NEAR EAST UNIVERSITY

INSTITUTE OF APPLIED
AND SOCIAL SCIENCES

# IDENTIFICATION OF FUZZY SYSTEMS

HUSSAM ELDIN HAMRAWI

Master Thesis

Department of Computer Engineering

Nicosia- 2003

Hussam Eldin Hamrawi : Identification of Fuzzy Systems

Approval of the Graduate School of Applied and Social Sciences

Prof. Dr. Fakhraddin Mamedov
Director

We certify this thesis is satisfactory for the award of the Degree of Master of Science in Computer Engineering

Examining Committee in charge:

Assist. Prof. Dr. Kadri Buruncuk, Committee Chairman, Electrical and Electronic Engineering Department, NEU

[illegible]

Assist. Prof. Dr. Firudin Muradov, Committee Member, Computer Engineering Department, NEU

[illegible]

Assist. Prof. Dr. Ali Denker, Committee Member, Computer Engineering Department, NEU

[illegible]

Assoc. Prof. Dr. Rahib Abiyev, Supervisor, Computer Engineering Department, NEU

[illegible]

# ACKNOWLEDGEMENTS

*"First, thank **Allah** for what ever he gives.*

*Second I would like to thank my supervisor Assoc. Prof. Dr. Rahib Abiyev for his valuable advice, encouragement and endless support.*

*Third, I would like to acknowledge special thank to the Near East University for offering me a suitable environment during my study.*

*I would like to dedicate my research to my family who are always motivating me with all their love.*

*I gratefully acknowledge the role of my friends those who sat behind me during the hard times.*

*Finally, Special thank to the Jury members who offering me this great opportunity to present my thesis"*

## ABSTRACT

In the practice many approaches have been used to model industrial processes from large number of numerical input and output data. Obtaining mathematical models of the system, allow us to investigate the process and develop efficient control. For this reason identification (i.e. obtaining a mathematical model) of these processes is one of important problems in engineering science.

Some of technological processes are characterized by uncertainty of environment, fuzziness of information. In order to identify these processes the use of fuzzy techniques for identification becomes important.

Thesis is devoted to the fuzzy identification of technological processes using fuzzy methodologies. To solve this problem, the state of application problems of fuzzy technology for identification of technological processes is given. The deterministic identification algorithms, such as batch and recursive least squares algorithms are described. The use of fuzzy version of these algorithms is presented. Also the applications of optimal output predefuzzification clustering and nearest neighborhood clustering algorithms for system identification are described.

Finally the use of optimal output predefuzzification-clustering algorithm for identification of static and dynamic systems is described. The simulation of fuzzy identification of static and dynamic systems is realized using Matlab Package.

# TABLE OF CONTENTS

# INTRODUCTION

*'The closer one looks at a "real" world problem, the fuzzier becomes its solution. Stated informally, the essence of this principle is that, as the complexity of a system increases, our ability to make precise and yet significant statements about its behavior diminishes until a threshold beyond which precision and significance (relevance) become almost mutually exclusive characteristics.' (L. Zadeh)*

This throws light on the place of fuzziness in the models we wish to construct. Fuzzy identification and modeling methodologies have been successfully used in a number of real-world applications. The Takagi–Sugeno model has often been employed in the modeling and identification of nonlinear technical processes from data. Examples are the modeling of a multilayer incinerator [17], a converter in a steel-making process, or a glass-melting furnace [18]. Biotechnology and ecology are typical examples of areas where conventional modeling techniques do not give satisfactory results. Fuzzy modeling has been used in a number of applications, such as Penicillin–G conversion [19] , prediction of river water flow [20], enzymatic soil removal in washing processes [21], or modeling of algae growth in lakes [20].

Fuzzy identified models can be used in the design of automatic controllers, for instance in train operation, combustion control, or pressure control [3]. Fuzzy models can also serve as decision support systems to assist operators [22], or can be used to clone the operators based on traces of their behavior.

Fuzzy modeling is a framework in which different modeling and identification methods are combined, providing, on the one hand, a transparent interface with the designer or the operator and, on the other hand, a flexible tool for nonlinear system modeling and control, comparable with other nonlinear black-box techniques. The rule-based character of fuzzy models allows for a model interpretation in a way that is similar to the one humans use.

The aim of this thesis is to examine how to design, construct and use fuzzy systems for identification. The basic problem to be studied here is how to construct a fuzzy system from numerical data. If the numerical data is plant input-output data obtained from an experiment, we may identify a fuzzy system model of the plant. This may be useful for simulation purposes and sometimes for use in a controller. On the other hand, the data may come from other sources, and a fuzzy system may be used to provide for a parameterized nonlinear function that fits the data by using its basic interpolation capabilities. For instance, suppose that we have a human expert who controls some process and we observe how he or she does this by observing what numerical plant input the expert picks for the given numerical data that he or she observes. Suppose further that we have many such associations between "decision-making data." The methods in this thesis will examine how to construct rules for a fuzzy model from this data (i.e. identify a model from the human-generated decision making data).

Thesis consists of an introduction, five chapters, conclusion, references and appendixes.

In Chapter 1 the identification problem and state of application of fuzzy technology for identification of different industrial processes is given.

In Chapter 2 using least mean square methods the estimation of parameters values of mathematical models of systems is considered. The description of batch least squares methods and recursive squares methods are given.

In Chapter 3 the description of fuzzy batch and recursive least squares methods is given and a simple demonstration example is provided.

In Chapter 4 two techniques for training fuzzy systems based on clustering are given. The first uses "c-means clustering" and least squares to train the premises and consequents, respectively, of the Takagi-Sugeno fuzzy system; while the second uses a nearest neighborhood technique to train standard fuzzy systems.

In Chapter 5 the fuzzy identification of static and dynamic systems are considered, two applications to a course performance system and a tank water level system are analyzed.

In conclusion the important results obtained from the thesis are given.

# CHAPTER ONE

# STATE OF APPLICATION PROBLEMS OF FUZZY TECHNOLOGY FOR IDENTIFICATION OF TECHNOLOGICAL PROCESSES

## 1.1 Overview

In practice conventional system identification of technological processes are often developed via simple models of the plant behavior that satisfy the necessary assumptions, and via the tuning of relatively simple linear or nonlinear processes. Regardless, it is well understood that heuristic enter the conventional modeling design process as long as we are concerned with the actual implementation of the system. It must be acknowledged, moreover, that conventional system identification approaches that use appropriate heuristics to tune the design have been relatively successful.

Fuzzy models provide a formal methodology for representing, manipulating, and implementing a human's heuristic knowledge about how a system is responding.

In this chapter the concept of systems identification and modeling and the relevance of these systems to application and practice are previewed, a review of other researches and methods concerning the field of fuzzy identification is considered.

## 1.2 Systems Identification and Modeling

Developing mathematical models of real systems is a central topic in many disciplines of engineering and science. Models can be used for simulations, analysis of the system's behavior, better understanding of the underlying mechanisms in the system, design of new processes, or design of controllers.

Traditionally, modeling is seen as a conjunction of a thorough understanding of the system's nature and behavior, and of a suitable mathematical treatment that leads to a usable model. This approach is usually termed "white-box" (physical, mechanistic, first-principle) modeling.

However, the requirement for a good understanding of the physical background of the problem at hand proves to be a severe limiting factor in practice, when complex and poorly understood systems are considered. Difficulties encountered in conventional white-box modeling can arise, for instance, from poor understanding of the underlying phenomena, inaccurate values of various process parameters, or from the complexity of the resulting model. A complete understanding of the underlying mechanisms is virtually impossible for a majority of real systems. However, gathering an acceptable degree of knowledge needed for physical modeling may be a very difficult, time-consuming and expensive or even impossible task. Even if the structure of the model is determined, a major problem of obtaining accurate values for the parameters remains. It is the task of system identification to estimate the parameters from data measured on the system. Identification methods are currently developed to a mature level for linear systems only. Most real processes are, however, nonlinear and can be approximated by linear models only locally.

A different approach assumes that the process under study can be approximated by using some sufficiently general "black-box" structure used as a general function approximator. The modeling problem then reduces to postulating an appropriate structure of the approximator, in order to correctly capture the dynamics and nonlinearity of the system. In black-box modeling, the structure of the model is hardly related to the structure of the real system. The identification problem consists of estimating the parameters of the model. If representative process data are available, black-box models usually can be developed quite easily, without requiring process-specific knowledge. A severe drawback of this approach is that the structure and parameters of these models usually do not have any physical significance. Such models cannot be used for analyzing the system's behavior otherwise than by numerical simulation, cannot be scaled up or down when moving from one process scale to another, and therefore are less useful for industrial practice.

There is a range of modeling techniques that attempt to combine the advantages of the white-box and black-box approaches, such that the known parts of the system are modeled using physical knowledge, and the unknown or less certain parts are approximated in a black-box manner, using process data and black-box modeling structures with suitable approximation properties. These methods are often denoted as hybrid, semi-mechanistic or "gray-box" modeling.

A common drawback of most standard modeling approaches is that they cannot make effective use of extra information, such as the knowledge and experience of engineers and operators, which is often imprecise and qualitative in its nature. The fact that humans are often able to manage complex tasks under significant uncertainty has stimulated the search for alternative modeling and control paradigms. So-called "intelligent" modeling and control methodologies, which employ techniques motivated by biological systems and human intelligence to develop models and controllers for dynamic systems, have been introduced.

These techniques explore alternative representation schemes, using, for instance, natural language, rules, semantic networks or qualitative models, and possess formal methods to incorporate extra relevant information. Fuzzy modeling and control are typical examples of techniques that make use of human knowledge and deductive processes.

## 1.3 Identification of Technological Processes using Fuzzy Technology

Conventional system theory relies on crisp mathematical models of systems, such as algebraic and differential or difference equations. For some systems, such as electro-mechanical systems, mathematical models can be obtained. This is because the physical laws governing the systems are well understood. For a large number of practical problems, however, the gathering of an acceptable degree of knowledge needed for physical modeling is a difficult, time-consuming and expensive or even impossible task.

In the majority of systems, the underlying phenomena are understood only partially and crisp mathematical models cannot be derived or are too complex to be useful. Examples of such systems can be found in the chemical or food industries, biotechnology, ecology, finance, sociology, etc. A significant portion of information about these systems is available as the knowledge of human experts, process operators and designers. This knowledge may be too vague and uncertain to be expressed by mathematical functions. It is, however, often possible to describe the functioning of

systems by means of natural language, in the form of if-then rules. Fuzzy rule-based systems can be used as knowledge-based models constructed by using knowledge of experts in the given field of interest. From this point of view, fuzzy systems are similar to expert systems studied extensively in the "symbolic" artificial intelligence.
Adequate processing of imprecise information, precise numerical computation with conventional mathematical models only makes sense when the parameters and input data are accurately known.

As this is often not the case, a modeling framework is needed which can adequately process not only the given data, but also the associated uncertainty. The stochastic approach is a traditional way of dealing with uncertainty. However, it has been recognized that not all types of uncertainty can be dealt with within the stochastic framework. Various alternative approaches have been proposed, fuzzy logic and set theory being one of them.

Transparent (gray-box) modeling and identification, Identification of dynamic systems from input-output measurements is an important topic of scientific research with a wide range of practical applications. Many real-world systems are inherently nonlinear and cannot be represented by linear models used in conventional system identification.

Recently, there is a strong focus on the development of methods for the identification of nonlinear systems from measured data. Fuzzy models are one of the most popular model structures used. From the input-output view, fuzzy systems are flexible mathematical functions which can approximate other functions or just data (measurements) with a desired accuracy.

There are a number of work and research done in the field of fuzzy systems identification and here are some of them:

In [6] fuzzy relational identification builds a relational model describing system's behavior by a nonlinear mapping between its variables. The fuzzy relational algorithm based on simplified max-min relational equation was introduced. This algorithm presents an adaptation method applied to gravity-center of each fuzzy set based on error integral value between measured and predicted system's output, and uses the concept of time-variant universe of discourses. The identification algorithm also includes a method to attenuate noise influence in extracted system's relational model using a fuzzy filtering mechanism.

In [9] a fuzzy system can be constructed to interpolate between input-output data to provide an approximation for the function that is implicitly defined by the input-output data pair associations. This method shows how to choose the input-output data pairs so that accurate function approximation can be achieved with fuzzy systems. Using this insight the technique that is called uniform training on which input sequences are chosen to produce good training data sets is achieved. Also, the technique for function approximation via fuzzy systems is called modified learning from examples where the membership functions are specified and rules are added to try to achieve pre-specified function approximation accuracy

In [16] the use of fuzzy set theory and fuzzy logic to construct an annual time-series for the (unobservable) underground economy over a period of time was introduced. Two input variables are used – the effective tax rate and an index of the degree of regulation. The resulting underground economy time-series is compared with one previously constructed by the second author using a structural "Multiple Indicators, Multiple Causes" (MIMIC) model.

In [10] two different methods of fuzzy identification of a class of nonlinear systems are developed. This is applicable to systems with unknown and partially known mathematical models. The class of systems considered is nonlinear in output but linear in input. In the first method, a gray box model is considered. The nominal values of parameters of the nonlinear system are assumed to be known. The unknown nonlinear function is identified off-line by choosing a suitable fuzzy relational model and the parameters of the nonlinear system are updated on-line using recursive least square (RLS) algorithm. In the second method, a black box model is considered. The nonlinear plant is identified on-line by choosing a suitable linear model using RLS in stage-1 and the residual nonlinear part is identified in stage-2 using fuzzy identification. The control input is then calculated based on the identified nonlinear model using weighted one step ahead control method.

In [8] the methods of pre-processing for structure identification of fuzzy models are developed. There are two approaches the first approach uses the statistical method of Principal Component Analysis (PCA). The second one uses a clustering technique called autonomous mountain-clustering method. The statistical method of Principal Component Analysis helps to select the variables that dominate the system dynamics. Besides, this method contributes to design fuzzy models with better performance. The second approach identifies the fuzzy model order. That is, the method identifies the number of membership functions attributed to each variable, as well as their position and width. So, the autonomous mountain-clustering eliminates the usual "trial-and-error" mechanism.

In [7] a simplified incremental type of cause-effect models for additive MISO type dynamical processes is proposed and analyzed. Dynamics of the process is expressed by three groups of parameters: gains, memory lengths and shapes of the specially introduced cause-effect relations membership functions. These functions represent in a fuzzy manner the degree of relationship between the past time changes of the respective input and the current change of the process output. The total model of the dynamical system is identified from experimental data by different modifications of the Least Mean Squares algorithm for each group of parameters separately. The specially introduced indirect LSM algorithm is able to reduce significantly the size of the problem by identifying one-dimensional fuzzy model that represents indirectly the cause-effect relation for the dynamics.

In [14] a fuzzy modeling framework has been developed for the utilization of a priori knowledge. The proposed modeling approach transforms the different types of information into the structure of the model (fuzzy rule base), constraints defined on the parameters and variables, dynamic local model or data, and steady-state data or model. This modeling step is followed by an optimization procedure based on these transformed information. The method describes one element of this framework that was developed to use prior knowledge in constrained adaptation of the rule consequences of Takagi-Sugeno fuzzy models.

In [11] the resilient propagation (RPROP), an efficient nonlinear optimization technique, for parameter identification is used in order to achieve a fast Takagi-Sugeno modeling (FTSM) that is suited to model high-dimensional data sets containing a large number of data.

Identification and defining accurate mathematical model of technological systems allows us to develop efficient control for these processes which increase the quality of output products.

## 1.4 Summary

Analyses of industrial processes show that for effective control of these processes it is necessary to define accurate models of these processes. In this chapter the state of application problem of fuzzy identification for technological processes is given.

# CHAPTER TWO
# SYSTEM IDENTIFICATION USING LMS METHODS

## 2.1 Overview

This chapter starts by precisely defining the function approximation problem, in which it is needed to synthesize a function to approximate another function that is inherently represented via a finite number of input-output associations (i.e., it is only known how the function maps a finite number of points in its domain to its range).

Then the batch and recursive least squares methods are introduced for constructing a linear system to match some input-output data.

## 2.2 Function Approximation Problem

Given some function

$$g : \bar{X} \rightarrow \bar{Y}$$

where $\bar{X} \subset \Re^n$ and $\bar{Y} \subset \Re^n$ ,in order construct a system

$$f : x \rightarrow y$$

where $x \subset \bar{X}$ and $y \subset \bar{Y}$ are some domain and range of interest, by choosing a parameter vector $\theta \subset \Re^n$ so that

$$g(x) = f(x/\theta) + e(x) \tag{2.1}$$

for all $x = [x_1, x_2, ..., x_n]^T \in \bar{X}$ where the approximation error $e(x)$ is as small as possible.

If we want to refer to the input at time k, we will use $x(k)$ for the vector and $x_j(k)$ for its $j^{th}$ component.

Assume that all the information available to choose the parameters $\theta$ of the system $f(x/\theta)$ is part of the function $g$ in the form of a finite set of input-output data pairs (i.e. the functional mapping implemented by g is largely unknown). The $i^{th}$ input-output data pair from the system g is denoted by $(x^i, y^i)$ where $x^i \in \bar{X}, y^i \in \bar{Y}$. We let $x^i = [x_1^i, x_2^i, \ldots, x_n^i]^T$ represent the input vector for the $i^{th}$ data pair. Hence, $x_j^i$ is the $j^{th}$ element of the $i^{th}$ data vector (it has a specific value and is not a variable). We call the set of input-output data pairs the training data set and denote it by

$$G = \{(x^1, y^1), \ldots, (x^M, y^M)\} \subset \bar{X} \times \bar{Y} \tag{2.2}$$

where M denotes the number of input-output data pairs contained in $G$.

To get a graphical picture of the function approximation problem, see Figure 2.1. This clearly shows the challenge; it can certainly be hard to come up with a good function $f$ to match the mapping g when we know only a little bit about the association between X and Y in the form of data pairs G. Moreover, it may be hard to know when we have a good approximation; that is, when $f$ approximates g over the whole space of inputs X.

For the function approximation problem we consider a simple example. Suppose that, $n = 2, X \subset \Re^2, Y = [0,10]$, and $g : X \rightarrow Y$. Let M $= 3$ and the training data set

$$G = \{(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, 1), (\begin{bmatrix} 2 \\ 4 \end{bmatrix}, 5)(\begin{bmatrix} 3 \\ 6 \end{bmatrix}, 6)\} \tag{2.3}$$

which partially specifies $g$ as shown in figure 2.2. The function approximation problem amount to finding a function $f(x/\theta)$ by manipulating $\theta$ so that $f(x/\theta)$ approximate $g$ as closely as possible. We will use this simple data set in several methods we developed in this thesis.

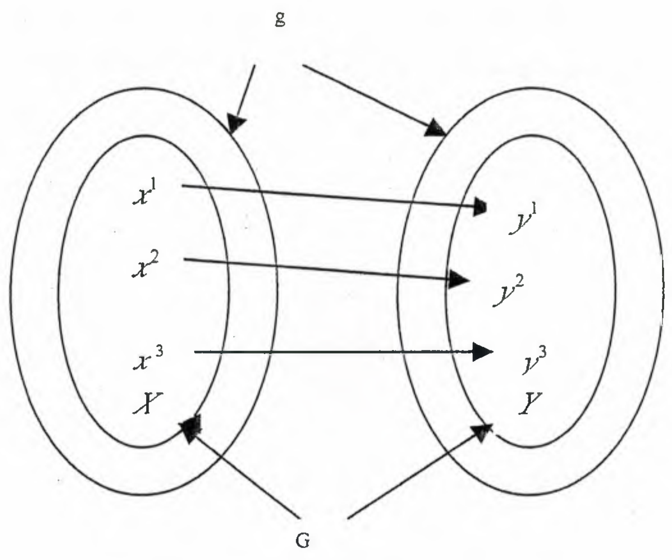

Figure 2.1 Function mapping with three .known input-output data pairs.

How do we evaluate how closely a fuzzy system $f(x/\theta)$ approximate the function $g(x)$ for all $x \in \overline{X}$ for a given $\theta$? Notice that

$$\sup_{x \in \overline{X}} \{ |g(x) - f(x/\theta)| \} \tag{2.4}$$

is a bound on the approximation error (if it exists).However, specification of such a bound requires that the function g be completely known; however, as stated above, we know only a part of $g$ given by the finite set $G$. Therefore, we are only available to evaluate the accuracy of approximation by evaluating the error between $g(x)$ and $f(x/\theta)$ at certain points $x \in \overline{X}$ given by *available* input-output data. We call this set of input-output data the test set and denote it as $\Gamma$ where

$$\Gamma = \{ (x^1, y^1), \ldots, (x^{M_\Gamma}, y^{M_\Gamma}) \} \subset \overline{X} \times \overline{Y} \tag{2.5}$$

Here $M_\Gamma$ denotes the number of known input-output data pairs contained within the test set. It is important to note that the input-output data pairs $(x^j, y^j)$ contained in $\Gamma$ may not be contained in G, or vice versa. It also might be the case that the test set is equal to the training set $(G = \Gamma)$ however; this choice is not always a good one. Most often we will want to test the system with at least some data that were not used to construct $f(x/\theta)$ since this will often provide a more realistic assessment of the quality of the approximation.

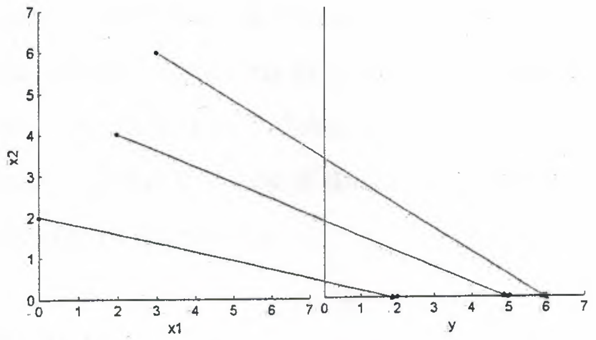

Figure 2.2 The training data G generated from the function $g$.

We see that, the evaluation of the error in approximation between $g$ and a system $f(x/\theta)$ based on a test set $\Gamma$ may or may not be a true measure of the error between $g$ and $f$ for every $x \in \bar{X}$, but it is the only evaluation we can make based on known information. Hence we can use measures like

$$\sum_{(x',y')\in\Gamma} (g(x') - f(x'/\theta))^2 \tag{2.6}$$

or

$$\sup_{x',y')\in\Gamma} \left\{ \left| g(x') - f(x'/\theta) \right| \right\} \tag{2.7}$$

to measure the approximation error. Accurate function approximation requires that some expression of this nature be small; however, this clearly does not guarantee perfect representation of g with $f$ since most often we cannot test that $f$ matches g over all possible input points.

We would like to emphasize that the type of function that we choose to adjust (i.e., $f(x/\theta)$) can have a significant impact on the ultimate accuracy of the approximator. For instance, it may be that a Takagi-Sugeno-Kang (or functional) fuzzy system will provide a better approximator than a standard fuzzy system for a particular application..

We think of $f(x/\theta)$ as a structure for an approximator that is parameterized by $\theta$. In this thesis we will study the use of fuzzy systems as approximators, and use a fuzzy system as the structure for the approximator. The choice of the parameter vector $\theta$ depends on, for example, how many membership functions and rules we use. Generally, we want enough membership functions and rules to be able to get good accuracy, but not too many since if our function is "over-parameterized" this can actually degrade approximation accuracy. Often, it is best if the structure of the approximator is based on some physical knowledge of the system.

## 2.3 System Identification

Many applications exist in the control and signal processing areas that may utilize nonlinear function approximation. One such application is system identification, which is the process of constructing a mathematical model of a dynamic system using experimental data from that system. Let $g$ denote the physical system that we wish to identify. The training set G is defined by the experimental input-output data.
In linear system identification, the following model is often used

$$y(k) = \sum_{i=1}^{\hat{q}} \theta_{a_i} y(k-i) + \sum_{i=0}^{\hat{p}} \theta_{b_i} u(k-i) \tag{2.8}$$

where $u(k)$ and $y(k)$ are the system input and output at time $k \geq 0$. Notice that we will need to specify appropriate initial conditions. In this case $f(x/\theta)$, which is not a fuzzy system, is defined by

$$f(x/\theta) = \theta^T x(k)$$

where

$$x(k) = [y(k-1), \ldots, y(k-\hat{q}), u(k), \ldots, u(k-\hat{p})]^T \tag{2.9}$$

$$\theta = \left[\theta_{a_1}, \ldots, \theta_{a_{\hat{q}}}, \theta_{b_0}, \ldots, \theta_{b_{\hat{p}}}\right]^T \tag{2.10}$$

Let $N = \hat{q} + \hat{p} + 1$ so that $x(k)$ and $\theta$ are $N \times 1$ vectors. Linear system identification amounts to adjusting $\theta$ using information from G so that $g(x) = f(x/\theta) + e(x)$ where $e(x)$ is small for all $x \in \overline{X}$. Similar to conventional linear system identification, for fuzzy identification we will utilize an appropriately defined "regression vector" $x$ as specified in Equation (2.9). And we will tune a fuzzy system $f(x/\theta)$ so that $e(x)$ is small.

Our hope is that since the fuzzy system $f(x/\theta)$ has more functional capabilities (as characterized by the universal approximation property) than the linear map defined in Equation (2.8), we will be able to achieve more accurate identification for nonlinear systems by appropriate adjustment of its parameters $\theta$ of the fuzzy system.

## 2.4 Batch Least Squares

We will introduce the batch least squares method to train fuzzy systems by first discussing the solution of the linear system identification problem. Let $g$ denote the physical system that we wish to identify. The training set $G$ is defined by the experimental input-output data that is generated from this system. In linear system identification, we can use a model

$$y(k) = \sum_{i=1}^{\hat{q}} \theta_{a_i} y(k-i) + \sum_{i=0}^{\hat{p}} \theta_{b_i} u(k-i)$$

where $u(k)$ and $y(k)$ are the system input and output at time $k$. In this case $f(x/\theta)$, which is not a fuzzy system, is defined by

$$f(x/\theta) = \theta^T x(k) \qquad (2.11)$$

where we recall that

$$x(k) = [y(k-1),\dots,y(k-\hat{q}),u(k),\dots,u(k-\hat{p})]^T$$

and

$$\theta = [\theta_{a_1},\ldots,\theta_{a_{\bar{q}}},\ldots,\theta_{b_0},\ldots,\theta_{b_{\bar{p}}}]^T$$

we have $N = \bar{q} + \bar{p} + 1$ so that $x(k)$ and $\theta$ are $N \times 1$ vectors, and often $x(k)$ is called the "regression vector".

Recall that system identification amounts to adjusting $\theta$ using information from $G$ so that $f(x/\theta) \approx g(x)$ for all $x \in X$. Often, to form $G$ for linear system identification we choose $x^i = x(i), y^i = y(i)$, and let $G = \{(x^i, y^i) : i = 1,2,\ldots,M\}$ to do this we will need appropriate initial conditions.

### 2.4.1 Batch Least Squares Derivation

In the batch least squares method we define

$$Y(M) = [y^1, y^2, \ldots, y^M]^T$$

to be a $M \times 1$ vector of output data where the $y^i : i = 1,2,\ldots,M$ come from $G$ (i.e., $y^i$ such that $(x^i, y^i) \in G$). We let

$$\Phi(M) = \begin{bmatrix} (x^1)^T \\ (x^2)^T \\ \vdots \\ (x^M)^T \end{bmatrix}$$

be a $M \times N$, matrix that consists of the $x^i$ data vectors stacked into a matrix (i.e., the $x^i$ such that $(x^i, y^i) \in G$) Let

$$\epsilon_i = y^i - (x^i)^T \theta$$

be the error in approximating the data pair $(x^i, y^i) \in G$ using $\theta$. Define

$$E(M) = [\in_1, \in_2, \ldots, \in_M]^T$$

so that

$$E = Y - \Phi\theta$$

choose

$$V(\theta) = (\frac{1}{2} E^T E)$$

to be a measure of how good the approximation is for all the data for a given $\theta$. We want to pick $\theta$ *to* minimize $V(\theta)$ Notice that $V(\theta)$ is convex in $\theta$ so that a local minimum is a global minimum.

Now, using basic ideas from calculus, if we take the partial of $V$ with respect to $\theta$ and set it equal to zero, we get an equation for $\hat{\theta}$, the best estimate (in the least squares sense) of the unknown $\theta$ . Another approach to deriving this is to notice that

$$2V = E^T E = Y^T Y - Y^T \Phi\theta - \theta^T \Phi^T Y + \theta^T \Phi^T \Phi\theta$$

then we "complete the square" by assuming that $\Phi\Phi^T$ is invertible and letting

$$2V = Y^T Y - Y^T \Phi\theta - \theta^T \Phi^T Y + \theta^T \Phi^T \Phi\theta + Y^T \Phi(\Phi^T \Phi)^{-1} \Phi^T Y - Y^T \Phi(\Phi^T \Phi)^{-1} \Phi^T Y$$

where we are simply adding and subtracting the same terms at the end of the equation).

Hence

$$2V = Y^T (1 - \Phi(\Phi^T \Phi)^{-1} \Phi^T) Y + (\theta - (\Phi^T \Phi)^{-1} \Phi^T Y)^T \Phi^T \Phi(\theta - (\Phi^T \Phi)^{-1} \Phi^T Y)$$

the first term in this equation is independent of $\theta$, so we cannot reduce $V$ via this term, so it can be ignored. Hence, to get the smallest value of $V$, we choose $\theta$ so that the second term is zero. We will denote the value of $\theta$ that achieves the minimization of $V$ by $\hat{\theta}$ and we notice that

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T Y \tag{2.12}$$

since the smallest we can make the last term in the above equation is zero. This is the equation for batch least squares that shows we can directly compute the least squares estimate $\hat{\theta}$ from the "batch" of data that is loaded into $\Phi$ and $Y$. If we pick the inputs to the system so that it is "sufficiently excited" , then we will be guaranteed that $\Phi^T\Phi$ is invertible; if the data come from a linear plant with known $\overline{q}$ and $\overline{p}$ then for sufficiently large $M$ we will achieve perfect estimation of the plant parameters.

In "weighted" batch least squares we use

$$V(\theta) = (\frac{1}{2}E^T WE) \tag{2.13}$$

where, for example, $W$ is a $M \times M$ diagonal matrix with its diagonal elements $w_i > 0$ for $i = 1,2,...,M$ and its off-diagonal elements equal to zero. These $w_i$ can be used to weight the importance of certain elements of $G$ more than others. For example, we may choose to have it put less emphasis on older data by choosing $w_1 < w_2 < ... < w_M$ when $x^2$ is collected after $x^1$, $x^3$ is collected after $x^2$, and so on. The resulting parameter estimates can be shown to be given by

$$\hat{\theta}_{wbls} = (\Phi^T W\Phi)^{-1}\Phi^T WY \tag{2.14}$$

To show this, we use Equation (2.13) and proceed with the derivation in the same manner as above.

As an example of how batch least squares can be used, suppose that we would like to use this method to fit a line to a set of data. In this case our parameterized model is

$$y = x_1\theta_1 + x_2\theta_2 \tag{2.15}$$

Notice that if we choose $x_2 = 1$, $y$ represents the equation for a line. Suppose that the data that we would like to fit the line to is given by

$$\left\{\left(\begin{bmatrix}1\\1\end{bmatrix},1\right),\left(\begin{bmatrix}2\\1\end{bmatrix},1\right),\left(\begin{bmatrix}3\\1\end{bmatrix},3\right)\right\}$$

Notice that to train the parameterized model in Equation (2.15) we have chosen $x_2^i = 1$ for $i = 1,2,3 = M$. We will use Equation (2.12) to compute the parameters for the line that best fits the data (in the sense that it will minimize the sum of the squared distances between the line and the data). To do this we let

$$\Phi = \begin{bmatrix}1 & 1\\2 & 1\\3 & 1\end{bmatrix}$$

and

$$Y = \begin{bmatrix}1\\1\\3\end{bmatrix}$$

Hence,

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T Y = \left(\begin{bmatrix}14 & 6\\6 & 3\end{bmatrix}\right)^{-1}\begin{bmatrix}12\\5\end{bmatrix} = \begin{bmatrix}1\\-\frac{1}{3}\end{bmatrix}$$

Hence, the line

$$y = x_1 - \frac{1}{3}$$

best fits the data in the least squares sense.

## 2.5 Recursive Least Squares

While the batch least squares approach has proven to be very successful for a variety of applications, it is by its very nature a "batch" approach (i.e., all the data are gathered, then processing is done). For small $M$ we could clearly repeat the batch calculation for increasingly more data as they are gathered, but the computations become prohibitive due to the computation of the inverse of $\Phi^T\Phi$ and due to the fact that the dimensions of $\Phi$ and $Y$ depend on $M$. Next, we derive a recursive version of the batch least squares method that will allow us to update our $\hat{\theta}$ estimate each time we get a new data pair, without using all the old data in the computation and without having to compute the inverse of $\Phi^T\Phi$. Since we will be considering successively increasing the size of $G$, and we will assume that we increase the size by one each time step, we let a time index $k = M$ and $i$ be such that $0 \leq i \leq k$ Let the $N \times N$ matrix

$$p(k) = (\Phi^T\Phi)^{-1} = \left(\sum_{i=1}^{k} x^i (x^i)^T\right)^{-1} \tag{2.16}$$

and let $\hat{\theta}(k-1)$ denote the least squares estimate based on $(k-1)$ data pairs $p(k)$ is called the "covariance matrix". Assume that $\Phi^T\Phi$ is nonsingular for all $k$.
We have

$$p^{-1}(k) = \Phi^T\Phi = \sum_{i=1}^{k} x^i (x^i)^T$$

so we can pull the last term from the summation to get

$$p^{-1}(k) = \sum_{i=1}^{k-1} x^i (x^i)^T + x^k (x^k)^T$$

and hence

$$p^{-1}(k) = p^{-1}(k-1) + x^k (x^k)^T \tag{2.17}$$

Now, using Equation (2.12) we have

$$\begin{aligned}\hat{\theta}(k) &= (\Phi^T\Phi)^{-1}\Phi^T Y \\ &= \left(\sum_{i=1}^{k} x^i (x^i)^T\right)^{-1} \sum_{i=1}^{k} x^i y^i \\ &= p(k)\left(\sum_{i=1}^{k} x^i y^i\right) \\ &= p(k)\left(\sum_{i=1}^{k-1} x^i y^i + x^k y^k\right) \qquad (2.18)\end{aligned}$$

Hence,

$$\hat{\theta}(k-1) = p(k-1)\left(\sum_{i=1}^{k-1} x^i y^i\right)$$

and so

$$p^{-1}(k-1)\hat{\theta}(k-1) = \sum_{i=1}^{k-1} x^i y^i$$

Now, replacing $p^{-1}(k-1)$ in this equation with the result in Equation (2.17), we get

$$(p^{-1}(k) - x^k (x^k)^T)\hat{\theta}(k-1) = \sum_{i=1}^{k-1} x^i y^i$$

Using the result from Equation (2.18), this gives us

$$\begin{aligned}\hat{\theta}(k) &= p(k)(p^{-1}(k) - x^k (x^k)^T)\hat{\theta}(k-1) + p(k)x^k y^k \\ &= \hat{\theta}(k-1) - p(k)x^k (x^k)^T \hat{\theta}(k-1) + p(k)x^k y^k \\ &= \hat{\theta}(k-1) + p(k)x^k (y^k - (x^k)^T \hat{\theta}(k-1)) \qquad (2.19)\end{aligned}$$

This provides a method to compute an estimate of the parameters $\hat{\theta}(k)$ at each time step

$k$ from the past estimate $\hat{\theta}(k-1)$ and the latest data pair that we received $(x^k, y^k)$. Notice that $(y^k - (x^k)^T\hat{\theta}(k-1))$ is the error in predicting $y^k$ using $\hat{\theta}(k-1)$ to update $\hat{\theta}$ in Equation (2.19) we need $p(k)$ , so we could use

$$p^{-1}(k) = p^{-1}(k-1) + x^k(x^k)^T \tag{2.20}$$

But then we will have to compute an inverse of a matrix at each time step (i.e., each time we get another set of data). Clearly this is not desirable for real-time implementation, so we would like to avoid this. To do so, recall that the "matrix inversion lemma" indicates that if A, C, and $(C^{-1} + DA^{-1}B)$ are nonsingular square matrices, then $A + BCD$ is invertible and

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

We will use this fact to remove the need to compute the inverse of $p^{-1}(k)$ that comes from Equation (2.20) so that it can be used in Equation (2.19) to update $\hat{\theta}$ . Notice that

$$\begin{aligned} P(k) &= (\Phi^T(k)\Phi(k))^{-1} \\ &= (\Phi^T(k-1) + \Phi(k-1) + x^k(x^k)^T)^{-1} \\ &= (p^{-1}(k-1) + x^k(x^k)^T)^{-1} \end{aligned}$$

if we use the matrix inversion with $A = p^{-1}(k-1), B = x^k, C = I, D = (x^k)^T$ we get

$$p(k) = p(k-1) - p(k-1)x^k(I + (x^k)^T p(k-1)x^k)^{-1}(x^k)^T p(k-1) \tag{2.21}$$

which together with

$$\hat{\theta}(k) = \hat{\theta}(k-1) + p(k)x^k(y^k - (x^k)^T\hat{\theta}(k-1)) \tag{2.22}$$

that was derived in Equation (2.19) is called the "recursive least squares (RLS) algorithm".

Basically, the matrix inversion lemma turns a matrix inversion into the inversion of a scalar (.i.e., the term $(I+(x^k)^T p(k-1)x^k)^{-1}$ is a scalar).

We need to initialize the RLS algorithm (i.e., choose $\hat{\theta}(0), p(0)$ One approach to do this is to use $\hat{\theta}(0)=0, p(0)=p_0$ where $p_0=\alpha 1$ for some large $\alpha>0$. This is the choice that is often used in practice. Other times, we may pick $p(0)=p_0$ but choose $\hat{\theta}(0)$ to be the best guesses that you have at what the parameter values are.

There is a "weighted recursive least squares" (WRLS) algorithm also. Suppose that the parameters of the physical system $\theta$ vary slowly. In this case it may be advantageous to choose

$$V(\theta,k)=\frac{1}{2}\sum_{i=1}^{k}\lambda^{k-i}(y^i(x^i)^T\theta)^2$$

where $0<\lambda\leq 1$ is called a "forgetting factor" since it gives the more recent data higher weight in the optimization (note that this performance index $V$ could also be used to derive weighted batch least squares). Using a similar approach to the above, you can show that the equations for WRLS are given by

$$p(k)=\frac{1}{\lambda}(I-p(k-1)x^k(\lambda I+(x^k)^T p(k-1)x^k)^{-1}(x^k)^T p(k-1) \tag{2.23}$$

$$\hat{\theta}(k)=\hat{\theta}(k-1)+p(k)x^k(y^k-(x^k)^T\hat{\theta}(k-1))$$

(Where when $\lambda=1$ we get standard RLS), this completes our description of the least squares methods. Next, we will discuss how they can be used to train fuzzy systems.

## 2.6 Summary

In this chapter the function approximation problem was stated and the deterministic batch and recursive least square method for system identification was described.

# CHAPTER THREE
# FUZZY IDENTIFICATION USING LMS METHODS

## 3.1 Overview

In this chapter the descriptions of fuzzy systems are given. The explanation of batch and recursive least squares methods for training fuzzy systems is described.

## 3.2 Fuzzy Systems

A static or dynamic system which makes use of fuzzy sets or fuzzy logic and of the corresponding mathematical framework is called a *fuzzy system.*

Fuzzy systems are of two types, numerical and rule-based which in combination together we get an excellent identification of a systems behavior merely depending on the choice of the best tuning mechanism that we are going to view in this thesis.

Mainly based on knowledge of experts, this knowledge is often formulated through fuzzy rule-based format. In the fuzzy rule-based input and output variables are often characterized by linguistic values. These linguistic values are described by membership functions.

The input-output data of a system formulate the numerical fuzzy system which we are going to overview thoroughly through out this thesis.

Here we can state some of the advantages of using fuzzy system:

- Ability to translate imprecise/vague knowledge of human experts.
- Simple, easy to implement technology.
- Software design and hardware implementation support.
- Results are easy to transfer from product to product.
- Smooth tuning behavior.

There are a number of ways that fuzzy sets can be involved in a system, such as:

- In the description of the system. A system can be defined, for instance, as a collection of if-then rules with fuzzy predicates, or as a fuzzy relation. An example of a fuzzy rule describing the relationship between a heating power and the temperature trend in a room may be:

**If** the heating power is high **then** the temperature will increase fast:

- In the specification of the system's parameters. The system can be defined by an algebraic or differential equation, in which the parameters are fuzzy numbers instead of real numbers. As an example consider an equation: $y = 3x_1 + 5x_2$ where 3 and 5 are fuzzy number "about three" and "about five", respectively, defined by membership functions. Fuzzy numbers express the uncertainty in the parameter values.
- The input, output and state variables of a system may be fuzzy sets. Fuzzy inputs can be readings from unreliable sensors ("noisy" data), or quantities related to human perception, such as comfort, beauty, etc. Fuzzy systems can process such information, which is not the case with conventional (crisp) systems.

A fuzzy system can simultaneously have several of the above attributes. In this thesis we will focus on the last type of systems, i.e., fuzzily described systems with crisp or fuzzy inputs.

## 3.3 Tuning Fuzzy Systems

It is possible to use the least squares methods described in the previous chapter to tune fuzzy systems either in a batch or real-time mode. In this chapter we will explain how to tune both standard and Takagi-Sugeno-Kang fuzzy systems that have many inputs and only one output. To train fuzzy systems with many outputs, simply repeat the procedure described below for each output.

### 3.3.1 Standard Fuzzy Systems

First, we consider a fuzzy system

$$y = f(x|\theta) = \frac{\sum_{i=1}^{R} b_i \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} \tag{3.1}$$

where $x = [x_1, x_2, ..., x_n]^T$ and $\mu_i(x)$ is defined as the certainty of the premise of the $i^{th}$ rule (it is specified via the membership functions on the input universe of discourse together with the choice of the method to use in the triangular norm for representing the conjunction in the premise). The $b_i; i = 1,2,..., R$ values are the centers of the output membership functions. Notice that

$$f(x/\theta) = \frac{b_1 \mu_1(x)}{\sum_{i=1}^{R} \mu_i(x)} + \frac{b_2 \mu_2(x)}{\sum_{i=1}^{R} \mu_i(x)} + ... + \frac{b_R \mu_R(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

and that if we define

$$\xi_i(x) = \frac{\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} \tag{3.2}$$

then

$$f(x|\theta) = b_1 \xi_1(x) + b_2 \xi_2(x) + ... + b_R \xi_R(x)$$

Hence, if we define

$$\xi(x) = [\xi_1(x), \xi_2(x), ..., \xi_R(x)]^T$$

and

$$\theta = [b_1, b_2 ..., b_R]^T$$

then

$$y = f(x|\theta) = \theta^T \xi(x) \tag{3.3}$$

we see that the form of the model to be tuned is only a slightly different form from the standard least squares case in Equation (2.11). In fact, if the $\mu_i$ are given, then $\xi(x)$ is given so that it is in *exactly* the right form for use by the standard least squares methods since we can view $\xi(x)$ as a known regression vector. Basically, the training data $x^i$ are mapped into $\xi(x)$ and the least squares algorithms produce an estimate of the best centers for the output membership function centers $b_i$.

This means that either batch or recursive least squares can be used to train certain types of fuzzy systems (ones that can be parameterized so that they are "linear in the parameters," as in Equation (3.3). All we have to do is replace $x^i$ with $\xi(x)$ in forming the $\Phi$ vector for batch least squares and in Equation (2.23) for recursive least squares.

Hence, we can achieve either on-line or off-line training of certain fuzzy systems with least squares methods.

If we have some heuristic ideas for the choice of the input membership functions and hence $\xi(x)$ then this method can be quite effective (Note that any known function can be used to replace any of the $\xi_i$ in $\xi(x)$ vector). It is found that some of the standard choices for input membership functions (e.g., uniformly distributed ones) work very well for some applications.

### 3.3.2 Takagi-Sugeno-Kang Fuzzy Systems

It is interesting to note that Takagi-Sugeno-Kang fuzzy systems can also be parameterized so that they are linear in the parameters; so that they can also be trained with either batch or recursive least squares methods. In this case, if we can pick the membership functions appropriately (e.g., using uniformly distributed ones), then we can achieve a nonlinear interpolation between the linear output functions that are constructed with least squares.

In particular a Takagi-Sugeno-Kang fuzzy system is given by

$$y = \frac{\sum_{i=1}^{R} g_i(x)\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

where

$$g_i(x) = a_{i,0} + a_{i,1}x_1 + \ldots + a_{i,n}x_n$$

Hence, using the same approach as for standard fuzzy systems, we note that

$$y = \frac{\sum_{i=1}^{R} a_{i,0}\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} + \frac{\sum_{i=1}^{R} a_{i,1}x_1\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} + \ldots + \frac{\sum_{i=1}^{R} a_{i,2}x_n\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

We see that the first term is the standard fuzzy system. Hence, use the $\xi(x)$ defined in Equation (3.2) and redefine $\xi(x)$ and $\theta$ to be

$$\xi(x) = [\xi_1(x), \xi_2(x), \ldots, \xi_R(x), x_1\xi_1(x), x_1\xi_2(x), \ldots, x_1\xi_R(x), \ldots, x_n\xi_1(x), x_n\xi_2(x), \ldots, x_n\xi_R(x)]^T$$

and

$$\theta = [a_{1,0}, a_{2,0} \ldots, a_{R,0}, a_{1,1}, a_{2,1} \ldots, a_{R,1}, \ldots, a_{1,n}, a_{2,n} \ldots, a_{R,n}]^T$$

so that

$$f(x/\theta) = \theta^T \xi(x)$$

represents the Takagi-Sugeno-Kang fuzzy system, and we see that it is linear in the parameters.

Just as for a standard fuzzy system, we can use batch or recursive least squares for training $f(x/\theta)$. To do this simply pick (a priori) the $\mu_i(x)$ and hence the $\xi_i(x)$ vector, process the training data $x^i$ where $(x^i, y^i) \in G$ through $\xi(x)$, and replace $x^i$

with $\xi(x^i)$ in forming the $\Phi$ vector for batch least squares, or in Equation (2.23) for recursive least squares.

Note that the above approach to training will work for any nonlinear that is linear in the parameters. For instance, if there are known nonlinearities in the system of the quadratic form, we can use the same basic approach as the one described above to specify the parameters of consequent functions that are quadratic.

## 3.4 Batch Least Squares Training of Fuzzy Systems

To answer the question of how to train fuzzy systems with batch least squares, we will consider how to tune the fuzzy system

$$f(x/\theta) = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}$$

(However, other forms may be used equally effectively). Here, $b_i$ is the point in the output space at which the output membership function for the $i^{th}$ rule achieves a maximum, $c_j^i$ is the point in the $j^{th}$ input universe of discourse where the membership function for the $i^{th}$ rule achieves a maximum, and $\sigma_j^i > 0$ is the relative width of the membership function for the $j^{th}$ input and the $i^{th}$ rule. Clearly, we are using center-average defuzzification and product for the premise and implication. Notice that the outer most input membership functions do not saturate as is the usual case in control.

We will tune $f(x/\theta)$ to interpolate the data set $G$ given in Equation (2.3) Choosing $R$ = 2 and noting that $n = 2$, we have $\theta = [b_1, b_2]^T$ and

$$\xi_i(x) = \frac{\prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)} \qquad (3.4)$$

Next, we must pick the input membership function parameters $c_j^i, i = 1,2, j = 1,2.$ one way to choose the input membership function parameters is to use the $x^i$ portions of the first $R$ data pairs in $G$. In particular, we could make the premise of rule $i$ have unity certainty if $x^i, (x^i, y^i) \in G$ is input to the fuzzy system, $i = 1,2,\ldots,R, R \leq M$; For instance, if $x^1 = [0,2]^T = [x_1^1, x_2^1]^T$ *and* $x^2 = [2,4]^T = [x_1^2, x_2^2]^T$, we would choose $c_1^1 = x_1^1 = 0, c_2^1 = x_2^1 = 2, c_1^2 = x_1^2 = 2, c_2^2 = x_2^2 = 4$.

Another approach to picking the $c_j^i$ is simply to try to spread the membership functions somewhat evenly over the input portion or the training data space. For instance, consider the axes on the left of Figure (2.2) where the input portions of the training data are shown for $G$. From inspection, a reasonable choice for the input membership function centers could be $c_1^1 = 1.5, c_2^1 = 3, c_1^2 = 3, c_2^2 = 5$ since this will place the peaks of the premise membership functions in between. The input portions of the training data pairs. In our example, we will use this choice of $c_j^i$.

Next we need to pick the spreads $\sigma_j^i$. To do this we simply pick $\sigma_j^i = 2, i = 1,2, j = 1,2$ as a guess that we hope will provide reasonable overlap between the membership functions. This completely specifies the $\xi_i(x)$ in Equation (3.4).

Let $\xi(x) = [\xi_1(x), \xi_2(x)]^T$. We have M = 3 for G, so we find

$$\Phi = \begin{bmatrix} \xi^T(x^1) \\ \xi^T(x^2) \\ \xi^T(x^3) \end{bmatrix} = \begin{bmatrix} 0.8634 & 0.1366 \\ 0.5234 & 0.4766 \\ 0.2173 & 0.7827 \end{bmatrix}$$

and $Y = [y^1, y^2, y^3]^T = [1,5,6]^T$. We use the batch least squares formula in Equation (3.3) to find $\theta = [0.3646, 8.1779]^T$, and hence our fuzzy system is $f(x/\theta)$.

To test the fuzzy system, note that at the training data

$$f(x^1/\hat{\theta}) = 1.4320$$
$$f(x^2/\hat{\theta}) = 4.0883$$
$$f(x^3/\hat{\theta}) = 6.4798$$

so that the trained fuzzy system maps the training data reasonably accurately ($x^3 = [3,6]^T$).

Next, we test the fuzzy system at some points not in the training data set to see how it interpolates. In particular, we find

$$f([1,2]^T/\hat{\theta}) = 1.8267$$
$$f([2.5,5]^T/\hat{\theta}) = 5.3981$$
$$f([4,7]^T/\hat{\theta}) = 7.3673$$

These values seem like good interpolated values considering Figure (2.2), which illustrates the data set $G$ for this example.

## 3.5 Recursive Least Squares Training of Fuzzy Systems

Here, we illustrate the use of the RLS algorithm in Equation (2.23) for training a fuzzy system to map the training data given in $G$ in Equation (2.3). First, we replace $x^k$ with $\xi(x^k)$ to obtain

$$P(x) = \frac{1}{\lambda}(1 - P(k-1)\xi(x^k)(\lambda I + (\xi(x^k))^T P(k-1)(\xi(x^k))^{-1}(\xi(x^k))^T)P(k-1)$$

$$\hat{\theta}(x) = \theta(k-1) + P(k)\xi(x^k)(y^k - (\xi(x^k))^T\hat{\theta}(k-1) \qquad (3.5)$$

and we use this to compute the parameter vector of the fuzzy system. We will train the same fuzzy system that we considered in the batch least squares example of the previous section, and we pick the same $c_j^i$ and $\sigma_j^i, i = 1,2, j = 1,2$ as we chose there so that we have the same.

$$\xi(x) = [\xi_1(x), \xi_2(x)]^T$$

for initialization of Equation (3.5), we choose

$$\hat{\theta}(0) = [2,5.5]^T$$

as a guess of where the output membership function centers should be. Another guess would be to choose $\hat{\theta}(0) = [0,0]^T$.Next, using the guidelines for RLS initialization, we choose

$$P[0] = \alpha I$$

where $\alpha = 2000$. We choose $\lambda = 1$ since we do not want to discount old data, and hence we use the standard (non-weighted) RLS.

Before using Equation (3.5) to find an estimate of the output membership function centers, we need to decide in what order to have RLS process the training data pairs $(x^i, y^i) \in G$. For example, you could just take three steps with Equation (3.5), one for each training data pair. Another approach would be to use each $(x^i, y^i) \in G, N_i$ times (in some order) in Equation (3.5) then stop the algorithm. Still another approach would be to cycle through all the data (i.e. $(x^1, x^1)$ first, $(x^2, y^2)$ second, up to $(x^M, y^M)$ then go back to $(x^1, y^1)$ and repeat), say, $N_{RLS}$ times. It is the last approach that we will use and we will choose $N_{RLS}$.

After using Equation (3.5) to cycle through the data $N_{RLS}$ times, we get the last estimate

$$\hat{\theta}(N_{RLS} - M) = \begin{bmatrix} 0.3647 \\ 8.1778 \end{bmatrix} \tag{3.6}$$

and

$$P(N_{RLS} - M) = \begin{bmatrix} 0.0685 & -0.429 \\ -0.0429 & 0.0851 \end{bmatrix}$$

Notice that the values produced for the estimates in Equation (3.6) are very close to the values we found with batch least squares—which we would expect since RLS is derived from batch least squares. We can test the resulting fuzzy system in the same way as we did for the one trained with batch least squares. Rather than showing the results, we simply note that since $\hat{\theta}(N_{RLS} - M)$ produced by RLS is very similar to the $\hat{\theta}$ produced by batch least squares, the resulting fuzzy system is quite similar, so we get very similar values for $f(x/\hat{\theta}(N_{RLS} - M))$ as we did for the batch least squares case.

## 3.6 Summary

In this chapter the description of fuzzy systems is described, both standard and Takagi-Sugeno-Kang fuzzy systems are presented.

The use of batch and recursive least squares to train fuzzy systems are explained.

# CHAPTER FOUR
# FUZZY IDENTIFICATION USING CLUSTERING METHODS

## 4.1 Overview

"Clustering" is the partitioning of data into subsets or groups based on similarities between the data. Here, we will introduce two methods to perform fuzzy clustering where we seek to use fuzzy sets to define soft boundaries to separate data in to groups. The methods here are related to conventional ones that have been developed in the field of pattern recognition.

We begin with a fuzzy "c-means" technique coupled with least squares to train Takagi-Sugeno-Kang fuzzy systems, and then we briefly study a nearest neighborhood method for training standard fuzzy systems. In the c-means approach, we continue in the spirit of the previous methods in that we use optimization to pick the clusters and, hence, the premise membership function parameters. The consequent parameters are chosen using the weighted least squares approach developed earlier. The nearest neighborhood approach also uses a type of optimization in the construction of cluster centers and, hence, the fuzzy system.

## 4.2 Clustering with Optimal Output Predefuzzification

In this section we will introduce the clustering with optimal output predefuzzification approach to train Takagi-Sugeno-Kang fuzzy systems. We do this via the simple example we have used in previous chapters.

### 4.2.1 Clustering for Specifying Rule Premises

Fuzzy clustering is the partitioning of a collection of data into fuzzy subsets or "clusters" based on similarities between the data and can be implemented using an algorithm called fuzzy c-means. Fuzzy c-means is an iterative algorithm used to find grades of membership $\mu_{ij}$ (scalars) and cluster centers $\underline{v}^j$ (vectors of dimension $n \times 1$) to minimize the objective function

$$J = \sum_{i=1}^{M} \sum_{j=1}^{R} (\mu_{ij})^m \left| x^i - \underline{v}^j \right|^2 \quad (4.1)$$

where $m > 1$ is a design parameter. Here, $M$ *is* the number of input-output data pairs in the training data set G, $R$ is the number of clusters (number of rules) we wish to calculate, $X^i$ for $i = 1,...,M$ is the input portion of the input-output training data pairs, $\underline{v}^j = [v_1^j, v_2^j, ..., v_n^j]^T$ for; $j$= 1,2,...,$R$ are the cluster centers, and $\mu_{ij}$ for $i = 1,...,M$ and $j$=1,..,. $R$ is the grade of membership of $x^i$ in the $j^{th}$ cluster. Also $|x| = \sqrt{x^T x}$ where $x$ is a vector, Intuitively, minimization of $J$ results in cluster centers being placed to represent groups (clusters) of data.

Fuzzy clustering will be used to form the premise portion of the If-Then rules in the fuzzy system we wish to construct. The process of "optimal output predefuzzification" (least squares training for consequent parameters) is used to form the consequent portion of the rules. We will combine fuzzy clustering and optimal output predefuzzification to construct multi-input single-output fuzzy systems. Extension to multi-input multi-output systems can be done by repeating the process for each of the outputs.

In this section we utilize a Takagi-Sugeno-Kang fuzzy system in which the consequent portion of the rule-base is a function of the crisp inputs such that If $H^j$.

Then

$$g_j(x) = a_{j;0} + a_{j;1}x_1 + ... + a_{j;n}x_n \quad (4.2)$$

where $n$ is the number of inputs and $H^j$ is an input fuzzy set given by

$$H^j = \{(x, \mu_{H^j}(x)) : x \in X_1 \times ... \times X_n\} \quad (4.3)$$

where $X_i$ is the $i^{th}$ universe of discourse, and $\mu_{H^j}(x)$ *is* the membership function associated with $H^j$ that represents the premise certainty for rule $j$; and $g_j(x) = \underline{a}_j^T \hat{x}$ where $\underline{a}_j = [a_{j,0}, a_{j,1}, \ldots, a_{j,n}]^T$ and $\hat{x} = [1, x^T]^T$ where $j = 1, \ldots, R$.

The resulting fuzzy system is a weighted average of the output $g_j(x)$ for $j = 1, \ldots, R$ and is given by

$$f(x|\theta) = \frac{\sum_{j=1}^{R} g_j(x)\mu_{H^j}(x)}{\sum_{j=1}^{R} \mu_{H^j}(x)} \tag{4.4}$$

where $R$ is the number of rules in the rule-base. Next, we will use the Takagi-Sugeno-Kang fuzzy model, fuzzy clustering, and optimal output predefuzzification to determine the parameters $\underline{a}_j$ and $\mu_{H^j}(x)$, which define the fuzzy system. We will do this via a simple example.

Suppose we use the example data set in Equation (2.3) that has been used in the previous sections. We first specify a "fuzziness factor" $m > 1$ which is a parameter that determines the amount of overlap of the clusters. If $m > 1$ is large, then points with less membership in the $j^{th}$ cluster have less influence on the determination of the new cluster centers. Next, we specify the number of clusters $R$ *we* wish to calculate. The number of clusters $R$ equals the number of rules in the rule-base and must be less than or equal to the number of data pairs in the training data set $G (i.e., R \leq M)$. We also specify the error tolerance $\epsilon_c > 0$, which is the amount of error allowed in calculating the cluster centers. We initialize the cluster centers $\underline{v}_0^j$ via a random number generator so that each component of $\underline{v}_0^j$ *is* no larger (smaller) than the largest (smallest) corresponding component of the input portion of the training data. The selection $\underline{v}_0^j$, although somewhat arbitrary, may affect the final solution.

For our simple example, we choose $m=2$ and $R=2$, and let $\epsilon_c = 0.001$. Our initial cluster centers were randomly chosen to be

$$\underline{v}_0^1 = \begin{bmatrix} 1.89 \\ 3.76 \end{bmatrix}$$

and

$$\underline{v}_0^2 = \begin{bmatrix} 2.47 \\ 4.76 \end{bmatrix}$$

so that each component lies in between $x_1^i$ and $x_2^i$, for i = 1,2, 3 (see the definition of $G$ in Equation (2.3)).

Next, we compute the new cluster centers $\underline{v}_0^j$ based on the previous cluster centers so that the objective function in Equation (4.1) is minimized. The necessary conditions for minimizing $J$ are given by

$$\underline{v}_{new}^j = \frac{\sum_{i=1}^{M} x^i (\mu_{ij}^{new})^m}{\sum_{i=1}^{M} (\mu_{ij}^{new})^m} \tag{4.5}$$

where

$$\mu_{ij}^{new} = \left[ \sum_{k=1}^{R} \left( \frac{\left| x^i - \underline{v}_{old}^j \right|^2}{\left| x^i - \underline{v}_{old}^k \right|^2} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{4.6}$$

for each $i$= 1,2,...,$M$ and for each $j$=1,2,...R such that $\sum_{j=1}^{R} \mu_{ij}^{new} = 1$ (and $|x|^2 = x^T x$). In Equation (4.6) we see that it is possible that there exists an $i$=1, 2,..., $M$ such that $\left| x^i - \underline{v}_{old}^j \right|^2 = 0$ for some $j$=1,2,...$R$. In this case the $\mu_{ij}^{new}$ is undefined. To fix this problem, let $\mu_{ij}$ for all $i$ be any nonnegative numbers such that $\sum_{j=1}^{R} \mu_{ij} = 1$ and $\mu_{ij} = 0$, if $\left| x^i - \underline{v}_{old}^j \right|^2 \neq 0$.

Using Equation (4.6) for our example with $\underline{v}^j_{old} = \underline{v}^j_0$, $j$ = 1,2, we find that $\mu^{new}_{11} = 0.6729, \mu^{new}_{12} = 0.3271, \mu^{new}_{21} = 0.9197, \mu^{new}_{22} = 0.0803, \mu^{new}_{31} = 0.2254, and \mu^{new}_{32} = 0.7746.$

We use these $\mu^{new}_{ij}$ from Equation (4.6) to calculate the new cluster centers

$$\underline{v}^1_{new} = \begin{bmatrix} 1.366 \\ 3.4043 \end{bmatrix}$$

and

$$\underline{v}^2_{new} = \begin{bmatrix} 2.8381 \\ 5.7397 \end{bmatrix}$$

using Equation (4.5).

Next, we compare the distances between the current cluster centers $\underline{v}^j_{new}$ and the pervious cluster centers $\underline{v}^j_{old}$ (which for the first step is $\underline{v}^j_0$). If $\left|\underline{v}^j_{new} - \underline{v}^j_{old}\right| < \in_c$ for all $j = 1,2,\ldots,R$ then the cluster centers $\underline{v}^j_{new}$ accurately represent the input data, the fuzzy clustering algorithm is terminated, and we proceed on to the optimal output defuzzification algorithm. Otherwise, we continue to iteratively use Equations (4.5) and (4.6) until we find cluster centers $\underline{v}^j_{new}$ that satisfy $\left|\underline{v}^j_{new} - \underline{v}^j_{old}\right| < \in_c$ for all $j = 1, 2,\ldots, R$.

For our example, $\underline{v}^j_{old}$= $\underline{v}^j_0$ and we see that $\left|\underline{v}^j_{new} - \underline{v}^j_{old}\right| = 0.6328$ for $j$ = $1$ and 0.6260 for $j$ =$2$. Both of these values are greater than $\in_c$. So we continue to update the cluster centers.

Proceeding to the next iteration, let $\left|\underline{v}^j_{old} = \underline{v}^j_{new}\right|$, $j$ =$1,2,\ldots, R$ from the last iteration and apply Equations (4.5) and (4.6) to find $\mu^{new}_{11}$=0.8233, $\mu^{new}_{12}$= 0.1767, $\mu^{new}_{21}$=0.7445, $\mu^{new}_{22}$= 0.2555, $\mu^{new}_{31}$ = 0.0593, and $\mu^{new}_{32}$= 0.9407 using the cluster centers calculated above, yielding the new cluster centers

$$\underline{v}^1_{new} = \begin{bmatrix} 0.9056 \\ 2.9084 \end{bmatrix}$$

and

$$\underline{v}^2_{new} = \begin{bmatrix} 2.8381 \\ 5.7397 \end{bmatrix}$$

Computing the distances between these cluster centers and the previous ones, we find that $\left|\underline{v}^j_{new} - \underline{v}^j_{old}\right| > \epsilon_c$, so the algorithm continues. It takes 14 iterations before the algorithm terminates (i.e., before we have $\left|\underline{v}^j_{new} - \underline{v}^j_{old}\right| \leq \epsilon_c = 0.001$, for all $j$=1.2,...,R). When it does terminate, name the final membership grade values $\mu_{ij}$ and cluster centers $\underline{v}^j$, $i$= 1,2,...,M, $j$=1,2,...,R.

For our example, after 14 iterations the algorithm finds $\mu_{11} = 0.9994$, $\mu_{12} = 0.0006$, $\mu_{21} = 0.1875$, $\mu_{22} = 0.8125$, $\mu_{31} = 0.0345$, $\mu_{32} = 0.9655$,

$$\underline{v}^1 = \begin{bmatrix} 0.0714 \\ 2.0725 \end{bmatrix}$$

and

$$\underline{v}^2 = \begin{bmatrix} 2.5854 \\ 5.1707 \end{bmatrix}$$

Notice that the clusters have converged so that $\underline{v}^1$ is near $x^1 = [0,2]^T$ and $\underline{v}^2$ lies in between $x^2 = [2,4]^T$ *and* $x^3 = [3,6]^T$.

The final values of $\underline{v}^j$, $j$ = 1, 2,..., $R$, are used to specify the premise membership functions for the $i^{th}$ rule. In particular, we specify the premise membership functions as

$$\mu_{H^j}(x) = \left[ \sum_{k=1}^{R} \left( \frac{\left| x - \underline{v}^j \right|^2}{\left| x - \underline{v}^k \right|^2} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{4.7}$$

$j$ = !. 2,..., $R$ where $\underline{v}^j$ ,,j= 1. 2..... $R$ are the cluster centers from the last iteration that uses Equations (4.5) and (4.6). It is interesting to note that for large values of m we get smoother (less distinctive) membership functions. This is the primary guideline to use in selecting the value of m; however, often a good first choice is m = 2. Next, note that $\mu_{H^j}(x)$ is a premise membership function that is different from any that we have considered. It is used to ensure certain convergence properties of the iterative fuzzy c-means algorithm described above. With the premises of the rules defined, we next specify the consequent portion.

### 4.2.2 Least Squares for Specifying Rule Consequents

We apply "optimal output predefuzzification" to the training data to calculate the function $g_j(x) = \underline{a}_j^T \hat{x}, j = 1,2,\ldots,R,$ for each rule (i.e., each cluster center), by determining the parameters $\underline{a}_j$ There are two methods you can use to find $\underline{a}_j$.

#### 4.2.2.1 Approach 1

For each cluster center $\underline{v}^j$, we wish to minimize the squared error between the function $g_j(x)$ and the output portion of the training data pairs. Let $\hat{x}^i = [1, (x^i)^T]^T$ where $(x^i, y^i) \in G$. We wish to minimize the cost function $J_j$, given by

$$J_j = \sum_{i=1}^{M} (\mu_{ij})^2 \left( y^i - (\hat{x}^i)^T \underline{a}_j \right)^2 \tag{4.8}$$

for each $j = 1,2,..., R$ where $\mu_{ij}$ is the grade of membership of the input portion of the $i^{th}$ data pair for the $j^{th}$ cluster that resulted from the clustering algorithm after it converged, $y^i$ is the output portion of the $i^{th}$ data pair $d^{(i)} = (x^i, y^i)$ , and the multiplication of $(\hat{x}^i)^T$ and $\underline{a}_j$ defines the output associated with the $j^{th}$ rule for the $i^{th}$ ' training data point.

Looking at Equation (4.8), we see that the minimization of $J_j$, via the choice of the $\underline{a}_j$ is a weighted least squares problem. From Equation (3.2), the solution $\underline{a}_j$, for $j$=1,2,…,$R$ to the weighted least squares problem is given by

$$\underline{a}_j = (\hat{X}^T D_j^2)^{-1} \hat{X}^T D_j^2 Y \tag{4.9}$$

where

$$\hat{x} = \begin{bmatrix} 1 \cdots 1 \\ x^1 \cdots x^m \end{bmatrix}^T$$

$$Y = \left[y^1,...,y^m\right]^T$$

$$D_j^2 = \left(diag\left(\left[\mu_{1j},...,\mu_{Mj}\right]\right)\right)^2$$

For our example the parameters that satisfy the linear function

$$g_j(x) = \underline{a}_j^T \hat{x}, j = 1,2$$

such that $J_j$ in Equation (4.8) is minimized were found to be $\underline{a}_1 = [3,2.999,-1]^T$ and $\underline{a}_2 = [3,3,-1]^T$ which are very close to each other.

#### 4.2.2.2 Approach 2

As an alternative approach, rather than solving $R$ least squares problems, one for each rule, we can use the least squares methods we used earlier to specify the consequent parameters of the Takagi-Sugeno-Kang fuzzy system. To do this, we simply parameterize the Takagi-Sugeno-Kang fuzzy system in Equation (4.4) in a form so that it is linear in the consequent parameters and of the form

$$f(x/\theta) = \theta^T \xi(x)$$

where $\theta$ holds all the $a_{i,j}$ parameters and $\xi$ is specified in a similar manner to how we did earlier. Now, we can use batch or recursive least squares methods to find $\theta$.

### 4.2.3 Testing the Approximator

Suppose that we use approach 1 to specify the rule consequents. To test how accurately the constructed fuzzy system represents the training data set G in Figure 2.2, suppose that we choose the test point $x'$ such that $(x', y') \notin G$. Specifically, we choose

$$x' = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

We would expect from Figure 2.2 that the output of the fuzzy system would lie somewhere between 1 and 5. The output is 3.9999, so we see that the trained Takagi-Sugeno-Kang fuzzy system seems to interpolate adequately. Notice also that if we let $x = x^i, i = 1,2,3.$ where $(x^i, y^i) \in G$, we get values very close to the $y^i, i = 1,2,$ 3, respectively. That is, for this example the fuzzy system nearly perfectly maps the training data pairs. We also note that if the input to the fuzzy system is $x = [2.5,5]^T$ the output is 5.5, so the fuzzy system seems to perform good interpolation near the training data points.

Finally, we note that the $\underline{a}_j$ will clearly not always be as close to each other as for this example. For instance, if we add the data pair $([4,5]^T, 5.5)$ to G (i.e., make $M = 4$), then the cluster centers converge after 13 iterations (using the same parameters $m$ and $\in_c$ as we did earlier). Using approach 1 to find the consequent parameters, we get

$$\underline{a}_1 = [-1.458, 0.7307, 1.2307]^T$$

and

$$\underline{a}_2 = [2.999, 0.00004, 0.5]^T$$

For the resulting fuzzy system, if we let $x = [1, 2]^T$ in Equation (4.4), we get an output value of 1.8378, so we see that it performs differently than the case for $M = 3$, but that it does provide a reasonable interpolated value.

## 4.3 Nearest Neighborhood Clustering

As with the other approaches, we want to construct a fuzzy estimation system that approximates the function $g$ that is inherently represented in the training data set $G$. We use singleton fuzzification, Gaussian membership functions, product inference, and center-average defuzzification, and the fuzzy system that we train is given by

$$f(x|\theta) = \frac{\sum_{i=1}^{R} A_i \prod_{j=1}^{n} \exp\left(-\left(\frac{x_j - v_j^i}{2\sigma}\right)^2\right)}{\sum_{i=1}^{R} B_i \prod_{j=1}^{n} \exp\left(-\left(\frac{x_j - v_j^i}{2\sigma}\right)^2\right)} \tag{4.10}$$

where $R$ is the number of clusters (rules), $n$ is the number of inputs

$$\underline{v}^j = [\underline{v}_1^j, \underline{v}_2^j, \ldots, \underline{v}_n^j]^T$$

are the cluster centers, $\sigma$ is a constant and is the width of the membership functions, and A, and B, are the parameters whose values will be specified below (to train a multi-output fuzzy system, simply apply the procedure to the fuzzy system that generates each output).

From Equation (4.10), we see that the parameter vector $\theta$ is given by

The optimal output predefuzzification method was applied first to a static system represented by a sinusoidal function, a course performance system and to a dynamic tank water level system. The results of identification were given. As a result of identification it was clear that fuzzy identification of system gives more accurate model than sub-space identification method.

# CONCLUSION

The fuzzy techniques are found to be one of the most effective tools for building mathematical models and thus to the identification of systems that are found to be characterized with uncertainty and fuzziness. In this thesis identification of fuzzy system is considered.

The state of application problems of fuzzy technology for identification of systems is given.

The function approximation problem was described and the use of conventional methods such as batch and recursive least squares are represented for identification of parameters of model.

The use of these conventional techniques for tuning fuzzy systems was considered and implied to train standard and Takagi-Sugeno fuzzy models.

The training of fuzzy models using clustering techniques are demonstrated, the use of optimal output predefuzzification method to train Takagi-Sugeno fuzzy models and nearest neighborhood clustering to train standard fuzzy models were represented.

The use of fuzzy identification for static and dynamic systems was described using the optimal output predefuzzification technique. Fuzzy identification of parameters of mathematical models of course performance system and tank water level system is represented. Using Matlab package the fuzzy identification of course performance system and tank water level system is carried out

The thesis demonstrated how to design, develop and implement fuzzy identification techniques to applications in order to get better models of systems. These models can be used in different applications in the fields of control, estimation, prediction, etc.

The results comparison of fuzzy identification with conventional identification demonstrates the efficiency of fuzzy technology.

For further studies we can see that there are some accuracy losses in some regions of our model, these undesirable variances should be evaluated in order to increase the efficiency of the system. This topic along with the incorporation of linguistic information in to our system should affect our model accuracy.

# REFERENCES

[1] Pedrycz W., *Fuzzy Modeling Paradigms and Practice*, Kluwer Academic Publishers , Polt, Dortrecht, The Netherlands, 1996..

[2] W. Pedrycz, *Fuzzy Control and Fuzzy Systems*, Research Studies Press/Wiley & Sons, London, 1989

[3] Terano, T., K. Asai and M. Sugeno, "*Applied Fuzzy Systems*", Boston: Academic Press, Inc., 1994

[4] Zadeh L. A.," *Fuzzy Sets, Information and Control*", vol 8,pp. 338-353,1976.

[5] J. Abonyi, R. Babuska, M. Setnes, H.B. Verbruggen, and F. Szeifert, "*Constrained parameter estimation in fuzzy modeling*". In Proceedings of FUZZ-IEEE'99, pages 951–956, Seoul, Korea, August 1999.

[6] P. J. Costa Branco, J. A. Dente, "*A new algorithm for on-line relational identification of nonlinear dynamic systems*", In Second IEEE Int. Conf. on Fuzzy Systems (IEEEFUZZ'93),pp.1073-1079, 1993.

[7] Gancho Vachkov and Toshio Fukuda, "*Simplified Fuzzy Model Based Identification of Dynamical Systems*", International Journal of Fuzzy Systems, Vol. 2, No. 4,pp.229-235 ,December 2000.

[8] S.-K. Sin and R. J. Defigueiredo, "*Fuzzy system design through fuzzy clustering and optimal predefuzzification,*" in 2nd IEEE Conference on Fuzzy Systems, San Fransico, California, pp. 190–195, 1993.

[9] Laukonen E.G., Passino K.M., "*Training Fuzzy Systems to Perform Estimation and Identification*" Engineering Applications of Artificial Intelligence, Vol. 8, No. 5, pp. 499–514, 1995.

[10] H. Tanaka and H. Ishibuchi, "*Identification of possibilistic linear systems by quadratic membership functions of fuzzy parameters,*" Fuzzy Sets and Systems, vol. 41, pp. 145–160, 1991.

[11] L.-X.Wang, "*Fuzzy systems are universal approximators*", in 1st IEEE conference on fuzzy systems, pp. 1163–1170, March 1992.

[12] Takagi T. and Sugeno M., "*Fuzzy Identification of Systems and its Applications to Modeling and Control*", IEEE *Trans.Systems,Man & Cybernetics* 15(1): 116–132, (1985).

[13] Kerimov K. M., Abiyev R. H., Melikov T. G., Fuzzy System for Determination of Oil-gas Fields Productivity/Geophysics News in Azerbaijan. Baku, N:1,pp.13-15, (1999).

[14] J. Abonyi R. Babuska, "Local and global identification and interpretation of parameters in Takagi–Sugeno fuzzy models", Delft University of Technology, Department of Information Technology and Systems Control Engineering Laboratory, The Netherlands.

[15] Kerimov K. M., Abiyev R. H., Melikov T. G., *"Determination of Oil and Gas Field Productivity in The Condition of Insufficiency and Fuzziness of Information"*, Second International Symposium on Mathematical and Computational Applications, Baku, September 1-3, (1999).

[16] Robert Draeseke and David E. A. Giles, *"A Fuzzy Logic Approach to Modelling the Underground Economy"*, Department of Economics, University of Victoria, Canada, 2000.

[17] Sugeno M. and Kang G.T., "*Fuzzy modeling and control of multilayer incinerator*". Fuzzy Sets and Systems 18, 329, 1986.

[18] Zhao, J., V. Wertz and R. Gorez, "*A fuzzy clustering method for the identification of fuzzy models for dynamical systems*", In 9th IEEE International Symposium on Intelligent Control, Columbus, Ohio, USA. 1994.

[19] Babuska R., H.J.L. van Can and H.B. Verbruggen, "*Fuzzy modeling of enzymatic Penicillin–G conversion*", In Preprints 13th IFACWorld Congress, Volume N, San Francisco, USA, pp. 479–484, (1996).

[20] Sugeno, M. and K. Tanaka, "*Successive identification of a fuzzy model and its application to prediction of a complex system*", Fuzzy Sets and Systems 42, pp. 315–334, 1991.

[21] Kaymak, U., "*Application of fuzzy methodologies to a washing process*", Chartered designer thesis, Delft University of Technology, Control Lab., Faculty of El. Eng., Delft, 1994.

[22].Hartog, den, M.H., R. Babuska, H.J.R. Deketh, M. Alvarez Grima, P.N.W. Verhoef and H.B. Verbruggen, "*Knowledge-based fuzzy model for performance prediction of a rock cutting trencher*", International Journal of Approximate Reasoning 16(1),pp. 43–66, 1997.

[23] Bektaş Ş., Mamedov F. and Khashman A., *The Graduate Studies: A Complete Guide*, Near East University press, Nicosia (2001).

# APPENDIX A

## Sinusoidal Function Identification

### Program

```
u = (0:0.02:1)';
y = sin(7*u);
i=2;
Par.c=i;
Dat.U=u;
Dat.Y=y;
[ym,VAF,FM] = Clust_LMS(Dat,Par,u,y);v(1)=0.0;
v(i)=VAF;
while(v(i)>v(i-1))
    i=i+1;
    Par.c=i;
    [ym,VAF,FM] = Clust_LMS(Dat,Par,u,y);
    v(i)=VAF;
    if (v(i)<=v(i-1))
        Par.c=i-1;
        [ym,VAF,FM] = Clust_LMS(Dat,Par,u,y);
        v(i-1)=VAF;
        plot(y);
        hold on;
        plot(ym,'-.r');
        break;
    end;
end;
```

### Fuzzy model parameters

```
FM =         Ts: 1

             ni: 1

             no: 1

              N: 51

            tol: 0.0100

           seed: 2.1181e+005

           date: [2003 5 25 22 22 41]

             ny: 0

             nu: 1

             nd: 0

           ante: 1

              c: 9
```

```
          m: 2
  InputName: []
 OutputName: []
      Alist: {[1]}
      Clist: {[1 2]}
        rls: {[5x1 double]}
        mfs: {[]}
         th: {[5x2 double]}
          s: {[5x2 double]}
          V: {[5x1 double]}
          P: {[2x2x5 double]}
       zmin: {[0 -0.9989]}
       zmax: {[1 0.9995]}
```

**VAF** = 99.8701%

**Model performance**

| *Input Data* | *Output Data* | *Model Output* |
|---|---|---|
| 0 | 0 | 0.0229 |
| 0.0200 | 0.1395 | 0.1254 |
| 0.0400 | 0.2764 | 0.2529 |
| 0.0600 | 0.4078 | 0.4052 |
| 0.0800 | 0.5312 | 0.5539 |
| 0.1000 | 0.6442 | 0.6570 |
| 0.1200 | 0.7446 | 0.7270 |
| 0.1400 | 0.8305 | 0.8078 |
| 0.1600 | 0.9001 | 0.9072 |
| 0.1800 | 0.9521 | 0.9791 |
| 0.2000 | 0.9854 | 0.9833 |
| 0.2200 | 0.9995 | 0.9708 |
| 0.2400 | 0.9940 | 0.9860 |
| 0.2600 | 0.9691 | 0.9964 |
| 0.2800 | 0.9252 | 0.9417 |
| 0.3000 | 0.8632 | 0.8396 |
| 0.3200 | 0.7843 | 0.7562 |
| 0.3400 | 0.6901 | 0.6919 |
| 0.3600 | 0.5823 | 0.6122 |
| 0.3800 | 0.4632 | 0.4927 |
| 0.4000 | 0.3350 | 0.3361 |
| 0.4200 | 0.2002 | 0.1710 |
| 0.4400 | 0.0616 | 0.0285 |

| | | |
|---|---|---|
| 0.4600 | 0.0783- | 0.0861- |
| 0.4800 | 0.2167- | 0.1923- |
| 0.5000 | 0.3508- | 0.3139- |
| 0.5200 | 0.4780- | 0.4585- |
| 0.5400 | 0.5959- | 0.6100- |
| 0.5600 | 0.7021- | 0.7388- |
| 0.5800 | 0.7946- | 0.8244- |
| 0.6000 | 0.8716- | 0.8702- |
| 0.6200 | 0.9315- | 0.8994- |
| 0.6400 | 0.9731- | 0.9383- |
| 0.6600 | 0.9957- | 0.9925- |
| 0.6800 | 0.9989- | 1.0310- |
| 0.7000 | 0.9825- | 1.0164- |
| 0.7200 | 0.9468- | 0.9515- |
| 0.7400 | 0.8926- | 0.8662- |
| 0.7600 | 0.8210- | 0.7834- |
| 0.7800 | 0.7333- | 0.7091- |
| 0.8000 | 0.6313- | 0.6355- |
| 0.8200 | 0.5169- | 0.5467- |
| 0.8400 | 0.3924- | 0.4282- |
| 0.8600 | 0.2602- | 0.2794- |
| 0.8800 | 0.1229- | 0.1152- |
| 0.9000 | 0.0168 | 0.0457 |
| 0.9200 | 0.1562 | 0.1917 |
| 0.9400 | 0.2925 | 0.3193 |
| 0.9600 | 0.4231 | 0.4299 |
| 0.9800 | 0.5454 | 0.5266 |
| 1.0000 | 0.6570 | 0.6124 |

# APPENDIX B

## Course Performance Identification

### Program

```
AA=4.0;BA=3.5;BB=3.0;CB=2.5;CC=2.0;DC=1.5;DD=1.0;FD=0.5;FF=
0.0;
inp=[AA BA BB CB CC DC DD FD FF];
out=[3 4 11 17 14 10 18 7 5];
u=inp';
y=out';
i=2;
Par.c=i;
Dat.U=u;
Dat.Y=y;
[ym,VAF,FM] = Clust_LMS(Dat,Par,u,y);v(1)=0.0;
v(i)=VAF;
while(v(i)>v(i-1))
    i=i+1;
    Par.c=i;
    [ym,VAF,FM] = Clust_LMS(Dat,Par,u,y);
    v(i)=VAF;
    if (v(i)<=v(i-1))
        Par.c=i-1;
        [ym,VAF,FM] = Clust_LMS(Dat,Par,u,y);
        v(i-1)=VAF;
        plot(y);
        hold on;
        plot(ym,'-.r');
        break;
    end;
end;
```

### Fuzzy model parameters

```
FM =        Ts: 1

            ni: 1

            no: 1

             N: 9

           tol: 0.0100

          seed: 2.0899e+005

          date: [2003 5 25 23 8 25]

            ny: 0

            nu: 1
```

```
         nd: 0
       ante: 1
          c: 5
          m: 2
  InputName: []
 OutputName: []
      Alist: {[1]}
      Clist: {[1 2]}
        rls: {[5x1 double]}
        mfs: {[]}
         th: {[5x2 double]}
          s: {[5x2 double]}
          V: {[5x1 double]}
          P: {[2x2x5 double]}
       zmin: {[0 3]}
       zmax: {[4 18]}
```

**VAF =** 100 %( c = 5), 20.55 %( c = 2).

**Model performance**

| *Input Data* | *Output Data* | *Model Output(c=2)* | *Model Output(c=5)* |
|---|---|---|---|
| 4.0000 | 3 | 6.6521 | 3.0000 |
| 3.5000 | 4 | 7.8280 | 4.0000 |
| 3.0000 | 11 | 9.3347 | 11.0000 |
| 2.5000 | 17 | 12.2907 | 17.0000 |
| 2.0000 | 14 | 14.0457 | 14.0000 |
| 1.5000 | 10 | 7.0070 | 10.0000 |
| 1.0000 | 18 | 9.3117 | 18.0000 |
| 0.5000 | 7 | 10.6981 | 7.0000 |
| 0 | 5 | 11.8320 | 5.0000 |

# APPENDIX C

## Tank Water Level System Identification

### Program

```
i=2;
c = i;
m = 1.2;
tol = 0.01;
Ts = 2;
FMtype = 2;
Ny = 1;
Nu = [1];
Nd = [1];
load tankdata1
skip = 1;
N = size(system,1);
N2 = floor(size(system,1)/2);

u = control(1:skip:N2);
y = system(1:skip:N2);

ue = control(N2+1:skip:N);
ye = system(N2+1:skip:N);


Dat.U = u; Dat.Y = y; Dat.Ts = Ts; Par.c = c;
Par.ante = FMtype; Par.tol = tol;
Dyn.Ny = Ny; Dyn.Nu = Nu; Dyn.Nd = Nd;

[ym,VAF,FM] = Clust_LMS(Dat,Par,ue,ye,Dyn);

v(1)=0.0;
v(i)=VAF;
while(v(i)>v(i-1))
    i=i+1;
    Par.c=i;
    [ym,VAF,FM] = Clust_LMS(Dat,Par,ue,ye,Dyn);
    v(i)=VAF;
    if (v(i)<=v(i-1))
        Par.c=i-1;
        [ym,VAF,FM] = Clust_LMS(Dat,Par,ue,ye,Dyn);
        v(i-1)=VAF;
        plot(ye);
        hold on;
        plot(ym,'-.r');
        break;
    end;
end;
```

**Fuzzy model parameters**

```
FM =          Ts: 2
              ni: 1
              no: 1
               N: 2000
             tol: 0.0100
            seed: 2.0908e+005
            date: [2003 5 26 2 48 12]
              ny: 1
              nu: 1
              nd: 1
            ante: 2
               c: 3
               m: 2.2000
       InputName: []
      OutputName: []
           Alist: {[1 2]}
           Clist: {[1 2 3]}
             rls: {[3x2 double]}
             mfs: {{1x2 cell}}
              th: {[3x3 double]}
               s: {[3x3 double]}
               V: {[3x2 double]}
               P: {[3x3x3 double]}
            zmin: {[0.0437 0.2000 0.0437]}
            zmax: {[0.9926 1 0.9926]}
```

**VAF** = 98.04 %.

**Model performance**

(Note: Fragment of data is given)

| I/P(C) | O/P(C) | I/P(T) | O/P(T) | O/P(M) |
|---|---|---|---|---|
| 0.7512 | 0.5038 | 0.5180 | 0.4938 | 0.4938 |
| 0.7610 | 0.5083 | 0.4679 | 0.4735 | 0.4785 |
| 0.8165 | 0.5156 | 0.4654 | 0.4521 | 0.4597 |
| 0.8721 | 0.5278 | 0.4628 | 0.4321 | 0.4419 |
| 0.8062 | 0.5387 | 0.5210 | 0.4162 | 0.4251 |
| 0.7403 | 0.5425 | 0.5792 | 0.4071 | 0.4149 |
| 0.6766 | 0.5397 | 0.5617 | 0.4006 | 0.4109 |
| 0.6128 | 0.5310 | 0.5442 | 0.3929 | 0.4055 |
| 0.6664 | 0.5224 | 0.5637 | 0.3859 | 0.3989 |
| 0.7201 | 0.5196 | 0.5832 | 0.3813 | 0.3945 |
| 0.7283 | 0.5199 | 0.6300 | 0.3803 | 0.3923 |
| 0.7366 | 0.5210 | 0.6768 | 0.3839 | 0.3946 |

| | | | | |
|---|---|---|---|---|
| 0.6932 | 0.5203 | 0.6084 | 0.3860 | 0.4012 |
| 0.6498 | 0.5155 | 0.5400 | 0.3814 | 0.4008 |
| 0.5899 | 0.5060 | 0.6132 | 0.3776 | 0.3941 |
| 0.5300 | 0.4913 | 0.6865 | 0.3810 | 0.3947 |
| 0.5429 | 0.4755 | 0.6915 | 0.3879 | 0.4022 |
| 0.5559 | 0.4619 | 0.6966 | 0.3948 | 0.4096 |
| 0.5733 | 0.4509 | 0.6750 | 0.4003 | 0.4170 |
| 0.5908 | 0.4423 | 0.6534 | 0.4033 | 0.4218 |
| 0.5375 | 0.4325 | 0.6992 | 0.4073 | 0.4243 |
| 0.4843 | 0.4184 | 0.7451 | 0.4154 | 0.4309 |
| 0.4930 | 0.4031 | 0.7665 | 0.4261 | 0.4413 |
| 0.5017 | 0.3899 | 0.7880 | 0.4381 | 0.4531 |
| 0.5063 | 0.3784 | 0.8346 | 0.4525 | 0.4660 |
| 0.5109 | 0.3681 | 0.8813 | 0.4704 | 0.4824 |
| 0.5423 | 0.3605 | 0.8494 | 0.4877 | 0.5021 |
| 0.5737 | 0.3564 | 0.8175 | 0.5007 | 0.5174 |
| 0.5266 | 0.3519 | 0.8274 | 0.5118 | 0.5286 |
| 0.4795 | 0.3432 | 0.8372 | 0.5231 | 0.5400 |
| 0.5363 | 0.3358 | 0.8468 | 0.5346 | 0.5515 |
| 0.5932 | 0.3344 | 0.8563 | 0.5462 | 0.5631 |
| 0.5302 | 0.3328 | 0.9039 | 0.5599 | 0.5748 |
| 0.4672 | 0.3252 | 0.9514 | 0.5773 | 0.5902 |
| 0.5452 | 0.3192 | 0.9263 | 0.5946 | 0.6090 |
| 0.6233 | 0.3211 | 0.9011 | 0.6084 | 0.6241 |
| 0.6458 | 0.3276 | 0.9129 | 0.6207 | 0.6358 |
| 0.6683 | 0.3357 | 0.9247 | 0.6334 | 0.6477 |
| 0.6339 | 0.3425 | 0.8409 | 0.6418 | 0.6600 |
| 0.5994 | 0.3455 | 0.7571 | 0.6415 | 0.6636 |
| 0.6805 | 0.3506 | 0.7955 | 0.6392 | 0.6590 |
| 0.7615 | 0.3630 | 0.8338 | 0.6407 | 0.6584 |
| 0.7661 | 0.3785 | 0.8085 | 0.6427 | 0.6614 |
| 0.7707 | 0.3932 | 0.7832 | 0.6421 | 0.6618 |
| 0.8045 | 0.4087 | 0.7693 | 0.6396 | 0.6598 |
| 0.8383 | 0.4262 | 0.7555 | 0.6360 | 0.6566 |
| 0.8135 | 0.4429 | 0.7579 | 0.6321 | 0.6524 |
| 0.7886 | 0.4559 | 0.7603 | 0.6286 | 0.6487 |
| 0.8561 | 0.4701 | 0.7698 | 0.6259 | 0.6455 |
| 0.9235 | 0.4899 | 0.7793 | 0.6243 | 0.6434 |
| 0.9014 | 0.5104 | 0.6883 | 0.6188 | 0.6423 |
| 0.8794 | 0.5274 | 0.5972 | 0.6048 | 0.6327 |
| 0.9170 | 0.5440 | 0.5269 | 0.5839 | 0.6153 |
| 0.9545 | 0.5632 | 0.4566 | 0.5575 | 0.5925 |
| 0.9486 | 0.5826 | 0.4609 | 0.5297 | 0.5647 |
| 0.9426 | 0.6003 | 0.4651 | 0.5041 | 0.5392 |
| 0.9166 | 0.6153 | 0.4856 | 0.4814 | 0.5158 |
| 0.8906 | 0.6268 | 0.5060 | 0.4623 | 0.4961 |
| 0.9807 | 0.6408 | 0.4619 | 0.4434 | 0.4796 |
| 1.0000 | 0.6599 | 0.4178 | 0.4216 | 0.4601 |

|  |  |  |  |  |
|---|---|---|---|---|
| 1.0000 | 0.6781 | 0.4693 | 0.4019 | 0.4378 |
| 0.9986 | 0.6953 | 0.5208 | 0.3886 | 0.4219 |
| 1.0000 | 0.7114 | 0.4797 | 0.3767 | 0.4119 |
| 1.0000 | 0.7266 | 0.4385 | 0.3618 | 0.3988 |
| 1.0000 | 0.7409 | 0.4300 | 0.3458 | 0.3827 |
| 1.0000 | 0.7545 | 0.4214 | 0.3302 | 0.3669 |
| 1.0000 | 0.7672 | 0.4257 | 0.3157 | 0.3514 |
| 1.0000 | 0.7793 | 0.4300 | 0.3029 | 0.3374 |
| 1.0000 | 0.7907 | 0.4278 | 0.2913 | 0.3247 |
| 1.0000 | 0.8015 | 0.4255 | 0.2805 | 0.3128 |
| 0.9868 | 0.8113 | 0.4004 | 0.2694 | 0.3014 |
| 0.9604 | 0.8184 | 0.3753 | 0.2569 | 0.2885 |
| 1.0000 | 0.8259 | 0.4311 | 0.2470 | 0.2741 |
| 1.0000 | 0.8348 | 0.4868 | 0.2435 | 0.2660 |
| 1.0000 | 0.8432 | 0.4696 | 0.2420 | 0.2636 |
| 1.0000 | 0.8512 | 0.4524 | 0.2390 | 0.2598 |
| 1.0000 | 0.8587 | 0.4833 | 0.2371 | 0.2547 |
| 0.9779 | 0.8652 | 0.5143 | 0.2382 | 0.2528 |
| 0.9787 | 0.8699 | 0.5263 | 0.2413 | 0.2539 |
| 0.9795 | 0.8745 | 0.5383 | 0.2452 | 0.2561 |
| 1.0000 | 0.8805 | 0.5676 | 0.2507 | 0.2593 |
| 1.0000 | 0.8865 | 0.5969 | 0.2585 | 0.2649 |
| 1.0000 | 0.8922 | 0.6032 | 0.2672 | 0.2730 |
| 1.0000 | 0.8976 | 0.6095 | 0.2758 | 0.2811 |
| 1.0000 | 0.9027 | 0.6292 | 0.2848 | 0.2892 |
| 1.0000 | 0.9076 | 0.6489 | 0.2949 | 0.2986 |
| 1.0000 | 0.9122 | 0.6443 | 0.3048 | 0.3092 |
| 1.0000 | 0.9166 | 0.6398 | 0.3134 | 0.3186 |
| 0.9893 | 0.9205 | 0.6541 | 0.3217 | 0.3269 |
| 0.9657 | 0.9222 | 0.6684 | 0.3307 | 0.3360 |
| 0.9290 | 0.9209 | 0.7662 | 0.3444 | 0.3459 |
| 0.8922 | 0.9161 | 0.8640 | 0.3664 | 0.3642 |
| 0.8778 | 0.9091 | 0.8510 | 0.3906 | 0.3904 |

# APPENDIX D

## Clustering Function

```
function [Ymd,Vf,FM,Mu,Z] = Clust_LMS(Dat,Par,Uin,Yin,Dyn)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin <5, Dyn = []; end;
if isfield(Dat,'U'); U = Dat.U;
elseif isfield(Dat,'X')'; U = Dat.X;
elseif isfield(Dat,'x')'; U = Dat.x;
elseif isfield(Dat,'u')'; U = Dat.u; else U = []; end;
if isfield(Dat,'Y'); Y = Dat.Y;
elseif isfield(Dat,'y'); Y = Dat.y; else Y = []; end;
Ninps = size(U,2); [Nd,NO] = size(Y);
if isfield(Dat,'Ts'); Ts = Dat.Ts; else Ts = 1; end;
if isfield(Dat,'file'); file = Dat.file; else file = ''; end;
if isfield(Dat,'N'); N = Dat.N; else N = Nd; end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ninps = size(U,2);
[Nd,NO] = size(Y);

if isfield(Dyn,'Ny'); ny = Dyn.Ny; else ny = zeros(NO,NO); end;
if isfield(Dyn,'Nu'); nu = Dyn.Nu; else nu = ones(NO,Ninps); end;
if isfield(Dyn,'Nd'); nd = Dyn.Nd; else nd = zeros(NO,Ninps); end;
if Ninps == 0, nu = zeros(NO,1); nd = nu; end;

if isfield(Par,'c'); c = Par.c; else c = 2; end;
if isfield(Par,'ante'); ante = Par.ante; else ante = 1; end;
if isfield(Par,'m'); m = Par.m; else m = 2; end;
if isfield(Par,'tol'); tol = Par.tol; else tol = 0.01; end;
if isfield(Par,'seed'); seed = Par.seed; else seed = sum(100*clock);
end;

if max(size(c)) == 1, c = c*ones(1,NO); end;
if max(size(m)) == 1, m = m*ones(1,NO); end;
if max(size(ante)) == 1, ante = ante*ones(1,NO); end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clstep = 1;
show = 0;
lstep = 1;
show = 1;
MFTYPE = 2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 1 : NO,

Alist = 1:(sum(ny(k,:))+sum(nu(k,:)));
Clist = [Alist Alist(length(Alist))+1];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nc = [0; cumsum(N(:))];
dat = [];
for b = 1:size(Nc,1)-1,
  if Ninps == 0,
    z = Y(Nc(b)+1:Nc(b+1),:);
  else
    z = [Y(Nc(b)+1:Nc(b+1),:) U(Nc(b)+1:Nc(b+1),:)];
  end;
  [yr,ur,yl]=regres([Y(Nc(b)+1:Nc(b+1),k)                    z],0,[ny(k,:)
nu(k,:)],[ones(1,NO).*(ny(k,:)>0) nd(k,:)]);
  dat = [dat;[ur yl]];                              % data to cluster
```

```
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ND = size(dat,1);
NI = size(dat,2)-1;
dat = dat(1:clstep:ND,:);
rand('seed',seed);
[mu,V,P] = gkfast(dat,c(k),m(k),tol,0);
[dum,ind] = sort(V(:,1));
mu = mu(:,ind);
V = V(ind,:);
P = P(:,:,ind);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(NI,NI,c(k));
for j = 1 : c(k),
  M(:,:,j)                                                                   =
det(P(Alist,Alist,j)).^(1/length(Alist))*inv(P(Alist,Alist,j));
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mfs = [];
if ante(k) == 2,

  range   = [min(dat(:,1:NI))'  max(dat(:,1:NI))'];
  perc = 50;
  safety = perc*0.01*(range(:,2) - range(:,1));
  rlimits = [range(:,1)-safety range(:,2)+safety];

  mfstep = round(ND/100); if mfstep < 1, mfstep = 1; end;
  OPT = foptions; OPT(14) = 1000;
  for i = 1 : NI,
    [ds,fs]=smooth(dat(1:mfstep:ND,i),mu(1:mfstep:ND,:));
    mf = mffit(ds,fs,MFTYPE,OPT,[1 0 0]);
    lim      =      mf(:,3)       ==      min(mf(:,3));      mf(lim,2:3)       =
ones(sum(lim),1)*[rlimits(i,1) rlimits(i,1)];
    if mf(lim,4) < rlimits(i,1), mf(lim,4) = rlimits(i,1); end;
    lim      =      mf(:,4)       ==      max(mf(:,4));      mf(lim,4:5)       =
ones(sum(lim),1)*[rlimits(i,2) rlimits(i,2)];
    if mf(lim,3) > rlimits(i,2), mf(lim,3) = rlimits(i,2); end;
    mfs{i} = mf;
  end;
end;
rls = (1:c(k))'*ones(1,NI);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x1 = ones(size(dat,1),1);
if ante(k) == 1,
   d = [];
      for j = 1 : c(k),
      xv = dat(:,Alist) - x1*V(j,Alist);
            if NI == 1,
            d(:,j) = M(:,:,j)*xv.^2;
         else
            d(:,j) = sum((xv*M(:,:,j).*xv)')';
         end;
      end;
      d = (d+1e-100).^(-1/(m(k)-1));
  dof = (d ./ (sum(d')'*ones(1,c(k))));

elseif ante(k) == 2,
  dof = dofprod(rls(:,Alist),mfs,dat(:,Alist));
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dc = [dat(:,1:NI) ones(size(dat(:,1)))];
```

```
[p,t,t,t,s] = suglms([dc(:,Clist)],dat(:,NI+1),dof,[],0);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
FM.Ts = Ts; FM.ni = Ninps; FM.no = NO; FM.N = Nd; FM.tol = tol;
FM.seed = seed; FM.date  = fix(clock);;
FM.ny = ny; FM.nu = nu; FM.nd = nd;
FM.ante = ante; FM.c = c; FM.m = m;
if isfield(Dat,'InputName'); FM.InputName = Dat.InputName;
else FM.InputName = []; end;
if isfield(Dat,'OutputName'); FM.OutputName = Dat.OutputName;
else FM.OutputName = []; end;

FM.Alist{k} = Alist; FM.Clist{k} = Clist; FM.rls{k} = rls; FM.mfs{k} =
mfs;
FM.th{k} = p; FM.s{k} = s; FM.V{k} = V(:,1:end-1); FM.P{k} = P;
FM.zmin{k} = min(dat); FM.zmax{k} = max(dat);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ymd,vf]=modelOP(Uin,Yin,FM);
Ymd=ymd;Vf=vf;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargout > 3, if NO > 1, Mu{k} = mu; else Mu = mu; end; end;
if nargout > 4, if NO > 1, Z{k} = dat; else Z = dat; end; end;

end;

if (show ~= 0),
  fprintf(1,['Done.\n']);
  drawnow
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mfs = mffit(x,f,mfini,OPT,flag)

if nargin < 4, OPT = foptions; end;
if ~all(size(OPT)), OPT = foptions; end;
if nargin < 5, flag = [1 0 0]; end;
delta = 0.1;

[sx,xi] = sort(x);
f = f(xi,:);
[m,n] = size(f);

if size(mfini,2) == 1,
  mftype = mfini.*ones(n,1);
  mfs = ones(n,1)*[sx(1) sx(1) sx(m) sx(m)];
  for i = 1 : n,
    peak(i) = mean(find(f(:,i)==max(f(:,i))));
  end;
  for j = 1 : n;
    for i = 1 : peak(j),
      if f(i,j) >= delta, minfl(j) = f(i,j); mfs(j,1) = sx(i); break;
end;
    end;
    for i = peak(j) : -1 : 1,
      if f(i,j) <= 1-delta, mfs(j,2) = sx(i); break; end;
    end;
    for i = peak(j) : m,
      if f(i,j) <= 1-delta, mfs(j,3) = sx(i); break; end;
    end;
    for i = m : -1 : peak(j),
      if f(i,j) >= delta, minfr(j) = f(i,j); mfs(j,4) = sx(i); break;
end;
    end;
    minx(j) = mfs(j,1); maxx(j) = mfs(j,4);
    if minfl(j) ~= 1, mfs(j,1) = mfs(j,2) - (mfs(j,2) - mfs(j,1)) ./
(1 - minfl(j)); end
    if minfr(j) ~= 1, mfs(j,4) = mfs(j,3) + (mfs(j,4) - mfs(j,3)) ./
(1 - minfr(j)); end
  end;
  mfs = [mftype mfs];
else
  mfs = mfini; mftype = mfini(:,1);
  minx = mfs(:,2); maxx = mfs(:,5);
end;

for i = 1 : n,
  if flag(1),
    p      =      fmins('mff',mfs(i,[2,5]),OPT,[],sx(sx>=minx(i)      &
sx<=maxx(i)),f(sx>=minx(i) & sx<=maxx(i),i),mfs(i,:),1);
    mfs(i,[2,5]) = p;
  end;
  if flag(2),
    p      =      fmins('mff',mfs(i,[3,4]),OPT,[],sx(sx>=minx(i)      &
sx<=maxx(i)),f(sx>=minx(i) & sx<=maxx(i),i),mfs(i,:),2);
    mfs(i,[3,4]) = p;
  end;
  if flag(3),
    p      =      fmins('mff',mfs(i,2:5),OPT,[],sx(sx>=minx(i)      &
sx<=maxx(i)),f(sx>=minx(i) & sx<=maxx(i),i),mfs(i,:),3);
    mfs(i,2:5) = p;
  end;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [p,ym,yl,ylm,ps,S,delta] = suglms(x,y,f,def,flag)


if nargin < 5, flag = 0; end;
if isempty(flag), flag = 0; end;
if nargin < 4, def = 0; end;
if isempty(def), def = 0; end;
if nargin < 3, f = ones(size(y)); end;
if isempty(f), f = ones(size(y)); end;
[mx,nx] = size(x); [mf,nf] = size(f); p = zeros(nf,nx); ps = p;
sumDOF = sum([zeros(1,mf);f'])';
sumDOF = sumDOF(:,ones(1,nf));
NoRule = (sumDOF == 0);
sumDOF = sumDOF + NoRule;
x1 = zeros(mx,nf*nx);
xx = zeros(nx,nf*mx);
x = x'; f1 = x(:); xx(:) = f1(:,ones(1,nf));
f1 = x1;
x1(:) = xx';

if nf == 1, flag = 1; end;

if flag == 0,

xx = f(:)./sumDOF(:);
f1(:) = xx(:,ones(1,nx));
x1 = f1.*x1;

if nargout > 5,
  [Q,R]=qr(x1);
  p(:) = R\Q'*y;
```

```
else
  p(:) = x1 \ y;
end;

if nargout > 1,
  yl = x'*p';
  ym = x1*p(:) + def.*NoRule(:,1);
  ylm = yl;
  mask = find(f < 0.2);
  ylm(mask) = NaN*ones(size(mask));
end;

else
for i = 1 : nf,
  f1 = f(:,i)*ones(1,nx);
  x1 = sqrt(f1).*x';
if nargout > 5,
  [Q,R]=qr(x1);
  p(i,:) = (R\Q'*(sqrt(f(:,i)).*y))';
else
  p(i,:) = (x1 \ (sqrt(f(:,i)).*y))';
end;

if nargout > 1,
  yl(:,i) = x'*p(i,:)';
  ym(:,i) = yl(:,i);
  ylm(:,i) = yl(:,i);
  mask = find(f(:,i) < 0.2);
  ylm(mask,i) = NaN*ones(size(mask));
end;

if nargout > 4,
  df = mx - nf*nx;
  Rr = x1'*x1;
  r = y - x1*p(i,:)';
  V=r'*r/df;
  M = V*inv(Rr);
  ps(i,:) = sqrt(diag(M))';
end;
end;
end;

if nargout > 5,
  S = [R(1:nf*nx,:); ...
      [df zeros(1,nf*nx-1)]; ...
      [norm(r) zeros(1,nf*nx-1)]];
end;

if nargout > 6,
  E = x1/R;
  e = sqrt(1+sum((E.*E)')');
  if df == 0
      delta = Inf*e;
  else
      delta = norm(r)/sqrt(df)*e;
  end
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [sx,fm] = smooth(x,f)
```

```
[sx,xi] = sort(x);
f = f(xi,:);
[m,n] = size(f);

b = [0.1367 0.1367];
a = [1 -0.7265];

for i = 1 : n,
  fi1 = filter(b,a,f(:,i),f(1,i));
  fi2 = filter(b,a,f(m:-1:1,i),f(m,i));
  fi(:,i) = (fi1 + fi2(m:-1:1)) / 2;
  fi(:,i) = fi(:,i)/max(fi(:,i));
  fi(1,i) = f(1,i);  fi(m,i) = f(m,i);
end;

lf = zeros(1,n); rf = lf;
left = zeros(size(f)); right = left;
for i = 1 : m,
  lf = max(lf,fi(i,:));
  rf = max(rf,fi(m-i+1,:));
  right(m-i+1,:) = rf;
  left(i,:) = lf;
end;
fm = min(left,right);

if nargout == 0, plot(sx,f,'o',sx,fm); end;

%**********************************************************
function [Ym,q,DOF,Yl,Ylm] = modelOP(U,Y,FM,Ymin,Ymax,show,H)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ts = FM.Ts;
Ninps = FM.ni;
NO = FM.no;
ny = FM.ny;
nu = FM.nu;
nd = FM.nd;
atype = FM.ante;
c = FM.c;
m = FM.m;
for k = 1 : NO,
  Nr(k) = size(FM.rls{k},1);
  Nm(k) = size(FM.mfs{k},1);
  Alist = FM.Alist{k};
  NI = length(Alist);
  P = FM.P{k}; M = zeros(NI,NI,c(k));
  for j = 1 : c(k),

    M(:,:,j)                                                          =
det(P(Alist,Alist,j)).^(1/length(Alist))*inv(P(Alist,Alist,j));

  end;
  FM.M{k} = M;
end;
NI = sum([ny'; zeros(1,NO)]) + sum([nu'; zeros(1,NO)]);

if isempty(Y), Y = zeros(size(U,1),NO); end;
if nargin < 4, Ymin = -inf*ones(1,NO); elseif isempty(Ymin), Ymin = -
inf*ones(1,NO); end;
if nargin < 5, Ymax =  inf*ones(1,NO); elseif isempty(Ymax), Ymax =
inf*ones(1,NO); end;
```

```
 if nargin < 6, show = 1; elseif isempty(show), show =  1; end;
 if nargin < 7, H = 0; elseif isempty(H), H =  0; end;

 DOF = zeros(size(U,1),sum(Nr));
 Yl  = DOF; Ylm  = Yl;
 cind = [0 cumsum(c)];
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 if max(max(ny)) == 0,    % static model

 for kk = 1 : NO,
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   Alist = FM.Alist{kk};
   Clist = FM.Clist{kk};
   p     = FM.th{kk};
   V     = FM.V{kk};
   M     = FM.M{kk};
   rls   = FM.rls{kk};
   mfs   = FM.mfs{kk};
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   M1 = zeros(NI(kk));
   V1 = ones(c(kk),1);
   x1 = ones(size(U,1),1);

   ante = U; cons = [ante ones(size(ante,1),1)];

   if atype(kk) == 1,
   for j = 1 : c(kk),
     xv = ante(:,Alist) - x1*V(j,Alist);
     M1 = M(:,:,j);
     if NI == 1,
       d(:,j) = M1*xv.^2;
     else
       d(:,j) = sum((xv*M1.*xv)')';
     end;
   end;
   d = (d+1e-100).^(-1/(m(kk)-1));
   dofe = (d ./ (sum(d')'*ones(1,c(kk))));
   elseif atype(kk) == 2,
     dofe = dofprod(rls(:,Alist),mfs,ante(:,Alist));
   end;

   [Ym(:,kk),Yl(:,cind(kk)+(1:c(kk))),Ylm(:,cind(kk)+(1:c(kk)))]        =
sugval(p(:,Clist),cons(:,Clist),dofe);
   DOF(:,cind(kk)+(1:c(kk))) = dofe;

end;

elseif H > 0,

N = size(Y,1);
Ym = Y;

for kk = 1 : NO,
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  Alist = FM.Alist{kk};
  Clist = FM.Clist{kk};
  p     = FM.th{kk};
  V     = FM.V{kk};
  M     = FM.M{kk};
  rls   = FM.rls{kk};
  mfs   = FM.mfs{kk};
```

```
    if Ninps == 0,
      z = Y;
    else
      z = [Y U];
    end;
    [yr,ur,y1]=regres([Y(:,kk)                                        z],0,[ny(kk,:)
 nu(kk,:)],[ones(1,NO).*(ny(kk,:)>0) nd(kk,:)]);
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    M1 = zeros(NI(kk));
    V1 = ones(c(kk),1);
    x1 = ones(size(ur,1),1);

    ante = ur; cons = [ante ones(size(ante,1),1)];

    if atype(kk) == 1,

    d = [];
    for j = 1 : c(kk),
      xv = ante(:,Alist) - x1*V(j,Alist);

      M1 = M(:,:,j);
      if NI == 1,
        d(:,j) = M1*xv.^2;
      else
        d(:,j) = sum((xv*M1.*xv)')';
      end;
    end;

    d = (d+1e-100).^(-1/(m(kk)-1));
    dofe = (d ./ (sum(d')'*ones(1,c(kk))));

    elseif atype(kk) == 2,
      dofe = dofprod(rls(:,Alist),mfs,ante(:,Alist));
    end;

    s1 = size(dofe,1);
    [Ym(N-s1+1:N,kk),Yl(N-s1+1:N,cind(kk)+(1:c(kk))),Ylm(N-
 s1+1:N,cind(kk)+(1:c(kk)))] = sugval(p(:,Clist),cons(:,Clist),dofe);
    DOF(N-s1+1:N,cind(kk)+(1:c(kk))) = dofe;

 end;

 else

 if Ninps > 0,
   k0 = max(max(max(max(ny))+1,max(max(nu))+max(max(nd))),2);
   if size(U,1) < k0, error(['Supply at least ' int2str(k0) ' data
 samples.']); end;
 else
   k0 = max(max(max(ny))+1,2);
 end;

 if Ninps > 0, kmax = size(U,1); else kmax = size(Y,1); end;

 Ym = Y;
 for k = k0 : kmax,

 for kk = 1 : NO,
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   Alist = FM.Alist{kk};
   Clist = FM.Clist{kk};
   p     = FM.th{kk};
   V     = FM.V{kk};
   M     = FM.M{kk};
   rls   = FM.rls{kk};
   mfs   = FM.mfs{kk};
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M1 = zeros(NI(kk));
V1 = ones(c(kk),1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   ante = [];
   for j = 1 : NO,
     ante = [ante Ym(k-1:-1:k-ny(kk,j),j)'];
   end;
   for j = 1 : Ninps,
     ante = [ante U(k-nd(kk,j):-1:k-nu(kk,j)-nd(kk,j)+1,j)'];
   end;
   cons = [ante 1];
   if atype(kk) == 1,
     xv = V1*ante(Alist) - V(:,Alist);
     d = [];
     for j = 1 : c(kk),
       M1 = M(:,:,j);
       d(j) = xv(j,:)*M1*xv(j,:)';
     end;
   d = (d+1e-100).^(-1/(m(kk)-1));
   dofe = (d ./ (sum(d')'*ones(1,c(kk))));

 elseif atype(kk) == 2,
     dofe = dofprod(rls(:,Alist),mfs,ante(Alist));
 end;

 ds = sum(dofe);
  NoRule = ds == 0;
  ds = ds + NoRule;
  yl = cons(Clist)*p(:,Clist)';
  ylm = yl;
  mask = find(dofe < max(dofe)*ones(1,Nr(kk)));
  ylm(mask) = NaN*ones(size(mask));

  Ym(k,kk) = yl*(dofe/ds)' + Ym(k-1,kk)*NoRule;
  Yl(k,cind(kk)+(1:c(kk))) = yl;
  Ylm(k,cind(kk)+(1:c(kk))) = ylm;
  DOF(k-1,cind(kk)+(1:c(kk))) = dofe;

  if Ym(k,kk) < Ymin(kk), Ym(k,kk) = Ymin(kk); end;
  if Ym(k,kk) > Ymax(kk), Ym(k,kk) = Ymax(kk); end;

 end;
end;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if size(Y,1)==size(Ym,1) & size(Ym,1)>1, q = vaf(Y,Ym); end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ym,yl,ylm,delta] = sugval(p,x,f,def,S)

if nargin < 4, def = 0; end;
if isempty(def), def = 0; end;
if nargin < 3, f = ones(size(x,1),size(p,1)); end;
if isempty(f), f = ones(size(x,1),size(p,1)); end;
```

```
[mx,nx] = size(x); [mf,nf] = size(f);
sumDOF = sum([zeros(1,mf);f'])';
sumDOF = sumDOF(:,ones(1,nf));
NoRule = sumDOF == 0;
sumDOF = sumDOF + NoRule;
yl = x*p';
ym = sum([zeros(1,mx); ...
        (yl.*f./sumDOF)'])' + def.*NoRule(:,1);
ylm = yl;
mask = find(f < max(f')'*ones(1,nf));
ylm(mask) = NaN*ones(size(mask));

if nargin > 4 & nargout > 3,
    nc = nf*nx;
    [ms,ns] = size(S);
    if (ms ~= ns+2) | (nc ~= ns)
        error('S matrix must be n+2-by-n where n = length(p(:))')
    end
    R = S(1:nc,1:nc);
    df = S(nc+1,1);
    normr = S(nc+2,1);

    x1 = zeros(mx,nf*nx);
    xx = zeros(nx,nf*mx);
    x = x'; f1 = x(:); xx(:) = f1(:,ones(1,nf));
    f1 = x1;
    x1(:) = xx';
    xx = f(:)./sumDOF(:);
    f1(:) = xx(:,ones(1,nx));
    x1 = f1.*x1;

    E = x1/R;
    e = sqrt(1+sum((E.*E)')');
    if df == 0
        delta = Inf*e;
    else
        delta = normr/sqrt(df)*e;
    end
end;
```