i\right) \\ &\quad -2\sum_{t=1}^{N}\frac{\partial \hat{y}(t)}{\partial p_b^a}\left(y(t)-\hat{y}(t)\right)w^i(t)\left(\mathbf{c}(t)-\mathbf{o}^i\right)^T\mathbf{B}_j\left(\mathbf{c}(t)-\mathbf{o}^i\right) \\ &\quad +2\sum_{k=1}^{N}\left(y(t)-\hat{y}(t)\right)\left(\hat{y}^i(t)-\hat{y}(t)\right)\frac{\partial w^i(t)}{\partial p_b^a}\left(\mathbf{c}(t)-\mathbf{o}^i\right)^T\mathbf{B}_j\left(\mathbf{c}(t)-\mathbf{o}^i\right) \end{aligned} \tag{5.12}$$

$$\begin{aligned} \frac{\partial^2 J}{\partial p_b^a \partial \mathbf{o}^i} &= \frac{\partial^2 J}{\partial \mathbf{o}^i \partial p_b^a} \\ &= 4\sum_{t=1}^{N}\frac{\partial \hat{y}(t)}{\partial p_b^a}\left(\hat{y}^i(t)-\hat{y}(t)\right)w^i(t)\mathbf{P}^i\left(\mathbf{c}(t)-\mathbf{o}^i\right) \\ &\quad +4\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)\frac{\partial \hat{y}(t)}{\partial p_b^a}w^i(t)\mathbf{P}^i\left(\mathbf{c}(t)-\mathbf{o}^i\right) \\ &\quad -4\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)\left(\hat{y}^i(t)-\hat{y}(t)\right)\frac{\partial w^i(t)}{\partial p_b^a}\mathbf{P}^i\left(\mathbf{c}(t)-\mathbf{o}^i\right) \\ &\quad -4\delta(i-a)\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)\left(\hat{y}^i(t)-\hat{y}(t)\right)w^i(t)\mathbf{B}_b\left(\mathbf{c}(t)-\mathbf{o}^i\right) \end{aligned} \tag{5.13}$$

$$\begin{aligned} \frac{\partial^2 J}{\partial p_b^a \partial \boldsymbol{\theta}^i} &= \frac{\partial^2 J}{\partial p_b^a \partial \boldsymbol{\theta}^i} \\ &= 2\sum_{t=1}^{N}\frac{\partial \hat{y}(t)}{\partial p_b^a}w^i(t)\mathbf{x}(t)-2\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)\frac{\partial w^i(t)}{\partial p_b^a}\mathbf{x}(t) \end{aligned} \tag{5.14}$$

$$\begin{aligned}\frac{\partial^2 J}{\partial \mathbf{o}^i \partial \mathbf{o}^a} &= 4\sum_{t=1}^{N}\left(\hat{y}^i(t)-\hat{y}(t)\right)w^i(t)\mathbf{P}^i\left(\mathbf{c}(t)-\mathbf{o}^i\right)\left(\frac{\partial \hat{y}(t)}{\partial \mathbf{o}^a}\right)^T \\ &+4\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)w^i(t)\mathbf{P}^i\left(\mathbf{c}(t)-\mathbf{o}_i\right)\left(\frac{\partial \hat{y}(t)}{\partial \mathbf{o}^a}\right)^T \\ &-4\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)\left(\hat{y}^i(t)-\hat{y}(t)\right)\mathbf{P}^i\left(\mathbf{c}(t)-\mathbf{o}_i\right)\left(\frac{\partial w^i(t)}{\partial \mathbf{o}^a}\right)^T \\ &+4\delta(i-a)\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)\left(\hat{y}^i(t)-\hat{y}(t)\right)w^i(t)\mathbf{P}^i\end{aligned} \tag{5.15}$$

$$\frac{\partial^2 J}{\partial \mathbf{o}^a \partial \boldsymbol{\theta}^i} = 2\sum_{t=1}^{N} w^i(t)\mathbf{x}(t)\left(\frac{\partial \hat{y}(t)}{\partial \mathbf{o}^a}\right)^T - 2\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)\mathbf{x}(t)\left(\frac{\partial w^i(t)}{\partial \mathbf{o}^a}\right)^T \tag{5.16}$$

$$\frac{\partial^2 J}{\partial \boldsymbol{\theta}^i \partial \boldsymbol{\theta}^a} = 2\sum_{t=1}^{N} w^i(t)\mathbf{x}(t)\mathbf{x}(t)^T w^a(t) \tag{5.17}$$

with

$$\frac{\partial w^i(t)}{\partial p_b^a} = -\frac{\partial TA^a(t)}{\partial p_b^a}\frac{w^i(t)}{\sum_{i=1}^{r} TA^i(t)} \qquad \frac{\partial w^i(t)}{\partial \mathbf{o}^a} = -\frac{\partial TA^a(t)}{\partial \mathbf{o}^a}\frac{w^i(t)}{\sum_{i=1}^{r} TA^i(t)} \tag{5.18}$$

$$\frac{\partial \hat{y}(t)}{\partial p_b^a} = \frac{\partial TA^a(t)}{\partial p_b^a}\frac{\hat{y}^i(t)-\hat{y}(t)}{\sum_{i=1}^{r} TA^i(t)} \qquad \frac{\partial \hat{y}(t)}{\partial \mathbf{o}^a} = \frac{\partial TA^a(t)}{\partial \mathbf{o}^a}\frac{\hat{y}^i(t)-\hat{y}(t)}{\sum_{i=1}^{r} TA^i(t)} \tag{5.19}$$

$$\frac{\partial TA^a(t)}{\partial p_b^a} = -TA^a(t)\left(\mathbf{c}(t)-\mathbf{o}^a\right)^T \mathbf{B}_b\left(\mathbf{c}(t)-\mathbf{o}^a\right) \tag{5.20}$$

$$\frac{\partial TA^a(t)}{\partial \mathbf{o}^a} = 2TA^a(t)\mathbf{P}^a\left(\mathbf{c}(t)-\mathbf{o}^a\right) \tag{5.21}$$

With the above computed gradient vector and Hessian matrix. Newton's method will be applied to optimize the model parameters by solving a sequential of quadratic optimization problems. The solving procedure is summarized in the Algorithm 5.1

Algorithm 5.1

0. Algorithm configuration: s=10, μ=2,ε=1e-3

1. *Initial guess:* $\mathbf{v}(0)= [\mathbf{o}(0), \mathbf{P}(0), \boldsymbol{\theta}(0)]$

Repeat

2. Newton method

Repeat

2.1. Evaluate $\boldsymbol{g}$ *and* $\boldsymbol{H}$

2.2. Compute the search direction: ($\Delta\mathbf{v} = \boldsymbol{H}^{-1}\boldsymbol{g}$)

2.3. Linear search for λ *and update:* $\mathbf{v}=\mathbf{v}+\lambda\Delta\mathbf{v}$

2.4 Stop if $|| \lambda\Delta\mathbf{v} ||_2 < \varepsilon$

3. Increase s: $s=\mu s$;

4. Stop if $m/s<\varepsilon$

The algorithm involves two loops. The inner loop uses Newton's method to solve an unconstrained optimization problem with a given *s*. The scalar *s* is increased by μ in the outer loop to make the performance index *J* more important. The algorithm stops when *s* is sufficiently large. The scalar *m* in the outer loop stopping criterion is the number of model parameters. In the convex optimization (Boyd & Vandenberghe, 2004), it is shown that *m*/*s* quantifies the quality of a suboptimal solution and defined as the upper bound of the difference between the true optimal function value and the actual solution.

5.1.2 Interpretation of Local Optimal Solutions

Based on the Algorithm 5.1, the scalar *s* becomes sufficiently large at the end, which lets performance index dominate the penalty term. It is then possible to derive solutions at equilibrium conditions by considering only performance index. By letting Equation (5.7) equal to zero, we then have

$$-2\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)w^{i}(t)\mathbf{x}(t)=0, \qquad i=1,\mathrm{L}\ ,M \tag{5.22}$$

A possibility is to let $w^i(t)$ be zero for all *t*. The trivial solution could be reached if $\mathbf{o}^i$ is set to be sufficiently far from all $\mathbf{c}(t)$, which as shown in Equations (3.7) and (3.23)

will make $TA^i(t)$ and $w^i(t)$ very small. Otherwise, the equilibrium condition will be satisfied in a complex way. With $\hat{y}(t)$ replaced by Equation (3.22), the equilibrium condition becomes

$$-2\sum_{t=1}^{N}\left(y(t)-\sum_{j=1}^{M}w^j(t)\hat{y}^j(t)\right)w^i(t)\mathbf{x}(t)=0 \tag{5.23}$$

which could be simplified if the following assumption holds

$$\begin{aligned} w^i(t)w^j(t)&\approx 0, && j\neq i\\ w^i(t)w^j(t)&\approx w^i(t), && j=i \end{aligned} \tag{5.24}$$

the assumption makes the cross product of weights in different rules negligible and let

$$w^i(t)\sum_{j=1}^{M}w^j(t)\hat{y}^j(t)\approx w^i(t)\hat{y}^i(t)$$

Roughly speaking, the assumption is satisfied if rules in a GTSK model are relatively independent with $w^i(t)\approx 0$ or $w^i(t)\approx 1$. With the assumption, the equilibrium condition is simplified to

$$-2\sum_{t=1}^{N}\left(y(t)-\hat{y}^i(t)\right)w^i(t)\mathbf{x}(t)\approx 0 \tag{5.25}$$

The approximated equilibrium condition could be interpreted as a result of solving the following weighted least square

$$J^i=\sum_{t=1}^{N}\left(y(t)-\hat{y}^i(t)\right)^2 w^i(t) \tag{5.26}$$

The equilibrium condition for centroids is achieved by letting Equation (5.8) equal to zero

$$-4\sum_{t=1}^{N}\left(y(t)-\hat{y}(t)\right)\left(\hat{y}^i(t)-\hat{y}(t)\right)w^i(t)\mathbf{P}^i\left(\mathbf{c}(t)-\mathbf{o}^i\right)=0 \tag{5.27}$$

Clearly, the trivial solution with all zero $w^i(t)$ due to a distant $\mathbf{o}^i$ is able to satisfy the equilibrium condition. The trivial solution is however undesired. Another possibility is to let the product of $(\hat{y}^i(t) - \hat{y}(t))w^i(t)$ equal to zero everywhere, which will be approximately satisfied if the assumption in Equation (5.24) is made again. The product is about zero if $w^i(t) \approx 0$. Otherwise, the expression of $(\hat{y}^i(t) - \hat{y}(t))$ is about zero if $w^i(t) \approx 1$. Therefore, if rules in a GTSK model are relatively independent to each other, the equilibrium condition for the centroid is approximately satisfied.

Similarly, the equilibrium condition on the shape matrix parameters could also be approximately satisfied if rules are assumed relatively independent.

5.1.3 Random Parameter Initialization

An important factor affecting a nonlinear optimization is the initial guesses of decision variable values. Often times, initial guesses are randomly set. However, for the proposed GTSK model, random initialization might result in trivial or even infeasible solutions. Algorithm 5.1 requires feasible initial guesses. Since there are only constraints on $\mathbf{P}^i$, users might initialize $\mathbf{P}^i$ as identity matrix and randomize $\mathbf{o}^i$ and $\boldsymbol{\theta}^i$ to avoid infeasible initializations.

Care needs also to be taken to initialize the centroid, $\mathbf{o}^i$ especially for higher dimensional antecedents in order to prevent trivial $w^i(t)$ (all $w^i(t)$ are close to zero). As discussed in Section 5.1.2, trivial $w^i(t)$ will immediately satisfy the equilibrium conditions for both antecedent and consequent parameters. An illustration is shown in Figure 5.1 with a collection of antecedent samples as [$y(t\text{-}3)$ $u(t\text{-}9)$]. $y(t\text{-}3)$ is between 50 and 160 while $u(t\text{-}9)$ is between 20 and 100. Define an area by [$100<y(t\text{-}3)<160$, $50<u(t\text{-}9)<100$] as shown as the dashed box in Figure 5.1. Anywhere in that box is claimed to be distant from all observed samples. The box covers about 34% of the entire antecedent space. Therefore, there is about 34% likelihood to generate a trivial random centroid. Even if nontrivial centroids are initialized, the optimization is still subject to local optimal

solutions. Many random trials are needed to increase the probability of obtaining a global solution. In (Iyer & Rhinehart, 1999), statistical analysis is provided to estimate number of random trials given the probability of convergence region for a global optimal solution.

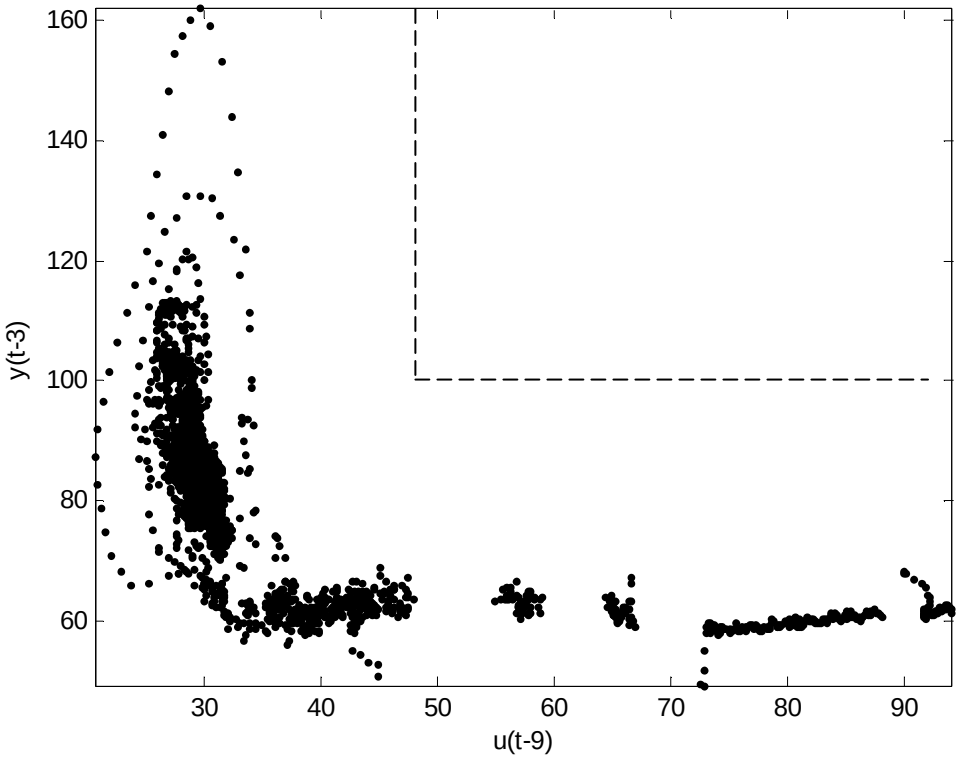


Figure 5.1. Antecedent space defined by antecedent variables $u(t\text{-}9)$ and $y(t\text{-}3)$

Alternatively, centroids might be randomly drawn from observed samples. This approach guarantees that every centroid is at least significantly expressed once. However, care has to be taken to make sure that drawn random centroids spread wide enough in order to cover the entire antecedent space effectively. Otherwise, it is possible that all drawn centroids are too concentrated. It could happen when distribution of data samples are significantly uneven over antecedent space. In Figure 5.1, there are 5000 points, 90% of them are distributed in the right-bottom corner. The rest of points are scattered in the both tails, assuming 200 and 300 points at both tails respectively. The likelihood of drawing one from the right-bottom corner is 90%. If the desired centroid distribution is to

have at least one centroid in each portion of data in Figure 5.1 (the top tail, the bottom-right corner, the bottom tail), the likelihood is L If N random centroids are drawn.

$$L = 1 - 0.9^N - 0.04^N - 0.06^N - \sum_{k=1}^{N-1} 0.9^{N-k} 0.04^k - \sum_{k=1}^{N-1} 0.9^{N-k} 0.06^k - \sum_{k=1}^{N-1} 0.04^{N-k} 0.06^k \tag{5.28}$$

Figure 5.2 shows the evaluation of the L in Equation (5.28) with respect to N. It indicates that least 45 random centroids need to be generated to assure the above mentioned initialization requirements with 0.99 likelihood. N will be increased if more centroids are required for the sparser areas, which is often necessary for data-rich-but-information-poor chemical processes, where a large amount of data is recorded at steady state operation. For describing a chemical process around a steady state condition, one linear model will be sufficient. Nonlinearity is observed during transition between operating conditions, which however only generate a limit amount of data although more rules are needed to describe nonlinear behavior. In practice, it is hard to tell if N is sufficiently large, one might have to try to find the right value.

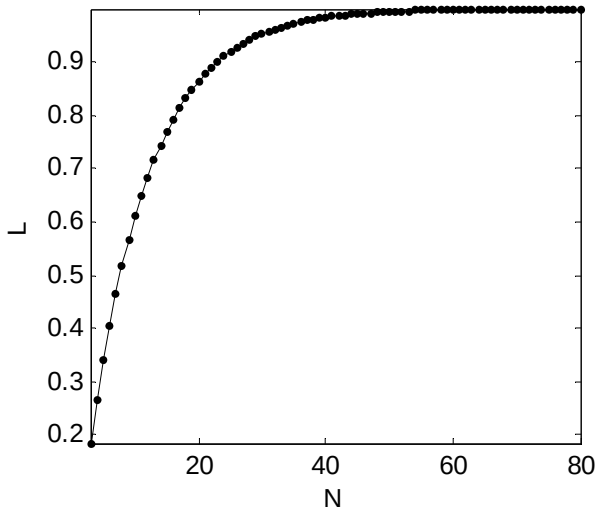


Figure 5.2. Evaluation of function in Equation (5.28)

5.2 Parameter Estimation for MIMO GTSK Models

Readers might choose to skip this section now and interested readers could come back to revisit this section when dealing with models.

The following constrained quadratic performance index is used for a MIMO GTSK model

$$\begin{aligned} &\underset{\mathbf{o}^i,\mathbf{P}^i,\boldsymbol{\theta}^i(1\le i\le M)}{\text{minimize}} \; J=\sum_{t=1}^{N}\left(\mathbf{y}(t)-\hat{\mathbf{y}}(t)\right)^T \mathbf{Q}\left(\mathbf{y}(t)-\hat{\mathbf{y}}(t)\right) \\ &\text{subject to} \\ &\quad \mathbf{P}^i>0 \end{aligned} \tag{5.29}$$

where, the weighting matrix Q is a positive definite diagonal matrix used to reflect the relative importance of each output or to make them comparable by adjusting scales

$$\mathbf{Q}=\begin{bmatrix} q_1 & \text{L} & 0 \\ \text{M} & \text{O} & \text{M} \\ 0 & \text{L} & q_n \end{bmatrix} \tag{5.30}$$

$\mathbf{Q}(i,j)$ is set to zero ($i\neq j$). Otherwise, y_i and y_j are coupled. Since $\mathbf{Q}$ is only a diagonal matrix, the performance index J can be decomposed as below

$$J=\sum_{k=1}^{n} J_k \tag{5.31}$$

with $J_k=q_k\sum_{t=1}^{N}\left(y_k(t)-\hat{y}_k(t)\right)^2$

The derivatives of J with respect to model parameters are then defined by

$$\frac{\partial J}{\partial \mathbf{o}}=\sum_{k=1}^{n}\frac{\partial J_k}{\partial \mathbf{o}};\quad \frac{\partial J}{\partial \mathbf{P}}=\sum_{k=1}^{n}\frac{\partial J_k}{\partial \mathbf{P}};\quad \frac{\partial J}{\partial_k \boldsymbol{\theta}}=\sum_{k=1}^{n}\frac{\partial J_k}{\partial_k \boldsymbol{\theta}} \tag{5.32}$$

where the derivative to ${}_k\boldsymbol{\theta}$ could be further simplified due to its local influence on J_k only

$$\frac{\partial J}{\partial_k \boldsymbol{\theta}} = \frac{\partial J_k}{\partial_k \boldsymbol{\theta}} \tag{5.33}$$

The second order derivatives are defined by

$$\begin{aligned}
&\frac{\partial^2 J}{\partial \mathbf{o}^2} = \sum_{k=1}^{n} \frac{\partial^2 J_k}{\partial \mathbf{o}^2}; \qquad \frac{\partial^2 J}{\partial \mathbf{o} \partial \mathbf{P}} = \sum_{k=1}^{n} \frac{\partial^2 J_k}{\partial \mathbf{o} \partial \mathbf{P}}; \qquad \frac{\partial^2 J}{\partial \mathbf{o} \partial_k \boldsymbol{\theta}} = \frac{\partial^2 J_k}{\partial \mathbf{o} \partial_k \boldsymbol{\theta}} \\
&\frac{\partial^2 J}{\partial \mathbf{P}^2} = \sum_{k=1}^{n} \frac{\partial^2 J_k}{\partial \mathbf{P}^2}; \qquad \frac{\partial^2 J}{\partial \mathbf{P} \partial_k \boldsymbol{\theta}} = \frac{\partial^2 J_k}{\partial \mathbf{P} \partial_k \boldsymbol{\theta}} \\
&\frac{\partial^2 J}{\partial^2_k \boldsymbol{\theta}} = \frac{\partial^2 J_k}{\partial^2_k \boldsymbol{\theta}}
\end{aligned} \tag{5.34}$$

It shows that the evaluation of the first and second derivatives for a single-output GTSK model is only needed. Simple arithmetic operations and matrix stacking would be able to recover the derivates and Hessian matrix for a MIMO GTSK model.

The Hessian matrix is then expected to have a diagonal sub-matrix in its right-bottom corner.

$$H = \begin{bmatrix}
\frac{\partial^2 J}{\partial \mathbf{o}^2} & \frac{\partial^2 J}{\partial \mathbf{o} \partial \mathbf{P}} & \frac{\partial^2 J}{\partial \mathbf{o} \partial_1 \boldsymbol{\theta}} & \text{L} & \frac{\partial^2 J}{\partial \mathbf{o} \partial_n \boldsymbol{\theta}} \\
\frac{\partial^2 J}{\partial \mathbf{P} \partial \mathbf{o}} & \frac{\partial^2 J}{\partial \mathbf{P}^2} & \frac{\partial^2 J}{\partial \mathbf{P} \partial_1 \boldsymbol{\theta}} & \text{L} & \frac{\partial^2 J}{\partial \mathbf{P} \partial_n \boldsymbol{\theta}} \\
\frac{\partial^2 J}{\partial_1 \boldsymbol{\theta} \partial \mathbf{o}} & \frac{\partial^2 J}{\partial_1 \boldsymbol{\theta} \partial \mathbf{P}} & \frac{\partial^2 J}{\partial^2_1 \boldsymbol{\theta}} & \text{L} & 0 \\
\text{M} & \text{M} & \text{M} & \text{O} & \text{M} \\
\frac{\partial^2 J}{\partial_n \boldsymbol{\theta} \partial \mathbf{o}} & \frac{\partial^2 J}{\partial_n \boldsymbol{\theta} \partial \mathbf{P}} & 0 & \text{L} & \frac{\partial^2 J}{\partial^2_n \boldsymbol{\theta}}
\end{bmatrix} \tag{5.35}$$

5.3 Overview of the Proposed Parameter Initialization

A constrained nonlinear optimization problem is described above to estimate model parameters. The performance of the optimization is subject to the quality of initial guesses of decision variable values. One might need to try many different random initializations and find an acceptable result. At the same time, randomization has to be

carefully conducted to avoid poor centroid locations and distributions. In addition to the initialization problem, it is assumed in the above elaboration that the number of rules, *M* is known. However, this number is unknown and should be related to the complexity of the functional behavior. In practice, determination of *M* often requires trials for an appropriate choice with balanced model accuracy and complexity.

The above optimization procedure takes a 'global' approach to estimate parameter values for a GTSK model and adjust all parameters simultaneously. This approach has the advantage to fully consider interactions among all parameters while suffers the above mentioned initialization difficulties. On the other hand, a GTSK model could be viewed a collection of rules. As shown in Figure 3.6, a GTSK model consists of 4 rules, where each rule is designated to an ellipsoidal area. An alternative approach to identify a GTSK model is then to identify its rules individually. A rule is identified if its antecedent and consequent parameters are estimated. As shown in Figure 3.6, antecedent identification will be to recognize an ellipsoid in terms of a centroid and a shape matrix. Consequent model identification is reduced to an estimation of a local linear model in the corresponding antecedent area. A rule is identified if it is known where the rule is needed in terms of a region in antecedent space. The antecedent space in this work is simply defined as a minimum hypercube that contains all antecedent samples. The problem is then to define regions in antecedent space to place rules. In this work, rule regions are generated out of an antecedent space by partition. An illustrating example for the Figure 3.6 is shown below, where four regions are defined by three linear splitting boundaries (dashed lines).

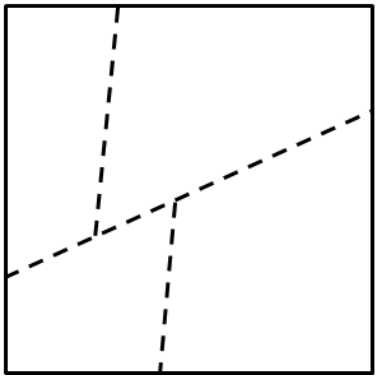


Figure 5.3. An antecedent space partitioned by three linear boundaries

In this work, boundaries are iteratively placed in an antecedent space as below.

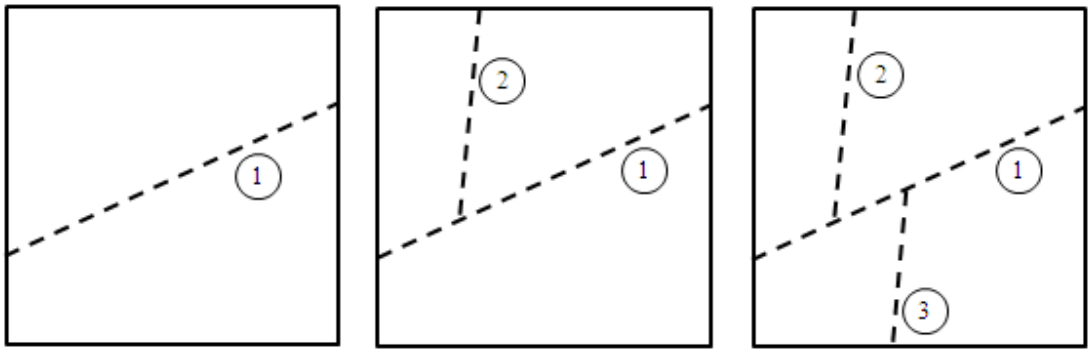
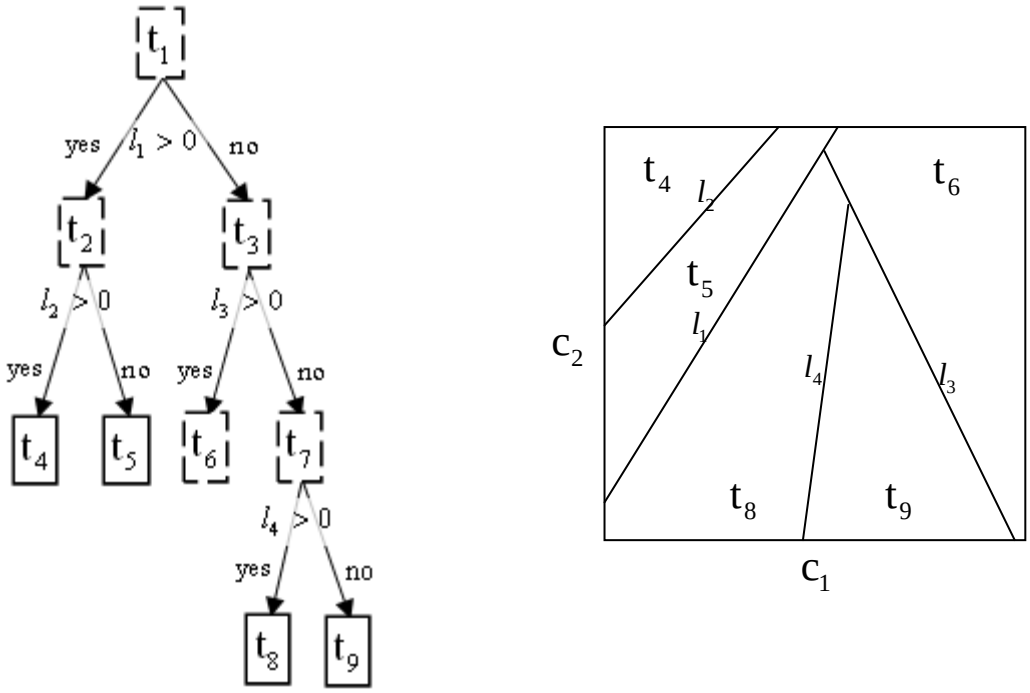


Figure 5.4. An iterative procedure to partition an antecedent space

The antecedent space partition procedure could also be represented by a regression tree (Breiman, Friedman, Olshen & Stone, 1984) as shown in Figure 5.5(a), where t_1 is the where the tree starts and is termed as a root node. Every tree has only one root that represents the original undivided antecedent space. Underneath t_1 is the first split boundary, a linear inequality, $\mathbf{s}^T\mathbf{c} \geq s_0$, which divides the space t_1 into two disjointed parts. To the left of t_1 is a branch node, t_2, which includes all the data fulfilling the inequality. The rest of data from t_1 is contained in another branch node t_3. Underneath t_2, another split boundary is presented that further divides t_2 into other two disjointed parts. Two nodes, t_4 and t_5 are then generated. No further splits are conducted on t_4 and t_5 that then make them terminal nodes. To the right of t_1, a similar splitting process is conducted, which results two branch nodes t_6 and t_7 as well as three terminal nodes t_5, t_8 and t_9.



(a) (b)

Figure 5.5. Antecedent space partition by a regression tree

The rectangle in Figure 5.5(b) outlines the range of variables c_1 and c_2 which are the two regressors identified as providing nonlinear functionality. These are the antecedent variables. Initially, the rectangle defines a space, t_1. The first split is indicated by the line labeled l_1, which split region t_1 into two regions which were labeled t_2 and t_3. However, region t_2, was split by line l_2, creating regions t_4 and t_5. Similarly, region t_3, was split by line l_3, creating regions t_6 and t_7. Then region t_7, was split by line l_4, creating regions t_8 and t_9.

Note that in this approach, it is no longer necessary to assume the number of rules, *M*. It is, however, determined along with the space partitioning procedure.

The concept of recursive space partition is also seen in (Nelles, 2001; Nelles & Isermann, 1996), where only boundaries along with axes are allowed and must pass the centroid of the space to be partitioned. In (Hartmann & Nelles, 2009; Nelles, 2006), a more general partition is defined in a sigmoid function to construct hierarchical models, which requires careful initializations of the smoothness of the sigmoid function, and

splitting position and direction to avoid trivial solutions. The partition defined in a sigmoid function could be considered as a 'soft' partition to be seen below. In this work, a 'sharp' partition is instead defined, analyzed and solved. In the meantime, the 'soft' partition is also investigated. The impact of the initial smoothness of a sigmoid function on a 'soft' partition is demonstrated to be complex and illustrated in Figure 5.29.

5.4 A Splitting and Regression Problem

5.4.1 Description of the Splitting and Regression Problem

The fundamental step to obtain an antecedent space partition is to solve a splitting and regression problem (SRP). An example SRP on a two dimensional antecedent space is illustrated in Figure 5.6. The objective is to minimize the modeling error of the partitioned data by the two linear models by placing a linear separation boundary (the bold dashed line) in the antecedent space, which results in two regions **A** and **B**. Each region has a local linear model. The two linear models shown use all relevant regressors, not just the two (c_1 and c_2) chosen to express nonlinear behavior. The separation boundary is chosen here to be linear, and is a function of c_1 and c_2.

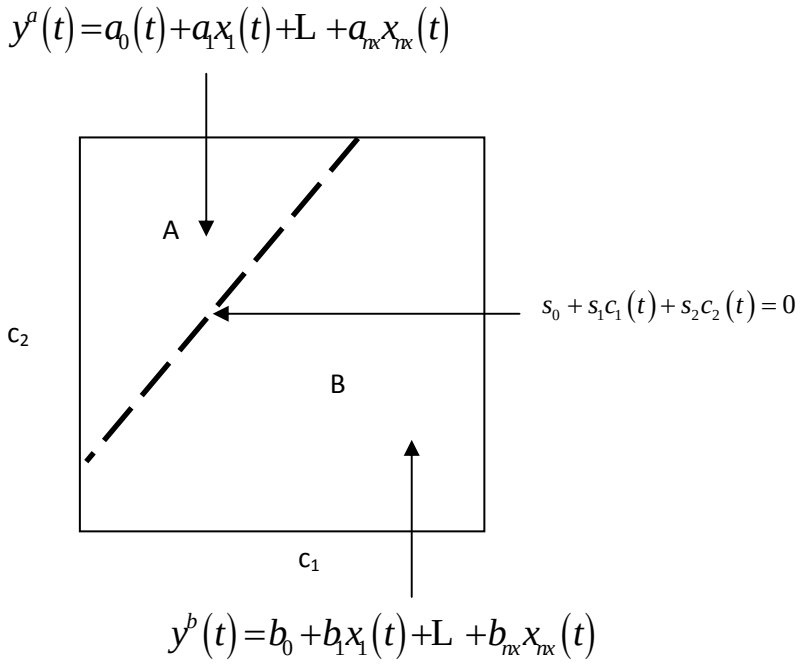


Figure 5.6. Parameters to be estimated in solving a SRP

The belongingness of data sample to region **A** is determined by $l(t)$ and $\varphi(t)$ as below

$$l(\mathrm{t}) = s_0 + s_1 c_1(\mathrm{t}) + \mathrm{L} \; + s_{nc} c_{nc}(\mathrm{t}) \tag{5.36}$$

$$\varphi(\mathrm{t}) = \begin{cases} 0, & l(\mathrm{t}) < 0 \\ 1, & l(\mathrm{t}) \geq 0 \end{cases} \tag{5.37}$$

where $s_0,\ldots,s_{\mathrm{nc}}$ defines a separation boundary in Figure 5.6. The value of $l(t)$ is $\sqrt{s_1^2 + \mathrm{L} \; + s_{nc}^2}$ times of distance of a point, $[c_1(t),\ldots,c_{\mathrm{nc}}(t)]$ to the linear separation boundary. However, Equation (5.37) implies that only the sign of $l(t)$ matters. In Figure 5.6, the points in category A have negative values for $l(t)$ while B category has positive $l(t)$. In Figure 5.6, two local linear models are

$$\begin{aligned} \mathrm{y}^a(\mathrm{t}) &= a_0 + a_1 x_1(\mathrm{t}) + \mathrm{L} \; + a_{nx} x_{nx}(\mathrm{t}) \\ \mathrm{y}^b(t) &= b_0 + b_1 x_1(\mathrm{t}) + \mathrm{L} \; + b_{nx} x_{nx}(\mathrm{t}) \end{aligned} \tag{5.38}$$

Combing Equation (5.37) with the Equation (5.38), the output prediction is then computed by

$$\hat{\mathrm{y}}(\mathrm{t}) = \left(1 - \varphi(\mathrm{t})\right) \mathrm{y}^a(\mathrm{t}) + \varphi(\mathrm{t}) y^b(\mathrm{t}) \tag{5.39}$$

The SRP is then solved by minimizing the following performance index J

$$\min_{\mathbf{s},\mathbf{a},\mathbf{b}} J = \sum_{t=1}^{N} \varepsilon^2(\mathrm{t}) \tag{5.40}$$

where, $\varepsilon(t) = y(t) - \hat{y}(t)$ is the residual, and parameter values to be estimated include **a** and **b** in Equation (5.38), and **s** in Equation (5.36),

5.4.2 SRP is Not a Clustering Problem

The problem described above includes a linear separation boundary and also a learning perspective. The SRP problem needs to minimize a performance index, which

makes it a supervised learning problem. If we only focus on the linear separation part, it seems to be a clustering problem to separate un-categorized data, which is a typical unsupervised learning problem and can be solved in different ways (Hastie, Tibshirani & Friedman, 2001). The following illustration is used to indicate the difference between a separation boundary due to an unsupervised learning and the boundary for the SRP. The function used for the illustration is defined in Equation (5.41) and Figure 5.7 shows a collection of random samples

$$y \equiv \begin{cases} y^1 = x_1 + 5x_2, & x_1 \geq x_2 \\ y^2 = 5x_1 + x_2, & x_1 < x_2 \end{cases} \tag{5.41}$$

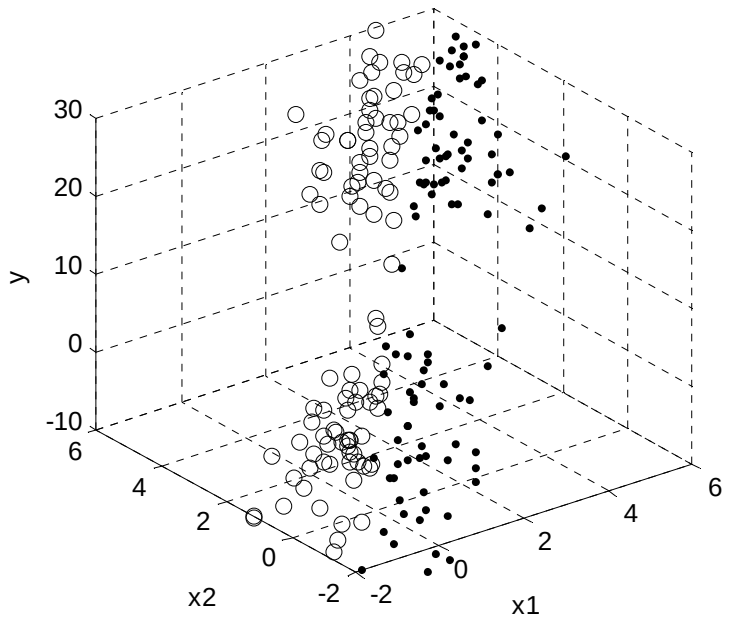


Figure 5.7. Data samples for Equation (5.41)

The antecedent space is shown in Figure 5.8.

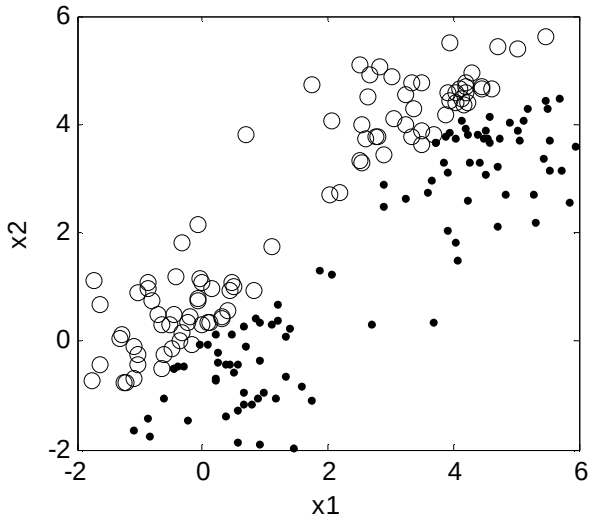


Figure 5.8. Data samples in antecedent space for Equation (5.41)

Figure 5.9 shows the result of an unsupervised learning, which separates data to two clusters based on their geometric distribution. This type of data segregation is however inconsistent with the underlying nonlinearity in the function. The desired data segregation due to the function definition is shown in Figure 5.10. Therefore, the problem to be solved is not purely an unsupervised learning problem. The boundary is not placed based on geometric distribution of data but on the function nonlinearity embedded in data.

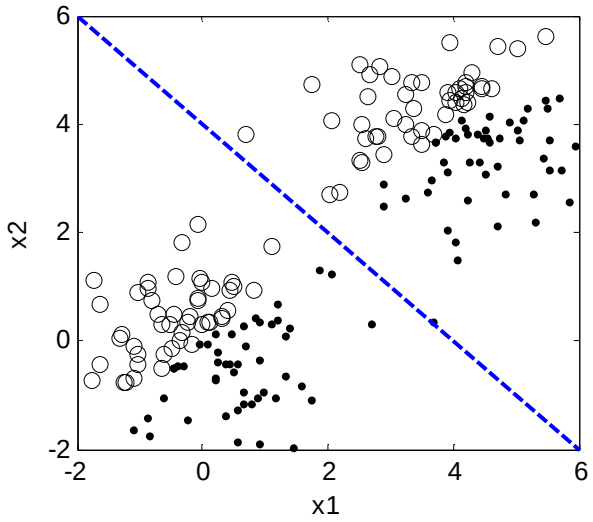


Figure 5.9. A linear boundary based on data distribution

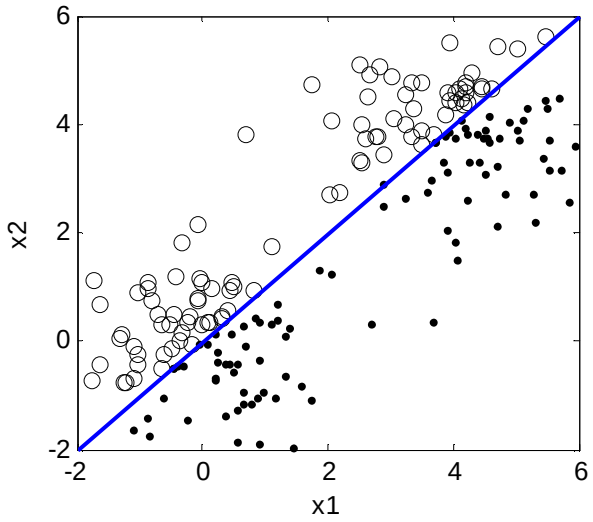


Figure 5.10. A linear boundary according to function nonlinearity

There are a number of methods proposed to initialize or identify a fuzzy model by unsupervised learning in either an input space or an input-output space (Dickerson & Kosko, 1996). Unsupervised learning is however relied on data distribution but not function nonlinearity. The above illustration shows why we should not do that.

5.4.3 Analysis of the Splitting and Regression Problem

The minimization problem in Equation (5.40) is nonlinear since the model parameters **a** or **b** are nonlinearly coupled with separation boundary parameters, **s**. The objective function in Equation (5.40) is discontinuous due to the discontinuity in the separation boundary in Equation (5.37). In order to derive more compact analytical expressions for first and second-order derivatives for analysis, Equation (5.37) is replaced by a sigmoid function

$$\varphi(t) = \frac{1}{1+e^{-\frac{l(t)}{\tau}}} \tag{5.42}$$

where, τ is introduced to adjust the 'sharpness' of the separation boundary. The impact of τ on Equation (5.42) is illustrated in Figure 5.11.

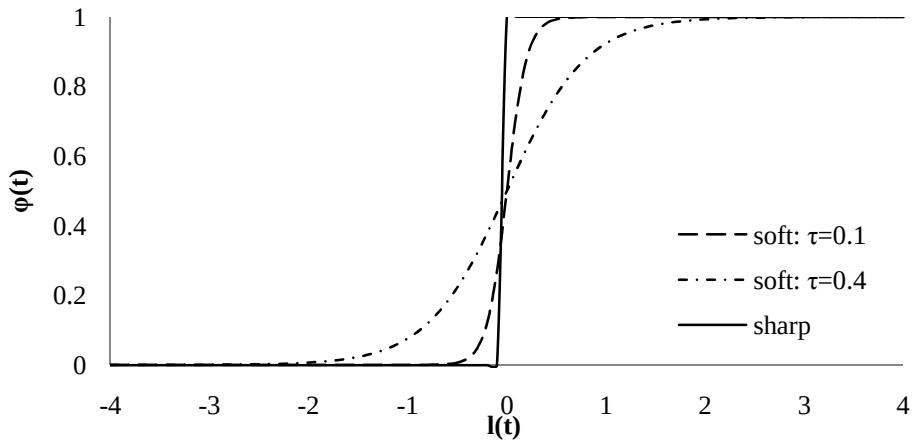


Figure 5.11. Illustration of Equation (5.42) with different τ

In this work, the original separation boundary in Equation (5.37) is called a "sharp" boundary. The modified one for analysis is a "soft" boundary. The sharp boundary is recovered from the soft one at τ approaches to zero.

With the *soft* separation boundary defined, it is then possible to compute the gradients defined by

$$\mathbf{g} = \left[\left(\frac{\partial J}{\partial \mathbf{a}} \right)^T \quad \left(\frac{\partial J}{\partial \mathbf{b}} \right)^T \quad \left(\frac{\partial J}{\partial \mathbf{s}} \right)^T \right]^T \tag{5.43}$$

where, **g** is a concatenation of three gradient vectors. Among them, for instance, the gradient of J to $\boldsymbol{a}$ is defined by

$$\frac{\partial J}{\partial \mathbf{a}} = \left[\frac{\partial J}{\partial a_0} \quad \mathrm{L} \quad \frac{\partial J}{\partial a_{nx}} \right]^T \tag{5.44}$$

The derivative of J to a_k is derived as below

$$\begin{aligned}\frac{\partial J}{\partial a_k} &= \sum_{t=1}^{N} \varepsilon(t) \frac{\partial \varepsilon(t)}{\partial a_k} \\ &= -\sum_{t=1}^{N} \varepsilon(t) \frac{\partial \hat{y}(t)}{\partial a_k} \\ &= -\sum_{t=1}^{N} \varepsilon(t)\left(1-\varphi(t)\right) \frac{\partial y^a(t)}{\partial a_k} \\ &= -\sum_{t=1}^{N} \varepsilon(t)\left(1-\varphi(t)\right) x_k(t)\end{aligned} \tag{5.45}$$

In the similar approach, the first order derivative of J to b_k is described

$$\frac{\partial J}{\partial b_k} = -\sum_{\mathrm{t}=1}^{N} \varepsilon(t)\varphi(\mathrm{t}) x_k(\mathrm{t}) \tag{5.46}$$

The first order derivative of J to s_k is computed as below

$$\frac{\partial J}{\partial s_k} = -\sum_{t=1}^{N} \varepsilon(t) y^a(t) \frac{\partial\left(1-\varphi(t)\right)}{\partial s_k} - \sum_{t=1}^{N} \varepsilon(t) y^b(t) \frac{\partial \varphi(t)}{\partial s_k} \tag{5.47}$$

with the following equality derived from Equation (5.42)

$$\frac{\partial \varphi(t)}{\partial s_k} = \varphi(t)\left(1-\varphi(t)\right) \frac{c_k(t)}{\tau} \tag{5.48}$$

the derivative to s_k is concluded by

$$\frac{\partial J}{\partial s_k} = \sum_{\mathrm{t}=1}^{N} \varepsilon(t)\varphi(\mathrm{t})\left(1-\varphi(\mathrm{t})\right) \frac{c_k(\mathrm{t})}{\tau} w(\mathrm{t}) \tag{5.49}$$

where $w(t) = y^a(t)-y^b(t)$ is the prediction difference between two local models.

The second-order derivative is collected in the following Hessian matrix,

$$\mathbf{H}=\begin{bmatrix} \dfrac{\partial^2 J}{\partial \mathbf{a}\partial \mathbf{a}^T} & \dfrac{\partial^2 J}{\partial \mathbf{a}\partial \mathbf{b}^T} & \dfrac{\partial^2 J}{\partial \mathbf{a}\partial \mathbf{s}^T} \\ \dfrac{\partial^2 J}{\partial \mathbf{b}\partial \mathbf{a}^T} & \dfrac{\partial^2 J}{\partial \mathbf{b}\partial \mathbf{b}^T} & \dfrac{\partial^2 J}{\partial \mathbf{b}\partial \mathbf{s}^T} \\ \dfrac{\partial^2 J}{\partial \mathbf{s}\partial \mathbf{a}^T} & \dfrac{\partial^2 J}{\partial \mathbf{s}\partial \mathbf{b}^T} & \dfrac{\partial^2 J}{\partial \mathbf{s}\partial \mathbf{s}^T} \end{bmatrix} \tag{5.50}$$

where for instance, H(1,2) is a *nx*+1 by *nx*+1 matrix as defined by

$$\frac{\partial^2 J}{\partial \mathbf{a}\partial \mathbf{b}^T}=\begin{bmatrix} \dfrac{\partial^2 J}{\partial a_0 \partial b_0} & \text{L} & \dfrac{\partial^2 J}{\partial a_0 \partial b_{nx}} \\ \text{M} & \text{O} & \text{M} \\ \dfrac{\partial^2 J}{\partial a_{nx} \partial b_0} & \text{M} & \dfrac{\partial^2 J}{\partial a_{nx} \partial b_{nx}} \end{bmatrix} \tag{5.51}$$

the definitions for other block matrices are similar. The explicit derivations of each matrix element are given as below

$$\frac{\partial^2 J}{\partial a_k \partial a_l}=\sum_{\mathrm{t}=1}^{N}\left(1-\varphi(\mathrm{t})\right)^2 x_k(\mathrm{t})\,x_l(\mathrm{t}) \tag{5.52}$$

$$\frac{\partial^2 J}{\partial a_k \partial b_l}=\sum_{\mathrm{t}=1}^{N}\varphi(\mathrm{t})\left(1-\varphi(\mathrm{t})\right)x_k(\mathrm{t})\,x_l(\mathrm{t}) \tag{5.53}$$

$$\begin{aligned}\frac{\partial^2 J}{\partial a_k \partial s_l}=&-\sum_{\mathrm{t}=1}^{N}\varphi(\mathrm{t})\left(1-\varphi(\mathrm{t})\right)\frac{c_l(\mathrm{t})}{\tau}x_k(\mathrm{t})\,w(\mathrm{t})\\ &+\sum_{\mathrm{t}=1}^{N}\varphi^2(\mathrm{t})\left(1-\varphi(\mathrm{t})\right)\frac{c_l(\mathrm{t})}{\tau}x_k(\mathrm{t})\,w(\mathrm{t})\\ &+\sum_{\mathrm{t}=1}^{N}\varepsilon(\mathrm{t})\varphi(\mathrm{t})\left(1-\varphi(\mathrm{t})\right)\frac{c_l(\mathrm{t})}{\tau}x_k(\mathrm{t})\end{aligned} \tag{5.54}$$

$$\frac{\partial^2 J}{\partial b_k \partial b_l}=\sum_{\mathrm{t}=1}^{N}\varphi^2(\mathrm{t})\,x_k(\mathrm{t})\,x_l(\mathrm{t}) \tag{5.55}$$

$$\frac{\partial^2 J}{\partial b_k \partial s_l} = -\sum_{\mathrm{t}=1}^{N} \varphi^2(\mathrm{t})\left(1-\varphi(\mathrm{t})\right)\frac{c_l(\mathrm{t})}{\tau}x_k(\mathrm{t})\,w(\mathrm{t}) - \sum_{\mathrm{t}=1}^{N} \varepsilon(\mathrm{t})\varphi(\mathrm{t})\left(1-\varphi(\mathrm{t})\right)\frac{c_l(\mathrm{t})}{\tau}x_k(\mathrm{t}) \tag{5.56}$$

$$\frac{\partial^2 J}{\partial s_k \partial s_l} = \sum_{\mathrm{t}=1}^{N} \varphi^2(\mathrm{t})\left(1-\varphi(\mathrm{t})\right)^2\frac{c_k(\mathrm{t})c_l(\mathrm{t})}{\tau^2}w^2(\mathrm{t}) + \sum_{\mathrm{t}=1}^{N} \varepsilon(\mathrm{t})\varphi(\mathrm{t})\left(1-\varphi(\mathrm{t})\right)^2\frac{c_k(\mathrm{t})c_l(\mathrm{t})}{\tau^2}w(\mathrm{t}) - \sum_{\mathrm{t}=1}^{N} \varepsilon(\mathrm{t})\varphi^2(\mathrm{t})\left(1-\varphi(\mathrm{t})\right)\frac{c_k(\mathrm{t})c_l(\mathrm{t})}{\tau^2}w(\mathrm{t}) \tag{5.57}$$

Once the gradients and Hessian matrix are obtained, it is then possible to analyze local solutions. The "soft" boundary is an approximation of the "sharp" boundary. As mentioned above, the "sharp" boundary is recovered from the "soft" one as τ approaches to zero. It is then possible to obtain the gradients and Hessian matrix for the "sharp" boundary by computing the limits of Equations (5.43) and (5.50) for the "soft" one.

$$\begin{aligned}\mathbf{g}_0 &= \lim_{\tau\to 0}\mathbf{g}\\ \mathbf{H}_0 &= \lim_{\tau\to 0}\mathbf{H}\end{aligned} \tag{5.58}$$

where, the following limit appearing in Equations (5.49, 5.53, 5.54, 5.56 and 5.57) needs to be evaluated

$$\begin{aligned}\lim_{\tau\to 0}\frac{\varphi(t)\left(1-\varphi(t)\right)}{\tau} &= \lim_{\tau\to 0}\frac{e^{-l(t)\tau^{-1}}}{\left(1+e^{-l(t)\tau^{-1}}\right)^2\tau}\\ &= \lim_{\tau\to 0}\frac{e^{-l(t)\tau^{-1}}}{\tau} \qquad \left(\text{let } u=\tau^{-1}\right)\\ &= \lim_{u\to\infty}\frac{u}{e^{ul(t)}} = \lim_{u\to\infty}\frac{(u)'}{\left(e^{ul(t)}\right)'}\\ &= \lim_{u\to\infty}\frac{1}{l(t)e^{ul(t)}} = 0\end{aligned} \tag{5.59}$$

Using above evaluation, the gradients are then reevaluated by

$$\frac{\partial J}{\partial a_k} = -\sum_{i=1}^{Na} \varepsilon\left(t_i^a\right) x_k\left(t_i^a\right) \tag{5.60}$$

$$\frac{\partial J}{\partial b_k} = -\sum_{j=1}^{Nb} \varepsilon\left(t_j^b\right) x_k\left(t_j^b\right) \tag{5.61}$$

$$\frac{\partial J}{\partial s_k} = 0 \tag{5.62}$$

Where $Na = \left\{ t^a \,\middle|\, \varphi\left(t^a\right) = 0 \right\}$ and $Nb = \left\{ t^b \,\middle|\, \varphi\left(t^b\right) = 1 \right\}$. *Na* collects data belonging to group A while *Nb* collects data in group B. Second-order derivatives are evaluated by

$$\frac{\partial^2 J}{\partial a_k \partial a_l} = \sum_{i=1}^{Na} x_k\left(t_i^a\right) x_l\left(t_i^a\right) \tag{5.63}$$

$$\frac{\partial^2 J}{\partial a_k \partial b_l} = 0 \tag{5.64}$$

$$\frac{\partial^2 J}{\partial a_k \partial s_l} = 0 \tag{5.65}$$

$$\frac{\partial^2 J}{\partial b_k \partial b_l} = \sum_{j=1}^{Nb} x_k\left(t_j^b\right) x_l\left(t_j^b\right) \tag{5.66}$$

$$\frac{\partial^2 J}{\partial b_k \partial s_l} = 0 \tag{5.67}$$

$$\frac{\partial^2 J}{\partial s_k \partial s_l} = 0 \tag{5.68}$$

Equilibrium solutions are defined if Equations (5.60) and (5.61) are zero. The equilibrium condition on s_k is automatically satisfied in Equation (5.62). One possible

solution is to have all $\varphi(t) = 0$ (or $\varphi(t) = 1$) for all t, then the equilibrium condition is

$$\sum_{t=1}^{N} \varepsilon(t)\, x_k(t) = 0$$

which is resulted from a least square estimation of model parameters for one linear model. The $\varphi(t)=0$ for all t implies that all data belong to group A and no data is in B. The corresponding Hessian matrix is then described by

$$\mathbf{H} = \begin{bmatrix} \mathbf{X}^T\mathbf{X} & 0 \\ 0 & 0 \end{bmatrix} \tag{5.69}$$

where, $\mathbf{X}$ is defined in Equation (5.1). $\mathbf{X}^T\mathbf{X}$ is positive semi-definite if $\mathbf{X}$ has linear independent columns, which is a reasonable assumption for a linear regression problem. $\mathbf{H}$ is hence positive semi-definite and the solution with $\varphi(t) = 0$ is stable. The same conclusion is also available for $\varphi(t) =1$. These two situations define trivial solutions for the SRP problem since no separation is obtained.

On the contrary, a non-trivial separation will have both zero and non-zero $\varphi(t)$. The equilibrium condition is described by

$$\sum_{i=1}^{Na} \varepsilon\left(t_i^a\right) x_k\left(t_i^a\right) = 0$$

$$\sum_{j=1}^{Nb} \varepsilon\left(t_j^b\right) x_k\left(t_j^b\right) = 0$$

where, two equations are independent to each other, each of which is satisfied if model parameters are estimated by a least square estimation. The corresponding Hessian matrix then becomes

$$\mathbf{H} = \begin{bmatrix} \mathbf{X}_A^T\mathbf{X}_A & 0 & 0 \\ 0 & \mathbf{X}_B^T\mathbf{X}_B & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.70}$$

where $\mathbf{X}_A$ denote the portion of $\mathbf{X}$ being assigned to model A and is described by

$$\mathbf{X_A} = \begin{bmatrix} 1 & x_1\left(t_1^a\right) & \text{L} & x_{nx}\left(t_1^a\right) \\ \text{M} & \text{M} & \text{O} & \text{M} \\ 1 & x_1\left(t_{Na}^a\right) & \text{L} & x_{nx}\left(t_{Na}^a\right) \end{bmatrix} \tag{5.71}$$

Then the Hessian matrix in Equation (5.70) is also positive semi-definite and indicates a stable solution.

Based on the above analysis, it can be concluded that the SRP has many local minima. A local minimum could be trivial if the separation happens outside the antecedent space. If the separation is placed inside, two local models are then obtained. It then provides a two-step procedure to reach an equilibrium solution starting from an arbitrary separation boundary followed by least square estimation on one or two models depending on the location of the boundary. In a searching space with many stable local minima, a gradient based optimization method, which optimizes both separation and local model parameters simultaneously, can easily get trapped. On the other hand, obtaining a local optimal solution is however often good enough to be expected in practice. In the following, we will follow a heuristic procedure to obtain a particular local optimal solution, which, as will be demonstrated in section 5.5.5, tends to be a global solution compared with solutions obtained from other methods.

5.5 Solving of the Splitting and Regression Problem

5.5.1 Initialization of Data Segregation

As analyzed in 5.4, the SRP problem has many local optimal solutions. A local solution is obtained when a random boundary is given. A trivial solution is obtained if the boundary is outside the antecedent space.

The SRP is solved in this work by a heuristic suboptimal approach. The heuristic approach is based on the assumption that the entire data set could be described by two local linear models. The entire data set is denoted by [**y C X**] in Equation (5.1).

A separation is specified by **s** defined in Equation (5.36). We then have the following expression for a separation. Given a separation defined by **s**, it results in a split of data [**y C X**] into **A** and **B** groups as [$\mathbf{y}_A$ $\mathbf{C}_A$ $\mathbf{X}_A$] and [$\mathbf{y}_B$ $\mathbf{C}_B$ $\mathbf{X}_B$] with definitions for group A as shown below. The definition for $\mathbf{X}_A$ is described in Equation (5.71), and $\mathbf{y}_A$ and $\mathbf{C}_A$ are defined by

$$\mathbf{y_A} = \begin{bmatrix} y\left(t_1^a\right) \\ \text{M} \\ y\left(t_{Na}^a\right) \end{bmatrix} \quad \mathbf{C_A} = \begin{bmatrix} c_1\left(t_1^a\right) & \text{L} & c_{nc}\left(t_1^a\right) \\ \text{M} & \text{O} & \text{M} \\ c_1\left(t_{Na}^a\right) & \text{L} & c_{nc}\left(t_{Na}^a\right) \end{bmatrix} \tag{5.72}$$

The model with two underlying linear models are defined by

$$\begin{aligned} &\mathbf{y_A} = \mathbf{X_A a} + \mathbf{e_A} \\ &\mathbf{y_B} = \mathbf{X_B b} + \mathbf{e_B} \\ &\text{with} \\ &\mathbf{y_A} = \left[y^a(1) \quad \text{L} \quad y^a\left(N_A\right) \right]^T \\ &\mathbf{y_B} = \left[y^b(1) \quad \text{L} \quad y^b\left(N_B\right) \right]^T \\ &\mathbf{e_A} : N\left(0, \sigma_{\mathbf{A}}^2 \mathbf{I}\right); \qquad \mathbf{e_B} : N\left(0, \sigma_{\mathbf{B}}^2 \mathbf{I}\right) \end{aligned} \tag{5.73}$$

The corresponding model parameters **a** and **b** are estimated.

$$\begin{aligned} \hat{\mathbf{a}} &= \left(\mathbf{X_A^T X_A}\right)^{-1} \mathbf{X_A^T y_A} \\ \hat{\mathbf{b}} &= \left(\mathbf{X_B^T X_B}\right)^{-1} \mathbf{X_B^T y_B} \end{aligned} \tag{5.74}$$

the residual for model **A** could then be evaluated by

$$\boldsymbol{\varepsilon}_{\mathbf{A}} = \mathbf{y_A} - \mathbf{X_A}\hat{\mathbf{a}} \tag{5.75}$$

substituting Equations (5.74) and (5.75) for $\hat{\mathbf{a}}$ and $\mathbf{y}_A$ the residual terms is then described

$$\begin{aligned}\boldsymbol{\varepsilon}_A &= \mathbf{X}_A\mathbf{a} + \mathbf{e}_A - \left(\mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\right)\left(\mathbf{X}_A\mathbf{a} + \mathbf{e}_A\right) \\ &= \left(\mathbf{I} - \mathbf{X}_\mathbf{A}\left(\mathbf{X}_\mathbf{A}^T\mathbf{X}_\mathbf{A}\right)^{-1}\mathbf{X}_\mathbf{A}^T\right)\mathbf{e}_\mathbf{A}\end{aligned} \tag{5.76}$$

The residual is then used to compute a quadratic performance criterion, $\mathbf{J}_\mathrm{A}$ for model A by

$$J_A = E\left[\boldsymbol{\varepsilon}_A^T\boldsymbol{\varepsilon}_A\right] \tag{5.77}$$

where, $\boldsymbol{\varepsilon}^\mathrm{T}\,\boldsymbol{\varepsilon}$ is equal to the trace of a matrix $\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\mathrm{T}$.

$$J_A = E\left[Tr\left(\boldsymbol{\varepsilon}_A\boldsymbol{\varepsilon}_A^T\right)\right] \tag{5.78}$$

with definition of $\boldsymbol{\varepsilon}_\mathrm{A}$ in Equation (5.76)

$$\begin{aligned}J_\mathbf{A} &= E\left[Tr\left(\left(\mathbf{I} - \mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\right)\mathbf{e}_A\mathbf{e}_A^T\left(\mathbf{I} - \mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\right)\right)\right] \\ &= E\left[Tr\left(\left(\mathbf{I} - \mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\right)^2\mathbf{e}_A\mathbf{e}_A^T\right)\right]\end{aligned} \tag{5.79}$$

where the cyclic operation in Trace is used to obtain the above equality. In addition, it can be verified that

$$\begin{aligned}&\left(\mathbf{I} - \mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\right)^2 \\ &\quad= \mathbf{I} - 2\mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T + \mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T \\ &\quad= \mathbf{I} - 2\mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T + \mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T \\ &\quad= \mathbf{I} - \mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\end{aligned}$$

then J_A is expressed in terms of $\mathbf{X}_\mathrm{A}$ and σ_A by.

$$
\begin{aligned}
J_{\mathbf{A}} &= E\left[Tr\left(\left(\mathbf{I}-\mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\right)\mathbf{e}_A\mathbf{e}_A^T\right)\right] \\
&= Tr\left(\left(\mathbf{I}-\mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\right)E\left[\mathbf{e}_A\mathbf{e}_A^T\right]\right) \\
&= Tr\left(\left(\mathbf{I}-\mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\right)\sigma_A^2\mathbf{I}\right) \\
&= Tr\left(\mathbf{I}-\mathbf{X}_A\left(\mathbf{X}_A^T\mathbf{X}_A\right)^{-1}\mathbf{X}_A^T\right)\sigma_A^2
\end{aligned}
\tag{5.80}
$$

where, the Trace term is evaluated as below

$$
\begin{aligned}
&Tr\left(\mathbf{I}-\mathbf{X}_{\mathbf{A}}\left(\mathbf{X}_{\mathbf{A}}^T\mathbf{X}_{\mathbf{A}}\right)^{-1}\mathbf{X}_{\mathbf{A}}^T\right) \\
&\quad = Tr(I)-Tr\left(\mathbf{X}_{\mathbf{A}}\left(\mathbf{X}_{\mathbf{A}}^T\mathbf{X}_{\mathbf{A}}\right)^{-1}\mathbf{X}_{\mathbf{A}}^T\right) \\
&\quad = Na-Tr\left(\left(\mathbf{X}_{\mathbf{A}}^T\mathbf{X}_{\mathbf{A}}\right)^{-1}\mathbf{X}_{\mathbf{A}}^T\mathbf{X}_{\mathbf{A}}\right) \\
&\quad = Na-Tr\left(I_{nx+1}\right) \\
&\quad = Na-nx-1
\end{aligned}
$$

then

$$
\begin{aligned}
J_{\mathbf{A}} &= \left(Na-nx-1\right)\sigma_{\mathbf{A}}^2 \\
&\approx Na\sigma_{\mathbf{A}}^2
\end{aligned}
\tag{5.81}
$$

where, it is assumed that that Na >> nx.

In the same manner, the performance criterion for model **B** is described by $J_{\mathbf{B}} = Tr\left(\mathbf{I}-\mathbf{X}_{\mathbf{B}}\left(\mathbf{X}_{\mathbf{B}}^T\mathbf{X}_{\mathbf{B}}\right)^{-1}\mathbf{X}_{\mathbf{B}}^T\right)\sigma_{\mathbf{B}}^2$ and approximated by $J_{\mathbf{B}} \approx Nb\sigma_{\mathbf{B}}^2$. Then the quadratic performance is expressed as a weighted sum of σ_{A} and σ_{B} by

$$
\begin{aligned}
J &= J_{\mathbf{A}} + J_{\mathbf{B}} \\
&\approx N_{\mathbf{A}}\sigma_{\mathbf{A}}^2 + N_{\mathbf{B}}\sigma_{\mathbf{B}}^2
\end{aligned}
\tag{5.82}
$$

where, if $\varphi(t)$ in Equation (5.37) is known, $N_{\mathbf{A}}$ and $N_{\mathbf{B}}$ can be calculated by

$$N_{\mathbf{A}} = \sum_{t=1}^{N} \varphi(t), \;\; N_{\mathbf{B}} = N - \sum_{t=1}^{N} \varphi(t) \tag{5.83}$$

additionally, the unknown $\sigma_{\mathbf{A}}^2$ and $\sigma_{\mathbf{B}}^2$ are to be replaced by their estimates by

$$\begin{aligned} \hat{\sigma}_{\mathbf{A}}^2 &= \frac{\sum_{t=1}^{N} \left[\varphi(t) \left(y(t) - \mu_{\mathbf{A}} \right) \right]^2}{\sum_{t=1}^{N} \varphi(t)} \\ \hat{\sigma}_{\mathbf{B}}^2 &= \frac{\sum_{t=1}^{N} \left[\left(1 - \varphi(t) \right) \left(y(t) - \mu_{\mathbf{B}} \right) \right]^2}{\sum_{t=1}^{N} \left(1 - \varphi(t) \right)} \end{aligned} \tag{5.84}$$

where $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ are unknown means of $y_{\mathbf{A}}$ and $y_{\mathbf{B}}$ in groups **A** and **B**. Substituting Equations (5.83) and (5.84) to Equation (5.82), the minimization problem to be solved is described by

$$\begin{aligned} &\underset{\mu_{\mathbf{A}}, \mu_{\mathbf{B}}, \varphi(t)}{\text{minmize}} \; J = \sum_{t=1}^{N} \varphi^2(t) \left(y(t) - \mu_{\mathbf{A}} \right)^2 + \left(1 - \varphi(t) \right)^2 \left(y(t) - \mu_{\mathbf{B}} \right)^2 \\ &\text{subject to} \\ &\qquad \varphi(t) = 0, 1; \qquad\qquad t = 1, \mathrm{L} \;, N \end{aligned} \tag{5.85}$$

where, there are N+2 decision variables, N belongingness values, $\varphi(t)$, $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$. Since the $\varphi(t)$ are not coupled, it can be solved individually by solving a simple optimization problem for the objective J(t) if and $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ are assumed to be known

$$J(t) = \varphi^2(\mathrm{t}) \left(y(t) - \mu_{\mathbf{A}} \right)^2 + \left(1 - \varphi(\mathrm{t}) \right)^2 \left(y(\mathrm{t}) - \mu_{\mathbf{B}} \right)^2 \tag{5.86}$$

Where

$$\begin{aligned} \frac{\partial J(t)}{\partial \varphi(t)} &= 2\varphi(t) \left(y(t) - \mu_{\mathbf{A}} \right)^2 - 2\left(1 - \varphi(t) \right) \left(y(t) - \mu_{\mathbf{B}} \right)^2 \\ \frac{\partial^2 J(t)}{\partial \varphi^2(t)} &= 2\left(y(t) - \mu_{\mathbf{A}} \right)^2 + 2\left(y(t) - \mu_{\mathbf{B}} \right)^2 \end{aligned}$$

By equating the first-order derivative to zero, $\varphi(t)$ is then solved by

$$\varphi(t) = \frac{\left(y(t) - \mu_{\mathbf{B}}\right)^2}{\left(y(t) - \mu_{\mathbf{A}}\right)^2 + \left(y(t) - \mu_{\mathbf{B}}\right)^2} \tag{5.87}$$

Where the second-order derivative is always positive assuming that $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ are not equal to $y(t)$ at the same time. It then verifies that the solution of $\varphi(t)$ in Equation (5.87) is a global optimal solution for the $J(t)$ in Equation (5.86) minimizes $J(t)$.

Combining Equations (5.85) and (5.87) defines the minimization problem in terms of $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ only by.

$$\underset{\mu_{\mathbf{A}},\mu_{\mathbf{B}}}{\text{minmize}}\ J = \sum_{t=1}^{N} \frac{\left(y(t) - \mu_{\mathbf{A}}\right)^2 \left(y(t) - \mu_{\mathbf{B}}\right)^2}{\left(y(t) - \mu_{\mathbf{A}}\right)^2 + \left(y(t) - \mu_{\mathbf{B}}\right)^2} \tag{5.88}$$

the objective function in Equation (5.88) has only two decision variables $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$, which is to be found using a Newton's method. The first order derivatives of J to $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ are computed by

$$\begin{aligned} \frac{\partial J}{\partial \mu_{\mathbf{A}}} &= \sum_{t=1}^{N} \frac{-2\left(y(t) - \mu_{\mathbf{A}}\right)\left(y(t) - \mu_{\mathbf{B}}\right)^4}{\left(\left(y(t) - \mu_{\mathbf{A}}\right)^2 + \left(y(t) - \mu_{\mathbf{B}}\right)^2\right)^2} \\ \frac{\partial J}{\partial \mu_{\mathbf{B}}} &= \sum_{t=1}^{N} \frac{-2\left(y(t) - \mu_{\mathbf{A}}\right)^4\left(y(t) - \mu_{\mathbf{B}}\right)}{\left(\left(y(t) - \mu_{\mathbf{A}}\right)^2 + \left(y(t) - \mu_{\mathbf{B}}\right)^2\right)^2} \end{aligned} \tag{5.89}$$

And the second-order derivatives are described by

$$\frac{\partial^2 J}{\partial \mu_{\mathbf{A}}^2} = \sum_{t=1}^{N} \frac{2\left(y(t)-\mu_{\mathbf{B}}\right)^6 - 6\left(y(t)-\mu_{\mathbf{A}}\right)^2\left(y(t)-\mu_{\mathbf{B}}\right)^4}{\left(\left(y(t)-\mu_{\mathbf{A}}\right)^2 + \left(y(t)-\mu_{\mathbf{B}}\right)^2\right)^3}$$

$$\frac{\partial^2 J}{\partial \mu_{\mathbf{B}}^2} = \sum_{t=1}^{N} \frac{2\left(y(t)-\mu_{\mathbf{A}}\right)^6 - 6\left(y(t)-\mu_{\mathbf{B}}\right)^2\left(y(t)-\mu_{\mathbf{A}}\right)^4}{\left(\left(y(t)-\mu_{\mathbf{A}}\right)^2 + \left(y(t)-\mu_{\mathbf{B}}\right)^2\right)^3} \tag{5.90}$$

$$\frac{\partial^2 J}{\partial \mu_{\mathbf{A}} \partial \mu_{\mathbf{B}}} = \sum_{t=1}^{N} \frac{8\left(y(t)-\mu_{\mathbf{A}}\right)^3\left(y(t)-\mu_{\mathbf{B}}\right)^3}{\left(\left(y(t)-\mu_{\mathbf{A}}\right)^2 + \left(y(t)-\mu_{\mathbf{B}}\right)^2\right)^3}$$

Using the gradient and Hessian matrix, a version of Newton's method modified for non-convex quadratic problem in (Han, Pardalos & Ye, 1992) is used to minimize J ,and find $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ since it is possible that resultant Hessian matrix might be indefinite (containing both positive and negative eigenvalues).

Once J is minimized, $\varphi(t)$ is determined by Equation (5.87) and automatically lies between 0 and 1. The resultant $\varphi(t)$ takes any value within 0 and 1 instead of 0 and 1 only as defined in Equation (5.37). The following Equation (5.91) will convert the $\varphi(t)$ to a two-value indicator (0,1)

$$\varphi(t) = \begin{cases} 0, & \varphi(t) < 0.5 \\ 1, & \varphi(t) \geq 0.5 \end{cases} \tag{5.91}$$

which assigns each data sample to either group **A** or **B**.

5.5.2 Solving for a Linear Boundary

Note the solving procedure mentioned above does not use a linear separation boundary. $\varphi(t)$ is obtained by minimizing J in Equation (5.88) but not confined to a linear separation boundary defined in Equation (5.36). Now the problem to be solved comes down to find a linear boundary segregating data with known categories, 0 and 1 due to Equation (5.91). There are many ways to place a linear separation boundary in data with known classifications. Perceptron neural network, logistic regression and linear discriminate are all possible methods to find a linear separation boundary. However,

these methods are only effective if the classification problem is linear separable.

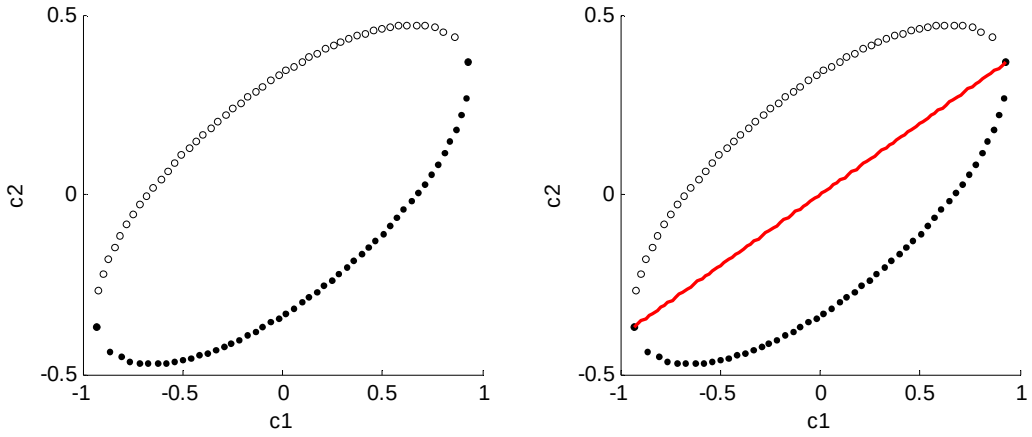
Multi-layer perceptrons (Hagan, Demuth & Beale, 2002) can be used for linear inseparable classifications assuming the number of linear boundaries is known. Linear regression can be used to fit a linear separation model for a two-value function. The resultant separation boundary is often not robust. A more robust approach way to find a linear separation boundary is by solving a support vector machine (SVM) (Hastie, Tibshirani & Friedman, 2001). The following version of SVM is used in this work to find the linear separation parameters **s** based on obtained $\varphi(t)$

$$\begin{aligned}
&\text{minimize } \sum_{k=0}^{nc} s_k^2 + r\left(\sum_{i=1}^{N_a} \xi_i^a + \sum_{j=1}^{N_b} \xi_j^b\right) \\
&\text{subject to} \\
&\quad s_0 + s_1 c_1\left(t_i^a\right) + \mathrm{L} \; + s_{nc} c_{nc}\left(t_i^a\right) \geq 1 - \xi_i^a, \qquad i = 1\mathrm{L} \;\; N_a \\
&\quad s_0 + s_1 c_1\left(t_j^b\right) + \mathrm{L} \; + s_{nc} c_{nc}\left(t_j^b\right) \leq \xi_j^b - 1, \; j = 1\mathrm{L} \;\; N_b \\
&\quad \xi_i^a, \xi_i^a \geq 0
\end{aligned} \tag{5.92}$$

where 'slack' variables ξ are introduced to take care of misclassification if the problem is non-separable. A misclassification is indicated by $\xi > 1$. The scalar r is used to penalize the total amount of misclassification.

In implementing the above procedure to find a separation boundary in practice, one practical problem is encountered when a trivial solution is obtained via solving the SVM. The trivial solution is defined by letting all separation parameters be zero. One possible situation to have a trivial solution is when the problem is equally mixed. A different approach is then taken to find a separation boundary if a zero boundary is obtained out of the SVM.

The following several examples show progressively how a trivial solution is obtained. Figure 5.12(a) shows a linearly separable example; the obtained separation boundary due to SVM is shown in Figure 5.12(b). In fact, the obtained boundary is same as that due to a linear discriminate method.



(a) (b)

Figure 5.12. A linear boundary generated for liner separable data

A little mixed example (linear non-separable) is shown in Figure 5.13, where 5 solid dots are mixed with circles. The solid separation boundary is due to a SVM solution and the dashed line is due to a linear discriminate method. Two methods can be compared based on the number of misclassifications. The SVM method performances better with 10 misclassification than the linear discriminate with 16 misclassifications.

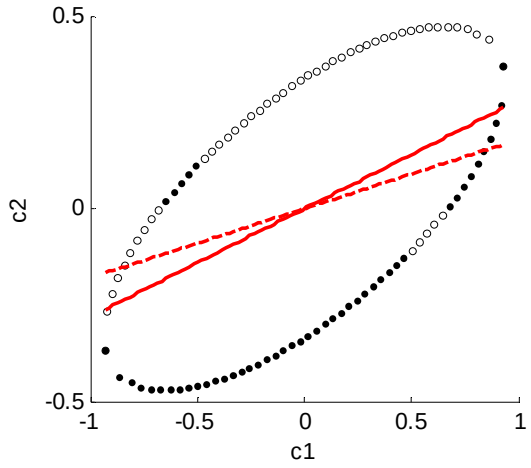


Figure 5.13. A linear non-separable case;
(solid line by SVM, dashed line by liner discriminate)

A more mixed or non-separable case is shown in Figure 5.14, where a set of dots

are followed by a set of equal number circles. The pattern then repeats.

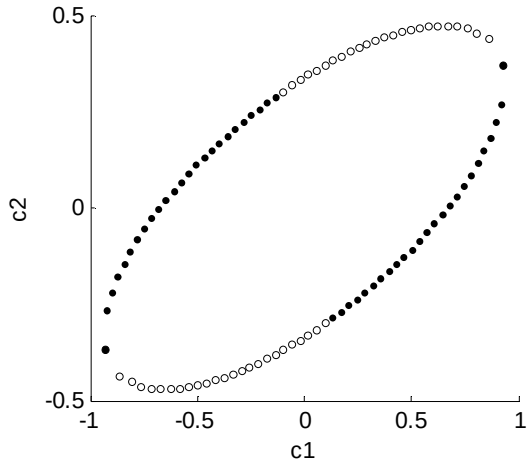


Figure 5.14. A liner non-separable example with equally mixed points

The resultant values of separation parameters are all zeros by either SVM or linear discriminate method. The dots and circles are equally mixed. Any linear boundary through the center of region will end up with same number misclassification. In the objective function of Equation (5.92), the penalty term for the non-separability is a constant. Therefore, the only quantity can be minimized is the norm of separation parameter vector. Its minimum is zero with all separation parameters being zero.

On the other hand, it is clear from Figure 5.14 that two categories of data exist, dots and circles. Separation has to be defined. In this situation, a technique based on a special type of neural network, liner vector quantization (LVQ) (Hagan, Demuth & Beale, 2002) is used to find a suitable linear boundary. LVQ is a clustering technique used to recognize clusters in the categorized data. A separation boundary could be defined by connecting centers of two clusters for different categories. Figure 5.15 shows the result of the implementation of LVQ for the problem in Figure 5.14. As shown, there are 10 clusters (triangles) recognized for dots and 10 clusters (stars) for circles. There are hence totally 100 possible linear separation boundaries. The best one is reported as the found separation boundary.

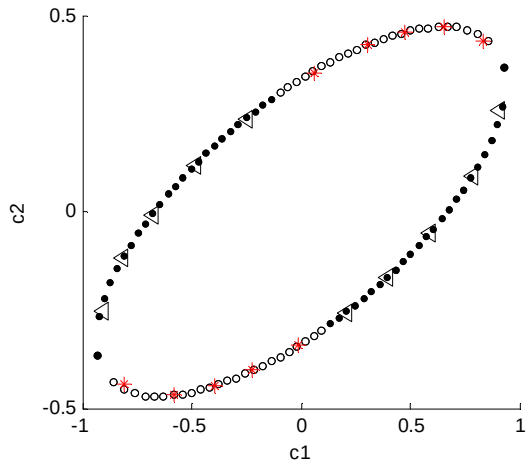


Figure 5.15. Clusters found by LVQ for data in Figure 5.14

5.5.3 Boundary Refinement

The solved **s** is then applied to Equations (5.36) and (5.37) to update $\varphi(t)$, which is now confined a linear separation boundary. The resultant $\varphi(t)$ defines a split, [$\mathbf{y}_A$ $\mathbf{C}_A$ $\mathbf{X}_A$] and [$\mathbf{y}_B$ $\mathbf{C}_B$ $\mathbf{X}_B$]. Then **a** and **b** are estimated by Equation (5.74). It then is able to evaluate residuals $\boldsymbol{\varepsilon}_\mathbf{A}$ and $\boldsymbol{\varepsilon}_\mathbf{B}$ explicitly by Equation (5.75). The belongingness values of $\varphi(t)$ are then updated by minimizing the following J with replacement of $(y(t)-\mu_\mathbf{A})$ and $(y(t)-\mu_\mathbf{B})$ in Equation (5.88) by $\varepsilon_\mathbf{A}(t)$ and $\varepsilon_\mathbf{B}(t)$

$$J = \sum_{t=1}^{N} \varphi^2(t)\,\varepsilon_\mathbf{A}^2(t) + \left(1-\varphi(t)\right)^2 \varepsilon_\mathbf{B}^2(t) \tag{5.93}$$

where, $\varphi(t)$ is solved by

$$\varphi(t) = \frac{\varepsilon_\mathbf{B}^2(t)}{\varepsilon_\mathbf{A}^2(t) + \varepsilon_\mathbf{B}^2(t)} \tag{5.94}$$

The new $\varphi(t)$ is then converted to 0 and 1 by Equation (5.91) and the SVM is solved again. Subsequently, **a** and **b** are re-estimated. The flowchart in Figure 5.16 illustrates the procedure to solve the SRP.

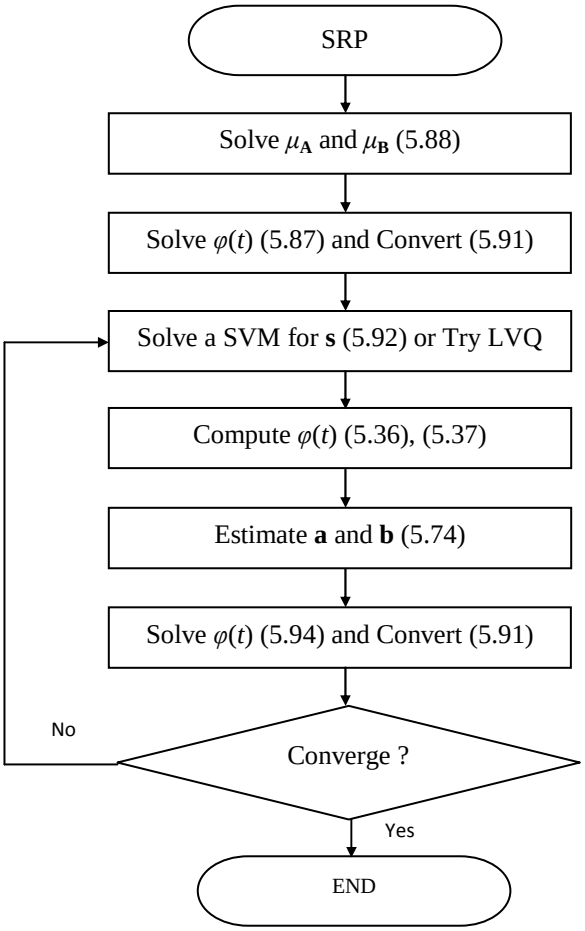


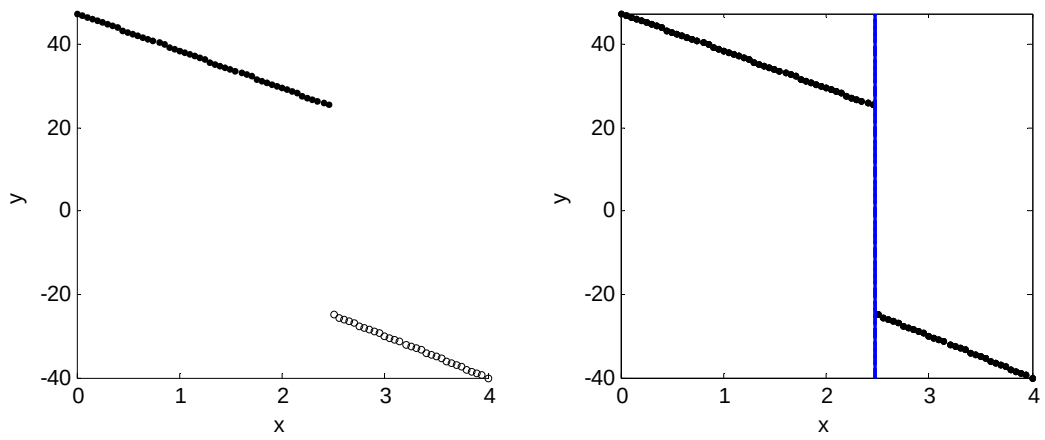
Figure 5.16. Flowchart for solving a SRP

5.5.4 Testing and Demonstration

The following examples are used to demonstrate how to implement the proposed technique to solve a SRP in practice. The first example is a piecewise linear function defined as below and illustrated in Figure 5.17(a)

$$y \equiv \begin{cases} y^1 = -9x + 47.5 & 0 \le x \le 2.5 \\ y^2 = -10x & 2.5 < x \le 4 \end{cases} \tag{5.95}$$

where, the separation is at x=2.5. Solving the problem of Equation (5.88), the solved μ_A and μ_B are -32.4915 and 36.5139 and the φ due to Equation (5.87) separate the function is shown in Figure 5.17(a), where dots and circles represent two different groups.



(a) (b)

Figure 5.17. a) Initialization of data segregation for Equation (5.95);
b) A linear separation boundary found for the initial data segregation

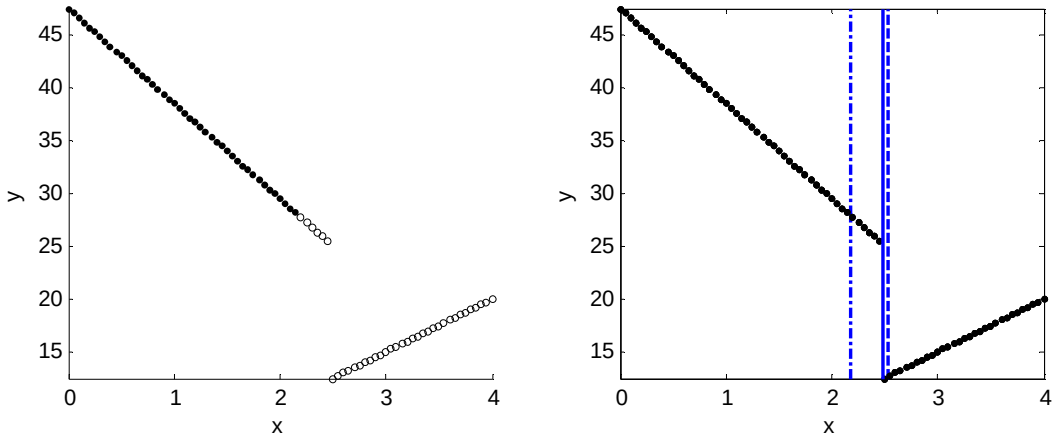
The initial separation is consistent with the piece-wise function. The resultant separation boundary is shown as the vertical line in Figure 5.17(b).This problem has a very particular piece-wise function, which has very distinct values in each region. The problem is actually solved at the first iteration. The initial data segregation is consistent with the underlying function nonlinearity.

The second example is defined by

$$y \equiv \begin{cases} y^1 = -9x + 47.5 & 0 \le x \le 2.5 \\ y^2 = -5x & 2.5 < x \le 4 \end{cases} \tag{5.96}$$

where, the difference to the first example is in the second linear function. The separation is also at x=2.5. The found optimal $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ of y, are 17.4742, 38.5195. The resultant segregation of data is shown in Figure 5.18(a), where the segregation is not totally consistent with the desired separation according to the function definition. 5 circles before x=2.5 should be dots. The misclassification illustrates the mismatch between an unsupervised learning and the desired classification. The first separation boundary by solving a SVM is shown as the dot-dashed vertical line (the leftmost one) in Figure 5.18(b), which separates circles from dots. Then two linear models are obtained. One of

linear models actually (dots) matches the true model exactly since dots are all resulted from one linear function. Residuals are computed after two linear models are obtained and the separation boundary is then updated, which is shown as the dashed vertical line (the rightmost one) in Figure 5.18(b). Clearly, it is closer to the desired solution at x=2.5 than the initial boundary. The dashed line resulted in a better separation and two better local models. Using the improved local models, residuals are updated, which in turn results in another step of improvement of separation boundary. The third separation is shown as the solid vertical line (middle one) in Figure 5.18(b). The solution is at x = -2.4757. In this simulation, further iteration results in no improvement. Actually, there are infinite number of global solutions between two margin points, the last point from the left line equation and first point from the right one. The resultant one is due to the SVM solution, which is expected to be robust with equal distance between two margin points.



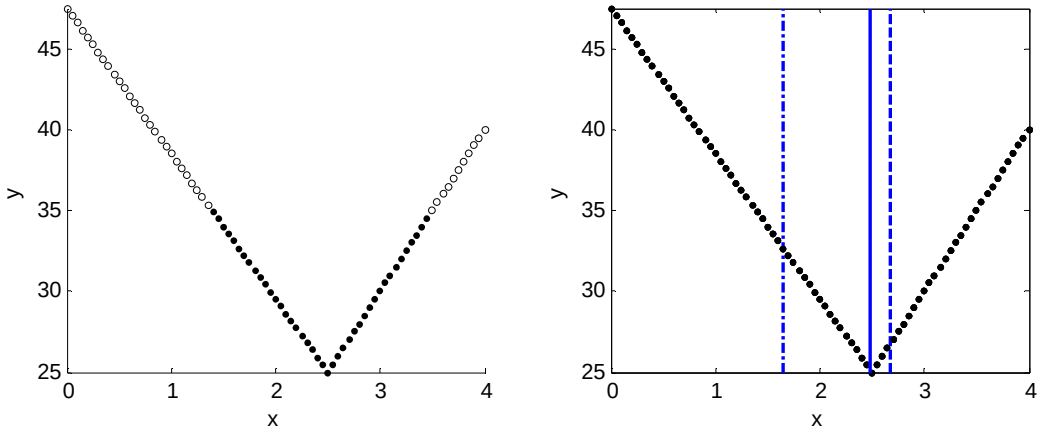
(a) (b)

Figure 5.18. a) Initialization of data segregation for Equation (5.96)
b) Initial linear boundary and its variation over iteration

The third example is more confusing at the initial step than the first two and defined by

$$y \equiv \begin{cases} y^1 = -9x + 47.5 & 0 \le x \le 2.5 \\ y^2 = -5x & 2.5 < x \le 4 \end{cases} \tag{5.97}$$

The initial separation is shown in Figure 5.19(a) with two recognized centers $\mu_{\mathbf{A}}$ = 35.1792 and $\mu_{\mathbf{B}}$ = 34.7254, which separates high value *y* from low values. However, the initial separation does not match the underlying nonlinearity in the piecewise function. The initial boundary solved is the dot-dashed line (the leftmost one) shown in Figure 5.19(b). Another iteration brings the separation boundary to the right of x=2.5 (the dashed line in Figure 5.19(b). The final separation boundary is shown as the solid vertical line at x=-2.4757 in Figure 5.19(b).



(a) (b)

Figure 5.19. a) Initialization of data segregation for Equation (5.97)
b) Initial linear boundary and its variation over iteration

Note that the nonlinear optimization problem in Equation (5.88) is subject to the initial guesses of $\mu_{\mathbf{A}}$ and $\mu_{\mathbf{B}}$ (a common problem for all nonlinear optimization problems). Figure 5.20 shows an initial separation due to estimated $\mu_{\mathbf{A}}(0)$ = 35.4836 and $\mu_{\mathbf{B}}(0)$ = 57.9810. It appears that a linear boundary might not be needed since all data points appear to belong to one category with only two dark dots are observed in the upper-left corner in Figure 5.20. Solving a SVM based on the initial categorization results in a trivial separation with s_0=-1, s_1 = 0, which means no separation. Clearly, the initial categorization of data is not consistent with the underlying nonlinear function.

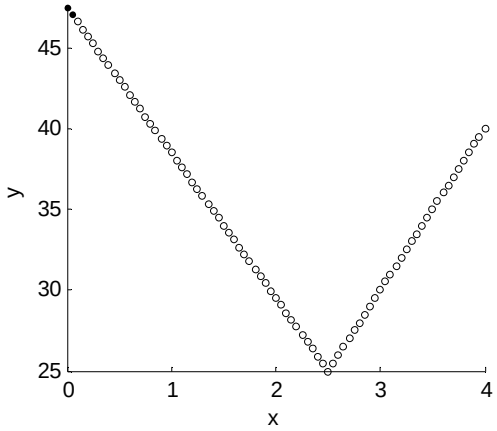


Figure 5.20. An initial data segregation for Equation (5.97) fails a SVM solver

As mentioned above, a LVQ based method will be applied when SVM fails. For this case, a linear quantization vector (LVQ) is solved to recognize some clusters in each category. The result is shown in Figure 5.21, where one cluster (star) is identified for the two solid dots and 16 clusters (triangles) are identified for the circles. Given the solved LVQ, the next step is to try all possible separation boundaries. One boundary at x=0.9160 defined by two clusters of x=0.0312 (star) and x=1.8 (triangle) is shown in Figure 5.21. In this case, 16 separation boundaries are tried (one star and 16 triangles).

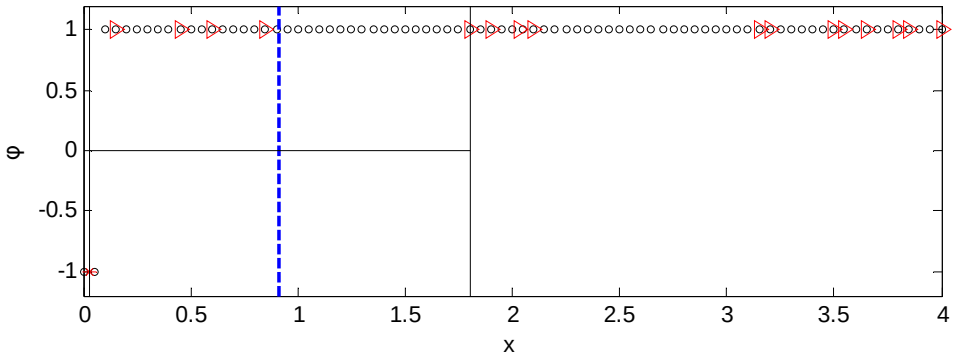


Figure 5.21. Clusters recognized using LVQ for the initial segregation in Figure 5.20

The best of 16 trials is shown as the dot-dashed vertical line (the leftmost) in Figure 5.22. The dashed and solid linear boundaries are obtained in the next two iterations. Convergence is obtained at x = -2.4757.

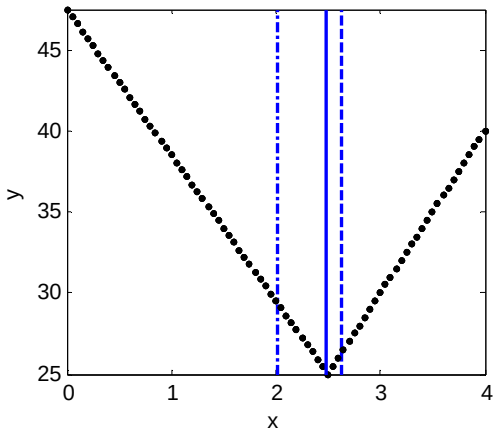


Figure 5.22. Initial boundary from clusters in Figure 5.21 and its variation in iterations

As shown in Figure 5.23 is a SRP applied to a linear piecewise function with three pieces. The resultant separation is the solid vertical line (the leftmost one) in Figure 5.23. One can imagine that subsequent steps will be to solve two SRPs for data on both sides of the first separation. Following the procedure, an antecedent space is progressively partitioned.

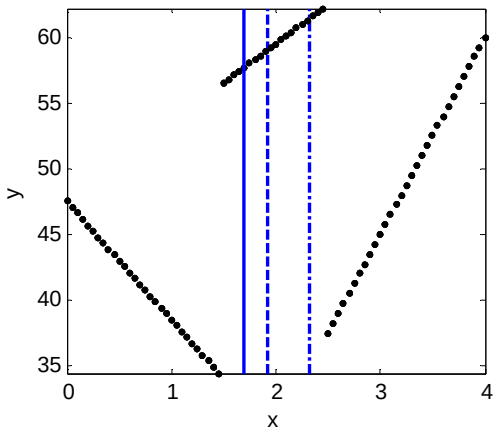


Figure 5.23. Liner boundary solved for a three-piece piecewise function

Figure 5.24 shows results for a quadratic function after eight iterations. Unlike the piece-wise linear functions, it is hard to tell how ‘optimal’ the solution is. Solutions for

the two-piece piecewise linear functions can be easily verified as global optimal solutions since obtained separation matches the separations defined in original functions.

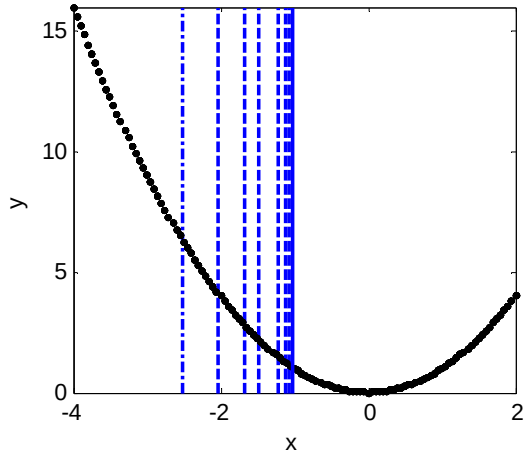


Figure 5.24. Linear boundary solved for a quadratic function

The sum of squared error (SSE) with respect to different separation boundary locations is shown in Figure 5.25 for the quadratic function. As shown, the optimization problem appears to have a 'global' minimum around 1, which matches the converged solution shown in Figure 5.24.

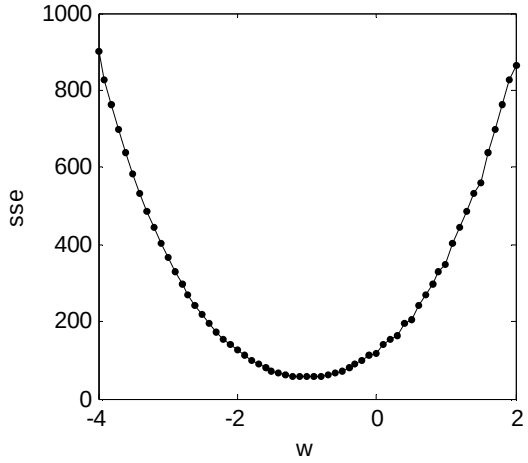


Figure 5.25. SSE with respect to the separation locations for the quadratic function

Figure 5.26 shows a one-period of Sin function, where the convergence is obtained at -2.5741. The solution is also shown in Figure 5.27 for the performance function (SSE) with respect to separation. The performance function is more complex than that in Figure 5.25. As shown in Figure 5.27, the resultant separation boundary is at the right edge of the valley of the performance function.

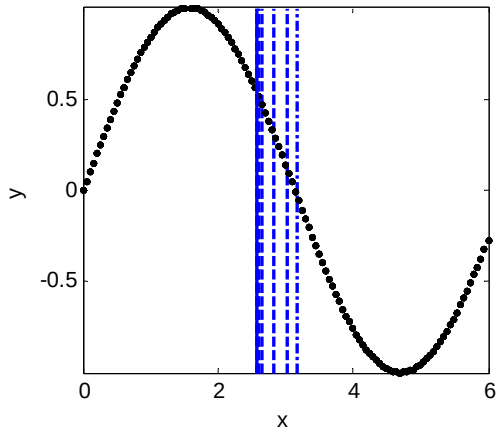


Figure 5.26. Initial linear boundary and its variation over iteration

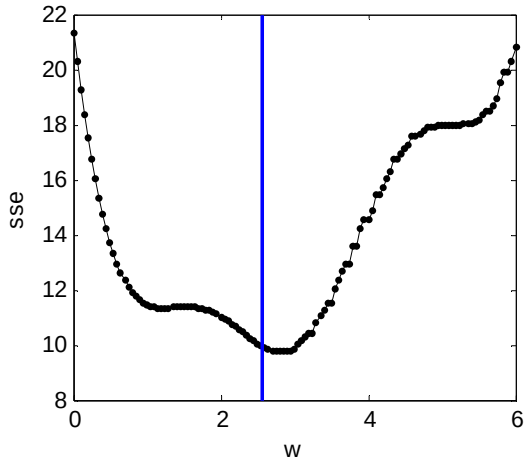


Figure 5.27. SSE with respect to the separation locations for Sin(x)

5.5.5 Comparison to Other Methods

In this section, the above mentioned SRP solving procedure is compared to two other methods. One is to use Newton's method to optimize the separation parameters, **s**, and local model parameters **a** and **b** simultaneously using a "soft" boundary. The other one is the Nelder-Mead method to search for separation parameters only. The following comparison is based on the piece wise linear function in Equation (5.97) and the function defined in Equation (5.98)

a. Newton's method to solve a SRP

The first and second order derivatives for using a Newton's method are described in Equations (5.43) and (5.50). One tuning factor is τ for the sharpness of a boundary, which has to be carefully chosen for a satisfactory result. The parameters, s_0, **a** and **b** are randomly initialized. In order to avoiding out-of-antecedent-space initial separation boundaries, the parameter s_1 is set such at the initial separation boundary location is at x=3.2040. The following gradients (**g**) and Hessian matrix (**H**) are evaluated for a very small τ = 1e-6 (τ = 0 will give indefinite evaluations numerically)

$$\mathbf{g} = \begin{bmatrix} -2110.3 & -2963.9 & -558.41 & -2032.4 & 0 & 0 \end{bmatrix}^T$$

$$\mathbf{H} = \begin{bmatrix} 65 & 104 & 0 & 0 & 0 & 0 \\ 104 & 223.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 58 & 0 & 0 \\ 0 & 0 & 58 & 211.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

where the gradients and Hessian matrix to **s** are all zeros, which verifies the derivations in Equations (5.62, 5.64, 5.65, 5.67 and 5.68). Therefore, separation parameters cannot be updated. The performance indexes over iterations are

J = [45745 215.05 215.04]

which implies that Newton's method converges after one step. The separation boundary is still at x=3.2024 and the improvement is achieved only by adjusting the local model parameters **a** and **b**. The procedure is same as to conduct two linear regressions on both sides of a random linear separation boundary. Although the boundary is far from the desired, the solution is still a local optimum.

The following **g** and **H** are evaluated at τ = 0.1, where evaluations for separation parameters become significant.

$$\mathbf{g} = \begin{bmatrix} -2105.2 & -2956.8 & -564.29 & -2041.2 & 3800.6 & 1179.6 \end{bmatrix}^T$$

$$\mathbf{H} = \begin{bmatrix} 62.58 & 96.573 & 1.9995 & 6.4058 & 1860.8 & 577.7 \\ 96.573 & 200.78 & 6.4058 & 20.588 & 6012.7 & 1860.8 \\ 1.9995 & 6.4058 & 14.421 & 52.615 & -1987.3 & -617.12 \\ 6.4058 & 20.588 & 52.615 & 192.74 & -6419.8 & -1987.3 \\ 1860.8 & 6012.7 & -1987.3 & -6419.8 & -13106 & -2892.3 \\ 577.7 & 1860.8 & -617.12 & -1987.3 & -2892.3 & -534.21 \end{bmatrix}^T$$

The performance index over iterations is shown in Figure 5.28 Newton's method converges after 5 iterations.

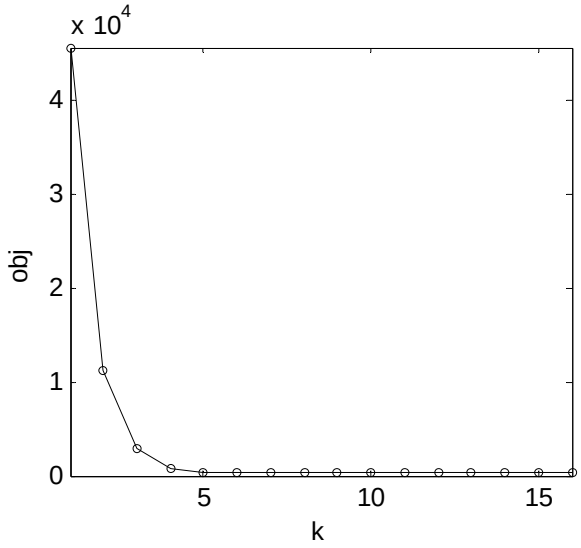


Figure 5.28. Objective function converges using Newton's method to solve a SRP

The separation boundary converges x=3.2680 and the final performance index is 250.9310, which is however larger than 215.04 due to τ =1e-6. Several tests are conducted on various τ values and the results are collected in Table 5.1. Figure 5.29 shows the converged objective function values with respect to τ. If an extra layer of optimization is introduced to optimize the scalar τ, Figure 5.29 implies that the optimization will be subject to local optimal solutions.

Table 5.1. Solution for a SRP using different τ values

τ	obj(final)	Separation boundary		τ	obj(final)	Separation boundary
1e-6	215.0415	3.2040		0.6	2.7419	2.6838
1e-3	215.0421	3.2000		0.7	0.0003	2.5062
0.01	181.7927	3.2000		0.8	0.0014	2.5004
0.1	250.9310	3.2680		0.9	28.4969	2.6215
0.2	155.3974	2.0837		1.0	46.8515	3.2997
0.3	3.8066	2.7486		1.5	8.5687	3.0385
0.4	1.3362	2.5848		2.0	9.6278	3.0246
0.5	16.3559	2.7133				

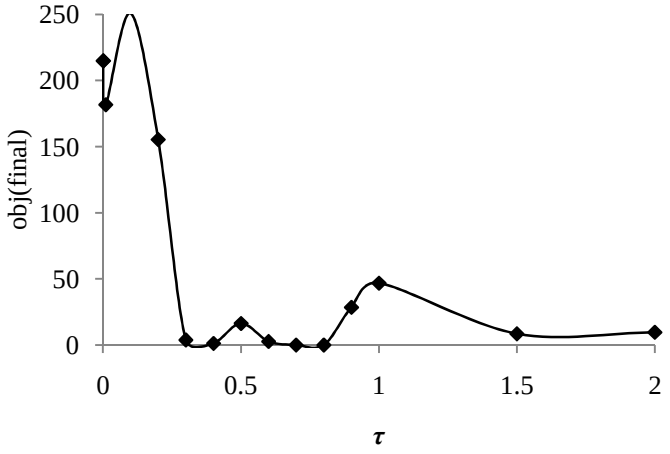


Figure 5.29. Converged objective function value with respect to τ

b. Nelder-Mead method to solve a SRP

A Nelder-Mead method searches for separation boundary parameters only. For each tried separation boundary, two local linear models are then estimated by least square regression. Figure 5.30 is the pseudo code for the Nelder-Mead to solve a SRP.

$\alpha = 1,\ \rho = 2,\ \gamma = 0.5,\ \sigma = 0.5,\ \varepsilon = 0.001$
While (1)
    / / ordering all the vertices
    $J(\mathbf{s}_1) \le J(\mathbf{s}_2) \le L \le J(\mathbf{s}_m)$
    IF $J(\mathbf{s}_m) - J(\mathbf{s}_1) \le \varepsilon$ Then
        return
    End IF
    / / compute the center of the best m-1 vertices
    $\mathbf{s}_0 = \frac{1}{m-1} \sum_{k=1}^{m-1} \mathbf{s}_k$
    //compute the refletion of the worst vertex to the center
        $\mathbf{s}_r = \mathbf{s}_0 + \alpha(\mathbf{s}_0 - \mathbf{s}_m)$
    IF $J(\mathbf{s}_1) < J(\mathbf{s}_r)$ And $J(\mathbf{s}_r) < J(\mathbf{s}_{m-1})$ Then
        replace $\mathbf{s}_m$ by $\mathbf{s}_r$
    Else IF $J(\mathbf{s}_r) < J(\mathbf{s}_1)$ Then
        / / compute the expansion vertex
        $\mathbf{s}_e = \mathbf{s}_0 + \gamma(\mathbf{s}_0 - \mathbf{s}_m)$
        IF $J(\mathbf{s}_e) < J(\mathbf{s}_r)$ Then
            replace $\mathbf{s}_m$ by $\mathbf{s}_e$
        Else
            replace $\mathbf{s}_m$ by $\mathbf{s}_r$
        End IF
    Else
        / / contraction
        $\mathbf{s}_c = \rho\mathbf{s}_r + (1-\rho)\mathbf{s}_0$
        IF $J(\mathbf{s}_c) \le J(\mathbf{s}_r)$ Then
            Replace $\mathbf{s}_m$ by $\mathbf{s}_c$
        Else
            //Shrink
            For each $\mathbf{s}_i$ $(i = 1, L\ , m)$
                $\mathbf{s}_i = \mathbf{s}_1 + \sigma(\mathbf{s}_i - \mathbf{s}_1)$
            End For
        End IF
    End IF
End While

Figure 5.30. Nelder-Mead algorithm to solve a SRP

where, m is the number of vertices and defined by nc+1.

Unlike the above mentioned Newton's method, the Nelder-Mead adjusts only separation parameters while the former optimizes both separation parameters and local model parameters at the same time. There are many factors affecting a Nelder-Mead method such as values for α, ρ and γ in the pseudo code. More importantly, the Neader-Mead method is also subject to initial guesses. Shown in Figure 5.31 is the performance index with respect to the location of a separation boundary. The performance is defined by SSE error reduction by having two local linear models

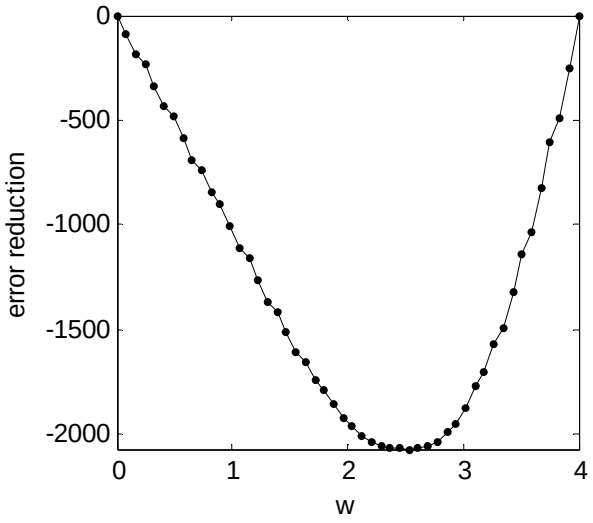


Figure 5.31. SSE with respect to the separation locations for Equation (5.97)

Figure 5.31 shows that the problem to be solved by the Nelder-Mead method has only one local optimal solution, which is also the global solution. As expected, the Nelder-Mead method should be able to locate the global optimal solution. Figure 5.32 shows 50 trials of the Nelder-Mead starting from random initial guesses, where global solution is found 48 times.

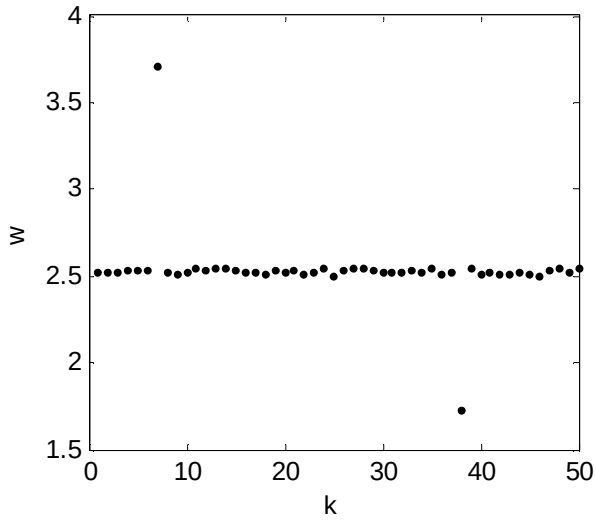


Figure 5.32. Separation locations for Equation (5.97) by Nelder-Mead method

The second function to be tried is defined as below (Zhang, Chen, Ansari & Shi, 2004) and plotted in Figure 5.33

$$y = (1-0.5x)\sin(2\pi x) + (1+0.4x)\cos(\pi x) + (1+0.1x)\sin(3\pi x); \quad 0 \le x \le 5 \qquad (5.98)$$

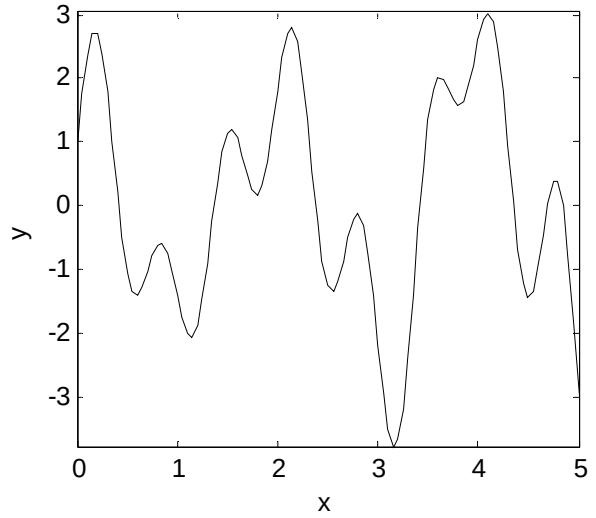


Figure 5.33. Illustration of the function in Equation (5.98)

The performance index with respect to the location of separation boundary is shown in Figure 5.34, where several local optimal solutions are observed

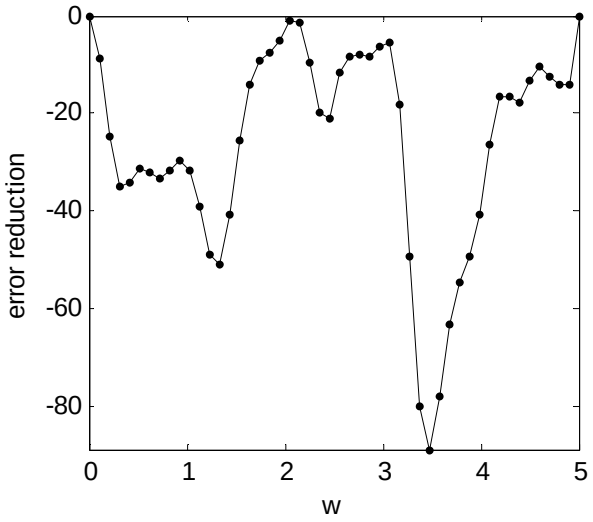


Figure 5.34. SSE with respect to separation locations for Equation (5.98)

Figure 5.35 shows the solutions obtained by the Nelder-Mead method out of 50 trials. Among them, 15 solutions are around the global solution at w=3.47.

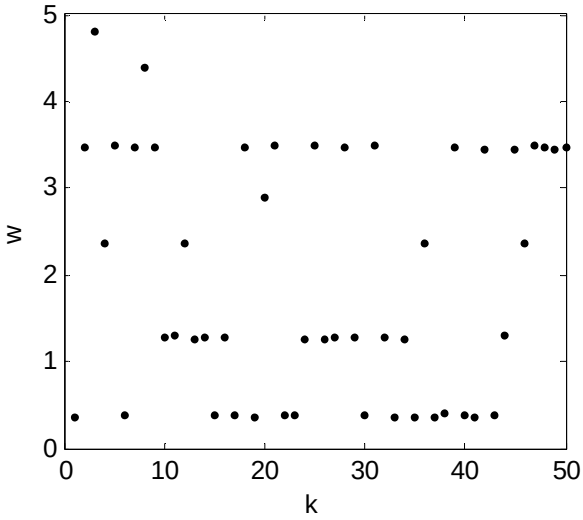


Figure 5.35. Separation locations for Equation (5.98) by Nelder-Mead method

The 50 trials by solving the SRP using the proposed procedure are shown in Figure 5.36. The scattering of solutions shown in Figure 5.35 is not observed in Figure 5.36. Instead, two groups of solutions could be visually recognized. There are 35 solutions around the global solution. The other 15 solutions concentrate around w = 3.7 and a little away from the global solution.

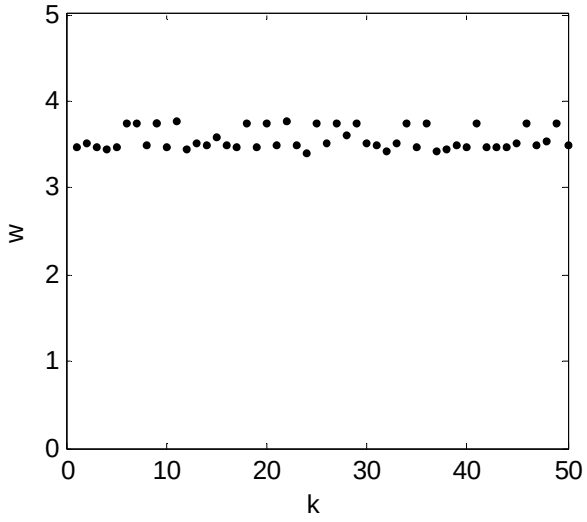


Figure 5.36. Separation locations for Equation (5.98) by the proposed SRP solver

As a conclusion, the proposed solving procedure for a SRP is more robust and problem independent. The Newton's method depends the 'sharpness' factor, τ, whose impact on the algorithm is shown complex. Direct search methods such as the Nelder-Mead method are subject to algorithm configurations and could get trapped by local optimal than the proposed SRP solver.

5.6 Extension to Multiple-Output Processes

Readers might choose to skip this section and come back for details when dealing with MIMO models.

The above SRP is for single-output models. Several functions need to be extended for models with multiple outputs. One of them is the performance index in Equation (5.85), which is redefined for multiple outputs by

$$\underset{\mu_{\mathbf{A}},\mu_{\mathbf{B}},\varphi(t)}{\text{minmize}}\ J=\sum_{t=1}^{N}\varphi^2(t)\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{A}}\right)^T\mathbf{R}_{\mathbf{A}}\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{A}}\right)+\left(1-\varphi(t)\right)^2\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)^T\mathbf{R}_{\mathbf{B}}\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)$$

subject to

$$\varphi(t)=0,1; \qquad t=1,\mathrm{L}\ ,N \tag{5.99}$$

where scalar $y(t)$ is replaced by a vector $\mathbf{y}(t)$ with dimension of n. Scalars of $\boldsymbol{\mu}_{\mathbf{A}}$ and $\boldsymbol{\mu}_{\mathbf{B}}$ are also replaced by their vector versions. Two diagonal weighting matrices $\mathbf{R}_{\mathbf{A}}$ and $\mathbf{R}_{\mathbf{B}}$ are introduced to adjust the scale of each output in each group (all weights are positive numbers).

$\varphi(t)$ in Equation (5.87) is then solved by

$$\varphi(t)=\frac{\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)^T\mathbf{R}_{\mathbf{B}}\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)}{\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{A}}\right)^T\mathbf{R}_{\mathbf{A}}\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{A}}\right)+\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)^T\mathbf{R}_{\mathbf{B}}\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)} \tag{5.100}$$

and described by

$$\varphi(t)=\frac{E_B}{E_A+E_B} \tag{5.101}$$

with

$$E_{\mathbf{A}}=\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{A}}\right)^T\mathbf{R}_{\mathbf{A}}\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{A}}\right)$$
$$E_{\mathbf{B}}=\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)^T\mathbf{R}_{\mathbf{B}}\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)$$

It can also be verified that $\varphi(t)$ in Equation (5.100) minimizes the objective funtion in Equation (5.99). With the definition of $\varphi(t)$ by $\boldsymbol{\mu}_{\mathbf{A}}$ and $\boldsymbol{\mu}_{\mathbf{B}}$, the optimization problem is converted to a problem with only decision variables of $\boldsymbol{\mu}_{\mathbf{A}}$ and $\boldsymbol{\mu}_{\mathbf{B}}$. The first-order derivatives of J to $\boldsymbol{\mu}_{\mathbf{A}}$ and $\boldsymbol{\mu}_{\mathbf{B}}$ are described by

$$\frac{\partial J}{\partial \boldsymbol{\mu}_{\mathbf{A}}} = \sum_{t=1}^{N} 2\varphi(t)\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{A}}} E_A + \varphi^2(t)\frac{\partial E_A}{\partial \boldsymbol{\mu}_{\mathbf{A}}} - 2\left(1-\varphi(t)\right)\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{A}}} E_{\mathbf{B}}$$
$$\frac{\partial J}{\partial \boldsymbol{\mu}_{\mathbf{B}}} = \sum_{t=1}^{N} 2\varphi(t)\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{B}}} E_A + \left(1-\varphi(t)\right)^2\frac{\partial E_{\mathbf{B}}}{\partial \boldsymbol{\mu}_{\mathbf{B}}} - 2\left(1-\varphi(t)\right)\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{B}}} E_{\mathbf{B}} \tag{5.102}$$

The second-order derivatives are described by

$$\frac{\partial^2 J}{\partial \boldsymbol{\mu}_{\mathbf{A}}^2} = \sum_{t=1}^{N} 2\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{A}}}\left(\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{A}}}\right)^T \left(E_{\mathbf{A}} + E_{\mathbf{B}}\right) + 4\varphi(t)\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{A}}}\left(\frac{\partial E_{\mathbf{B}}}{\partial \boldsymbol{\mu}_{\mathbf{A}}}\right)^T + 4E_{\mathbf{B}}\left[\frac{\varphi(t)E_{\mathbf{A}} - \left(1-\varphi(t)\right)E_{\mathbf{B}}}{\left(E_{\mathbf{A}} + E_{\mathbf{B}}\right)^2}\right] R_{\mathbf{A}}$$
$$+16E_{\mathbf{B}}\frac{\varphi(t)E_{\mathbf{A}} + \varphi(t)E_{\mathbf{B}} - E_{\mathbf{B}}}{\left(E_{\mathbf{A}} + E_{\mathbf{B}}\right)^3} R_{\mathbf{A}}\left(y(t)-\boldsymbol{\mu}_{\mathbf{A}}\right)\left(y(t)-\boldsymbol{\mu}_{\mathbf{A}}\right)^T R_{\mathbf{A}}{}^T + 2R_{\mathbf{A}}\varphi^2(t) \tag{5.103}$$

$$\frac{\partial^2 J}{\partial \boldsymbol{\mu}_{\mathbf{B}}^2} = \sum_{t=1}^{N} 2\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{B}}}\left(\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{B}}}\right)^T \left(E_{\mathbf{A}} + E_{\mathbf{B}}\right) + 4\left(1-\varphi(t)\right)\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{B}}}\left(\frac{\partial E_{\mathbf{B}}}{\partial \boldsymbol{\mu}_{\mathbf{B}}}\right)^T + 4E_{\mathbf{A}}\left[\frac{\varphi(t)E_{\mathbf{A}} - \left(1-\varphi(t)\right)E_{\mathbf{B}}}{\left(E_{\mathbf{A}} + E_{\mathbf{B}}\right)^2}\right] R_{\mathbf{B}}$$
$$+16E_{\mathbf{A}}\frac{E_{\mathbf{B}} - \varphi(t)E_{\mathbf{A}} - \varphi(t)E_{\mathbf{B}}}{\left(E_{\mathbf{A}} + E_{\mathbf{B}}\right)^3} R_{\mathbf{B}}\left(y(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)\left(y(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)^T R_{\mathbf{B}}{}^T + 2R_{\mathbf{B}}\left(1-\varphi(t)\right)^2 \tag{5.104}$$

$$\frac{\partial^2 J}{\partial \boldsymbol{\mu}_{\mathbf{A}}\partial \boldsymbol{\mu}_{\mathbf{B}}} = \sum_{t=1}^{N} 2\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{A}}}\left(\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{B}}}\right)^T \left(E_{\mathbf{A}} + E_{\mathbf{B}}\right) + 2\varphi(t)\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{B}}}\left(\frac{\partial E_{\mathbf{A}}}{\partial \boldsymbol{\mu}_{\mathbf{A}}}\right)^T - 2\left(1-\varphi(t)\right)\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{A}}}\left(\frac{\partial E_{\mathbf{B}}}{\partial \boldsymbol{\mu}_{\mathbf{A}}}\right)^T$$
$$+\left[8\frac{\left(1-\varphi(t)\right)E_{\mathbf{B}} - \varphi(t)E_{\mathbf{A}}}{\left(E_{\mathbf{A}} + E_{\mathbf{B}}\right)^2} + 16\frac{\varphi(t)E_{\mathbf{A}}E_{\mathbf{B}} - \left(1-\varphi(t)\right)E_{\mathbf{B}}^2}{\left(E_{\mathbf{A}} + E_{\mathbf{B}}\right)^3}\right] R_{\mathbf{A}}\left(y(t)-\boldsymbol{\mu}_{\mathbf{A}}\right)\left(y(t)-\boldsymbol{\mu}_{\mathbf{B}}\right)^T R_{\mathbf{B}}{}^T \tag{5.105}$$

where, the derivatives of $E_{\mathbf{A}}$ and $E_{\mathbf{B}}$ to μ_A and $\mu_{\mathbf{B}}$ are defined respectively

$$\frac{\partial E_A}{\partial \boldsymbol{\mu}_{\mathbf{A}}} = -2\mathbf{R}_1\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{A}}\right)$$
$$\frac{\partial E_B}{\partial \boldsymbol{\mu}_B} = -2\mathbf{R}_2\left(\mathbf{y}(t)-\boldsymbol{\mu}_{\mathbf{B}}\right) \tag{5.106}$$

and the derivatives of $\varphi(t)$ to $\boldsymbol{\mu}_{\mathbf{A}}$ and $\boldsymbol{\mu}_{\mathbf{B}}$ are defined by

$$\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{A}}} = \frac{-E_{\mathbf{B}}}{\left(E_{\mathbf{A}} + E_{\mathbf{B}}\right)^2} \frac{\partial E_A}{\partial \boldsymbol{\mu}_{\mathbf{A}}}$$
$$\frac{\partial \varphi(t)}{\partial \boldsymbol{\mu}_{\mathbf{B}}} = \frac{E_{\mathbf{B}}}{\left(E_{\mathbf{A}} + E_{\mathbf{B}}\right)^2} \frac{\partial E_{\mathbf{B}}}{\partial \boldsymbol{\mu}_{\mathbf{B}}} \tag{5.107}$$

The problem is non-convex. The same optimizer using the revised Newton's method for non-definite quadratic problems is used.

5.7 Recursive Partition by Growing a Binary Tree

The above mentioned SRP finds a linear separation boundary. A tree growth procedure is defined to recursively solve SRPs in obtained regions, which at the end defines a partition in an antecedent space. The procedure should end when stopping criteria are satisfied. As shown in Figure 5.23, it is clear to observe that one more SRPs on either side of the first separation boundary is required to complete the partition. Then, the growth procedure stops when modeling error is zero. The simple stopping criterion is only suitable for a piece-wise linear model. For a nonlinear model as shown in Figure 5.24, the tree growth cannot be stopped by the zero-modeling-error stopping criterion given sufficient number of data points in each region for parameter estimation. Practically, the growth has to be stopped at least for the minimal number of data points in a region to estimate local model parameters.

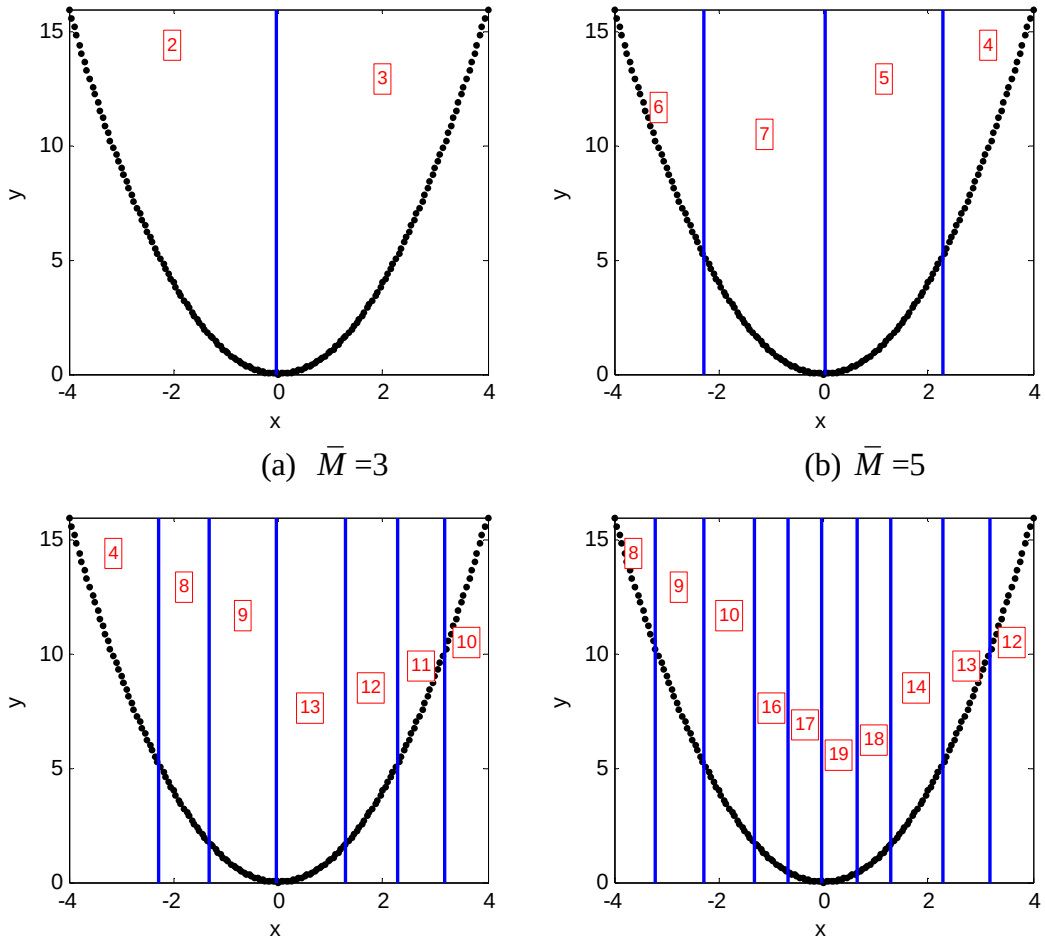
In this work, a scalar α_M is used to determine if a splitting is acceptable. The threshold number determines the minimum number of data points in a region. A splitting is rejected if either resultant region contains less than α_M data points. The threshold number is not directly set by users but resulted from a predefined number, $\bar{M}$.

$$\alpha_M = \frac{N}{\bar{M}} \tag{5.108}$$

where N is the number of data points and $\bar{M}$ could be roughly interpreted as the anticipated maximum number of regions (rules). It is expected that the number $\bar{M}$ is

more relevant for users' anticipation of the modeling complexity, number of rules. Lack of fit should be expected if the $\bar{M}$ is chosen too small while a too large $\bar{M}$ will result in over fit. Trials could be taken to find a suitable $\bar{M}$. More aggressively, a linear search could be conducted to find an optimal $\bar{M}$.

Different $\bar{M}$ are tried for function $y=x^2$ over [-4,4]. The results for $\bar{M}$ = 3,5,10 and 15 are shown in Figure 5.37. Table 5.2 collects the number of regions and SSE for each $\bar{M}$. Without any split, the SSE is 3755.37. The reduction rate of SSE is 93.72 % at $\bar{M}$ = 3 with one split. Another 5.81% improvement is achieved at $\bar{M}$ = 5 with another two splits. 0.38% improvement is gained at $\bar{M}$ = 10 with another three splits. Trials could be made for different $\bar{M}$.



(a) $\bar{M}$ =3 (b) $\bar{M}$ =5

(c) $\bar{M}$ =10 (d) $\bar{M}$ =15

Figure 5.37. Antecedent partition using different $\bar{M}$

Table 5.2. The number of rules and SSE resulted from different $\bar{M}$

$\bar{M}$	3	5	10	15	20
M	2	4	7	10	14
SSE	235.67	17.16	2.94	0.9428	0.1526

In this work, $\bar{M}$ is not searched. Instead, a large $\bar{M}$ is chosen on purpose, which might result in a 'large' model with 'too' many rules'. Then a tree trim procedure is conducted to cut off unnecessary tree branches to reduce model complexity.

5.8 Removal of Insignificant Partitions by Trimming a Tree

As mentioned above, an appropriate $\bar{M}$ is needed to generate a suitable size GTSK model with reasonable number of rules. Trials could be made to find a proper $\bar{M}$. In this work, $\bar{M}$ is not tried. Instead, a 'large' $\bar{M}$ is used, which will purposely over-partition an antecedent space. By doing that, the problem to be solved can only be over-fitting but not under-fitting. Subsequently, some regions in the over-partitioned spaces are merged via removing some unnecessary boundaries, which has the least model improvement per model complexity efficiency. Therefore, the under-fitting and over-fitting are addressed in two stages.

Using a large $\bar{M}$ could also be considered as an attempt to find a 'global' solution out of one obtained in a step-wise manner. Ideally, the partition problem should be solved by considering all separation boundaries together in order to get a global optimal partition in terms of both modeling complexity and errors. Rather than attempting to solve such a difficulty problem, the recursive procedure in this work is to solve a separation a time. Together, separations from each step build up the solution. The resultant solution is a step-wise partition, which is expected to be different from a solution obtained from 'global' procedure if it ever exists. If a large $\bar{M}$ is used, it is hoped that the resultant step-wise solution contains a 'global' solution. If considering a tree structure, the 'global' optimal tree is contained in the excessively large tree due to a large $\bar{M}$. The problem

remains to be solved is to find the 'global' tree by removing unnecessary branches and leaves from the 'big' tree.

A tree trim procedure is then operated to remove unnecessary branches. Branches to be removed should have low model improvement per model complexity efficiency. As shown in Figure 5.38(a), there are three branches with branch nodes t_2, t_3 and t_7. A branch is denoted by Bt, for instance, branch Bt$_3$ extracted from Figure 5.38(a) is shown in Figure 5.5.

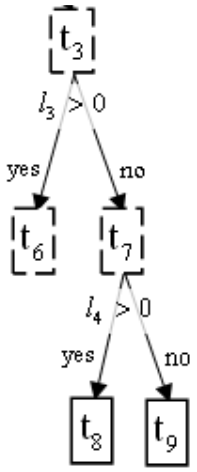


Figure 5.38. The branch Bt$_3$ from Figure 5.5(a)

A branch Bt is defined as a set of leaf nodes that are decedents of Bt. For instance, Bt$_3$ in Figure 5.38 is defined by $Bt_3 = [t_6, t_8, t_9]$.

At the tree-growth stage, the node t_3 is split into two nodes t_6 and t_7. The split is accepted if the modeling error is reduced, and t_6 and t_7 contain sufficient amount of data points. Therefore, the comparison is only made between t_3 and its two immediate decedents. The comparison can be extended to include later generation decedents. As shown in Figure 5.38, node t_3 is split into 3 leaf nodes. An extended comparison could be made to evaluate if the split of t_3 to $[t_6,t_8,t_9]$ is necessary. However, the extended comparison is only applicable when the branch Bt_3 for t_3 is known. It is why the following procedure is implemented after the tree-growth procedure is finished.

A performance index for a branch *B*t is defined by

$$R(Bt) = \sum_{\tau \in Bt} R(\tau) \tag{5.109}$$

Where $R(\tau)$ is the SSE of the local linear model for the node, τ. Its regularization with considering model complexity is defined by

$$R_\alpha(Bt) = R(Bt) + \alpha|Bt| \tag{5.110}$$

where |*B*t| represents the complexity of branch *Bt*. A regularization performance index is also defined for the branch node t

$$R_\alpha(t) = R(t) + \alpha|t| \tag{5.111}$$

where |t| is the complexity for the model to the node t. In this work, the complexity is simply defined as the number models in a branch. Therefore, |t| is always 1 since it contains only one model while |Bt| is the number of leaf nodes.

The branch *B*t will be kept (all splits are accepted) if $R_\alpha(Bt) < R_a(t)$. The inequality however depends on α, which reaches a critical α, α_c(t), when $R_\alpha(Bt) = R_a(t)$. The variable α_c (t) is hence defined by (5.112)

$$\alpha_c(t) = \begin{cases} \dfrac{R(t) - R(Bt)}{|Bt| - |t|}, & t \text{ is a branch node} \\ \infty, & t \text{ is a leaf or root node} \end{cases} \tag{5.112}$$

The critical value α_c(t) hence reveals the performance improvement per complexity increment efficiency for the branch node, *t*. Clearly, larger α_c(t) is preferred and less efficient branch should be removed. At every step, α_c for all branch nodes are computed. The branch node with the minimum α_c is defined by t_p

$$a_c\left(t_p\right)=\min_{t\in Bt_1}\frac{R(t)-R(Bt)}{|Bt|-|t|} \tag{5.113}$$

where, Bt_1 is a abuse of the branch notation and represents the entire tree (a branch from the root node). The branch Bt_p is then hypothetically removed. Then, α_c is reevaluated for all left branch nodes, and another t_p is found and hypothetically removed. The procedure continues until the root node, t_1, is reached. It is shown that $\alpha_c(t_p)$ value will be monotonically decreasing (Breiman, Friedman, Olshen & Stone, 1984), which implies that worse branches are removed first and the removal sequence is optimal.

The above procedure will generate a sequence of $\alpha_c(t_p)$, which is the minimum in each step. Nodes with α_c (t_p) lower than a threshold number will be actually removed and a trimmed tree is then obtained. In this work, the threshold number is tried for an appropriate level of complexity.

The corresponding tree structure for the Figure 5.37(c) is shown in Figure 5.39, where the number under each box is the sum of squared error and solid boxes are for leaf nodes.

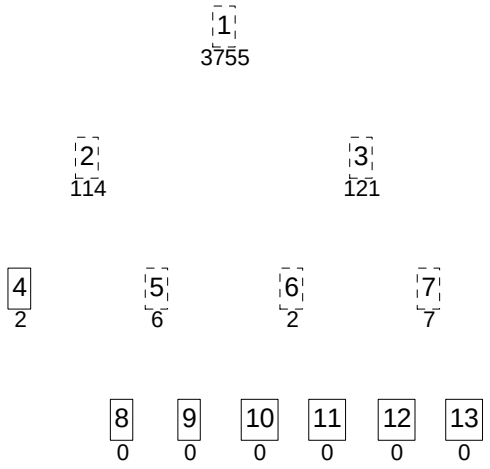


Figure 5.39. The tree structure for the antecedent partition in Figure 5.37 (c)

Table 5.3. The value α_c for branch nodes shown in Figure 5.39

α_c	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}
1	∞	55.69	40.12	∞	5.90	1.70	6.62	∞	∞	∞	∞	∞	∞
2	∞	55.69	59.32	∞	5.90	∞	6.62	∞	∞	∞	∞	∞	∞
3	∞	105.48	59.32	∞	∞	∞	6.62	∞	∞	∞	∞	∞	∞
4	∞	105.48	112.03	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	112.03	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

As shown in Figure 5.39, there are 5 branch nodes, t_2, t_3, t_5, t_6 and t_7. At the first step, the minimum α_c is found for t_6 with 1.7032. Then branch Bt_6 is hypothetically removed. The removal is simply operated by changing the branch node t_6 to a leaf node. At next step, the branch node with minimum α_c is t_5 with 5.8971. The procedure continues until all branch nodes are hypothetically removed. The largest α_c is 112.03 and its 10%, 11.20 is set as the threshold number to remove insignificant branch nodes. In this example, branches underneath branch nodes t_5, t_6, and t_7 will be permanently removed. The resultant trimmed tree is shown in Figure 5.40(a). The corresponding splitting is shown in Figure 5.40(b), where light-colored vertical lines represented removed splits. The result is same as that shown in Figure 5.41(b) with $\bar{M} = 5$.

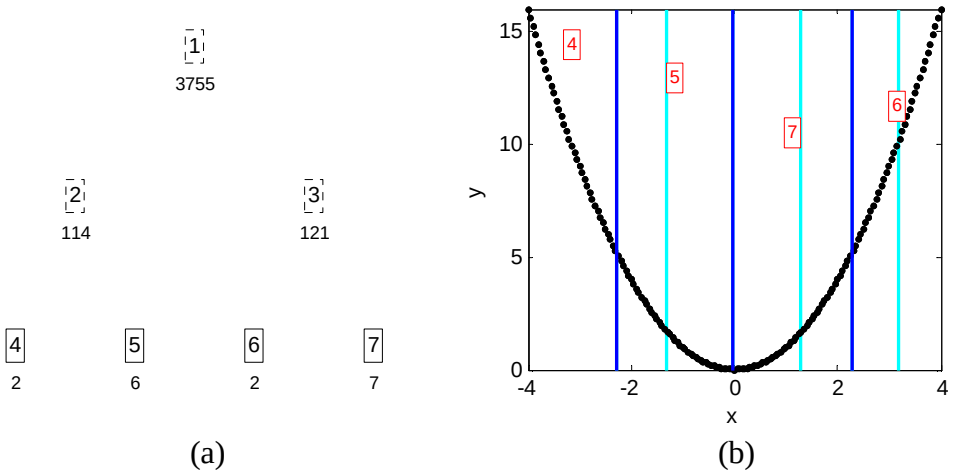


Figure 5.40. Antecedent space partition after removing splits under branch nodes t_5, t_6 and t_7 in Figure 5.39; light lines represent removed splits

The results for trimmed trees due to $\bar{M}$ = 15 and 20 are also shown in Figure 5.41 for comparison, where the threshold is also set as the 10% of the largest α_c. It is observed that trimmed trees are identical regardless the value of $\bar{M}$. Therefore, an excessive large $\bar{M}$ could be used to generate a large tree and a tree-trim procedure is used to remove unnecessary branches. Certainly, more computation is needed for generating a bigger tree, which however gives a better chance to contain an 'optimal' tree.

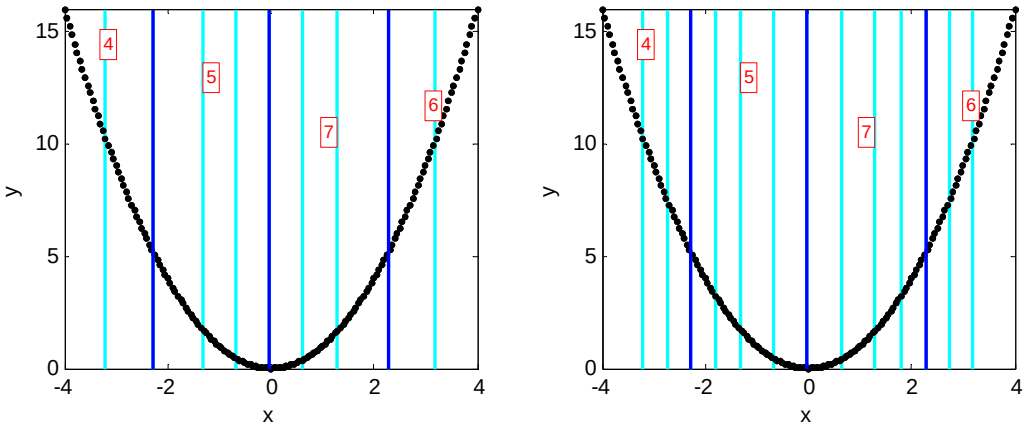


Figure 5.41. Antecedent space partitions after remove some unimportant splits (light lines) for a) Figure 5.37 (c); b) Figure 5.37 (d)

5.9 Rule Antecedent Parameter Estimation

The tree growth procedure generates a number of separation boundaries that partition the antecedent space. Given a partitioned antecedent space, there are many views on recognizing a local region. One way is to consider the local region as a polyhedron consisting of several separation boundaries. Another way is to consider the local region to be a set of points. Each way has its corresponding methods to identify centers and ellipsoids. Within a polyhedron, a maximum volume ellipsoid could be found. A minimum volume ellipsoid could be found containing a set of points. Both problems can be solved efficiently by convex optimization (Boyd & Vandenberghe, 2004). A dynamic search approach in (Pronzato, Wynn & Zhigljabsky, 2000) can also be used to

identify ellipsoids.

The above mentioned techniques are sound choices. However, and perhaps unnecessarily, this work also considers the quality of each data point. The quality is related to the prediction error for each data sample. For instance, the solid dots in Figure 5.42 represent data points with small residuals while the circles represent data points with larger residuals.

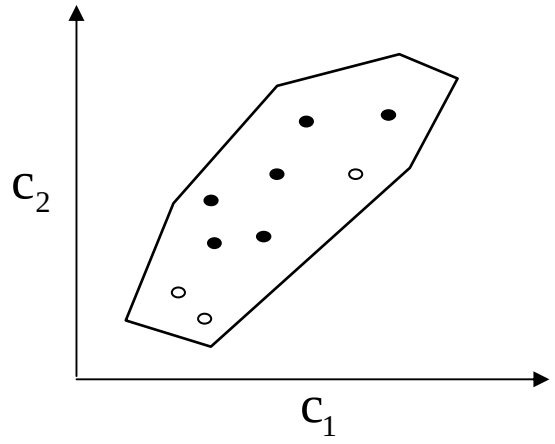


Figure 5.42. A local region in an antecedent space; dark dots represent data points with smaller residuals while circles represent points with higher residuals

A rule antecedent in fact represents the region where the consequent model is expected to be accurate. It is then reasonable to use only data samples with smaller residuals to estimate the antecedent parameters. There are many approaches for weighting the importance of data. This work uses a simple approach, where weighting is defined by the residual

$$\beta_i^r = \exp\left(-\frac{N^r\left(\varepsilon_i^r\right)^2}{\left(\boldsymbol{\varepsilon}^r\right)^T\boldsymbol{\varepsilon}}\right) \tag{5.114}$$

where N^r is the number of data points in region r. The script (r,i) represents the i^{th} data in region r. β_i^r reaches the highest value at 1 when ε_i^r is zero.

The centroid $\mathbf{o}^r$ is estimated by

$$\mathbf{o}^r = \frac{\sum_{i=1}^{N^r} \beta_i^r \mathbf{c}_i^r}{\sum_{i=1}^{N^r} \beta_i^r} \tag{5.115}$$

and the matrix $\mathbf{P}^r$ is defined by its inverse

$$\left(\mathrm{P}^r\right)^{-1} = \frac{\sum_{i=1}^{N^r} \beta_i^r \left(\mathbf{c}_i^r - \mathbf{o}^r\right)\left(\mathbf{c}_i^r - \mathbf{o}^r\right)^T}{\sum_{i=1}^{N^r} \beta_i^r} \tag{5.116}$$

CHAPTER VI

RESULTS FOR TESTING PROBLEMS

The objective of Chapter 6 is to test the proposed procedure to create GTSK models for function approximation in Section 6.1 and nonlinear dynamic process modeling in Section 6.2. The GTSK models to be created use the generalized antecedent structure proposed in Chapter 3. In modeling nonlinear dynamic processes, the dimension of a GTSK model (both antecedent and consequent dimensions) is specified by the determined dynamic orders and detected nonlinear components in Chapter 4. The model parameters are determined by parameter estimation procedure presented in Chapter 5.

6.1 Function approximation

Function 1

The first function to be approximated is defined by

$$y = 3x(x-1)(x-1.9)(x+0.7)(x+1.8), \qquad -2.1 \le x \le 2.1 \qquad (6.1)$$

Function 1 is used in (Dickerson & Kosko, 1996) as a primary example to demonstrate a function approximation procedure using GTSK models. The procedure starts initializing membership functions for both *x* and *y* by projecting recognized ellipsoidal patches onto *x*-*y* coordinates. The patch reorganization is an unsupervised learning procedure. Following the heuristic initialization, model parameters are refined using a steepest decent optimizer. The algorithm in (Dickerson & Kosko, 1996) works fine for Function 1. As demonstrated in Section 5.4.2, unsupervised learning might result in inappropriate initialization since it uses measures based on data distribution rather on nonlinearity.

Since the function has only one input, it should be included in both antecedent and consequent. There are 412 points uniformly sampled from the function. The scalar $\bar{M}$ in Equation (5.108) is set to 50, which implies that a region should no longer be split if it contains less than 412/50 ≈ 8 data points.

With the above configuration, 30 branch nodes are generated, each of which is associated with an efficiency index, α_c defined in Equation (5.113). Values of α_c for all branch nodes are shown in Figure 6.1.

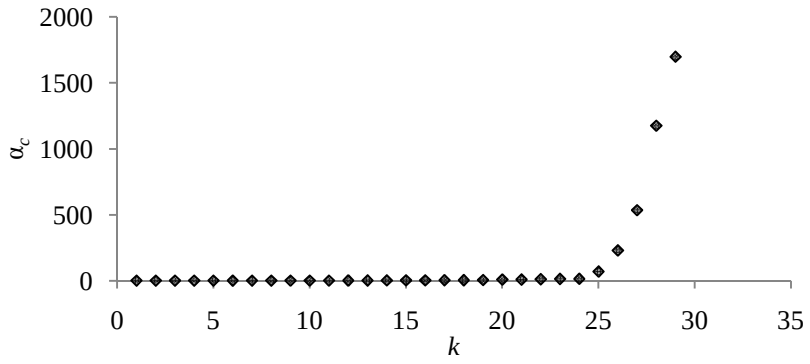


Figure 6.1. Values of α_c for antecedent space partition for Equation (6.1)

At this point, it should be subject to users' judgment to select an approximate value level in α_c to discard unimportant splits. In this testing, we choose to keep first 5 branch nodes. Among them, the lowest α_c value is 68.70, to which the next lower α_c value is 13.80. The resultant antecedent space partition is shown in Figure 6.2, which also shows the membership function initialization for an 8-rule GTSK model. The membership functions are initialized using Equations (5.115) and (5.116)

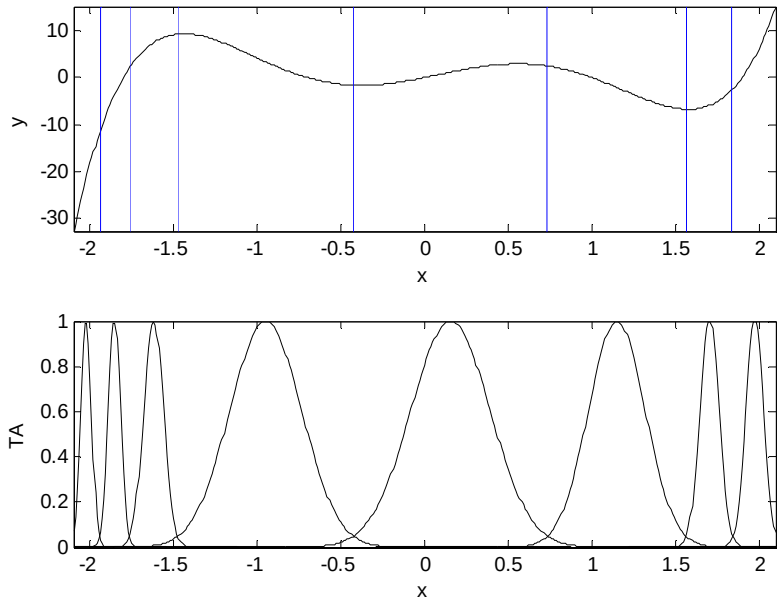


Figure 6.2. Antecedent space partition and TAs based on Equation (6.1)

The initialized GTSK model is fully described in Equation (6.2)

$$
\begin{aligned}
&\textbf{IF } \left(x \text{ is in } R^1\left(-2.0, 470.6\right)\right) \textbf{ THEN } y^1 = 133.59x + 248.59 \\
&\textbf{IF } \left(x \text{ is in } R^2\left(-1.9, 371.5\right)\right) \textbf{ THEN } y^2 = 76.37x + 137.48 \\
&\textbf{IF } \left(x \text{ is in } R^3\left(-1.6, 142.9\right)\right) \textbf{ THEN } y^3 = 24.53x + 46.57 \\
&\textbf{IF } \left(x \text{ is in } R^4\left(-1.0, 10.9\right)\right) \textbf{ THEN } y^4 = -12.65x - 8.34 \\
&\textbf{IF } \left(x \text{ is in } R^5\left(0.2, 8.9\right)\right) \textbf{ THEN } y^5 = 4.97x + 0.12 \\
&\textbf{IF } \left(x \text{ is in } R^6\left(1.2, 17.4\right)\right) \textbf{ THEN } y^6 = -12.70x + 12.42 \\
&\textbf{IF } \left(x \text{ is in } R^7\left(1.7, 153.3\right)\right) \textbf{ THEN } y^7 = 15.10x - 31.23 \\
&\textbf{IF } \left(x \text{ is in } R^8\left(2.0, 177.8\right)\right) \textbf{ THEN } y^8 = 68.57x - 130.30
\end{aligned}
\tag{6.2}
$$

where, R^1(-2.0,470.6) defines the region for the first rule (the leftmost in Figure 6.2) with o^1 = -2.0 and P^1 = 470.6. Both **o** and **P** are introduced in Equation (3.7). In the first rule, the linear consequent model is, y^1 = 133.59x+248.59. Note that the linear consequent model might not necessarily represent the local behavior of the original nonlinear

function due to linearization. The interpretation of linear consequent models depends on the interactions in rules and is discussed later in detail

The GTSK model is then used to approximate the function. The approximation is shown as dashed line in Figure 6.3. The mean squared error (MSE) for the approximation is 0.21, which is lower than that mentioned in (Dickerson & Kosko, 1996).

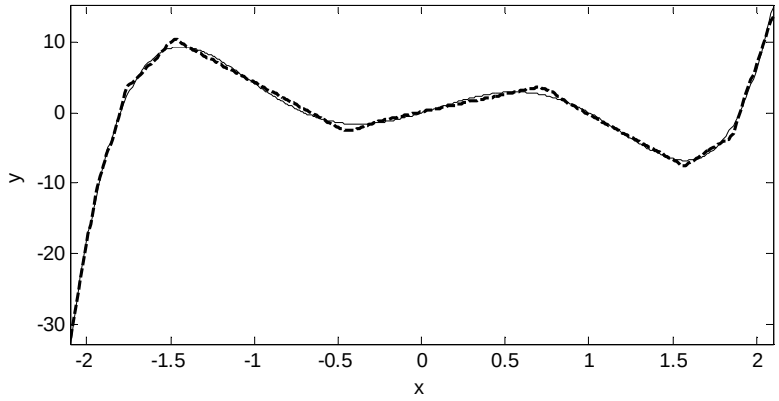


Figure 6.3. Function approximation by the 8-rule GTSK model in Figure 6.2

Figure 6.4 shows for each rule the normalized truth of antecedent, *w* defined in Equation (3.23), which could be used to visualize the interaction between rules and local interpretability in each rule. For instance, the 4^{th} rule almost works alone for *x* between -1.4 and -0.6, where the valve of *w* for the 4^{th} rule is about one. Therefore, the consequent model in the 4^{th} rule could be interpreted as a local linear approximation for the nonlinear function over the above mentioned region. Following the similar procedure, it is possible to interpret consequent models in all rules as local linear approximation for the nonlinear function and identify the approximation region respectively. Interactions between rules are signified by the value *w* a little far away from both 0 and 1. For instance, interaction between the 4^{th} and 5^{th} rules is observed for *x* between -0.6 and -0.2, where there are about 15 points with the value of *w* between 0.2 and 0.8. The assumption made on *w* in Equation (5.24) would not hold due to the presence of many interactions in rules. Therefore, it might be possible to use Newton's method (Algorithm 5.1) to further adjust model parameters to reduce the approximation error.

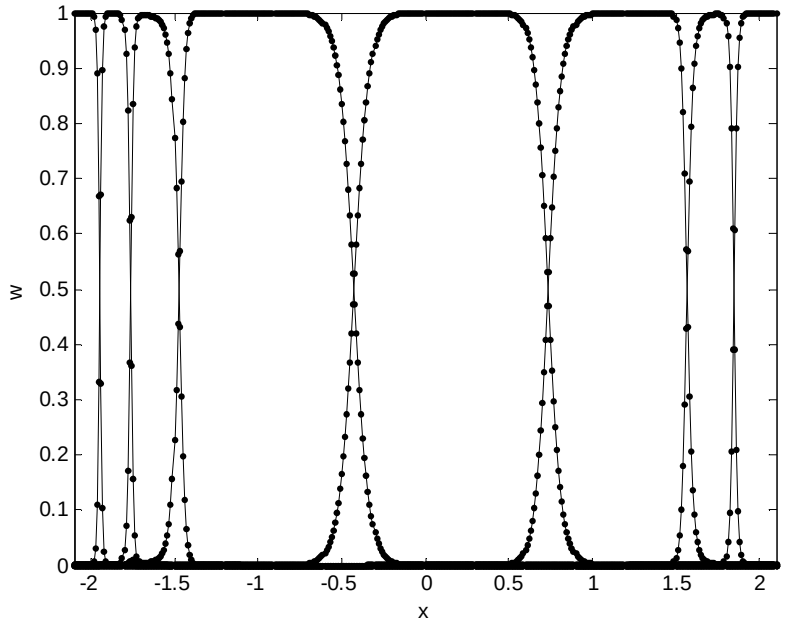


Figure 6.4. Normalized TAs for those in Figure 6.2

Figure 6.5 shows the optimized membership functions by Algorithm 5.1 starting from the above initialization. The resultant function approximation is shown in Figure 6.6 with the MSE reduced to 0.12. The improvement in terms of MSE is clear. Some noticeable large approximation error in Figure 6.3 (around x=-1.5, -0.5, 0.7) are significantly reduced. The approximation in Figure 6.6 becomes also smoother, which is due to the increase of overlap between adjacent membership functions. For instance, the 4^{th} and 5^{th} (from the left) membership functions in Figure 6.5 share a significant portion of overlap, which is not observed in Figure 6.3. The increase of overlapping is also observed in other adjacent membership function pairs, between 2^{nd} and 3^{rd}, and between 7^{th} and 8^{th}.

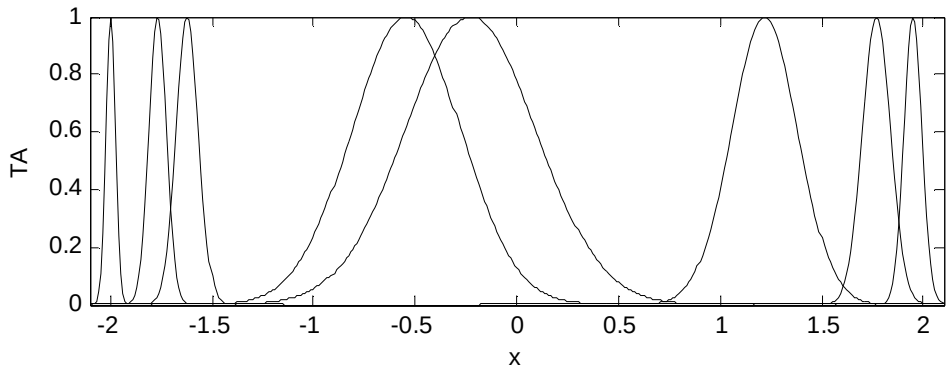


Figure 6.5. Optimized TAs from initialization in Figure 6.2

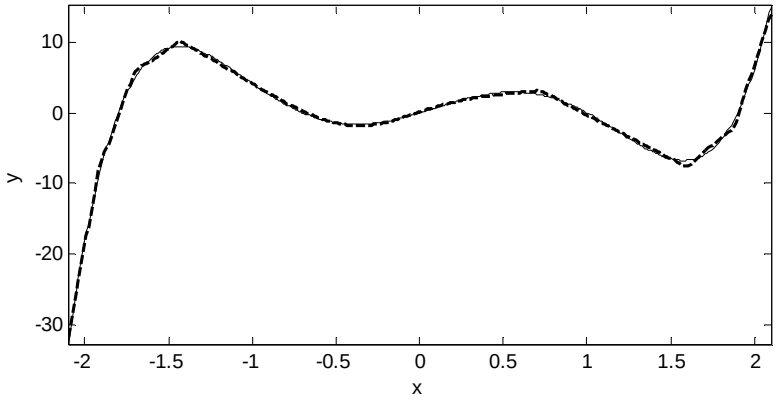


Figure 6.6. Function approximation by the optimized 8-rule GTSK model

The MSE reduction is achieved at the cost of interaction increase between rules. Rules resulted from the initialization shown in Figure 6.3 are relatively independent. The independence can be verified by the value *w* in Figure 6.4, which is close to 1 for the majority of data. The independence implies that the behavior of each rule represents the local behavior of the GTSK model. In other words, each rule is locally interpretable with respect to the GTSK model. In Figure 6.5, membership functions are more coupled. The increased interactions between rules are evidently observed in Figure 6.7. Rule 4 and 5 become less interpretable in terms of local behavior of the GTSK model. Both rules need to be considered together to explain a perhaps local quadratic behavior.

In general, approximation error and model interpretability are two conflicting goals. The illustrated interaction increase in rules should be expected in general when model parameters are optimized by the Newton's method (Algorithm 5.1), which will result in GTSK models consisting of less interpretable rules due to poor modularity. On the other hand, one might be able to preserve interpretability by forcing a certain distance between centroids or limiting the overlap between membership functions.

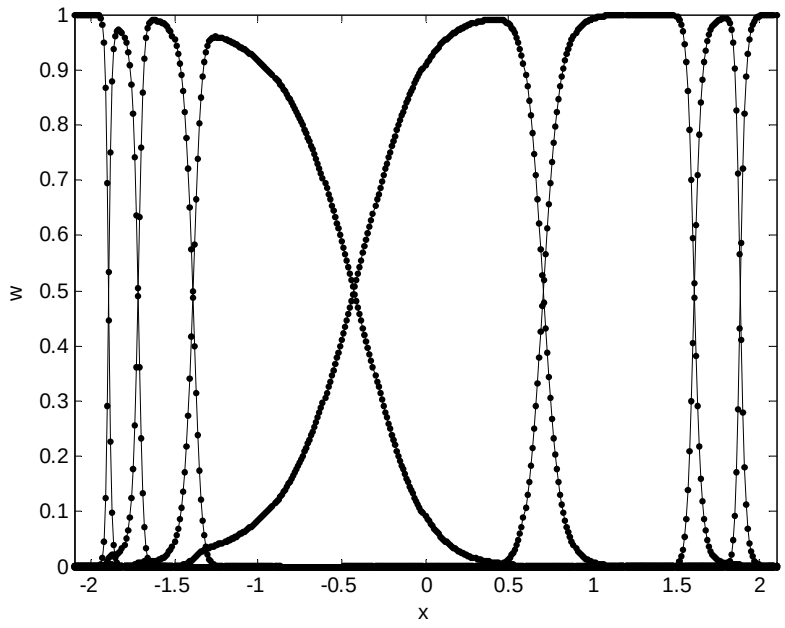


Figure 6.7. Normalized TAs for those in Figure 6.5

The two-stage parameter estimation procedure in Chapter 5 is also compared with the following one with random initialization. Figures 6.8 and 6.9 show the best result out of 50 trials. It represents a typical undesired result, stronger overlap but higher MSE (0.41).

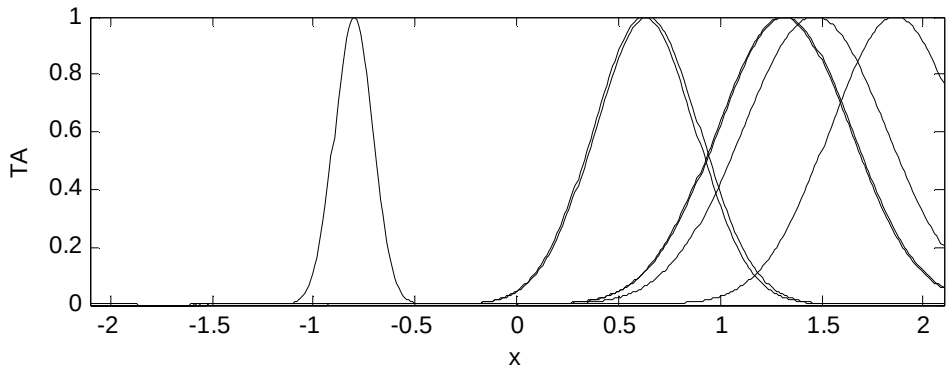


Figure 6.8. Optimized TAs starting from random initialization

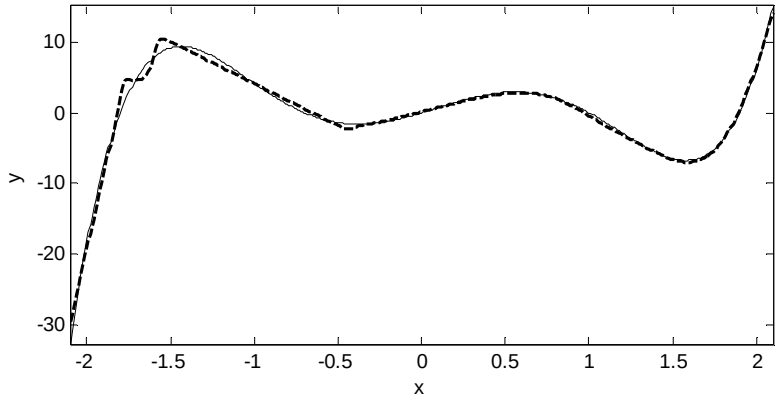


Figure 6.9. Function approximation by the 8-rule GTSK model in Figure 6.8

In approximating Function 1, the α_c is chosen to give an 8-rule GTSK model in order to compare the 8-rule fuzzy model in (Dickerson & Kosko, 1996). Certainly, one might need to have several trials to decide an appropriate value. In the following two function approximation examples, we will demonstrate what one may expect when the number of rules is progressive increased in a GTSK model.

Function 2 (Zhang, Chen, Ansari & Shi, 2004)

The second function to be tested is defined by

$$y = \sin\left(4\pi\sqrt{x}\right) + \left(1 + 0.4x\right)\cos\left(\pi x\right), \qquad 0 \le x \le 5 \qquad (6.3)$$

Figure 6.10 shows four different antecedent space partitions and membership function initializations. The partition in Figure 6.10(b) has one more split than that in Figure 6.10 (a). The additional split is added to the second region in Figure 6.10 (b), which then generates two linear approximations. One more split is added in Figure 6.10 (c) to its leftmost region, which exhibit strong nonlinear behavior. In Figure 6.10 (d) two more splits are added to split the 2nd and 9th regions in Figure 6.10 (c). It is observed in Figure 6.10 (d) that more splits are placed in the left part of the function. The function is uneven in terms of nonlinear behavior in different regions. Its left part is more nonlinear than its other parts. Therefore, the obtained partition is desired, which distribute rules according to nonlinearity.

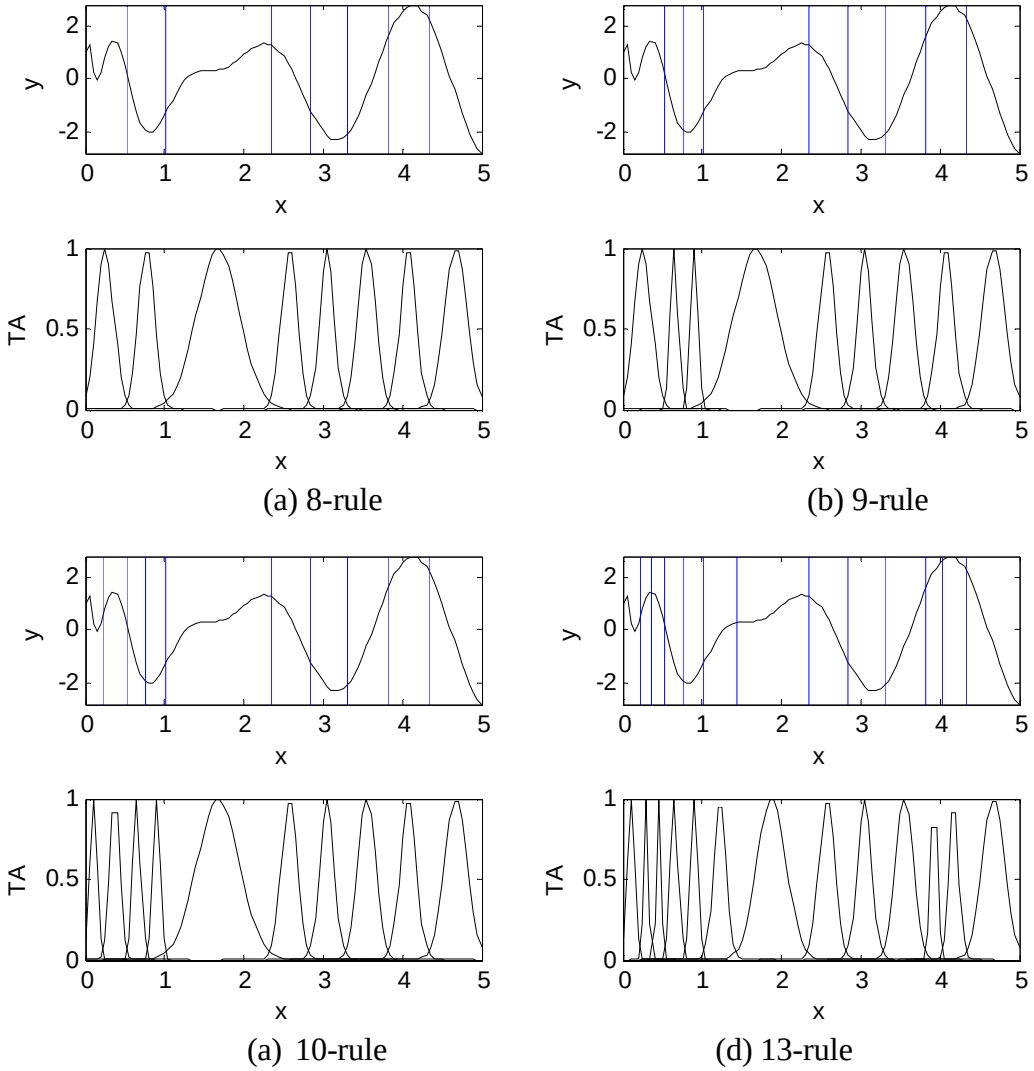


Figure 6.10. Antecedent space partition and TAs on Equation (6.3)

Function 3

The third function to be tested is a two-dimensional quadratic function.

$$y = x_1^2 + x_2^2, \qquad -2 \le x_1, x_2 \le 2 \tag{6.4}$$

There are 441 points uniformly sampled. With $\bar{M}$ set at 100, there are 55 branch nodes generated and their corresponding alphas are shown in Figure 6.11.

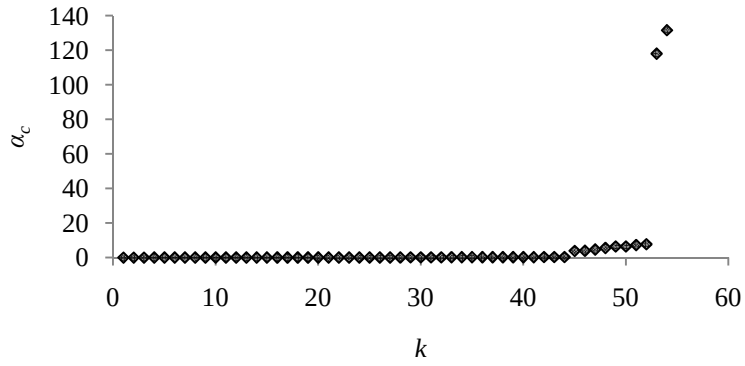


Figure 6.11. Values of α_c for antecedent space partition for Equation (6.4)

The values of the first two α_cs is much higher than others. It would be reasonable to keep both if either one is to be kept. Figure 6.12 shows the resultant antecedent space and the corresponding antecedents in terms of ellipsoids with *TA*=0.05

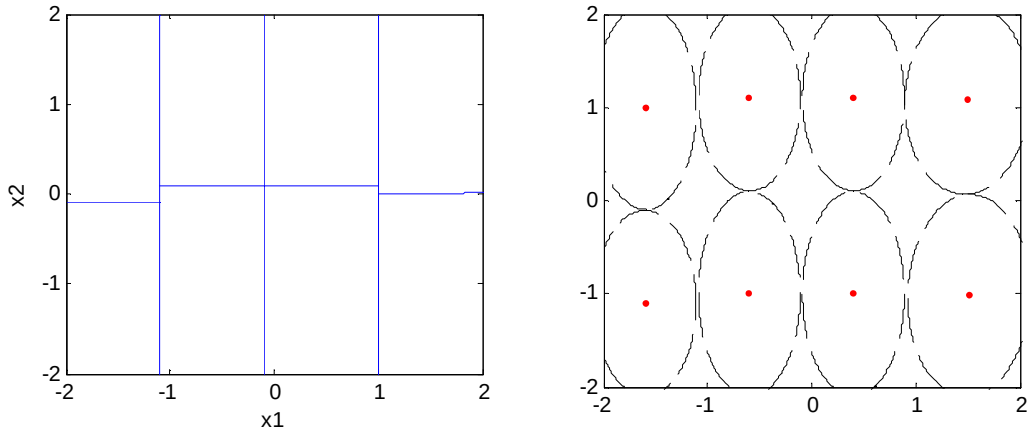


Figure 6.12. a) Antecedent space partition by α_c > 117; b) Ellipsoids (*TA* = 0.05)

Figure 6.13 shows the normalized truth of antecedent for all rules, where limited interactions are observed. It indicates that rules have good interpretability for the local behavior of the nonlinear function. In this example, the optimization by Algorithm 5.1 reduces the MSE from 0.125 to 0.121 (3.2% improvement). The negligible improvement is probably due to the distribution of *w* for each rule. Figure 6.13 shows that the values of *w* for each rule are either high or low. The values of *w* for the rule in left-front corner are plotted in Figure 6.14, where 424 out of 441 points have *w* outside the range of (0.1, 0.9). Other 17 points cluster around either 0.8 or 0.2. Not much intermediate values are observed for *w*. The observation might be able to make the assumption in Equation (5.24) approximately hold. It then indicates that the initialization almost reaches a local solution.

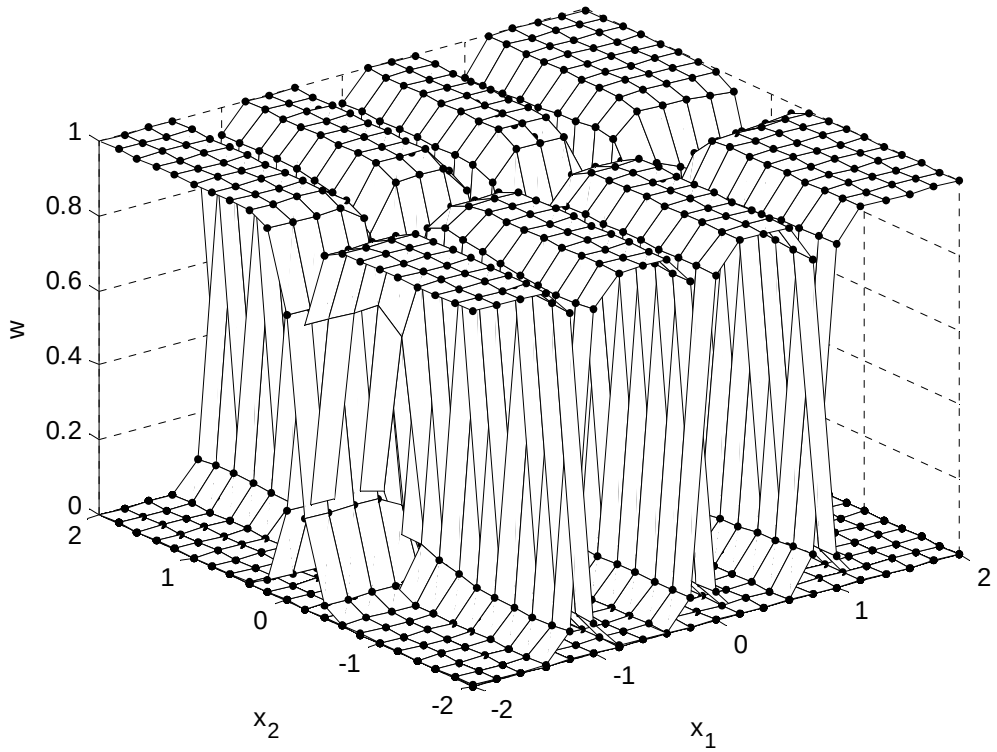


Figure 6.13. Normalized TAs for those in Figure 6.12

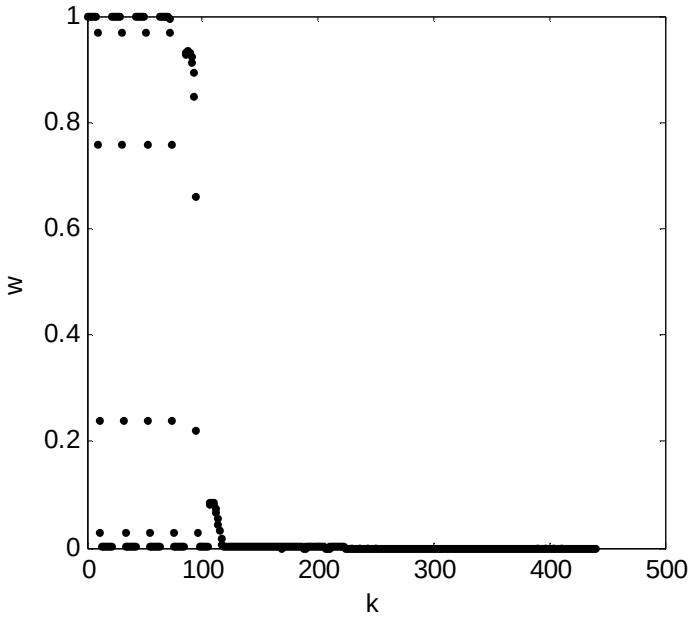


Figure 6.14. Normalized TAs for the left-front rule in Figure 6.13

The obtained model is an 8-rule GTSK model, which approximates the quadratic function using 8 planes. The approximation is shown in Figure 6.15 with MSE of 0.125.

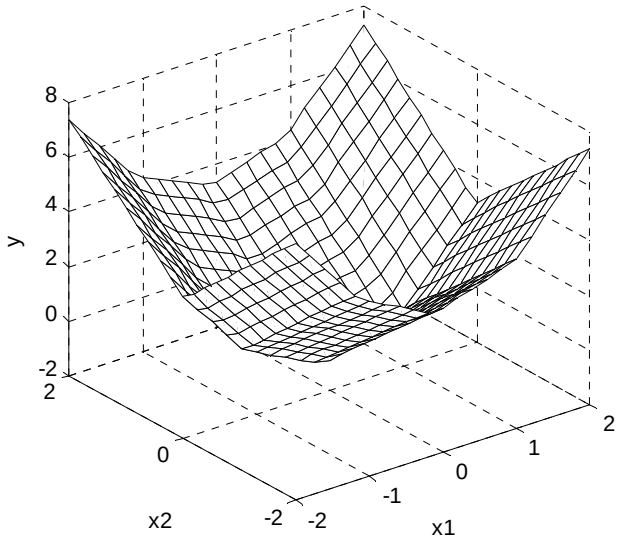


Figure 6.15. Quadratic function approximation by the GTSK model in Figure 6.12

Certainly, one can manage to discard the branch node with the second highest α_c in Figure 6.11. The resultant partition is shown in Figure 6.16, which is uneven and only have the right portion of the antecedent space partitioned. It then suggests to keeping branch nodes with like α_c values.

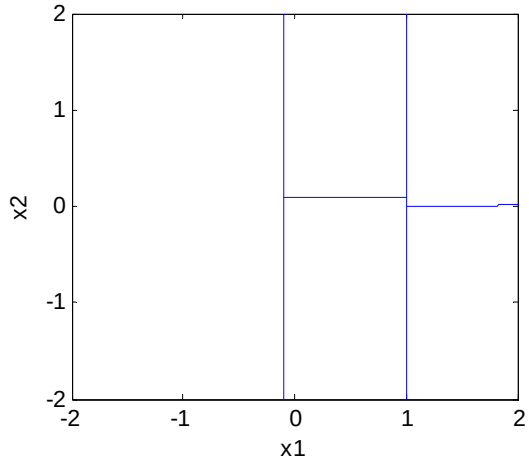


Figure 6.16. Antecedent space partition by $\alpha_c > 130$

The same procedure is practiced if users manage to increase the number of rules. Figure 6.17 shows the values of the rest of α_c and clearly indicates two groups with difference at least one order of magnitude. It suggests that one should keep all α_c between 3 and 8, if any of them is going to be kept.

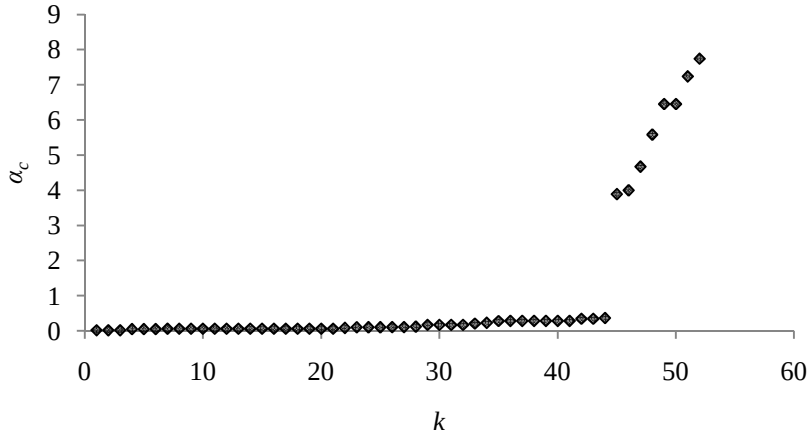


Figure 6.17. A portion of α_c in Figure 6.11 with values less than 118

The resultant antecedent partition is shown in Figure 18, which adds an additional split to each region in Figure 6.12. The observation is reasonable. Unlike Function 2 in Equation (6.3) whose nonlinearity is uneven, the two-dimensional quadratic function is uniformly nonlinear in every direction. Due to the uniformity, the antecedent space should be evenly partitioned. The increased rules will enable the GTSK model to approximate function in a finer scale. The 16 recognized antecedents are shown in Figure 6.18(a) and Figure 6.18(b) shows the approximation by the 16-rule GTSK model with MSE of 0.0153.

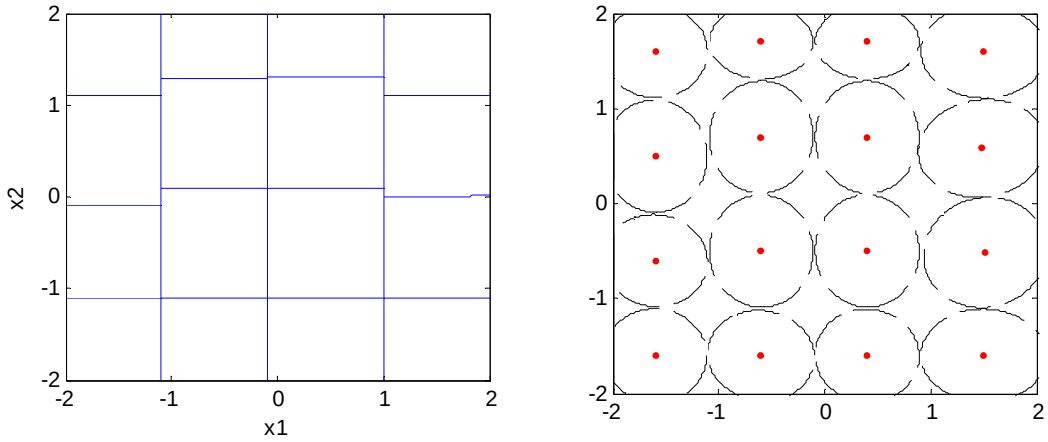


Figure 6.18. a) Antecedent space partition by $\alpha_c > 3$; b) Ellipsoids ($TA = 0.05$)

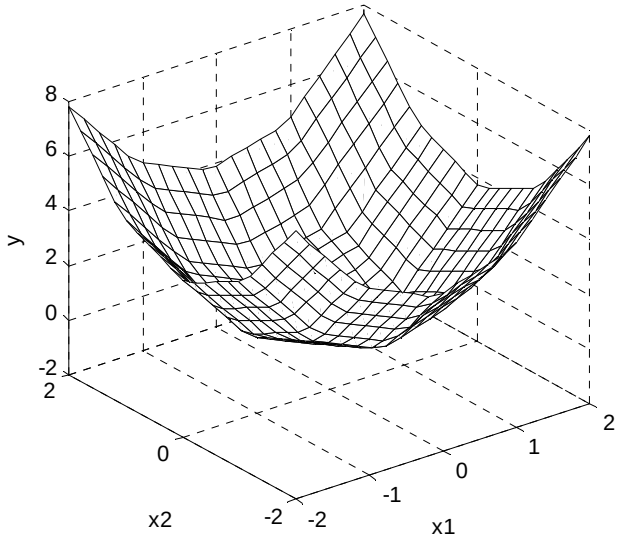


Figure 6.19. Quadratic function approximation by the GTSK model in Figure 6.18

One might follow the above procedure to further increase the number of rules by including branch nodes with smaller α_c as shown in Figure 6.20. The distinction between different levels is not as clear as shown in Figure 6.11 and 6.17. One might need try several values and find an appropriate one

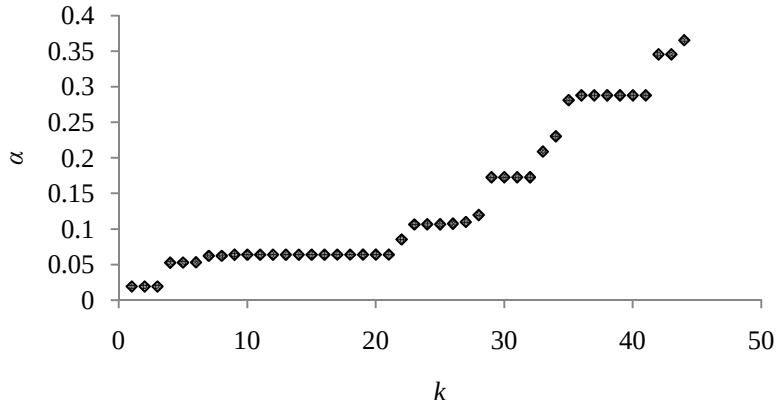


Figure 6.20. A portion of α_c shown in Figure 6.11 with values less than 3

It is also observed that splits in Figures 6.12 and 6.18 are along with the coordinate directions. The observation is reasonable since the symmetric quadratic function is uniformly nonlinear in all directions.

The above procedure is compared against the following one with random initializations. It is found that some GTSK models due to random initialization produce smaller MSE. One of typical good approximation result is shown in Figure 6.21 with MSE of only 0.0051.

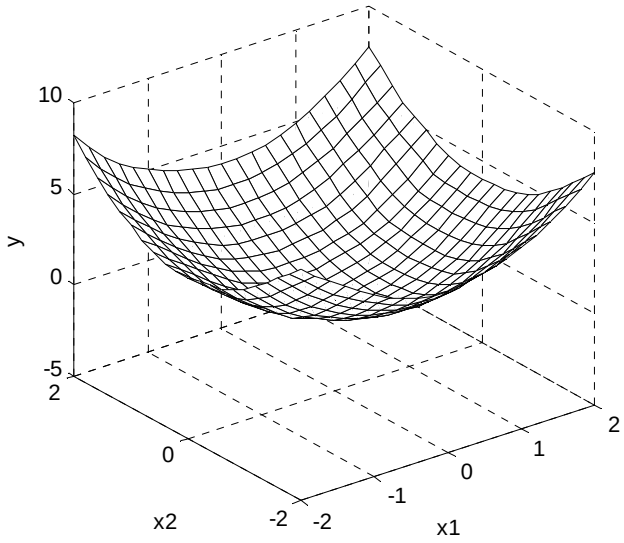


Figure 6.21. Quadratic function approximation by the GTSK model in Figure 6.22

The corresponding antecedents of 16 rules due to random initialization are shown in Figure 6.22, where very strong and complex coupling among rules are observed. Modularity in rules does not seem to exist and interpretation of rules with respect to local behavior of the model is impossible.

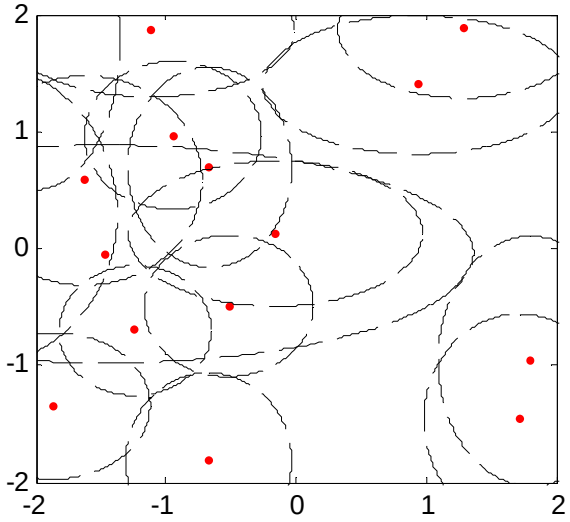


Figure 6.22. Optimized *TA*s for a 16-rule GTSK model from random initialization

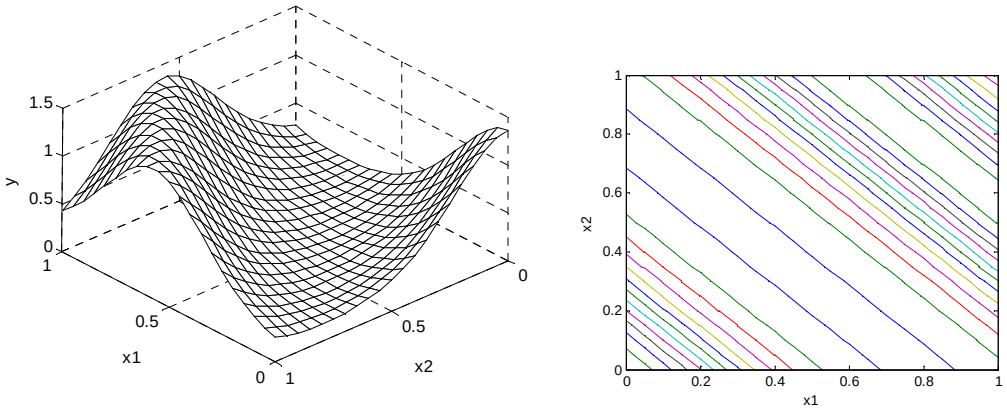
Function 3 is uniformly nonlinear in all directions. The resultant ellipsoids shown in Figures 6.12 and 6.18 are oriented along with coordinates. The next example will demonstrate how ellipsoids are to be oriented if the function is unevenly nonlinear in different directions.

Function 4 (Zhang, Chen, Ansari & Shi, 2004)

The fourth function to be approximated is defined below and shown in Figure 6.23(a)

$$y = \frac{1}{2}\exp\left(\cos\left(4\left(x_1 + x_2\right)\right)\right), \quad 0 \le x_1, x_2 \le 1 \tag{6.5}$$

Figure 6.23(b) is the contour plot of the function, which shows that the function behaves linearly along the main-diagonal direction from (0,1) to (1,0).



(a) (b)

Figure 6.23. Illustration of the function in Equation (6.5) and its contour plot

A total of 441 points are uniformly sampled. With $\bar{M}$ of 50, there are 29 branch nodes generated and their corresponding α_c are shown in Figure 6.24.

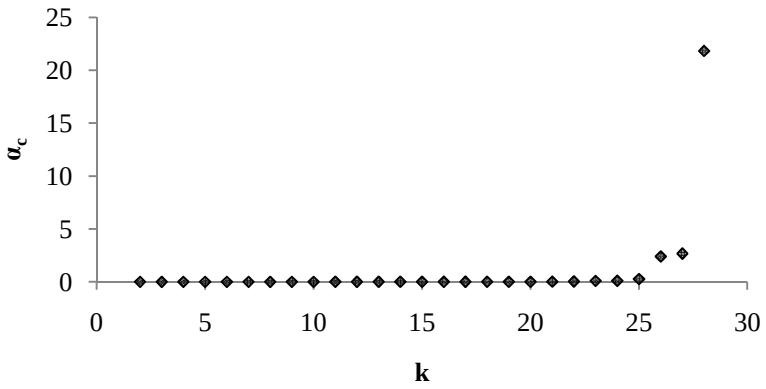
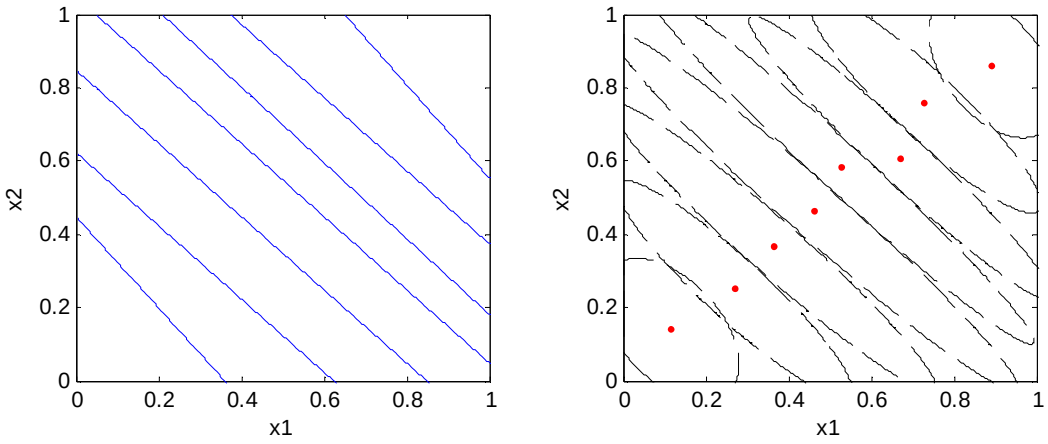


Figure 6.24. Values of α_c for antecedent space partition on Equation (6.5)

By including the first 5 branch nodes, the resultant antecedent partition is shown in Figure 6.25. The partition slices the antecedent space along the main diagonal direction, which matches the nonlinear orientation shown in Figure 6.23(b). The corresponding initialization of rule antecedents is shown in Figure 6.25. The approximation due to the 8-rule GTSK model has a MSE of 0.0015.



(a) (b)

Figure 6.25. a) Antecedent space partition by $\alpha_c > 0.1$;b) Ellipsoids (*TA*=0.05)

Improvement of MSE is achieved by further tuning the model parameters using Algorithm 5.1. Obtained rules are shown in Figure 6.26, where centroids of rules are

significantly changed. However, the direction of each antecedent is still kept in the main diagonal direction while the length and width of each ellipsoid are changed.

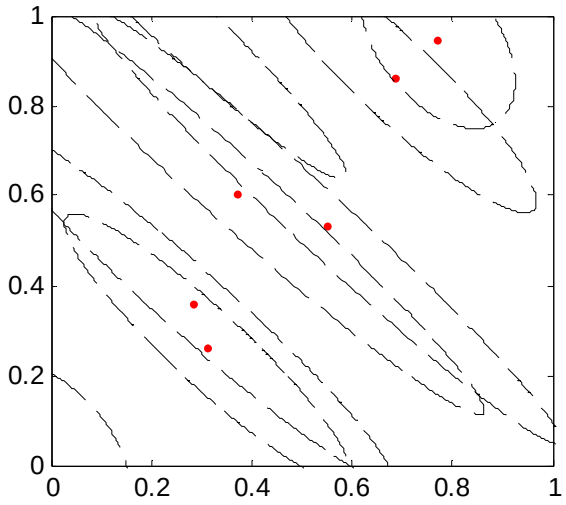


Figure 6.26. Optimized TAs of a 8-rule GTSK model for Equation (6.5)

The resultant function approximation is shown in Figure 6.27 with reduced MSE of 0.0003. Again, the reduction of MSE is at the cost of interpretability in individual rules.

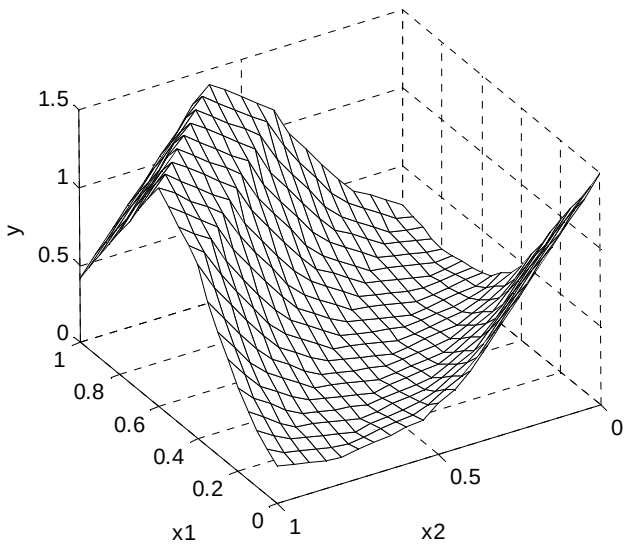


Figure 6.27. Approximation of Equation (6.5) by the GTSK model in Figure (6.26)

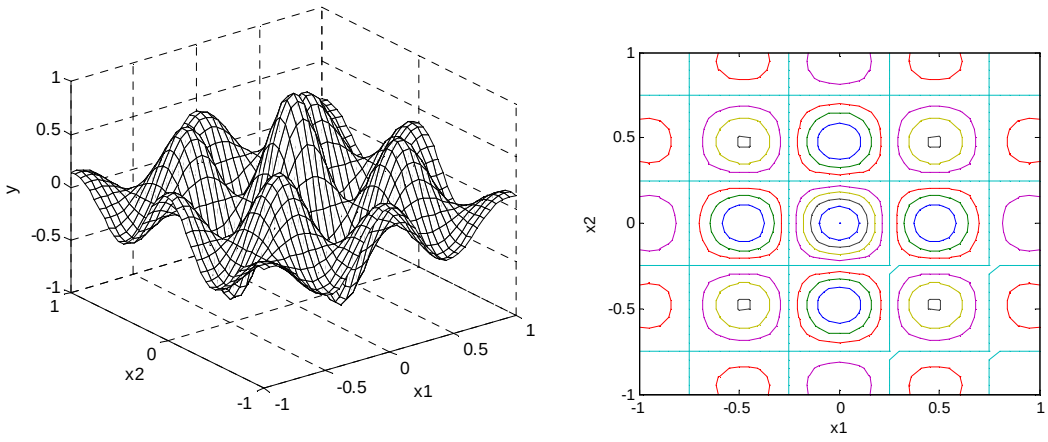
The local models in the above four examples are linear. In fact, there is no restriction on types of local models. Roughly speaking, one should expect better approximation and less number of rules if more complex local models are used. In the next example, linear and quadratic local models are compared.

Function 5

The fifth function (Zhang, Chen, Ansari & Shi, 2004) to be approximated is defined

$$y = \cos\left(2\pi x_1\right)\cos\left(2\pi x_2\right)\mathrm{e}^{-\left(x_1^2+x_2^2\right)}, \qquad -1 \le x_1, x_2 \le 1 \tag{6.6}$$

The function and its contour plot are shown in Figure 6.28.



(a) (b)

Figure 6.28. Illustration of function in Equation (6.6) and its contour plot

In this example, there are 1681 points sampled from the function. With $\bar{M}$ of 100, there are 53 branch nodes are generated. The values of α_c for all branch nodes are shown in Figure 6.29.

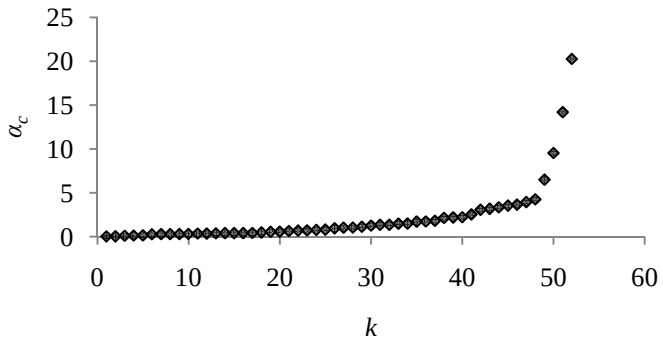
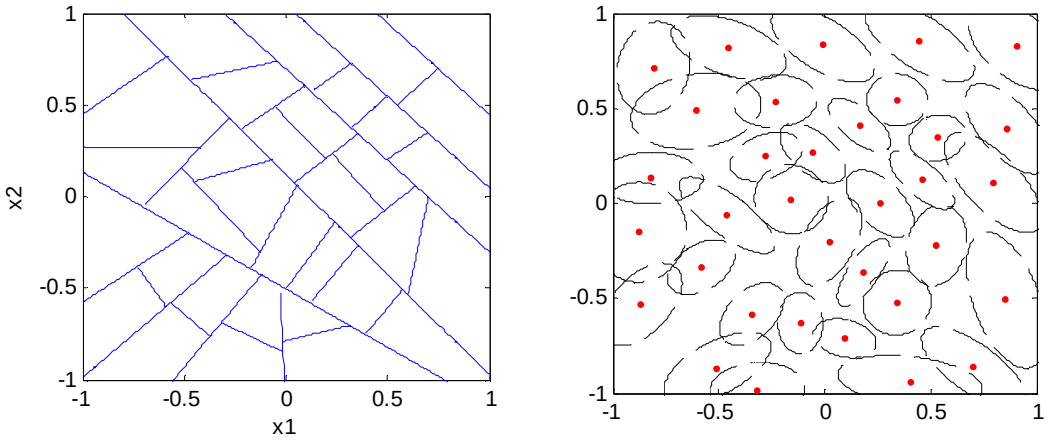


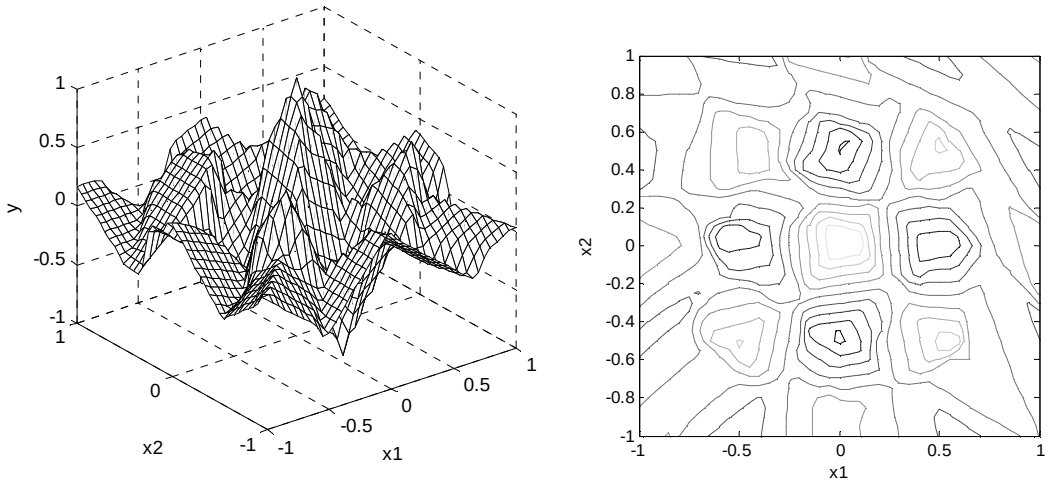
Figure 6.29. Values of α_c for antecedent space partition on Equation (6.6)

Figure 6.30 shows the obtained antecedent partition by accepting branch nodes with alpha greater than 0.81. A 34-rule GTSK model is then initialized. Figure 6.30 shows the final result after implementing Algorithm 5.1 to tune model parameters. The function approximation and corresponding contour plot are shown in Figure 6.31. The MSE for the function approximation is 0.0069.



(a) (b)

Figure 6.30.a) Antecedent space partition by α_c>0.81; b) Ellipsoids (*TA*=0.05)



(a) (b)

Figure 6.31. Function approximation by the model in Figure 6.30 and the contour

One could further increase the number of rules to reduce the approximation error. Alternatively, users might increase the complexity of local models. In the flowing example, quadratic local models are used instead. The obtained α_c values for all branch nodes are shown in Figure 6.32.

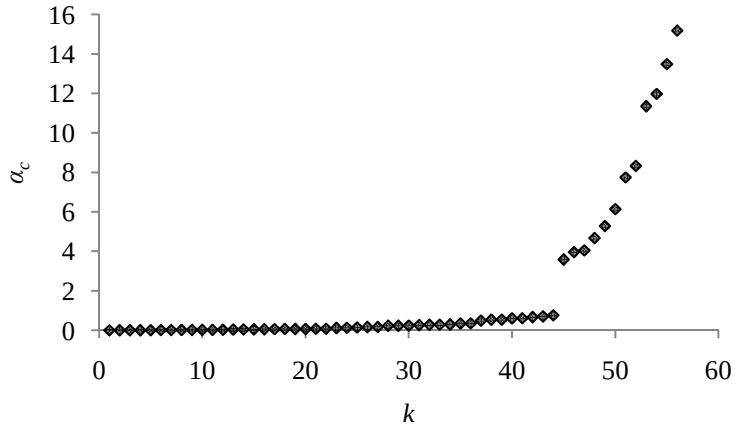
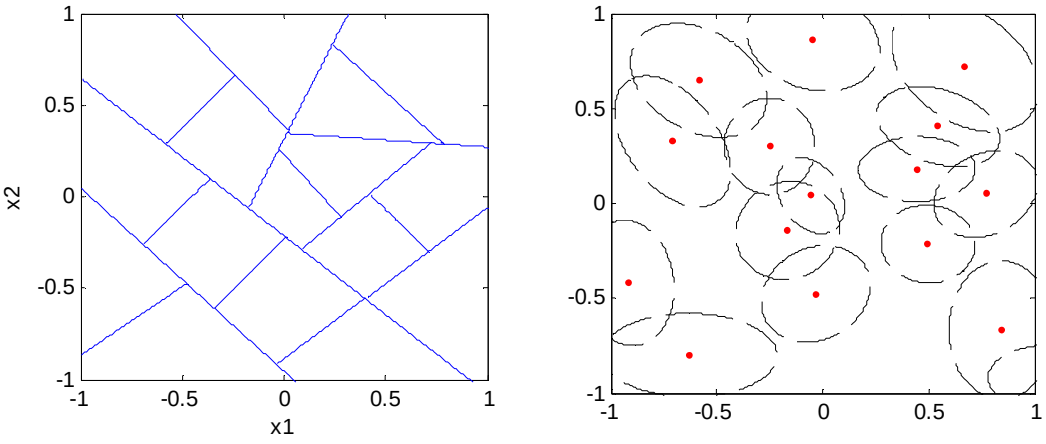


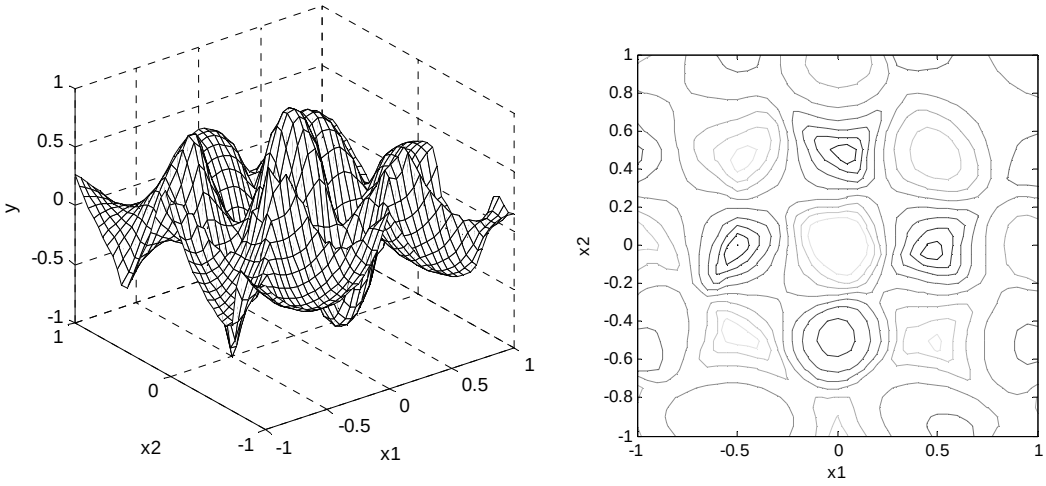
Figure 6.32. Values of α_c for antecedent space partition for Equation (6.6) with quadratic local models

With threshold for α_c set at 1.5, the following partition is obtained in Figure 6.33(a). The optimized antecedents are shown in Figure 6.33(b). The resultant function approximation and contour plot are shown in Figure 6.34. The MSE is 0.0049.



(a) (b)

Figure 6.33.a) Antecedent space partition by α_c >1.5; b) Ellipsoids (*TA*=0.05)



(a) (b)

Figure 6.34. Function approximation by the model in Figure 6.33 and the contour

The number of model parameters in the 16-rule GTSK model with quadratic local models is 176. On the other hand, the number of parameters is 272 in the 34-rule GTSK model with local linear models. It indicates that using more complex local models can significantly reduce the number of rules and overall model parameters while improving function approximation performance. However, complex local models might be difficult for interpretation, which is however subject to users' knowledge.

The above testing focuses on function approximation. The following several testing will be about nonlinear dynamic modeling, which is actually not very much different from function approximation in this work since the dynamic model structure is restricted to ARX structure. Users then have full access to all model inputs. The structure information for a nonlinear dynamic model is assumed known and determined in Chapter 4 by the proposed order determination technique. The antecedent variables are also selected in Chapter 4. In several following examples, we will validate the antecedent variable selection made in Chapter 4. For the convenience of presentation, we might reproduce some equations in Chapter 4. It is observed in Section 6.1 that modularity and local interpretability in initialized rules are reduced by further parameter tuning using Newton's method due to interaction increase between rules. In the following testing, results are based on parameter estimates extracted from partitioned antecedent space.

6.2 Dynamic Nonlinear Modeling

Model 1 (Narendra & Parthasarathy, 1990)

$$y(t) = 0.3y(t-1) + 0.6y(t-2) + 0.6\sin\left(\pi u(t-1)\right) + 0.3\sin\left(3\pi u(t-1)\right) + 0.1\sin\left(5\pi u(t-1)\right) + e(t) \tag{6.7}$$

The order determination was conducted on Model 1 in Chapter 4. The determined regressors are [$y(t\text{-}1)$ $y(t\text{-}2)$ $u(t\text{-}1)$] in Table 4.8. The detected nonlinear component is $u(t\text{-}1)$ that will be the antecedent variable. In order to verify the choice of antecedent variable, the following experiment is conducted to try different antecedent variables. The experiment result is collected in Table 6.1. The performance is evaluated by the sum of square error (SSE) between the output y and its prediction. The SSE without any splitting is 1544. The first row of Table 6.1 have the number of rules and resultant SSE due to

having only $u(t\text{-}1)$ in the antecedent. In order to compare each choice of antecedent variable fairly, each resultant GTSK model is configured to have the same number of rules; 3 in this experiment. It is observed in Table 6.1, the best choice of antecedent variable is $u(t\text{-}1)$. The other two choices, either $y(t\text{-}1)$ or $y(t\text{-}2)$, barely reduce the SSE. The experiment is then able to validate the choice of $u(t\text{-}1)$ as the antecedent variable.

Table 6.1.Trials of antecedent variables for Model 1 in Equation (6.7)

Antecedent	Number of rules	SSE
$u(t\text{-}1)$	3	1325
$y(t\text{-}1)$	3	1540
$y(t\text{-}2)$	3	1538

A GTSK model could include different number of rules by accepting different levels of α_c. In the following, each choice of number of rules is validated by a separate data set (validation data set). The results are collected in Table 6.2. The training data include 5000 samples while validation data include 3000 samples. The 'Model Error' is the sum of training and validation MSE. It is observed that Model Error start increasing when M is over 8. Based on the experiment results in Table 6.2, it actually makes no difference by choosing M as 7 or 8. In the following illustration, M=8 is chosen.

Table 6.2. Trials of a GTSK model for Model 1

M	MSE		Model Error
	Training	Validation	
2	0.272	0.276	0.547
3	0.265	0.269	0.534
4	0.259	0.261	0.520
5	0.253	0.257	0.510
6	0.252	0.257	0.509
7	0.251	0.256	0.507
8	0.251	0.256	0.507
9	0.251	0.259	0.509

Having $u(t\text{-}1)$ in the antecedent with M=8, the resultant antecedent partition and membership function initializations are shown in Figure 6.35, where the number in each region indicates the order that regions are generated in a binary tree.

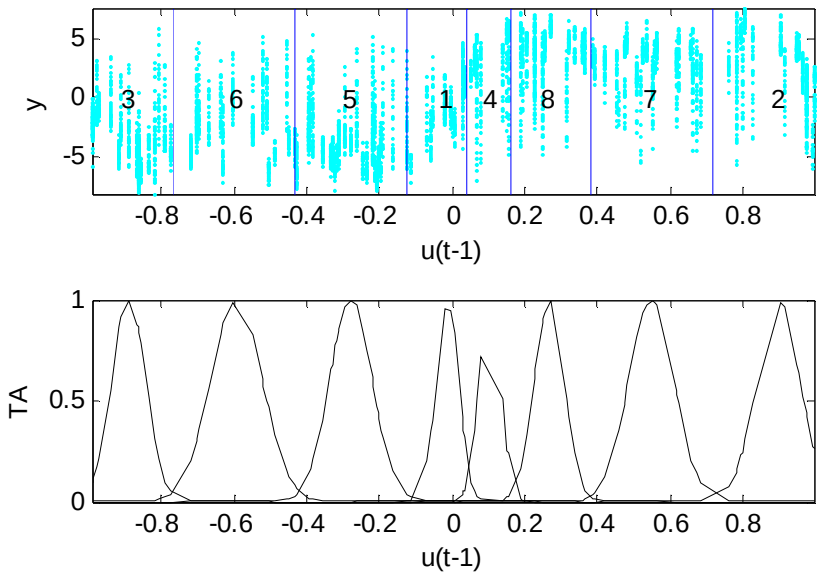


Figure 6.35. Antecedent space partition and TAs for Model 1

Figure. 6.36 shows the separations in the nonlinear part of Model 1, g($u(t\text{-}1)$), the sum of three Sine functions of $u(t\text{-}1)$, which behaves relatively linearly in local regions.

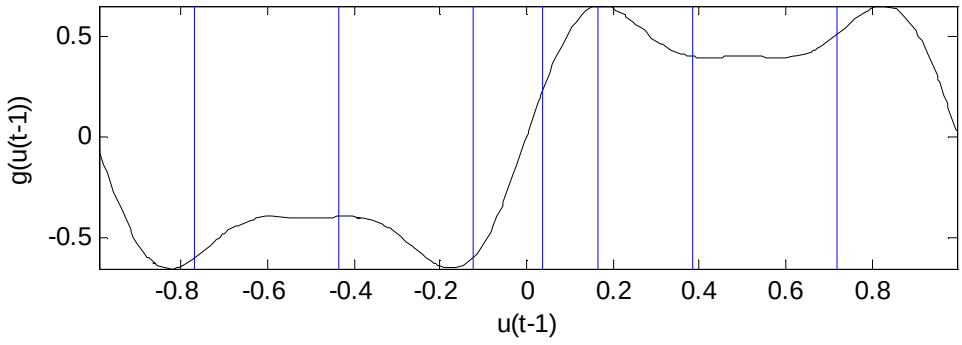


Figure 6.36. The separation boundaries shown for the nonlinear part in Model 1

The resultant GTSK model is fully described in Equation (6.8) and listed from the left to right in Figure 6.35.

$$
\begin{aligned}
&3:\textbf{IF}\ \left(u(t-1)\ \text{is in}\ R^3(-0.9,227.5)\right)\ \textbf{THEN}\ \hat{y}^3(t)=-3.02+0.34y(t-1)+0.56y(t-2)-2.88u(t-1)\\
&6:\textbf{IF}\ \left(u(t-1)\ \text{is in}\ R^6(-0.6,100.0)\right)\ \textbf{THEN}\ \hat{y}^6(t)=-0.33+0.31y(t-1)+0.60y(t-2)+0.11u(t-1)\\
&5:\textbf{IF}\ \left(u(t-1)\ \text{is in}\ R^5(-0.3,145.5)\right)\ \textbf{THEN}\ \hat{y}^5(t)=-0.82+0.29y(t-1)+0.61y(t-2)-1.06u(t-1)\\
&1:\textbf{IF}\ \left(u(t-1)\ \text{is in}\ R^1(-0.0,533.3)\right)\ \textbf{THEN}\ \hat{y}^1(t)=-0.03+0.30y(t-1)+0.60y(t-2)+5.32u(t-1)\\
&4:\textbf{IF}\ \left(u(t-1)\ \text{is in}\ R^4(0.1,567.1)\right)\ \textbf{THEN}\ \hat{y}^4(t)=\ 0.22+0.26y(t-1)+0.63y(t-2)+3.25u(t-1)\\
&8:\textbf{IF}\ \left(u(t-1)\ \text{is in}\ R^8(0.3,299.6)\right)\ \textbf{THEN}\ \hat{y}^8(t)=\ 0.78+0.30y(t-1)+0.61y(t-2)-1.13u(t-1)\\
&7:\textbf{IF}\ \left(u(t-1)\ \text{is in}\ R^7(0.5,129.0)\right)\ \textbf{THEN}\ \hat{y}^7(t)=\ 0.31+0.31y(t-1)+0.58y(t-2)+0.19u(t-1)\\
&2:\textbf{IF}\ \left(u(t-1)\ \text{is in}\ R^2(0.9,141.1)\right)\ \textbf{THEN}\ \hat{y}^2(t)=\ 2.72+0.33y(t-1)+0.58y(t-2)-2.64u(t-1)
\end{aligned}
\tag{6.8}
$$

It is observed in Equation (6.8) that coefficients for *u*(t-1) experiences both magnitude variation and sign change. However, coefficients for either *y*(t-1) or *y*(t-2) do not seem to vary too much. It seems that the variation in the coefficient for *u*(t-1) is sufficient to verify the nonlinearity of the model. A more detail address of coefficient value variation across rules needs to however consider the variance of model parameter estimates. The covariance of local model parameters is estimated by

$$
Cov\left(\hat{\boldsymbol{\theta}}^i\right)=\left(\hat{\sigma}^i\right)^2\left(\left(\mathbf{X}^i\right)^T\mathbf{X}^i\right)^{-1} \tag{6.9}
$$

where the matrix $\mathbf{X}^i$ collects all regressors in region i. $\left(\hat{\sigma}^i\right)^2$ is the variance estimate for the noise in region i and is computed via the residuals, ε^i by

$$
\left(\hat{\sigma}^i\right)^2=\frac{1}{N^i}\sum_{t=1}^{N^i}\left(\varepsilon^i(t)\right)^2 \tag{6.10}
$$

where N^i is the data number in region i. The 95% confidence interval for θ_j^i is defined by

$$
\theta_j^i=\hat{\theta}_j^i\pm 1.96Cov\left(\hat{\theta}_j^i,\hat{\theta}_j^i\right) \tag{6.11}
$$

Equation (6.11) is for Gaussian distribution for a known variance. One might use Student distribution if variance is an estimate. The difference could however be ignored

for sufficient number of data, likely over 50 data points (Box, Jenkins & Reinsel, 1994). The coefficients and their 95% confidence interval for 8 rules are shown in Figure 6.37.

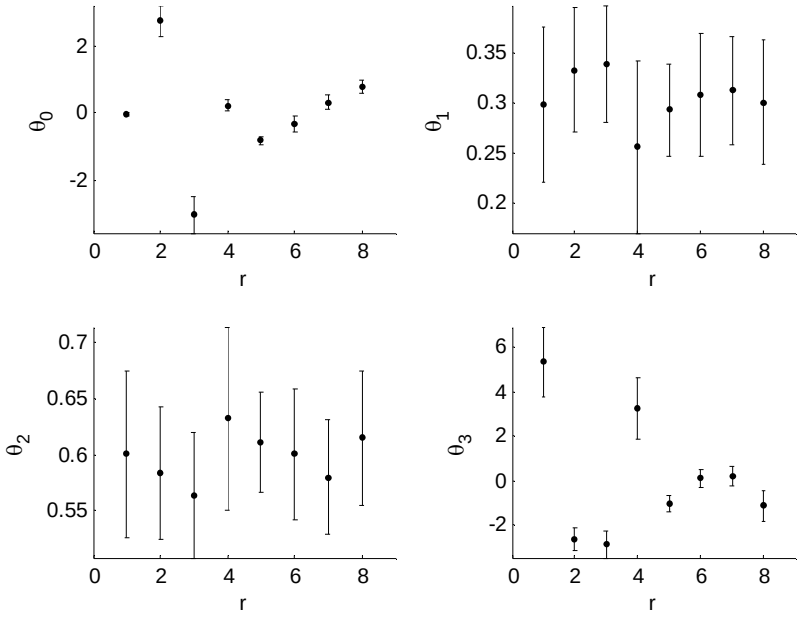


Figure 6.37. Coefficients for local models in the GTSK model in Figure 6.35

In Figure 6.37, θ_0 to θ_3 are coefficients for regressors, 1, $y(t\text{-}1)$, $y(t\text{-}2)$ and $u(t\text{-}1)$. Strong variation is observed for both θ_0 and θ_3. The variation in θ_3 indicates a change of model behavior in different regions. On the other hand, the confidence intervals for θ_1 in different rules have overlaps. The same phenomenon is also observed for θ_2. The observations might imply constant coefficients for regressors, $y(t\text{-}1)$ and $y(t\text{-}2)$ for all rules, which then suggests that it might be unnecessary to include $y(t\text{-}1)$ and $y(t\text{-}2)$ in a GTSK model. It is then possible to simply the structure of the GTSK model as a hybrid with an explicit linear structure.

$$y(t) = a_1 y(t-1) + a_2 y(t-2) + f\left(u(t-1)\right) \tag{6.12}$$

The obtained GTSK model is compared to other modeling possibilities. In this work, a radial basis network model (RB) and a feed-forward neural network model (FFNN) are considered. In order to have a common basis for comparison, the architecture for each model is chosen such that the number of parameter in each model is close. In the comparison, 5000 data points are used to obtain the model parameters that gives a

'training error' and 2500 data points are used to give a 'validation error'. Both 'training' and 'validation' errors are summed-square of residuals. The comparison detail is collected in Table 6.3.

Table 6.3 Comparison of the GTSK with RB and FFNN for Model 1

Model	Architecture	Training (SSE)	Validation (SSE)	# of Parameters
GTSK	8	1253.6	639.7	48
FFNN	(3,5,6,1)	1252.9	639.2	63
FFNN	(3,2,12,1)	1256.6	640.6	57
FFNN	(3,8,4,1)	1254.2	647.4	73
RB	11	4564.8	3209.6	56

In Table 6.3, the architecture for GTSK is the number of rules. In the FFNN models, the architecture represents the number of inputs, number of neurons in each of two hidden layers, and the number of outputs. The architecture in the RB model is the number of neurons. The RB model gives the highest training and validation errors. On the other hand, there is no significant difference between GTSK and FFNN.

Training a neural network is a nonlinear optimization process. In practice, one often has to try many times of training from random initialization to obtain an acceptable solution. The result in Table for each FFNN is the best out of 50 trials while GTSK needs only one trial. In addition, the architecture information for a FFNN is not automatically available. In practice, one needs to try different architecture, and for each multiple regressors to find the probably best model. Three attempts are revealed in Table 6.3.

The GTSK model is more informative than a FFNN model. Parameter values in a FFNN model can hardly reveal any knowledge about the process to be described. Observed in Figure.6.37, the values of local model coefficients indicate to decouple $y(t\text{-}1)$ and $y(t\text{-}2)$ from a nonlinear function of $u(t\text{-}1)$.

Model 3 (Narendra & Parthasarathy, 1990)

$$y(t) = \frac{y(t-1)}{1+y(t-1)^2} + u(t-1)^3 + e(t) \tag{6.13}$$

The determined order is defined by ny=1,nu=0 and d=1. The result of nonlinear component detection indicates that both $u(t\text{-}1)$ and $y(t\text{-}1)$ should be included in antecedents. The antecedent space is shown in Figure 6.38

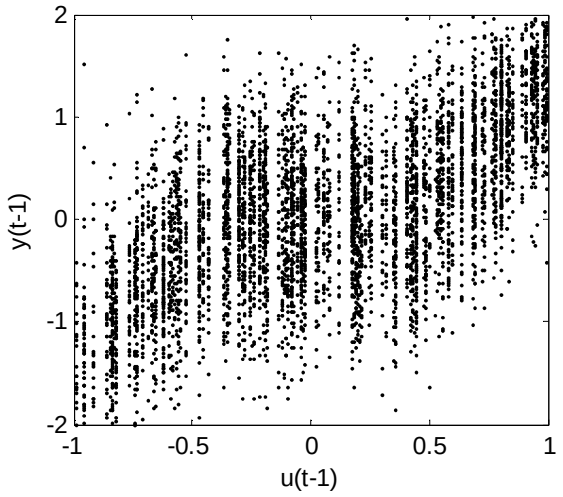


Figure 6.38. Two-dimension antecedent space for Model 3

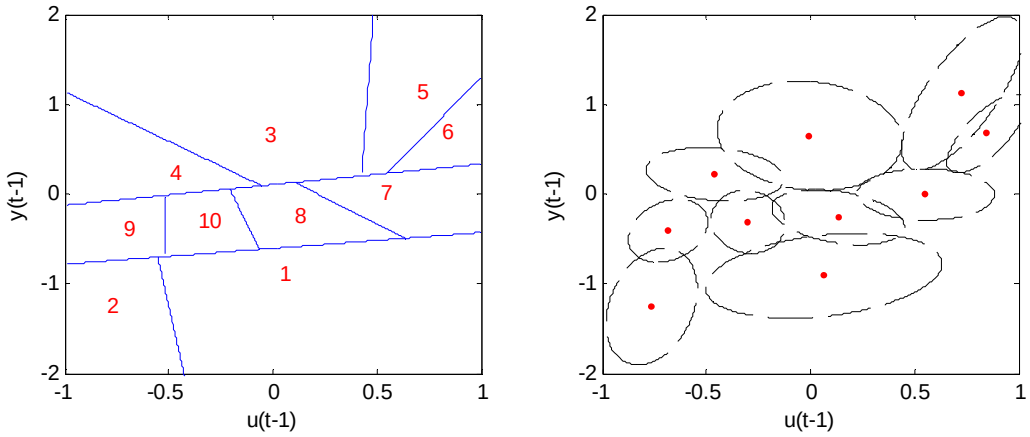
The results for trials of GTSK models with different complexity are collected in Table 6.4, where the minimum Model Error is due to a 10-rule GTSK model.

Table 6.4. Trials of a GTSK model for Model 3

	MSE		
M	Training	Validating	Model Error
2	0.277	0.278	0.555
3	0.268	0.271	0.540
4	0.265	0.266	0.530
5	0.261	0.264	0.525
6	0.259	0.262	0.521
8	0.257	0.260	0.516
9	0.256	0.260	0.516
10	0.255	0.259	0.514
11	0.255	0.260	0.515

Figure 6.39(a) shows the antecedent space partition with 10 regions. Figure

6.39(b) shows the ellipsoids with *TA* = 0.05 for initialized 10 rules.



(a) (b)

Figure 6.39. a) Antecedent space partition by $\alpha_c > 10$; b) Ellipsoids (*TA*=0.05)

The estimated local model coefficients and their 95% confidence interval are shown in Figure 6.40 for the 10-rule GTSK model. θ_0, θ_1 and θ_2 are coefficients for regressors, 1, $y(t\text{-}1)$ and $u(t\text{-}1)$. It is found after comparing each pair of local models that the rule 8 and 10 might have same local models. Confidence intervals for each pair of corresponding local model coefficients have overlap in rule 8 and 10. The observation could be verified by observing Figure 6.39. Regions 8 and 10 are next to each at about the same level of $y(t\text{-}1)$, which makes both have about the coefficient for $y(t\text{-}1)$. On the other hand, region 8 and 10 contain data with opposite signs on $u(t\text{-}1)$ around 0. The term of $u(t\text{-}1)^3$ in Equation (6.11) may be expressed by $\theta_2(t)u(t\text{-}1)$ with $\theta_2(t) = u(t\text{-}1)^2$, which eliminates the effect of signs in $u(t\text{-}1)$.

Based on the above comparison, one may decide to merge rules 8 and 10 to one rule. The merge can be easily operated by remove the line boundary between 8 and 10. Note that the merge operation on regions 8 and 10 only is not possible by choosing a different level of α_c since both regions are resulted from different branch nodes.

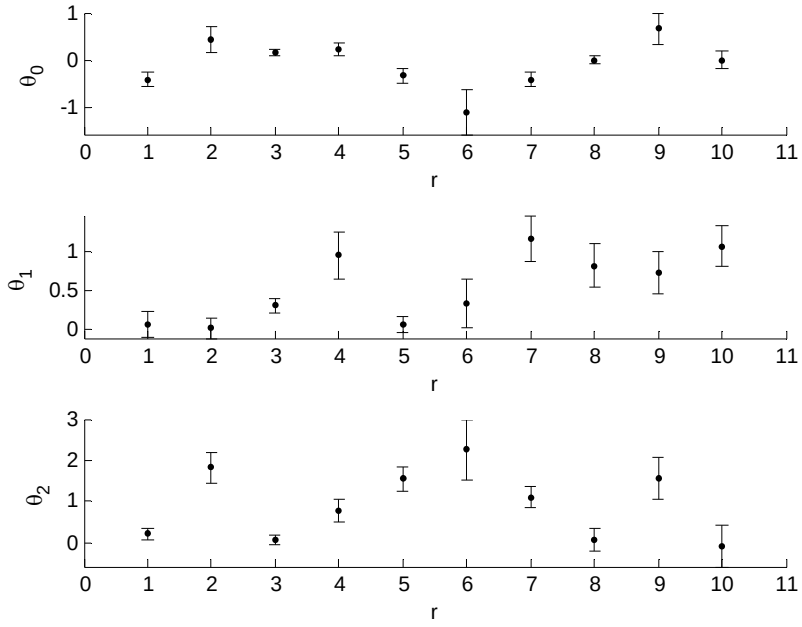


Figure 6.40. Coefficients for local models in the GTSK model in Figure 6.39

The selected antecedent variables $u(t\text{-}1)$ and $y(t\text{-}1)$ are due to nonlinear component detection in Chapter 4. In the following, experiments are conducted to try other antecedents with different complexity to verify the result. The comparison is collected in Table 6.5. Note the MSE without split is 0.289.

Table 6.5.Trials of antecedent variables for Model 3

Antecedent	Number of rules	MSE (Training)
$y(t\text{-}1)$	5	0.271
$y(t\text{-}1)$	19	0.266
$u(t\text{-}1)$	11	0.267

Reduction in MSE is observed for each trial. However, the maximum improvement in MSE is achieved for the GTSK model with the two-dimensional antecedent. In Table 6.3, the training MSE for a 10-rule GTSK model is 0.255, which is smaller than those obtained for either a 19-rule model with antecedent variable, $y(t\text{-}1)$ or a 11-rule model with antecedent variable $u(t\text{-}1)$.

The above two nonlinear modeling examples use the results on order determination and nonlinear component detection from Chapter 4 to construct GTSK models. It is noticed that the results for Model 1 and 3 from Chapter 4 match the 'truth'. In Chapter 4, we also mentioned the 'mistakes' that the order determination could make such as the missing of $u(t\text{-}2)$ for Model 4. Also, the detected nonlinear components contain only the most dominating one such as the $y(t\text{-}2)$ for Model 4. In the following the example, a GTSK model for Model 4 based on determined orders and detected nonlinear components will be created and compared with one based on the 'truth'.

Model 4 (Narendra & Parthasarathy, 1990)

$$y(t)=\frac{y(t-1)\,y(t-2)\,y(t-3)\,u(t-2)\big(y(t-3)-1\big)+u(t-1)}{1+y(t-3)^2+y(t-2)^2}+e(t) \qquad (6.14)$$

The dynamic order analysis in Chapter 1 determines the following values, ny=3, nu=0 and d=1 as shown in Table 4.8. The detected nonlinear component is $y(t\text{-}2)$.

The trial for different level of complexity is collected in Table 6.6, which suggests an 8-rule GTSK model for its minimum Model Error, although other choices for M being 6 and 7 might be also acceptable.

Table 6.6. Trials of a GTSK model for Model 4

M	MSE		Model Error
	Training	Validating	
2	0.0037	0.0041	0.0078
3	0.0031	0.0035	0.0066
4	0.0029	0.0032	0.0061
5	0.0029	0.0031	0.0060
6	0.0028	0.0031	0.0059
7	0.0028	0.0031	0.0059
8	0.0028	0.0031	0.0058
9	0.0028	0.0031	0.0059

Figure 6.41 shows the resultant antecedent partition and membership functions. The resultant parameter estimates for local models are shown in Figure 6.42 along with the 95% confidence interval.

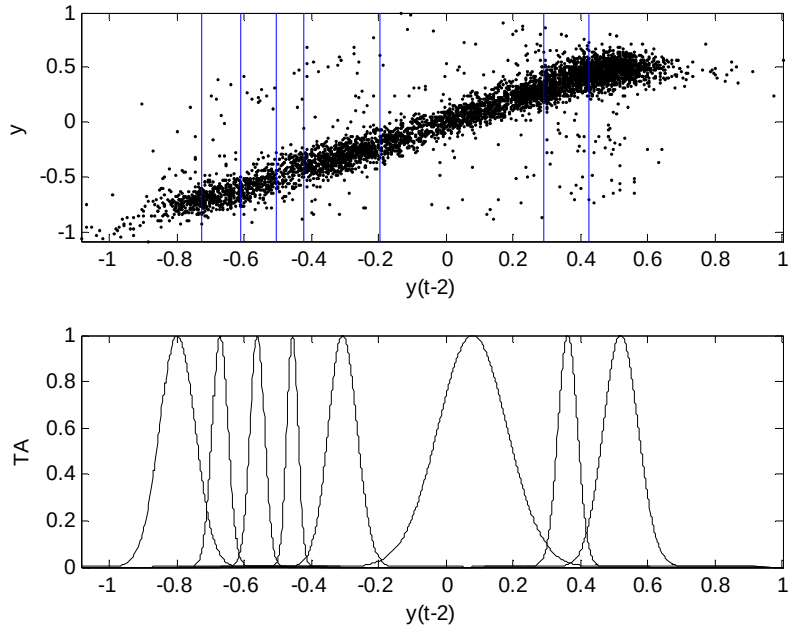


Figure 6.41. Antecedent space partition and TAs for Model 4

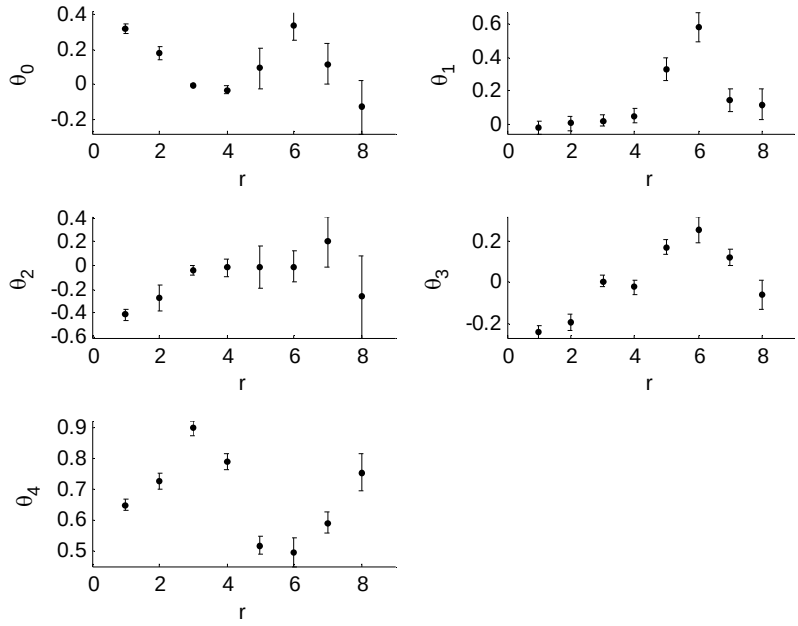


Figure 6.42. Coefficients for local models in the GTSK model in Figure 6.41

The comparison is made to build a GTSK model with *ny*=3, *nu*=1 and *d* =1, which are 'truth' in the Model 4. The antecedent variables are [*u*(*t*-1) *u*(*t*-2) *y*(*t*-1) *y*(*t*-2) *y*(*t*-3)] since they are all nonlinearly expressed in Model 4. The trial results for GTSK models with different complexity are collected in Table 6.7.

Table 6.7. Trials of a GTSK model for Model 4 with all regressors included

M	MSE		Model Error
	Training	Validating	
2	0.0036	0.0039	0.0075
4	0.0032	0.0035	0.0067
5	0.0029	0.0032	0.0067
6	0.0032	0.0034	0.0066
7	0.0031	0.0034	0.0065
8	0.0031	0.0034	0.0065
11	0.0031	0.0034	0.0065
12	0.0031	0.0034	0.0065

The maximum number of rules is 12 due to the choice of $\bar{M}$ being 50. It is interesting to note at first that the number of rules does not change much when antecedent dimension is increased from 1 to 5. Antecedent variables have different values in each rule. For instance, the antecedent variable *u*(*t*-1) has 12 levels. In the conventional TSK fuzzy models, 12 levels implies 12 fuzzy subsets for *u*(*t*-1). In the conventional combinatorial antecedent structure, one might expect to create a fuzzy model out of 5^{12} possible rules. The example shows that the generalized antecedent structure can largely improve the capability of modeling by efficiently representing an antecedent space.

More importantly in this example is to observe that the minim Model Error in Table 6.7 is higher than that in Table 6.6. The model does not become better by using 'true' dynamic orders. It provides a piece of evidence to show that the order determination and nonlinear component detection techniques in Chapter 4 are appropriate to provide the structure information for dynamical nonlinear modeling.

The two-phase flow

There are two sets of input-output definitions for the two phase flow process. In the following test, the input is taken as the air flowrate measurement and the output is the pressure drop measurement. According the Table 4.12, the determined order is defined by *ny*=2, *nu*=0 and *d*=1. The antecedent variables are detected nonlinear components *y*(*t*-2) and *u*(*t*-1). Figure 6.43 shows the antecedent space.

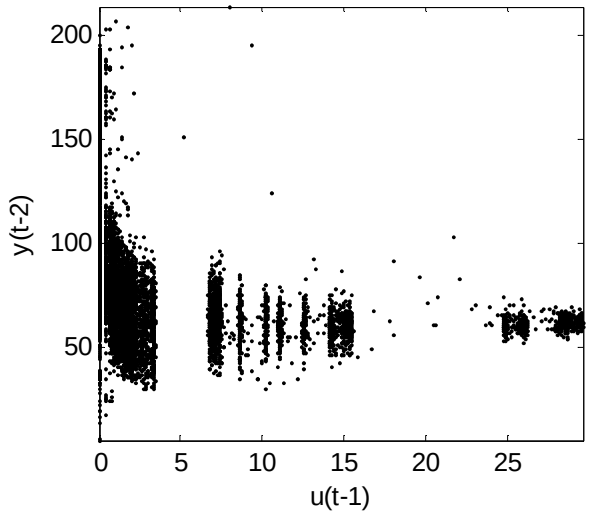


Figure 6.43. Two-dimension antecedent space for the two-phase process

The training data set include 8830 samples. The validation data set include 3000 samples shown in Figure 6.44.

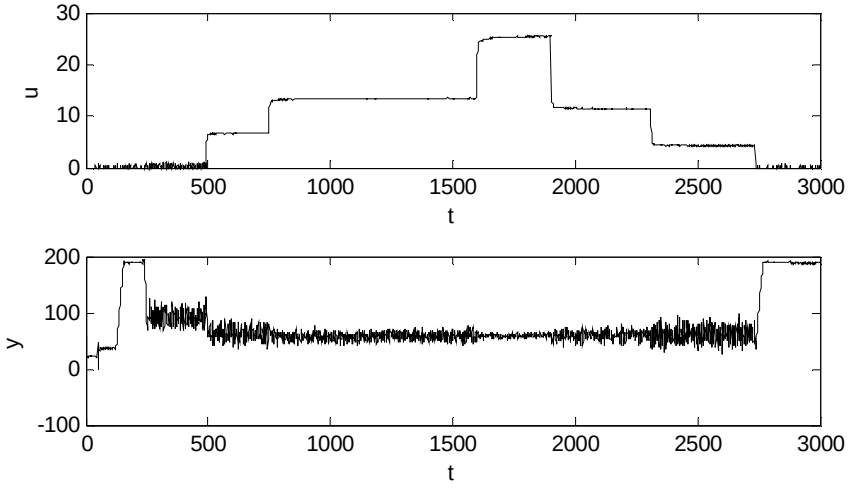


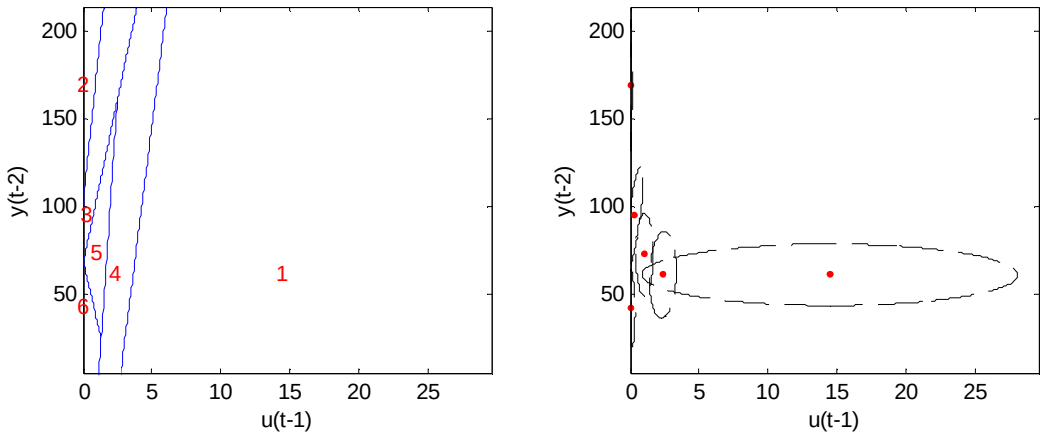
Figure 6.44. Validation data set for the two-phase flow process

The trial results for different model complexity are collected in Table 6.8, where the 6-rule model has the minimum Model Error.

Table 6.8. Trials of a GTSK model for the two phase process

M	MSE		Model Error
	Training	Validating	
1	1.74E+02	1.23E+02	2.97E+02
5	1.50E+02	9.52E+01	2.45E+02
6	1.46E+02	9.72E+01	2.43E+02

Figure 6.45(a) shows the obtained partition of the antecedent space into 6 regions. The initialized truth of antecedent with *TA* = 0.05 is illustrated in Figure 6.45(b).



(a) (b)

Figure 6.45. a) Antecedent space for two phase flow process; b) Ellipsoids (*TA*=0.05)

The coefficients for local models are shown in Figure 6.46.

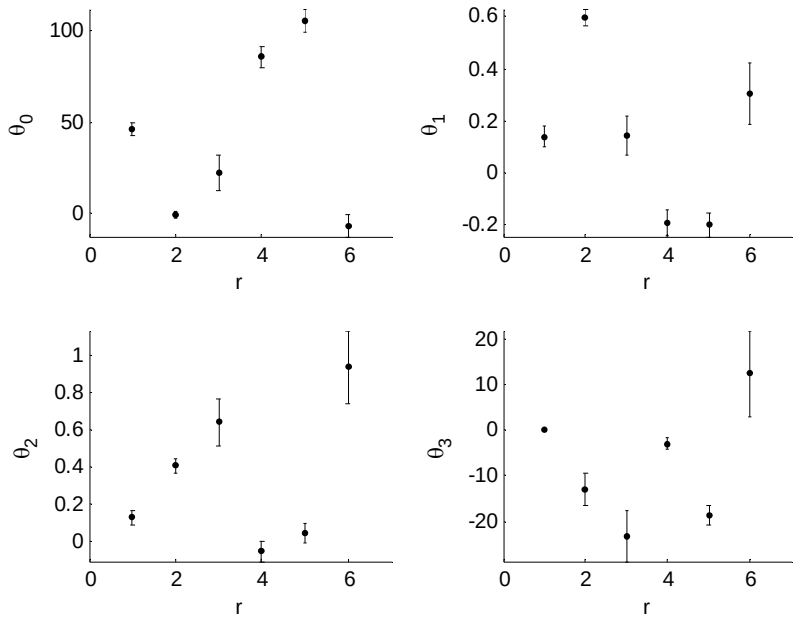


Figure 6.46. Coefficients for local models in the GTSK model in Figure 6.45

The nonlinearity of the process could be verified by the evident coefficient variation across rules shown in Figure 6.46. Rule 1 covers the most of the antecedent space and describes the process behavior operated under high air flowrate. High air flow blows water out of the vertical pipe creating an annular flow pattern. Varying the air flowrate when water is out barely affects the pressure drop. The negligible effect is reflected by the small coefficient value of θ_3 in Rule 1. When the process is operated in an intermediate air flowrate, with air and water coexisting in the pipe, varying the air flowrate will affect the density of the air-water mix, which in turn affects the pressure drop. The process behavior observed in intermediate air flowrate is primarily described by Rule 4. The other Rules, 2, 3, 5 and 6 describe the process behavior operated under low air flowrate and transition behavior from intermediate air flowrate to low. When the air flowrate is further decreased from the intermediate region, not only the density of the air-water mix is changed but also water starts accumulating in the pipe. With the increased water holdup, pressure drop is increased. Part of the water accumulation operation is described by Rule 3 and 5. Rule 6 features low pressure drop and low air flowrate. The low pressure drop is due to previous high air flowrate conditions, which

blows water out of pipe. Therefore, Rule 6 describes the transitional behavior from high to low air flowrate. Rule 2 is also featured with low air flowrate but it has high pressure drop. Therefore, Rule 2 describes the process behavior of further reducing the airflow rate when a certain amount of water has been accumulated in the pipe.

Distillation Column

The parameter value estimation for the distillation column is also based on the previously determined dynamic order and selected nonlinear component candidate sets in Chapter 3. The parameter value estimation is conducted for each output.

Overhead Concentration, x_D (= y_1)

For the output y_1, the determined regressors are [$y_1(t\text{-}1)$ $y_1(t\text{-}2)$ $y_1(t\text{-}3)$ $y_2(t\text{-}3)$ $u_1(t\text{-}3)$], and antecedent variables are to be chosen from [$y_1(t\text{-}2)$ $y_2(t\text{-}3)$] or [$y_2(t\text{-}3)$]. Five thousand data samples in the training set are used to identify a GTSK model and another set of 3000 are used for validation. Both data set are used to compare different choices of antecedent variables and model complexity in terms of number of rules. The result is summarized in Table 6.9.

Table 6.9. Training and validation results for the GTSK model on y_1

Antecedent	No. of Rules	Training (MSE)	Validation (MSE)	Total MSE
[$y_1(t\text{-}2)$ $y_2(t\text{-}3)$]	2	2.71e-5	2.76e-5	5.48e-5
	3	2.68e-5	2.75e-5	5.43e-5
	4	2.65e-5	2.74e-5	5.40e-5
	6	2.62e-5	2.85e-5	5.47e-5
[$y_2(t\text{-}3)$]	2	2.70e-5	2.80e-5	5.50e-5
	3	2.67e-5	2.76e-5	5.43e-5
	4	2.65e-5	2.77e-5	5.42e-5
	5	2.64e-5	2.77e-5	5.41e-5
	10	2.62e-5	2.81e-5	5.43e-5

Observed from Table 6.9, the best one is a 4-rule GTSK model with both $y_1(t\text{-}2)$ and $y_2(t\text{-}3)$ included in the antecedent. The antecedent space is shown in Figure 6.47. The corresponding antecedent space partition and the truth of antecedent at TA =0.05 for each rule are shown in Figure 6.48.

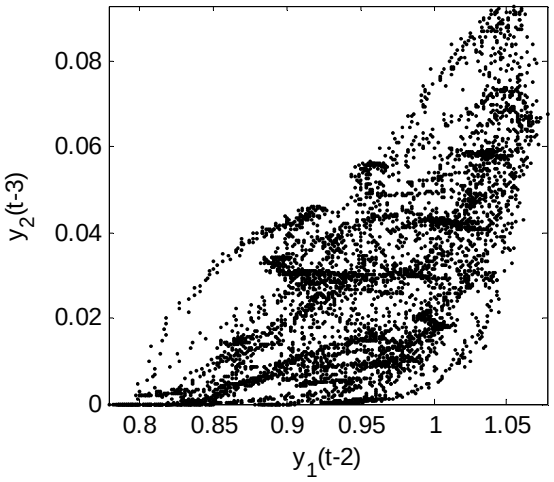


Figure 6.47. Two-dimension antecedent space for y_1 of the distillation column

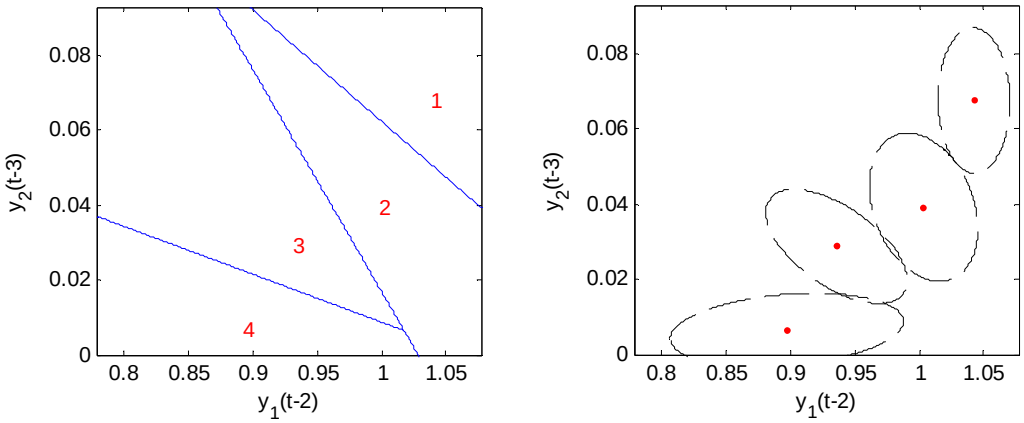


Figure 6.48. a) Antecedent space partition for y_1 of the distillation column;
b) Ellipsoids (TA=0.05)

Figure 6.49 shows the local model coefficients and their 95% confidence interval. It seems possible to simplify the GTSK model by merging rule 2 and rule 3 due to the same argument mentioned above to merge rule 8 and 10 in Figure 6.39.

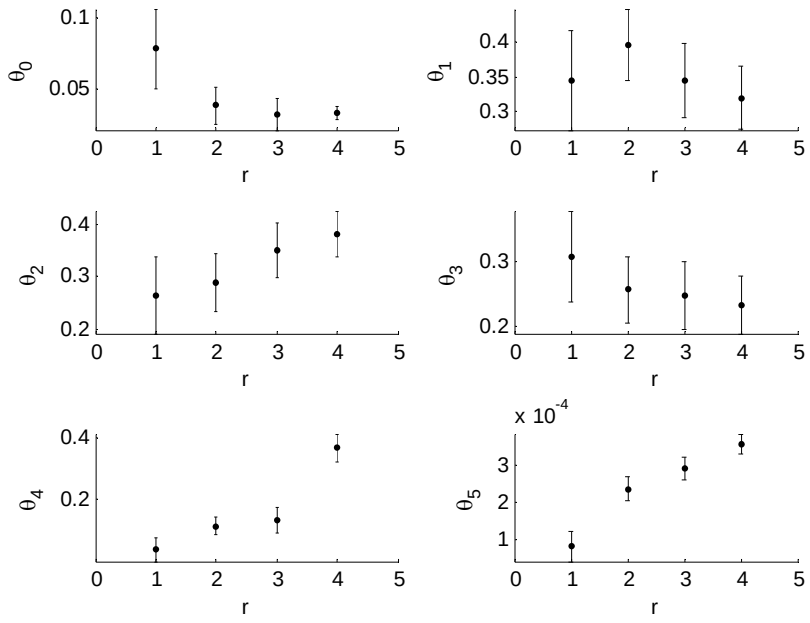


Figure 6.49. Coefficients for local models in the GTSK model in Figure 6.48

Confidence interval overlaps are observed for θ_1 and θ_3 across all rules. The observation may indicate constant coefficients for regressors $y_1(t\text{-}1)$ and $y_1(t\text{-}3)$. Then a hybrid model structure may be defined

$$y(t) = a_1 y_1(t-1) + a_2 y_1(t-3) + f\left(y_1(t-2), y_2(t-3), u_1(t-3)\right) \qquad (6.15)$$

Overlap of confidence interval is also observed for θ_2. One might also decide that coefficient for $y_1(t\text{-}2)$ is a constant across all rules. However, a constant coefficient to $y_1(t\text{-}2)$ is unable to take $y_1(t\text{-}2)$ out of the nonlinear part and add another linear term like $a_3 y_1(t\text{-}2)$ since the regressor, $y_1(t\text{-}2)$ is included in the antecedent. One possibility is to take $y_1(t\text{-}2)$ out of antecedent such as the second best model in Table 6.9. It is a 5-rule model with only one antecedent variable $y_2(t\text{-}3)$. The corresponding antecedent partition is shown in Figure 6.50. Confidence interval overlap for θ_1, θ_2 and θ_3 is observed in Figure 6.51. Since $y_1(t\text{-}2)$ is no longer included in the antecedent, it is then possible to take $y_1(t\text{-}2)$ out of the nonlinear part in Equation (6.13) and redefine a hybrid model by

$$y(t) = a_1 y_1(t-1) + a_2 y_1(t-2) + a_3 y_1(t-3) + f\left(y_2(t-3), u_1(t-3)\right) \qquad (6.16)$$

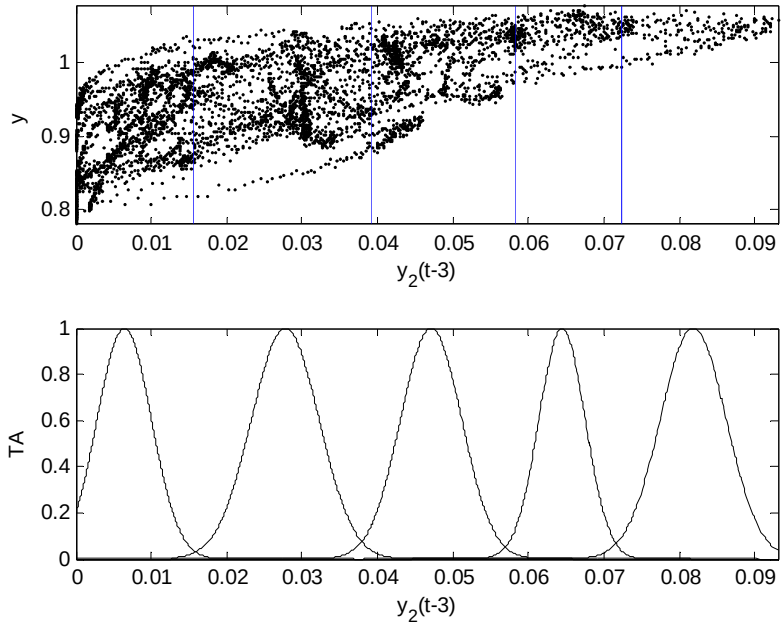


Figure 6.50. Antecedent space partition and TAs for y_1 of the distillation column

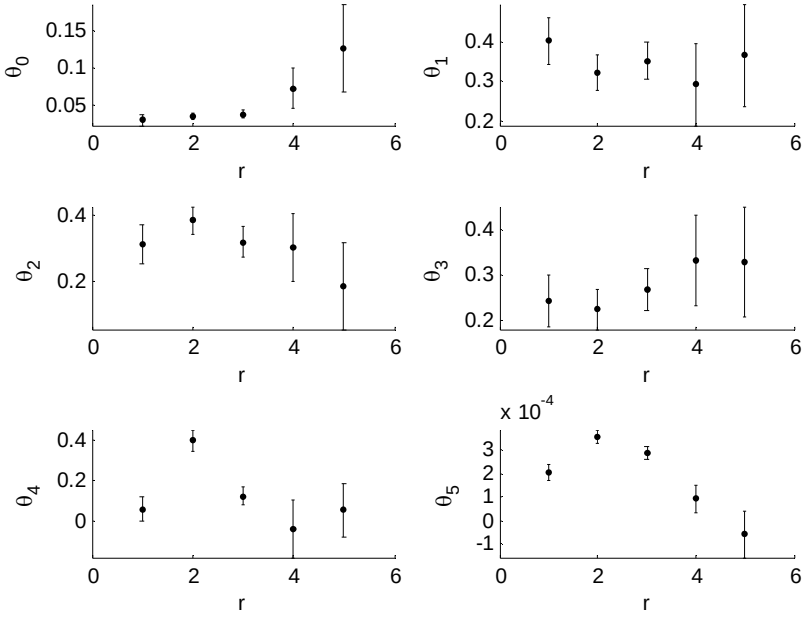


Figure 6.51. Coefficients for local models in the GTSK model in Figure 6.50

Bottom Concentration, x_B (= y_2)

The same procedure is also applied to the output, y_2. For the output y_2, the determined regressors are [$y_2(t\text{-}1)$ $y_2(t\text{-}2)$ $u_1(t\text{-}3)$ $u_2(t\text{-}1)$]. There are two sets of antecedent variables to be compared; [$y_2(t\text{-}1)$ $u_2(t\text{-}1)$] and [$u_2(t\text{-}1)$]. The result is summarized in Table 6.10.

Table 6.10. Training and validation results for the GTSK model on y_2

Antecedent	No. of Rules	Training (MSE)	Validation (MSE)	Model Error
[$y_2(t\text{-}1)$ $u_2(t\text{-}1)$]	1	1.36e-7	1.10e-7	2.46e-7
	2	1.17e-7	9.70e-8	2.14e-7
	3	1.13e-7	1.05e-7	2.19e-7
[$u_2(t\text{-}1)$]	1	1.36e-7	1.10e-7	2.46e-7
	2	1.17e-7	9.80e-8	2.15e-7
	3	1.14e-7	1.05e-7	2.19e-7

Observed from Table 6.10, the best one is a 2-rule model with both $y_2(t\text{-}1)$ and $u_2(t\text{-}1)$ as antecedent variables. The next choice is a 2-rule model with only $u_2(t\text{-}1)$ as the antecedent variable. We first explore the best choice. The resultant antecedent space partition and the truth of antecedent at *TA* =0.05 are shown in Figure 6.52.

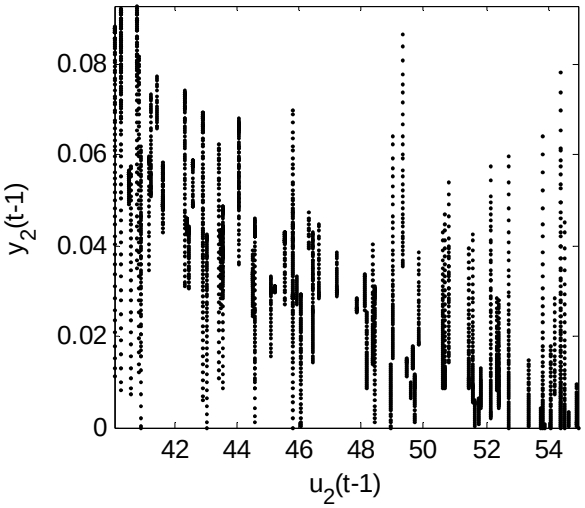


Figure 6.52. Two-dimension antecedent space for y_2 of the distillation column

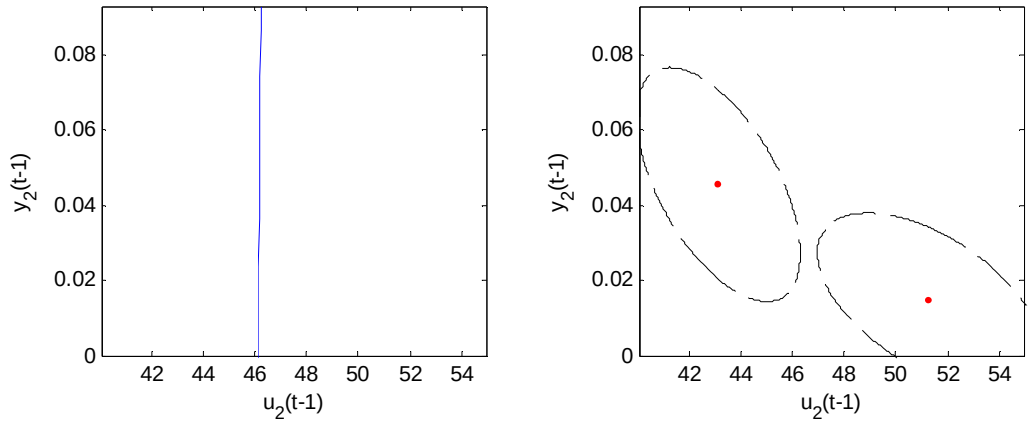


Figure 6.53. a) Antecedent space partition output y_2 of the distillation column;
b) Ellipsoids (*TA*=0.05)

It is observed in Figure 6.53, the separation boundary is almost vertical. The observation suggests a lower antecedent dimension with only $u_2(t\text{-}1)$, which in this case matches the second best choice in Table 6.9. Figure 6.54 shows the resultant antecedent space partition with only one antecedent variable, $u_2(t\text{-}1)$. Figure 6.55 shows the local model coefficients and their 95% confidence interval for the 2-rule model.

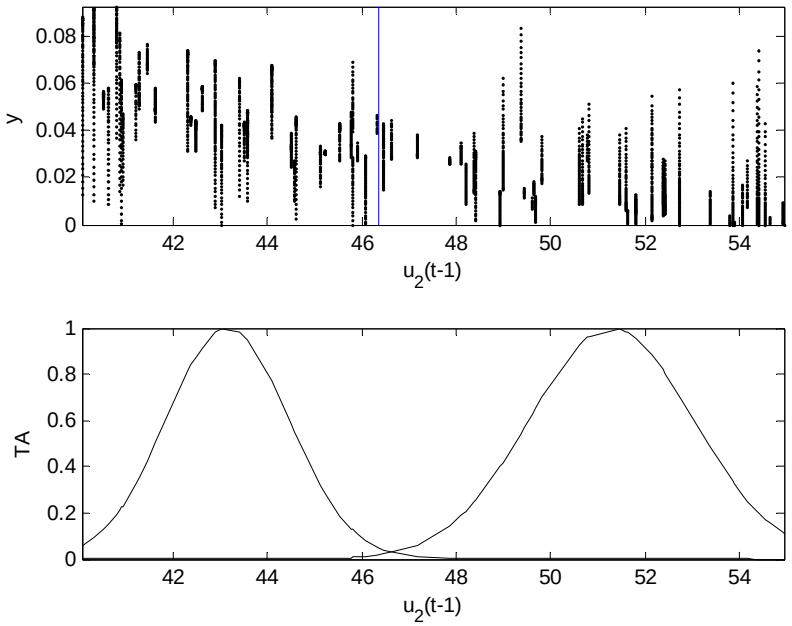


Figure 6.54. Antecedent space partition and TAs for y_2 of the distillation column

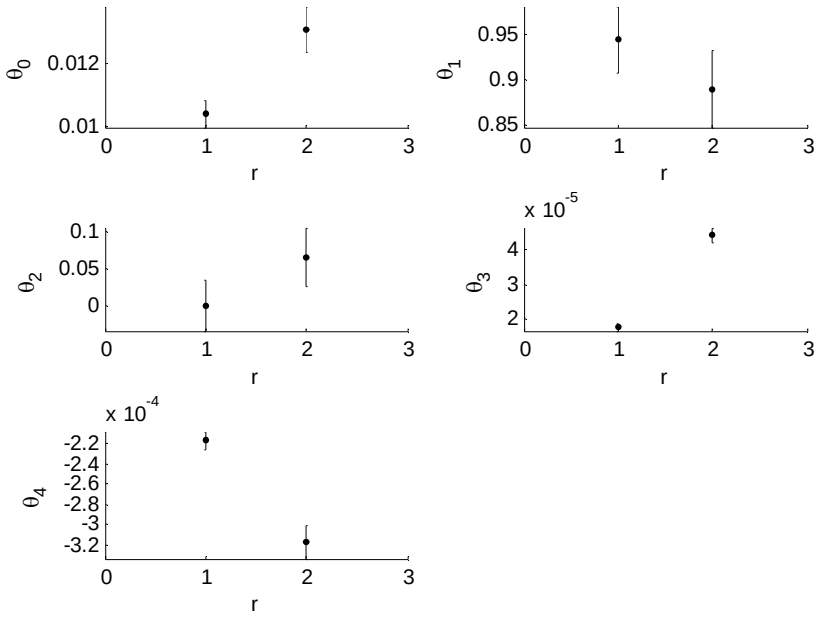


Figure 6.55. Coefficients for local models in the GTSK model in Figure 6.50

Figure 6.55 shows that the nonlinearity in the 2-rule model is due to nonlinear coupling between $u_1(t\text{-}3)$ and $u_2(t\text{-}1)$. Coefficients for regressors, $y_2(t\text{-}1)$ and $y_2(t\text{-}2)$ could be considered as constants. The following hybrid structure could then be defined for output, y_2.

$$y(t) = a_1 y_2(t-1) + a_2 y_2(t-2) + f\left(u_1(t-3), u_2(t-1)\right) \tag{6.17}$$

A MIMO (2,2) Model

The above procedure treats each output individually. Output y_1 is described by a 5-rule model, where the only antecedent variable is $y_2(t\text{-}3)$. The antecedent space partition is shown in Figure 6.50. Output y_2 is described by a 2-rule model, which has an antecedent variable $u_2(t\text{-}1)$ with the antecedent space partition shown in Figure 6.54. These two GTSK models could then be considered a MIMO(2,2) model with two inputs and outputs. It is possible to construct a more compact MIMO (2,2) GTSK model from the obtained two GTSK models. The construction starts by compounding the antecedent space. The antecedent space in a MIMO(2,2) GTSK model will have dimension 2, which

includes antecedent variables from both single-output GTSK models.

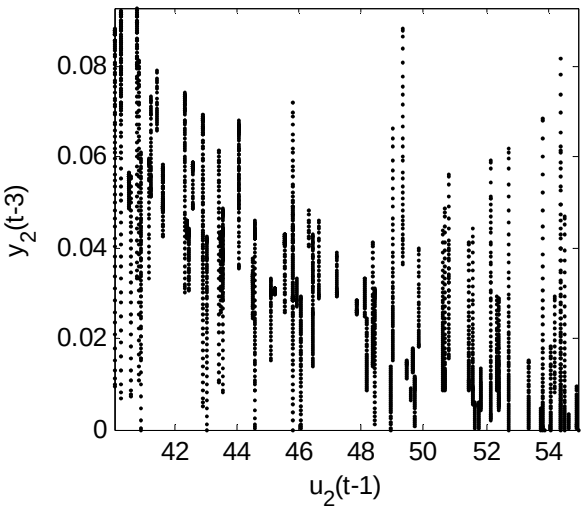


Figure 6.56. Two-dimension antecedent space for the MIMO(2,2) GTSK model

The extended antecedent space ($y_2(t\text{-}3)$, $u_2(t\text{-}1)$) will be partitioned by linear boundaries resulted from an exhaustive combination of obtained linear boundaries for both antecedent variables $y_2(t\text{-}3)$ and $u_2(t\text{-}1)$. Figure 6.57 shows the partitioned antecedent space for the MIMO(2,2) model.

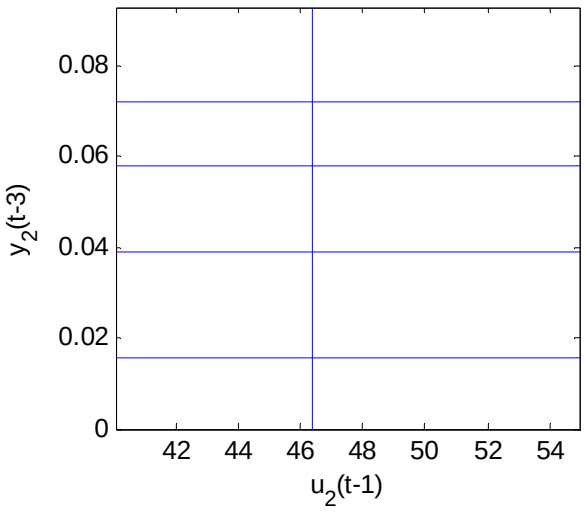


Figure 6.57. Antecedent space partition for the MIMO(2,2) GTSK model

Local models for each region in Figure 6.56 will be taken from individual models respectively. Note that the above mentioned construction only provides a more compact model description but not extra modeling accuracy or interpretability. Actually, the interpretability is reduced. From either single-output GTSK model, it is clear to tell which regressor has the dominant affect on the nonlinearity for the corresponding output. In this case, $y_2(t\text{-}3)$ affects y_1 nonlinearly and $u_2(t\text{-}1)$ affects y_2 nonlinearly. The decoupled connection is however smeared in the MIMO format, one can only tell that both $y_2(t\text{-}3)$ and $u_2(t\text{-}1)$ are affecting y_1 and y_2 nonlinearly. The advantage of having a MIMO format is to provide a general model description for the subsequent analysis and applications.

If both individual models share same antecedent variables, one could create MIMO models directly by solving the MIMO version of SRP in Section 5.6. It is however not the case for the distillation column example.

CHAPTER VII

SUMMARY, CONCLUSIONS AND FUTURE RESEARCH RECOMMENDATIONS

7.1 Summary

In this work, a generalized antecedent structure is proposed to replace the conventional combinatorial structure in a TSK fuzzy model. One of new features in the proposed antecedent structure is the extra degree of freedom in angle, which makes it possible to rotate the active region of a rule. In this work, active regions have the shape of ellipsoids. The rotation improves the coverage efficiency of rules. The improvement is achieved by allowing active regions to be more flexibly shaped according to function nonlinearity, which replaced the forced shapes oriented along with coordinates in the conventional antecedent structure. As a consequence, the improved rule coverage efficiency is expected to extend the application of TSK fuzzy models to higher dimension problems.

Another feature in the proposed antecedent structure is the separation of antecedent dimension from the overall dimension for a GTSK model. The distinction is a direct effort to deal with “the curse of dimensionality” and makes it even possible to apply the conventional TSK fuzzy models to high dimension problems so long as the corresponding antecedent dimension is acceptable. More importantly, the dimension separation is made applicable in this work by the proposed method to detect nonlinear components, which defines antecedent variables and antecedent dimension.

One focus of this work is to use the resultant GTSK model featured with the proposed antecedent structure to model nonlinear dynamic processes. A systematic approach is provided to create a GTSK model from input-output data. The overall dimension of a GTSK model defined by dynamic orders is determined by a selection procedure based on the recursive estimation of spatially rearranged data using the proposed SNNR method. The recursive estimation on SNNR treated data is also used to detect nonlinear components, which in this work refer to the regressors having dominating impact on the nonlinear behavior of a process.

The parameter estimation for the GTSK model with the proposed antecedent structure is initialized by recognizing ellipsoids out of a partitioned antecedent space. The partition in this work is conducted recursively. In each step, a spliiting and regression problem is solved by the proposed procedure. It is shown at least that the solution is a local optimum for the defined problem. Model parameters can be further tuned by a Newton's method that solves a constrained optimization problem. Constraints are imposed on the positive definiteness of shape matrices in the proposed antecedent structure.

7.2 Conclusions

The proposed SNNR method rearranges time-sequenced raw data by spatial order. The SNNR treatment is demonstrated to be able to artificially reduce the parameter variation caused by nonlinearity. The effectiveness of SNNR is verified by the reduced MSE on rearranged output and its prediction. It should be noted that the SNNR used in this work is only to prepare raw data for subsequent analysis on dynamic order determination and nonlinear component detection. The reduced MSE due to SNNR by no means suggests an alternative approach for recursive estimation for better prediction. Simply, prediction is a temporal concept and only applicable for the time-sequenced data, using past observation to predict the future behavior. However, the time sequence is no longer preserved in SNNR treated data, where the computation using recursive estimation equations should not be interpreted as prediction.

The proposed dynamic order determination based on SNNR is able to discover influential regressors. The method is however not perfect and makes ‘mistakes’. However, it provides better results in terms of number of ‘mistakes’ and sensitivity to noise, compared to the method using time-sequenced data. Comparing to other methods like geometric method, the proposed method performs less affected by noise.

The nonlinear components detection finds the regressors that exhibit dominating impact on process nonlinearity. The obtained results are verified by comparing to testing models and further verified in Chapter 6 by trying different antecedent variables in GTSK models.

The proposed solving procedure for separation boundaries shows better performance than other solvers (Newton’s method and Nelder-Mead) in terms of locating a global optimal solution for a given multimodal optimization problem. However, the proposed solver is highly designated to the separation and regression problem defined in this work. It should not be understood that a better optimizer is offered to replace Newton’s or Nelder-Mead method in general.

Model parameters for antecedents and consequents are initialized once the antecedent space partition is achieved. The initialization uses only data confined in a recognized subspace to compute for the corresponding rule, centroid, shape matrix and local model coefficients. Therefore, it is not surprising to observe that initialized rules exhibit limited interactions, which make rules more modular and interpretable. The interpretability could be verified by comparing the behavior of a rule with the local behavior of the nonlinear model.

The overall modeling accuracy of a GTSK model could be improved by further adjusting model parameters in an optimization scheme, which is conducted in this work by solving a constrained optimization problem. As observed in this work, the improvement in terms of modeling accuracy is achieved by interaction increase between rules. The observation is intuitively reasonable and increased interaction can at least make the GTSK model behave smoother. On the other hand, interaction increase reduces

the modularity. A rule alone is not sufficient to describe the local behavior of a nonlinear process. Therefore, individual rules become less interpretable. Users should be aware of the effect of parameter optimization on modularity and interpretability. If preference is set on modeling accuracy, one might accept a less interpretable model. On the other hand, one might prefer a modular model if, for instance, model management is concerned. It is possible that the obtained model might be augmented by deleting obsolete rules or adding new rules in the model management phase. It is then desired that any alteration has only local impact, which is possible if coupling between modules is limited.

The proposed parameter estimates are much better in terms of modularity compared to those estimated based on random initialization. The rules in GTSK models resulted from optimization starting from random initialization barely retain any modularity.

The obtained GTSK models exhibit desired behavior with ellipsoids expressing the truth of antecedent oriented according to function nonlinearity. The rule distribution in a GTSK model is also reasonable. Rules are given to more nonlinear portion of a function or to approximate a nonlinear function in a finer scale. These observations imply that the complexity of the resultant GTSK models in this work is determined by function nonlinearity rather than problem dimension. This is desired behavior, which could be the basis to support applying GTSK models to high dimension problems.

The conventional interpretability in individual antecedent variables will be lost due to the additional degree of freedom that combines all antecedent variables. However, the interpretability of the antecedent as a whole is still meaningful. A rule antecedent can be interpreted as a function that defines active region for the consequent model. It is also shown that it is possible to regain the conventional interpretability by converting the new GTSK model into the conventional format by defining several new variables. Then, interpretation in new variables could be defined.

7.3 Future Research Recommendations

This work provides a systematic approach starting from detecting data structure

and ending at a GTSK model. Many aspects in this work could be further investigated.

In this work, the dynamic order determination method is limited to the ARX type of nonlinear dynamic processes. The limitation is due to the SNNR operation that needs access to measurements. It is desired that the order determination technique could be generalized to include a broader range of model structures, where lagged prediction or residuals might be included as candidate regressors to be tried. They are, however, unavailable from measurements directly. A recommended procedure is to start the generalization by first considering an ARX structure. The obtained prediction for the rearranged data could then be used to compute residuals. Then, the SNNR operation on prediction and residuals becomes possible.

The SNNR operation is this work is conducted in a brute-force manner, which finds the exact nearest neighbor to a point in each step by computing its distance to all other points and finding the minimum. Further investigation is desired to improve the efficiency of the SNNR operation.

The nonlinear component detection in this work uses an exhaustive search to try all possible combinations of regressors, which would cause scalability problem when dealing with high dimension problem. Therefore, improving the search method for nonlinear component detection is also worthy of further investigation.

The order determination method is not perfect. The order determination is based on spatially rearranged data. The rearrangement is however based on the assumption of negligible high-order influence of regressors on parameter variation. It is then a research focus in the future to relax the assumption by considering higher order influence, or it is more desired to find an approach to test the assumption.

This work used all variables up to a certain order, not just the sparse subset found as important. Next pursuit should explore using only the variables found to be regressors.

Algorithms to solve the separation and regression problems could also be a research topic. A heuristic method appearing in a recent paper by (Magnani & Boyd,

2009) might be an alternative to solve the problem and worthy of investigation.

This work suggested that the obtained GTSK models can be further simplified by merging parameter-like rules or being redefined as a hybrid including an explicit linear structure. The later practice could be unified with the dynamic order determination and nonlinear component detection to allow users to gain more insight into the model structure embedded in data.

Another future research topic is on model management to let the model automatically adjust according to the dynamic behavior variation of the process to be modeled. One could adapt the local model coefficients or modify the interaction between rules to eliminate the mismatch. It is also possible to add new rules if mismatch is caused by never observed behavior. It is desired in the future study to find a systematic approach to reduce the mismatch and preserve interpretability by minimum modification of a model via evaluating all possible modifications. Retaining interpretability will need extra constraints to restrict the interaction between rules.

The proposed modeling approach is tested on several benchmark problems and a laboratory scale process. A possible future investigation is to broaden its application to industrial scale problems. The application could focus on different aspects. GTSK models could be used only for prediction and monitoring. One might be interested only in finding the overall problem dimension. It is also possible to investigate the structure embedded in input-output data expressed by a hybrid structure with both linear and nonlinear parts. A very important application is to use obtained GTSK models to design controllers (Sala, Guerra & Babuška, 2005). There have been many different ways proposed to design fuzzy model based controllers; adaptive nonlinear control using feedback linearization (Feng 2002; Feng and Chen 2005; Qi and Brdys 2008), linear matrix inequalities based parallel distributed compensator (Tanaka & Wang, 2001), gain scheduling-like multiple model approach (Hunt & Johansen, 1997) and nonlinear model predictive control (Abonyi, Nagy & Szeifert, 2001; Fischer, Schmidt & Kavsek-Biasizzo, 1997; Huang, Lou, Gong & Edgar, 2000). A comprehensive investigation and comparison of these methods is desired.

REFERENCES

Abonyi, J., Nagy, L. & Szeifert, F. (2001). "Fuzzy model-based predictive control by instantaneous linearization." Fuzzy Sets and Systems 120(1): 109-122.

Andersen, H. C., Lotfi, A. & Westphal, L. C. (1998). "Comments on “Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems”." IEEE Transactions on Neural Networks 9(6): 1529-1531.

Box, G. E. P., Jenkins, G. M. & Reinsel, G. C. (1994). Time Series Analysis: Forecasting and Control. Englewood Cliffs, Prentice-Hall.

Boyd, S. P., Balakrishnan, V., Feron, E. & Ghaoui, L. E. (1994). Linear matrix inequalities in system and control theory. Philadelphia Society for Industrial and Applied Mathematics.

Boyd, S. P. & Vandenberghe, L. (2004). Convex Optimization. New York, Cambridge University Press.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984). Classification and regression trees. Belmont, Wadsworth International Group.

Cordon, O., Herrera, F., Gomide, F., Hoffmann, F. & Magdalena, L. (2001). Ten years of genetic fuzzy systems: current framework and newtrends. IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2001). Introduction to algorithms. New York, The MIT Press.

Dickerson, J. A. & Kosko, B. (1996). "Fuzzy function approximation with ellipsoidal rules." IEEE Transactions On Systems, Man, And Cybernetics - Part B: Cybernetics 26(4): 542 - 560.

Du, H. & Zhang, N. (2008). "Application of evolving Takagi–Sugeno fuzzy model to nonlinear system identification." Applied Soft Computing 8(1): 676-686.

Fischer, M., Nelles, O. & Isermann, R. (1998). "Adaptive predictive control of a heat exchanger based on a fuzzy model." Control Engineering Practice 6: 259-269.

Fischer, M., Schmidt, M. & Kavsek-Biasizzo, K. K. (1997). Nonlinear predictive control based on the extraction ofstep-response models from Takagi-Sugeno fuzzy systems. American Control Conference, Albuquerque, NM, USA.

Guenounou, O., Belmehdi, A. & Dahhou, B. (2009). "Multi-objective optimization of TSK fuzzy models." Expert Systems with Applications 36(4): 7416-7423.

Haber, R. & Unbehauen, H. (1990). "Structure identification of nonlinear dynamic systems - A survey on input/output approaches." Automatica 26(4): 651-677.

Hagan, M. T., Demuth, H. B. & Beale, M. H. (2002). Neural Network Design. Boston, PWS Pub.

Han, G. G., Pardalos, P. & Ye, Y. (1992). "On the Solution of Indefinite Quadratic Problems using an Interior-Point Algorithm." Informatica 3: 474-496.

Hartmann, B. & Nelles, O. (2009). On the smoothness in local model networks. American Control Conference. St. Louis.

Hastie, T., Tibshirani, R. & Friedman, J. (2001). The elements of statistical learning. New York, Springer.

He, X. & Asada, H. (1993). A new method for identifying orders of input-output models for nonlinear dynamic systems. Proceedings of the American Control Conference.

Huang, Y. L., Lou, H. H., Gong, J. P. & Edgar, T. F. (2000). "Fuzzy model predictive control." IEEE Transactions on Fuzzy Systems 8(6): 665-678.

Hunt, K. J. & Johansen, T. A. (1997). "Design and analysis of gain-scheduled control using local controller networks." International Journal of Control 66(5): 619 - 651.

Iyer, M. S. & Rhinehart, R. R. (1999). "A Method to Determine the Required Number of Neural Network Training Repetitions." IEEE Transactions on Neural Networks 10(2): 427 - 432.

Johansen, T. A. & Foss, B. A. (1993). "Constructing NARMAX models using ARMAX models." International Journal of Control 58(5): 1125-1153.

Kawamoto, S. (1992). An approach to stability analysis of second order fuzzy systems. First IEEE international conference on fuzzy systems.

Kosko, B. (1994). "Fuzzy systems as universal approximators." IEEE Transactions On Computers 43(11): 1329 - 1333.

Lee, E. S. & Zhu, Q. (1995). Fuzzy and evidence reasoning. Heidelberg, Physica-Verlag.

Lee, M. L., Chung, H. Y. & Yu, F. M. (2003). "Modeling of hierarchical fuzzy systems." Fuzzy Sets and Systems 138: 343-361.

Leith, D. J. & Leithead, W. E. (1999). "Analytic framework for blended multiple model systems using linear local models." International Journal of Control 72(7/8): 605 - 619.

Lin, C. (2008). "An efficient immune-based symbiotic particle swarm optimization learning algorithm for TSK-type neuro-fuzzy networks design." Fuzzy Sets and Systems 159(21): 2890-2909.

Lin, C. & Xu, Y. (2006). "A hybrid evolutionary learning algorithm for TSK-type fuzzy model design." Mathematical and Computer Modelling 43(5-6): 563-581.

Lind, I. & Liung, L. (2008). "Regressor and structure selection in NARX models using a structured ANOVA approach." Automatica 44: 383 - 395.

Liu, P. & Li, H. (2005). "Hierarchical TS fuzzy system and its universal approximation." Information Sciences 169: 279-303.

Ljung, L. (1999). System identification: Theory for the user. New York, Prentice Hall PTR.

Ljung, L. & Soderstrom, T. (1986). Theory and Practice of Recursive Identification. Cambridge, The MIT Press.

Magnani, A. & Boyd, S. P. (2009). "Convex piecewise-linear fittting." Optim Eng 10: 1-17.

Mamdani, E. H. (1974). Application of fuzzy algorithms for control of a simple dynamic plant. Proceedings IEEE.

Mastorocostas, P. A. & Theocharis, J. B. (2002). "A Recurrent Fuzzy-Neural Model for Dynamic System Identification." IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 32(2): 176-190.

Miller, A. J. (1990). Subset selection regression. New York, Chapman and Hall.

Molina, C., Sampson, N., Fitzgerald, W. J. & Niranjan, M. (1996). Geometric techniques for finding the embedding dimension of time series. Neural Networks for signal processing [1996] VI. Proceedings of the 1996 IEEE signal processing society workshop. 1: 161 - 169.

Narendra, K. S. & Parthasarathy, K. (1990). "Identification and control of dynamical systems using neural networks." IEEE transactions on neural network 1(1): 4 - 27.

Nelles, O. (2001). Nonlinear System Identification. Berlin, Springer.

Nelles, O. (2006). Axes-oblique partitioning strategies for local model networks. IEEE International Symposium on Intelligent Control. Munich, Germany: 2378-2383.

Nelles, O. & Isermann, R. (1996). Basis function networks for interpolation of local linear models. IEEE Conference on Decision and Control (CDC). Kobe, Japan: 470-475.

Ou, J. (2001). Grouped neural network model-predictive control and its experimental distillation application Chemical Engineering. Stillwater, Oklahoma State University. Ph.D.

Ou, J. & Rhinehart, R. R. (2002). "Grouped Neural Network Modeling for Model Predictive Control." ISA Transactions 41(2): 195 - 202.

Pomares, H., Rojas, I., González, J. & Prieto, A. (2002). "Structure Identification in Complete Rule-Based Fuzzy Systems." IEEE Transactions on fuzzy systems 10(3): 349 - 359.

Pronzato, L., Wynn, H. P. & Zhigljabsky, A. A. (2000). Dynamic search - Applications of dynamical systems in search and optimization. New York, Chapman & Hall.

Rhodes, C. & Morari, M. (1995). Determining the model order of nonlinear input/output systems directly from data. Proceedings of the American Control Conference.

Roger, J. S. & Sum, C. T. (1993). "Functional equivalence between radial basis function networks and fuzzy inference system." IEEE Transactions on Neural Networks 4(1): 156-159.

Sala, A., Guerra, T. M. & Babuška, R. (2005). "Perspectives of fuzzy systems and control." Fuzzy Sets and Systems 156(3): 432-444.

Seborg, D. E. & Henson, M. A. (1996). Nonlinear Process Control. New York, Prentice Hall.

Shorten, R., Smith, R. M.-., Bjorgan, R. & Gollee, H. (1999). "On the interpretation of local models in blended multiple model structures." International Journal of Control 72(7/8): 620 - 628.

Smith, R. M. & Johansen, T. A., Eds. (1997). Multiple Model Approaches to Modeling and Control. New York, Taylor & Francis.

Sugeno, M. & Kang, G. T. (1986). "Fuzzy modeling and control of multilayer incinerator." Fuzzy Sets and Systems 18: 329-346.

Takagi, T. & Sugeno, M. (1985). "Fuzzy identification of systems and its application to modeling and control." IEEE Transactions on Systems, Man and Cybernetics 15: 116-132.

Tanaka, K. & Wang, H. O. (2001). Fuzzy control systems design and alaysis: A linear matrix inequality approach. New York, John Wiley & Sons, Inc.

Vernieuwe, H., Baets, B. D. & Verhoest, N. E. C. (2006). "Comparison of clustering algorithms in the identification of Takagi–Sugeno models: A hydrological case study." Fuzzy Sets and Systems 157(21): 2876-2896.

Wang, N. & Yang, Y. (2009). "A fuzzy modeling method via Enhanced Objective Cluster Analysis for designing TSK model." Expert Systems with Applications 36(10): 12375-12382.

Yen, J. & Wang, L. (1999). "Simplifying fuzzy rule-based models using orthogonal transformation methods." IEEE Transactions On Systems, Man, And Cybernetics - Part B: Cybernetics 29(1): 13 - 24.

Young, P. (1984). Recursive estimation and time-series analysis: An introduction New York, Springer - Verlag.

Young, P. (1993). Time variable and state dependent modeling of non-stationary and nonliner time series. Development in time series analysis. T.S.Rao. New York, Chapman and Hall: 374 - 413.

Zadeh, A. (1965). "Fuzzy Sets." Information and Control 8: 338-353.

Zeng, X.-J. & Keane, J. A. (2005). "Approximation capabilities of hierarchical fuzzy systems." IEEE Transactions on Fuzzy Systems 13(5): 659-672.

Zhang, X. M., Chen, Y. Q., Ansari, N. & Shi, Y. Q. (2004). "Mini-max initialization for function approximation " Neurocomputing 57: 389-409.

VITA

Ming Su

Candidate for the Degree of

Doctor of Philosophy

Thesis: A GENERALIZED RULE ANTECEDENT STRUCTURE FOR TSK TYPE OF DYNAMIC MODELS: STRUCTURE IDENTIFICATION AND PARAMETER ESTIMATION

Major Field: Chemical Engineering

Biographical:

Personal Data: Born in Jiangyou, Sichuan, P. R. China on November 19, 1977, the son of Wenxin Su and Minghua Du.

Education: Received Bachelor and Master Degrees in Chemical Engineering from East China University of Science and Technology, China in May 2000 and May 2003, respectively. Completed the requirements for the Doctor of Philosophy in Chemical Engineering at Oklahoma State University, Stillwater, Oklahoma in December, 2009.

Experience: Employed as a graduate assistant in East China University of Science and Technology from 2000 to 2003. Employed as a research assistant at Oklahoma State University from Fall, 2003 to present. Employed as a teaching assistant at Oklahoma State University from Spring, 2004 to Spring, 2009.

Name: Ming Su Date of Degree: December, 2009

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: A GENERALIZED RULE ANTECEDENT STRUCTURE FOR TSK TYPE OF DYNAMIC MODELS: STRUCTURE IDENTIFICATION AND PARAMETER ESTIMATION

Pages in Study: 195 Candidate for the Degree of Doctor of Philosophy

Major Field: Chemical Engineering

Scope and Method of Study: A novel rule antecedent structure is proposed to generalize TSK type of dynamic fuzzy models to deal with the problem of curse of dimensionality in conventional TSK fuzzy models. The proposed antecedent structure uses only nonlinear variables, which directly reduce the number of possible rules by reducing antecedent dimension. Additionally, one more degree of freedom is introduced to design antecedents to cover an antecedent space more efficiently, which further reduces the number of rules. The resultant GTSK model is identified in two stages. A novel recursive estimation based on spatially rearranged data is used to determine the consequent and antecedent variables. Model parameter values are obtained from partitioned antecedent space, which is the result of solving a series of splitting and regression problems.

Findings and Conclusions: The proposed rule antecedent structure is able to substantially reduce the complexity in a TSK type of dynamic model. The proposed dynamic order determination and nonlinear component detection methods are tested to be able to identify model structures and shown to be less sensitive to noise than other methods. Instead of directly estimating model parameters, the proposed approach solves a series of splitting and regression problems to partition the antecedent space as well as compute the antecedent and consequent parameters. The resultant antecedent partition is meaningful. The boundaries divide an antecedent space into regions, within which a linear relation is valid. The resultant GTSK model is tested on several nonlinear dynamic processes and shown to be more interpretable and informative than other modeling methods without loss of accuracy.

ADVISER'S APPROVAL: _____