FUZZY ASSOCIATIVE MEMORY ARCHITECTURE

TING CHAN WAI

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2009

Fuzzy Associative Memory Architecture

by

Ting Chan Wai

A thesis submitted to the Nanyang Technological University in fulfillment of the requirement for the degree of Doctor of Philosophy
July 2008

Abstract

Artificial neural networks (ANNs) are systems that are deliberately constructed to make use of some organizational principles resembling those in the human brain. ANNs have a large number of highly interconnected processing elements (perceptrons) that usually operate in parallel and are configured in regular architectures. The collective behavior of an ANN, like a human brain, demonstrates the ability to learn, recall, and generalize from training patterns or data. They are good at tasks such as classification, function approximation, optimization, and data clustering [1]. The cerebellar model articulation controller (CMAC), a perceptron-like associative memory equipped with overlapping receptive field proposed by Albus [2], belongs to a special category of ANNs. It was first applied in the domain of control problems. During the past decades, its ability to capture nonlinear function has been demonstrated through many applications in control, function approximation and pattern recognition.

On the other hand, the development of fuzzy systems suffered from decades of controversy ever since the first fuzzy set theory proposed by Prof Lotfi A. Zadeh in 1965 [3]. It aims to alleviate difficulties in developing complex systems without mathematically analyzing the dynamics of the problem. It provides an intuitive channel between different facets of a problem; from quantitative aspect to qualitative aspect and vice versa. However, the development of fuzzy systems in the early days required the manual tuning of the system parameters based on observation of the system performance and this shortcoming has become a major criticism on the application of fuzzy set theory.

Over the years, similarities between ANNs and fuzzy systems attracted significant attentions from researchers from both fields to explore means to combine their individual advantages. The

synergy between artificial neural networks and fuzzy systems leads to the development of a neuro-fuzzy system which automatically identifies the parameters in a fuzzy system using neural network techniques. This thesis focuses on the development of such a synergy and addressed the theoretical problems with the following contributions:

A new class of CMAC, referred to as TSK^0-FCMAC (a localized self-organizing zero-ordered Takagi-Sugeno-Kang fuzzy inference system based on the CMAC structure) is proposed. The structures and learning algorithms of the proposed architecture are described in details. TSK^0-FCMAC employs a two-phase training algorithm. The rigid memory structure of conventional CMAC network is replaced with a novel clustering technique, Discrete Incremental Clustering (DIC). This technique enables the CMAC network to grow without prior knowledge of the number of clusters.

Many existing fuzzy CMACs are proposed without evidence of the convergence property. An investigation into the convergence characteristic of TSK^0-FCMAC is rigorously undertaken in this study. The mathematical formulation presented in this thesis provides a strong foundation for further investigation on the convergence characteristic of fuzzy CMACs with similar fuzzy inference scheme.

An extension of the proposed CMAC, TSK^1-FCMAC (based on the first-ordered Takagi-Sugeno-Kang fuzzy inference system) is also presented. This extension aims to improve the precision of its predecessor. The improvement in the memory requirement and utilization are realized by the use of a higher order function in the consequent part of the system.

The proposed neuro-fuzzy inference systems have been successfully applied in real life applications of three different areas; namely: 1) personalized drug delivery system (control), 2) rainfall runoff (regression) and, 3) embedded audio detection (classification). The ability of the proposed neuro-fuzzy inference systems in handling problems from these areas is demonstrated.

Thesis Supervisor: Dr. Quek Hiok Chai

Title: Assoc. Prof. in Division of Computer Science, School of Computer Engineering, Nanyang Technological University

Acknowledgements

I would like to thank my research advisor, Associate Professor Dr Quek Hiok Chai, for his consistent support and encouragement throughout the Ph.D. program. His guidance has given me fruitful experiences over the past eight years, during my undergraduate studies, my career in the industry, and my postgraduate research.

I also wish to express my appreciation to Associate Professor Ian McLoughlin for introducing me to this exciting research area. I would like to thank my colleagues, they are: Quah Kian Hong, Tung Whye Loon, Ang Kai Keng and Khoo Sui Yang, with whom I have discussed various research ideas. I am grateful to my best friend, Chong Kwen Siong, for contributing many constructive discussions. I gratefully express my appreciation to the Centre of Computational Intelligence, especially the laboratory technicians, Mr Tan Swee Huat and Mr Lau Boon Chee for providing the facilities for this research.

I would like to extend my gratitude to Nanyang Technological University for sponsoring my research scholarship.

I am indebted to my family for their consistent support. Finally, I would like to dedicate this work to my mother.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1. Motivation

Machine learning is concerned with the question of how to construct computer programs that automatically improve with experience [4]. Three keywords could be extracted from the above statement: "*automatically*", "*improve*" and "*experience*". The first keyword, "*automatically*", suggests the need of an algorithmic solution that is formally defined. The second keyword, "*improve*", signifies the requirement of a continuous learning capability for such solution and the last keyword, "*experience*", implies such learning capability is carried out through observation of historical examples. Alternatively, *machine learning* is also regarded as a subsidiary of artificial intelligence. To be intelligent, a system that is in a changing environment should have the ability to learn. If the system can learn and adapt to such changes, the system designer need not foresee and provide solutions for all possible situations [5]. That is, the needs for the system to memorize, understand and generalize.

The biologically-inspired *artificial neural networks* (ANN) [6, 7] and the fuzzy inference system based on the fuzzy set theory introduced by Professor Lotfi A. Zadeh [3] emerge as two plausible approaches to fulfill the needs in machine learning. Both approaches have their respective characteristics that are quite different and yet, complementary in nature. This attracts researchers from both fields to explore means to combine their individual advantages which lead to the development of neuro-fuzzy systems. This is also the motivation of our research and a summary of our research objective is presented in Figure 1-1. In the figure, the preferred and non-preferred characteristics of three existing systems are highlighted. The three existing systems are: 1) cerebellar model articulation controller (CMAC) proposed by Albus [2], 2) multilayer perceptron neural networks (MLP) and, 3) fuzzy systems. The two proposed architectures, TSK^0-FCMAC and TSK^1-FCMAC (localized self-organizing zero-ordered and first-ordered Takagi-Sugeno-Kang fuzzy inference system based on the CMAC structure) aim to combine the advantages from existing systems and, at the same time, complement the shortcomings of one another.

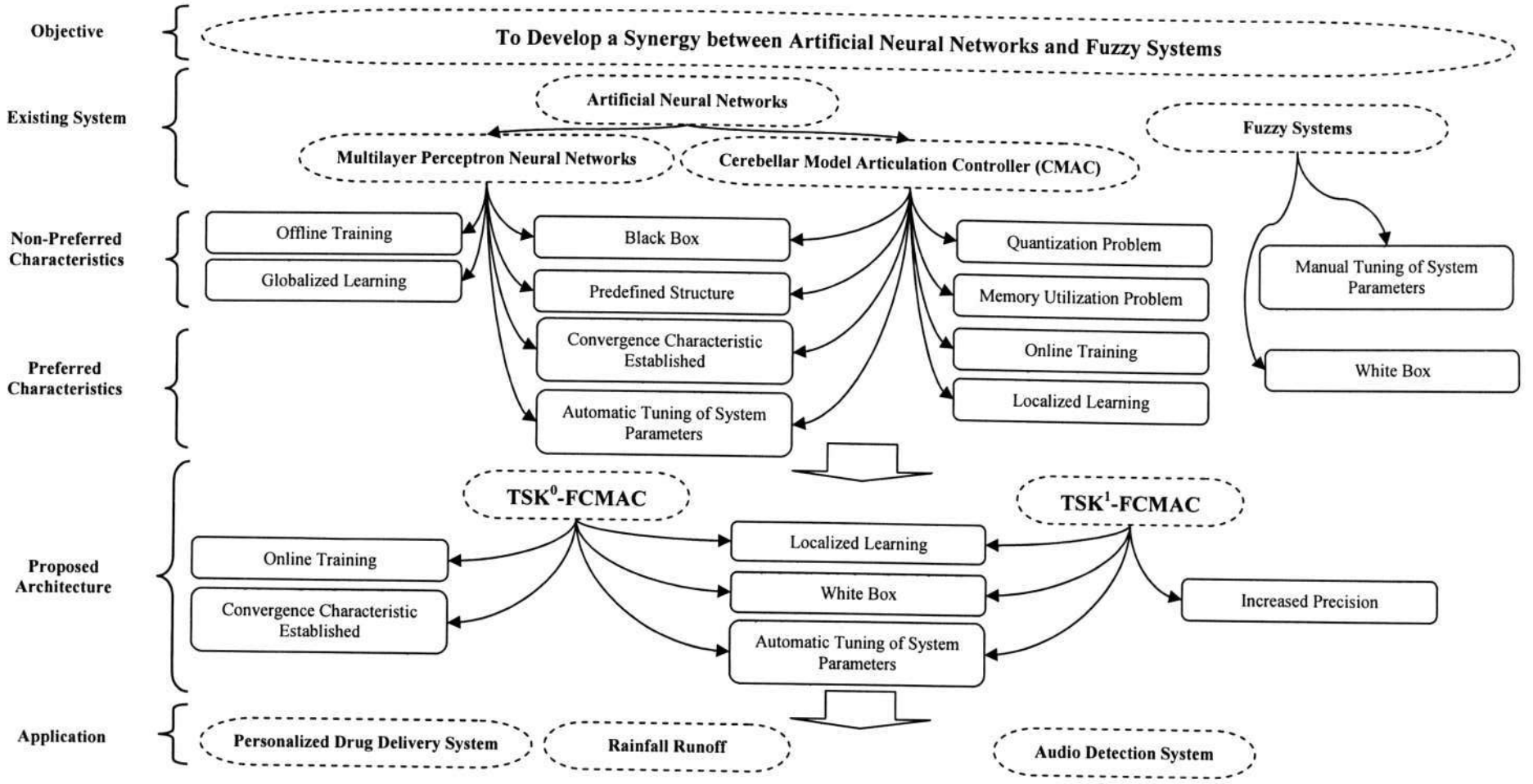


Figure 1-1: A Summary of Research Objective

1.2. Contributions of Dissertation

This thesis focuses on the development of a synergy between artificial neural networks and fuzzy systems and addressed the theoretical problems with the following contributions:

- A new class of CMAC, referred to as TSK^0-FCMAC (a localized self-organizing zero-ordered Takagi-Sugeno-Kang fuzzy inference system based on the CMAC structure), is proposed. The structures and learning algorithms of the proposed architecture are described in details. TSK^0-FCMAC employs a two-phase training algorithm. The rigid memory structure of conventional CMAC network is replaced with a novel clustering technique, Discrete Incremental Clustering (DIC). This technique enables the CMAC network to grow without prior knowledge of the number of clusters.

- An investigation into the convergence characteristic of TSK^0-FCMAC is rigorously undertaken. The mathematical formulation presented in this thesis provides a strong foundation for further investigation on the convergence characteristic of fuzzy CMAC with similar fuzzy inference scheme.

- An extension of the proposed CMAC, TSK^1-FCMAC (based on the first-ordered Takagi-Sugeno-Kang fuzzy inference system), is also presented. This extension aims to improve the precision of its predecessor. The improvement in the memory requirement and utilization are realized by the use of a higher order function in the consequent part of the system.

The proposed neuro-fuzzy inference systems have been successfully applied in real life applications of three different areas; namely: 1) personalized drug delivery system (control), 2) rainfall runoff (regression) and, 3) embedded audio detection (classification). The ability of the proposed neuro-fuzzy inference systems in handling problems from these areas is also demonstrated.

1.3. Organization of the thesis

This thesis is organized as follows:

- Chapter 2 presents a literature review on related systems and existing techniques. A brief introduction on fuzzy systems, artificial neural networks and cerebellar model articulation controller is first given, followed by discussions on the shortcomings and limitations on existing techniques.

- Chapter 3 presents the architecture and learning algorithm of the proposed TSK^0-FCMAC (a localized self-organizing zero-ordered Takagi-Sugeno-Kang fuzzy inference system based on the CMAC structure). The ability of the proposed architecture is demonstrated through applications on three benchmarking case-studies: 1) inverted pendulum problem (control), 2) Fisher's Iris classification (classification) and, 3) chaotic time series prediction (regression).

- Chapter 4 formally defines the mathematical model of the proposed TSK^0-FCMAC architecture. An investigation on the convergence characteristic of the model is rigorously undertaken.

- Chapter 5 presents an extension of TSK^0-FCMAC, the TSK^1-FCMAC (based on first-ordered Takagi-Sugeno-Kang fuzzy inference system). This extension aims to improve the precision of its predecessor by implementing a higher order function in the consequent part of the system. The proposed architecture has been applied to three benchmarking case-studies: 1) sinusoidal function/surface approximation (function approximation), 2) automobile MPG prediction (prediction) and, 3) two-spiral problem (classification).

- Chapter 6 to Chapter 8 present successful applications of the proposed architectures on three real-world applications; namely: 1) personalized drug delivery system (control), 2) rainfall runoff (regression) and, 3) embedded audio detection (classification).

- Chapter 9 concludes this research and provides recommendations for future directions.

Chapter 2 Literature Review

2.1. Fuzzy Systems

In 1979, Professor Lotfi A. Zadeh states that "*informally, by approximate or, equivalently, fuzzy reasoning, we mean the process or processes by which a possibly imprecise conclusion is deduced from a collection of imprecise premises. Such reasoning, is, for the most part, qualitative rather than quantitative in nature, and almost all of it falls outside the domain of applicability of classical logic*" [8]. This statement epitomizes the nature of life and draw attention to the existence of uncertainty in our everyday life. This uncertainty appears in the form of imprecision, vagueness and ill defined information and is not applicable in classical logic.

In classical logic, *modus ponens* states that, "*if a rule is true and the antecedent of the rule is true, then it can be inferred that the consequent of the rule is also true*". Yen [9] presented an interesting example to explain the connection between a fuzzy system and classical logic:

Given *Rule* 1, *Statement* 1 and *Statement* 2 are true.

Rule 1 : *IF the annual income of a person is greater than* 120*K*
THEN the person is rich
Statement 1 : *Jack's annual income is* 121*K*
Statement 2 : *Bob's annual income is* \$119,999

Based on *modus ponens*, classical logic can deduce that *Statement* 3 is also true.

Statement 3 : *Jack is rich*

People would usually say that Bob is somewhat rich. However *modus ponens* cannot infer whether Bob is rich or not using *Rule* 1 and *Statement* 2, even if Bob's annual income is only one dollar short. In fuzzy inference system, such limitation is removed by allowing the inferred conclusion to be modified by the degree to which the antecedent is satisfied, which, is the essence of a fuzzy inference system [9].

The term *fuzzy logic* has been used in two different senses. In a narrow sense, fuzzy logic refers to a logical system that generalizes classical two-valued logic for reasoning under uncertainty. In a broader sense, fuzzy logic refers to all of the theories and technologies that employ fuzzy sets,

which are classes with unsharp boundaries [9]. Figure 2-1 shows a fuzzy inference system, which is comprised of four principal components:

- A *fuzzifier* that transforms crisp measured data into suitable linguistic value;
- A *fuzzy rule base* that stores the empirical knowledge of the operation;
- An *inference engine* to simulate human decision making by performing approximate reasoning to achieve a desired output strategy and
- A *defuzzifier* to yield a non-fuzzy decision from an inferred fuzzy decision by the inference engine.

The fundamentals of fuzzy systems are described in the following section.

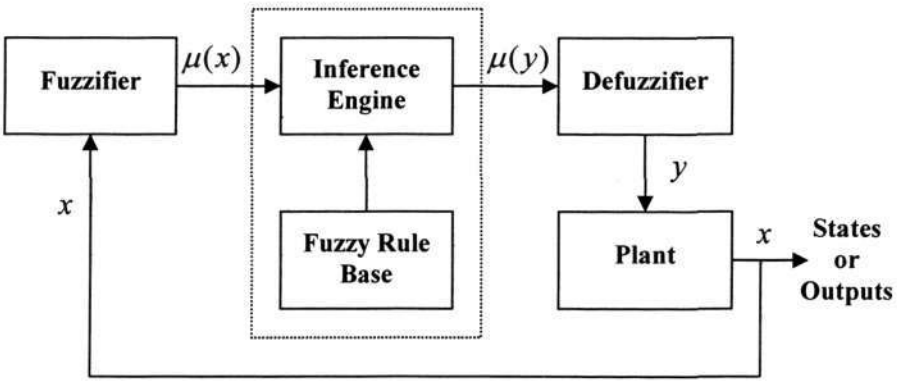


Figure 2-1: Fuzzy Inference System

2.1.1. Fuzzy Sets

The idea of fuzzy sets first occurred to Professor Lotfi A. Zadeh in 1964. He subsequently published the seminal paper on fuzzy sets in 1965 [3]. Based on this revolutionary idea, a new discipline has been formalized and it has attracted the attention from many researchers around the world. Detailed treatments of the fuzzy set theory can be found in [10-16]. *Fuzzy set* is a generalization of an ordinary set that allows the degree of membership for each element to range over the range of zero and one [1]. It generalizes the notion of membership from a binary categorization in classical set theory into one that allows partial membership [9]. The membership function μ_A is defined as

$$\mu_A = x_i \rightarrow [0,1], for\ \forall x_i \in X$$

where X is the *universe of discourse* and $\mu_A(x_i)$ is the membership value of element x_i defined by the fuzzy concept of A.

The fuzzy set A is defined as

$$A = \left\{ \left(x_i, \mu_A(x_i) \right) \middle| x_i \in X, \mu_A(x_i) \in [0,1] \right\}$$

or

$$A = \sum_{i=1}^{N} \frac{\mu_A(x_i)}{x_i}$$
$$= \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \ldots + \frac{\mu_A(x_N)}{x_N}$$

Where N is the number of elements in X. Figure 2-2 shows the graphical representation for fuzzy set A.

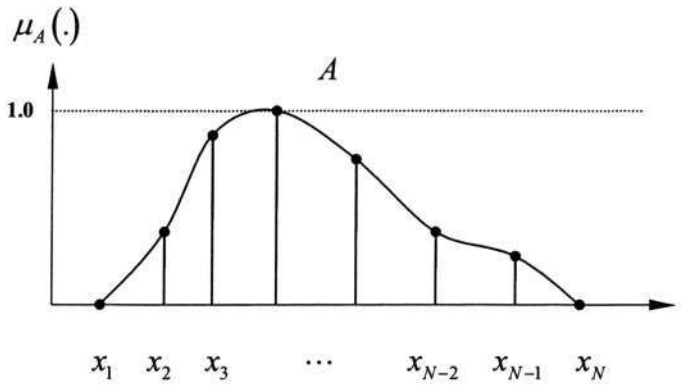


Figure 2-2: A Graphical Representation for Fuzzy Set A

2.1.1.1 Types of Membership Functions

Fuzzy set is represented with membership functions. Even though membership function of arbitrary shape can be defined, most fuzzy systems do not introduce non-parameterized membership functions. Instead, parameterized membership functions are introduced to reduce system design time and facilitate automated tuning of the system. The most commonly used

parameterized membership functions are triangles, trapezoids, bell curves, Gaussian, and sigmoidal functions. They are listed as follow:

1. Triangular Membership Function

$$triangle(x:a,b,c)=\begin{cases}(x-a)/(b-a) & a\le x\le b\\ (c-x)/(c-b) & b\le x\le c\\ 0 & ,otherwise\end{cases}$$

2. Trapezoidal Membership Function

$$trapezoid(x:a,b,c,d)=\begin{cases}(x-a)/(b-a) & a\le x<b\\ 1 & b\le x<c\\ (d-x)/(d-c) & c\le x<d\\ 0 & ,otherwise\end{cases}$$

3. Gaussian Membership Function

$$gaussian(x:m,\sigma)=\exp\left(\frac{(x-m)^2}{\sigma^2}\right)$$

4. Bell-shaped Membership Function

$$bell(x:a,b,c)=\frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}$$

5. Sigmoidal Membership Function

$$sigmoid(x:a,c)=\frac{1}{1+e^{-a(x-c)}}$$

2.1.1.2 Linguistic Variables

Similar to conventional set, a fuzzy set can be used to describe the value of a variable. For example, "This person is tall" uses a fuzzy set "tall" to describe the height of the particular person. The variable "height" in this example demonstrates an important concept in fuzzy logic: the *linguistic variables*. A linguistic variable enables its value to be described both qualitatively by a linguistic term (a symbol serving as the name of a fuzzy set) and quantitatively by a corresponding membership function (which expresses the meaning of the fuzzy set). The

linguistic term is used to express concepts and knowledge in human communication, whereas membership function is useful for processing numeric input data.

In the previous example regarding the height of a person, there are many other linguistic descriptions about the height such as "short", "medium", "marginal tall" and "very tall". One of the important concepts in fuzzy logic is that instead of enumerating all these different descriptions, they can be generated from a core set of linguistic terms using modifiers (e.g. "very", "more or less") and connectives (e.g. "and", "or"). In fuzzy logic, these modifiers are called *hedges*.

2.1.1.3 Possibility Distributions

Assigning a fuzzy set to a linguistic variable constrains the value of the variable. One major difference between classical logic and fuzzy set theory is that the notion of possible versus impossible values becomes a matter of degree. For instances, the membership function of a fuzzy set "tall" is shown in Figure 2-3. If fuzzy set "tall" is assigned to a person, the distribution about the possibility degree of the person's height is obtained.

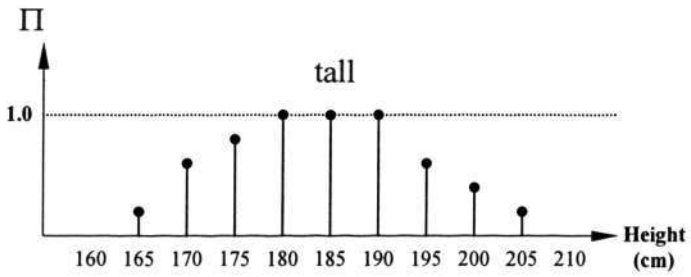


Figure 2-3: A possibility distribution of the fuzzy set "tall"

The possibility of the person is 175cm is 0.8, while the possibility of 180 through 190cm is 1. This assignment is denoted as:

$$\Pi_{Height(person)}(x) = \mu_{tall}(x)$$

where Π denotes a possibility distribution of the person's height, and x is a variable representing the person's height. For instance, if fuzzy set A is assigned to a variable X and is defined by A's membership function, this assignment can be denoted as:

$$\Pi_X(x) = \mu_A(x)$$

2.1.1.4 Defuzzification

The possibility distribution could be interpreted through *linguistic approximation* or through *defuzzification*. The former gives a qualitative interpretation, while the latter gives a quantitative summary and is more commonly used in fuzzy logic control. Defuzzification selects a single representative value that captures the crucial meaning of a given distribution. Three common defuzzification techniques are described in this section and the detailed descriptions are presented in [9].

1. Mean of Maximum (MOM)

The mean of maximum (MOM) defuzzification method calculates the average of those output values that have the highest possibility degrees. It can be expressed as

$$MOM(A) = \frac{\sum_{x \in P} x}{|P|}$$

where P is the set of output values x with highest possibility degree in A.

2. Center of Area (COA)

The center of area (COA) is also referred to as the center of gravity or the centroid. It is the most popular defuzzification technique. Unlike MOM, it takes into account the entire possibility distribution in calculating its representative point. It can be expressed as

$$COA(A) = \frac{\sum_x \mu_A(x) \times x}{\sum_x \mu_A(x)}$$

Similarly, if x is continuous, the result is

$$COA(A) = \frac{\int \mu_A(x) x dx}{\int \mu_A(x) dx}$$

3. The Height Method

The *height method* can be viewed as a two-step procedure. Given i fuzzy rules, each with consequent membership function C_i, the first step convert C_i into crisp consequent $y = c_i$ where c_i is the center of gravity of C_i. The centroid (COA) defuzzification is then applied to the crisp consequents. It can be expressed as

$$y = \frac{\sum_{i=1}^{M} w_i c_i}{\sum_{i=1}^{M} w_i}$$

where w_i is the degree to which the i th rule matches the input data. The main benefit of the height method is its simplicity. The calculation of c_i can be performed during compilation. Consequently, the only computation required during run-time is a normalized weighted sum.

2.1.2. Fuzzy If-then Rules

A fuzzy if-then rule associates a *condition* described using linguistic variables and fuzzy sets to a *conclusion*. The main feature of reasoning using these rules is its *partial matching* capability, which enables an inference to be made from a fuzzy rule even when the rule's condition is only partially satisfied. A fuzzy rule is the basic unit for capturing knowledge in many fuzzy systems. It consists of two components: an if-part (also referred to as the *antecedent*) and a then-part (also referred to as the *consequent*). The structure is shown as below:

IF <antecedent> THEN <consequent>

The structure of a fuzzy rule is identical to that of a conventional rule in artificial intelligence. The main difference lies in the content of the rule antecedent. The antecedent of a fuzzy rule describes an elastic condition (a condition that can be satisfied to a degree) while the antecedent of a conventional rule describes a rigid condition (a condition that is either satisfied or dissatisfied).

The consequent of fuzzy rules can be classified into three categories:

1. Crisp consequent: $IF \ldots THEN\ y = a$ where a is a non-fuzzy numeric value or a symbolic value.

2. Fuzzy consequent: $IF \ldots THEN\ y = A$ where A is a fuzzy set.

3. Functional consequent:

$$IF\ x_1\ is\ A_1\ AND\ x_2\ is\ A_2\ \ldots\ AND\ x_n\ is\ A_n\ THEN\ y = b_0 + \sum_{i=1}^{n} b_i \times x_i$$

where $b_0, b_1, \ldots, b_n$ are constants.

Each type of rule consequent has its merits. Fuzzy rule with crisp consequent can be processed more efficiently. A rule with a fuzzy consequent is easier to understand and more suitable for capturing imprecise human expertise while fuzzy rule with functional consequent can be used to approximate complex nonlinear models using only a small number of rules.

2.1.3. Fuzzy Rule-based Models

A fuzzy model is a model that is obtained by *fusing* multiple *local models* that are associated with fuzzy subspaces of the given input space. A *fuzzy subspace* is a region whose boundary allows a gradual transition from "inside the region" to "outside the region". Hence, a fuzzy subspace usually partially overlaps with its neighboring fuzzy subspaces. A fuzzy model contains a set of fuzzy subspaces that forms a fuzzy decomposition (also called *fuzzy partition*) of the input space. The result of fusing multiple local models is usually a *fuzzy conclusion*, which is often converted to a *final crisp output* through a *defuzzification* process. The four major concepts in fuzzy rule-based models are:

1. fuzzy partition
2. mapping of fuzzy sub-regions to local models
3. fusion of multiple local models
4. defuzzification

There are three types of fuzzy rule-based models for function approximation: (a) the Mamdani model [17], (b) the Takagi-Sugeno-Kang (TSK) model [18, 19], and (c) Kosko's addictive model (SAM) [20]. The inference scheme of SAM is similar to that of TSK model. Both of them use an inference analogous to the weighted sum to aggregate the conclusion of multiple rules into a final

conclusion. Therefore, they are referred to as *additive rule models*. In contrast, the Mamdani model combines inference results of rules using superimposition; hence, it is a *non-additive rule model*. All three models are described in this section.

2.1.3.1 The Mamdani Model

One of the most widely used fuzzy models in practice is the *Mamdani model* [17], which consists of the linguistic rules in the form of

$$R_k : IF\ x_1\ is\ A_{1,j}^{(k)}\ and\ ...\ and\ x_{N_{IP}}\ is\ A_{N_{IP},j}^{(k)}\ THEN\ y^{(k)}\ is\ C_l^{(k)} \tag{1}$$

Where,

N_{IP} is the total number of input;

x_i is the i th input;

$A_{i,j}^{(k)}$ is the j th fuzzy sets of the i th input that is connected to R_k;

$y^{(k)}$ is the output that is connected to R_k;

$C_l^{(k)}$ is the l th fuzzy set for the output that is connected to R_k.

Given inputs of the form:

$$x_1\ is\ A_1^{'}, x_2\ is\ A_2^{'}, \ldots, x_i\ is\ A_{N_{IP}}^{'}$$

The contribution of rule R_k to a Mamdani model's output is a fuzzy set whose membership function is computed by

$$\mu_{C_l^{'}}(y) = \left(\alpha_{1,j}^{(k)} \wedge \alpha_{2,j}^{(k)} \wedge \ldots \wedge \alpha_{N_{IP},j}^{(k)}\right) \wedge \mu_{C_l^{(k)}}(y) \tag{2}$$

$$\alpha_{i,j}^{(k)} = \sup_{x_i}\left(\mu_{A_i^{'}}(x_i) \wedge \mu_{A_{i,j}^{(k)}}(x_i)\right) \tag{3}$$

Where,

$\alpha_{i,j}^{(k)}$ is the matching degree between the input x_i and the fuzzy set

$A_{i,j}^{(k)}$.

and $\wedge$ denotes the "min" operator. The final output of the model is the aggregation of outputs from all N_R rules using the max operator:

$$\mu_C(y) = \max\left\{\mu_{C_1'}(y), \mu_{C_2'}(y), ..., \mu_{C_{N_R}'}(y)\right\} \quad (4)$$

2.1.3.2 The TSK Model

The Takagi-Sugeno-Kang (TSK) model was introduced in 1984 [18, 19]. The main motivation of this model is to reduce the number of rules required by Mamdani model, especially for high-dimensional problems. It consists of rules in the form of

$$R_k : IF\ x_1\ is\ A_{1,j}^{(k)}\ and\ x_2\ is\ A_{2,j}^{(k)}, THEN\ y_s^{(k)} = b_0^{(k)} + b_1^{(k)}x_1 + b_2^{(k)}x_2 \quad (5)$$

Where,

$A_{i,j}^{(k)}$ is the j th fuzzy set of the i th input that is connected to R_k;

$y_s^{(k)}$ is the output of k th fuzzy if-then rule at the presentation of the s th sample;

$b_j^{(k)}$ is the j th data contents of the k th fuzzy if-then rule at the presentation of the s th sample. $b_j^{(k)}$ is real-valued parameter.

The inputs to a TSK model are crisp (nonfuzzy) numbers. Therefore, the degree of matching for s th data samples (in the form of $x_1 = x_1', x_2 = x_2', ..., x_{N_{IP}} = x_{N_{IP}}'$) that matches the k th rule is typically computed using the min operator:

$$\alpha_s^{(k)} = \min\left\{\mu_{A_{1,j}^{(k)}}\left(x_1'\right), \mu_{A_{2,j}^{(k)}}\left(x_2'\right), ..., \mu_{A_{N_{IP},j}^{(k)}}\left(x_{N_{IP}}'\right)\right\} \quad (6)$$

or using product operator:

$$\alpha_s^{(k)} = \mu_{A_{1,j}^{(k)}}\left(x_1'\right) \times \mu_{A_{2,j}^{(k)}}\left(x_2'\right) \times ... \times \mu_{A_{N_{IP},j}^{(k)}}\left(x_{N_{IP}}'\right) \quad (7)$$

The total output of the model y_s is given as

$$y_s^{(k)} = \frac{\sum_{k=1}^{N_R} \alpha_s^{(k)} f^{(k)}\left(x_1, x_2, \ldots x_{N_{IP}}\right)}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} = \frac{\sum_{k=1}^{N_R} \alpha_s^{(k)} \left[b_0^{(k)} + b_1^{(k)} x_1 + b_2^{(1)} x_2\right]}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \quad (8)$$

Where,

N_R is the total number of rules;

$\alpha_s^{(k)}$ is the degree of matching between the k th fuzzy if-then rule and the s th sample.

2.1.3.3 Standard Additive Model

The Standard Additive Model (SAM) was introduced by B. Kosko in 1996 [20]. The structure of fuzzy rules in SAM is identical to that of the Mamdani model. In the Mamdani model, inference results of rules are combined using superimposition and they are non-additive. In contrast, the SAM model uses an inference scheme that is analogous to the weighted sum to aggregate the conclusion of multiple rules into one final conclusion. As a result, the SAM model is also referred to as the additive rule model. The rules are in the form of

$$R_k : IF\ x_1\ is\ A_{1,j}^{(k)}\ and\ \ldots and\ x_{N_{IP}}\ is\ A_{N_{IP},j}^{(k)}\ THEN\ y^{(k)}\ is\ C_l^{(k)} \quad (9)$$

Where,

N_{IP} is the total number of input;

x_i is the i th input;

$A_{i,j}^{(k)}$ is the j th fuzzy sets of the i th input that is connected to R_k;

$y^{(k)}$ is the output that is connected to R_k;

$C_l^{(k)}$ is the l th fuzzy set for the output that is connected to R_k.

Given crisp inputs $x_1 = x_1^{'}, x_2 = x_2^{'}, \ldots, x_{N_{IP}} = x_{N_{IP}}^{'}$, the output of the model is

$$y = Centoid\left(\sum_k \mu_{A_{1,j}^{(k)}}\left(x_1^{'}\right) \times \mu_{A_{2,j}^{(k)}}\left(x_2^{'}\right) \times \ldots \times \mu_{A_{N_{IP},j}^{(k)}}\left(x_{N_{IP}}^{'}\right) \times \mu_{C_l^{(k)}}\left(y^{(k)}\right) \right) \qquad (10)$$

Where centroid is the function that performs the centroid defuzzification. A comparison in terms of input types, operator and defuzification method between the three different models are shown in Table 2-1.

Table 2-1: Comparison between three types of rule-based model

	Mamdani Model	TSK Model	SAM Model
Inputs	Both crisp or fuzzy inputs	Crisp input	Crisp input
Operator used to combine the conclusions of fuzzy rules	Max	Addition	Addition
Defuzzification	Any	Not applicable	Centroid (center of area)

2.2. Artificial Neural Networks

The original inspiration for Artificial Neural Networks (ANN) [6, 7] was the desire to produce artificial systems having some means of intelligence that are computationally similar to those that the human brain routinely performs. An ANN composed of many simple processing elements operating in parallel, each possibly having a small amount of local memory. Elements are connected through communication channels (or connections) that usually carry numeric data (also called weights) as opposed to symbolic data. Each of these operates only on its local data and on the inputs it receives via the connections. The function of ANNs is then determined by the network structure, connection strengths, and the operations performed at computing elements or nodes [7]. Most ANNs have some sort of training rule whereby the weights of connections are adjusted so that they can learn from examples. In other words, ANNs are able to acquire, store and utilize experiential knowledge so that they can exhibit some capability for generalization beyond the training data [6]. Since ANNs learn the way human do, they can perform many tasks that rule-based systems cannot easily perform. General areas of application include classification, prediction, function approximation, optimization, control, knowledge acquisition, and decision

making. In this section, the discussion will focus on two types of ANNs: 1) multilayer perceptron network (MLP) and, 2) associative memory.

2.2.1. Multilayer Perceptron Network (MLP)

The multilayer perceptron network (also known as multilayer feedforward network) is the most popular connectionist model that has been successfully applied to resolve many complex real-world problems such as those consisting of non-linear decision boundaries. It is trained in a supervised manner using the error back-propagation algorithm based on the error correction learning rule [21]. It may be viewed as a generalization of an adaptive filtering algorithm: the delta learning rule [22, 23]. Hornik [24, 25] has proven that a three-layer multilayer feedforward network with one hidden layer can approximate virtually any function to any desired degree of accuracy provided sufficient hidden neurons are available. However, a well-known problem of MLP is that the back-propagation algorithm may converge to local minima. In addition, the structure of a MLP network is predefined before training samples are presented and hence, non-adaptive to changing environment. The internal operations of a multilayer feedforward neural network cannot be explained and are often referred to as black box operations.

Figure 2-4 illustrates an example of a multilayer feedforward neural network. It consists of three layer: input layer, hidden layer and output layer. For a given input-output pair $\left(\mathbf{x}_s, \mathbf{y}_{desired,s}\right)$, the back-propagation algorithm performs two steps of data processing. The input pattern $\mathbf{x}_s$ is first propagated in forward direction from the input layer to the output layer and network output $\mathbf{y}_s$ is generated. In the second step, the error signals computed from the difference between the desired network output $\mathbf{y}_{desired,s}$ and actual network output $\mathbf{y}_s$ are back-propagated from the output layer to the input layer to update their weights.

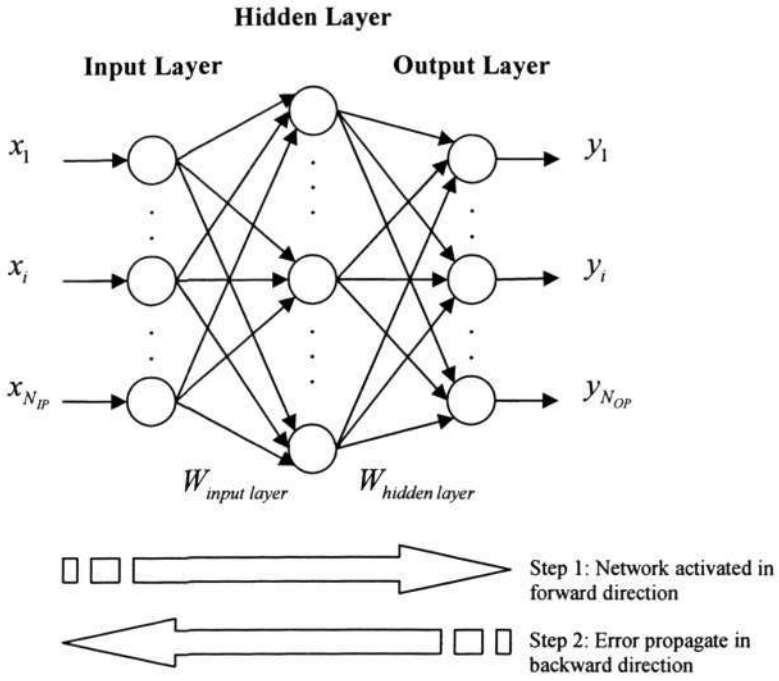


Figure 2-4: A Feedforward Multilayer Perceptron Network

2.2.2. Associative Memory

An associative memory can store a set of patterns as memories. When the associative memory is presented with a key pattern, it responds by producing whichever one of the stored patterns most closely resembles or relates to the key pattern. Hence, the recall is through association of the key pattern with the information memorized [1]. This is in contrast to the traditional *address-addressable memory* in digital computers where a pattern is recalled by its address. Therefore, associative memory is also referred to as *content-addressable memory*. Some important examples of associative memory are Hopfield network [26, 27], Boltzmann machine [28], bidirectional associative memory [29, 30] and cerebellar model articulation controller (CMAC) [2]. In particular, detailed descriptions on CMAC are presented in the following section.

2.3. Cerebellar Model Articulation Controller (CMAC)

2.3.1. Introduction

The *cerebellar model articulation controller* (CMAC), an associative memory proposed by Albus, is capable of performing localized generalization with very fast learning [2, 31, 32]. In one of Albus' papers, he described the training of CMAC as: "*CMAC, like the cerebellum in the brain after which it was modeled, does not operate by mathematically analyzing the dynamics of the*

control problem and then solving equations. Instead it operates by doing something, observing the results of the action, and then adjusting internal parameters in a direction calculated to improve the correspondence between what was called for and what actually occurred" [33]. CMAC estimates the desired output by taking the input states as an index to refer to a look-up table where the memory contents are addressed and stored. The ability of CMAC to approximate functions can be described in general terms as "*functions with similar outputs for similar inputs*" [34] or "*input vectors that are close in the input space will give outputs that are close*" [35]. It is a powerful and practical tool for nonlinear control that can be easily implemented. It has been found to be applicable in function approximation, pattern recognition and robotic control problem domains.

Conventional CMAC is a single input, single output neural network that assumes the integer valued vector inputs to be bounded within some range and these input vectors are associatively mapped to obtain the output. The basic idea behind the CMAC approach is to generate an approximation of the desired output. Due to its simple structure, the major advantages of the CMAC networks are the ability to generalize with good learning behavior. The serious local minimum problem that exists in multilayer neural network learning does not arise. In addition, it requires a small number of computations per output as compared to other traditional neural networks. The properties of CMAC include:

- Local generalization (Localized Learning)
- Incremental training (Online Training)
- Functional representation
- Output superposition
- Rapid algorithmic computation based on LMS training
- Fast practical hardware realization

Figure 2-5 shows the conceptual system structure of a CMAC network. The input space S is the set of all possible input vectors. Each point (vector) in S is mapped into a set of N_M points in

the conceptual memory M'. The mapping is done in such a way that input vectors that are close together in the space S have considerable overlap in their $N_{M'}$ memory locations, and vectors that are far apart have no overlap. The measure of distance in the S space is the sum of the absolute values of the differences of components of the two vectors. Memory M' is referred to as conceptual because it is normally too large to be practical. For example, if the input vector has 10 dimensions, each input dimension with 200 possible values, the size of M' is 200^{10}.

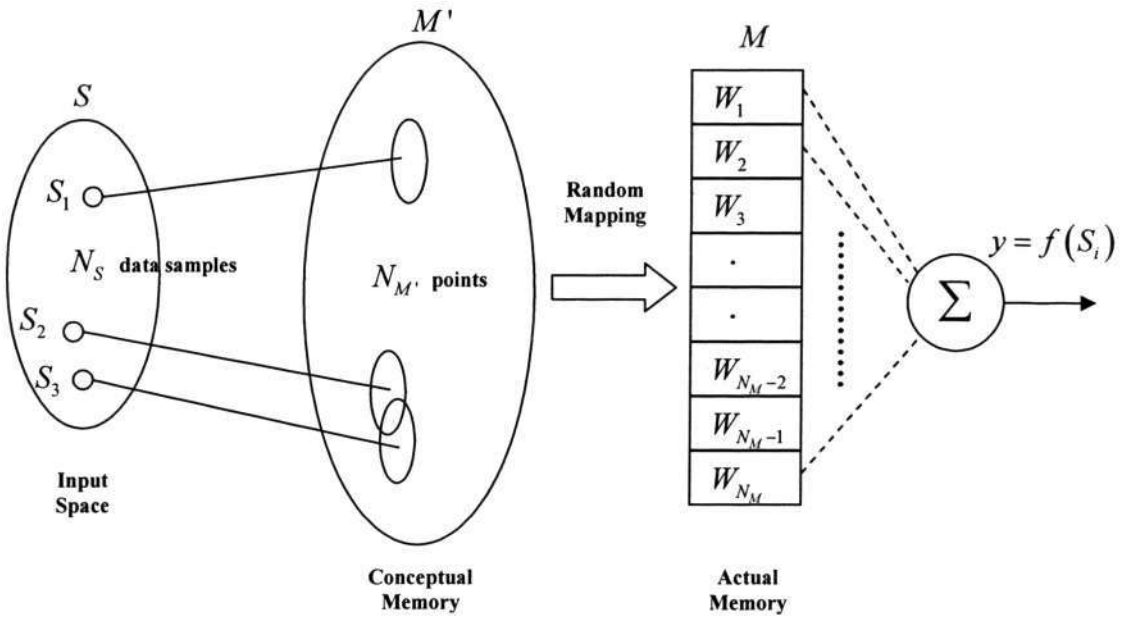


Figure 2-5: System Structure of CMAC Network

In order to obtain a reasonable size memory, the addresses for memory A are *hash coded* using a uniform random mapping scheme. The hash coding maps addresses in memory M' into memory M of size N_M that is chosen as convenient for a particular application. There is a finite probability that multiple entries in memory M' are mapped to the same location in memory M. This phenomenon is called *collision*.

The input-output mapping of CMAC can be divided into three phases: $S \rightarrow M' \rightarrow M \rightarrow R$. S is the input space. M' is the conceptual memory. M is the actual memory and R is the output space. The three phases are described as follow:

- *Phase 1:* $S \rightarrow M'$

The data sample $S_i = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{iN_{IP}} \end{bmatrix}^T$ is first quantized to $C_i = \begin{bmatrix} c_{i1} & c_{i2} & \cdots & c_{iN_{IP}} \end{bmatrix}^T$ with Eq (11).

$$c_{ij} = round\left[\left(\frac{x_{ij}}{x_{\max,j}}\right)\left(L_j - 1\right)\right] + 1 \qquad (11)$$

Where,

N_S	is the number of data samples;
N_{IP}	is the number of input dimension;
S_i	is the i th data sample;
x_{ij}	is the j th input of the i th data sample;
C_i	is the quantized vector of the i th data sample;
c_{ij}	is the quantized value for the j th input of the i th data sample;
$round(.)$	is the function that returns the rounded value;
$x_{\max,j}$	is the maximum value for the j th input;
L_j	is the quantization size of the j th input.

- *Phase 2:* $M' \rightarrow M$

The quantized vector $\mathbf{C} = \begin{bmatrix} C_1 & C_2 & \cdots & C_{N_S} \end{bmatrix}^T$ is mapped randomly to the actual memory $A = \begin{bmatrix} A_1 & A_2 & \cdots & A_{N_S} \end{bmatrix}^T$.

- *Phase 3:* $M \rightarrow R$

Final output $y_i \in R$ of CMAC for a given data sample S_i is

$$y_i = f(S_i) = A_i^T W_i \qquad (12)$$

Where,

$W_i = \begin{bmatrix} w_{i1} & w_{i2} & \cdots & w_{iN_{IP}} \end{bmatrix}^T$ are the weights stored in the i th memory location of the actual memory.

The weights are updated using *least mean square* (LMS) training rule from Widrow and Hoff [22, 23].

$$w_{ij}(t) = w_{ij}(t-1) + \lambda\left[y_{desired,i} - y_i(t)\right] \tag{13}$$

Where,

$w_{ij}(t)$ is the j th input of the i th data sample at the t th iteration;

λ is the learning parameter and $\lambda \in [0,1]$;

$y_{desired,i}$ is the desired output for y_i;

$y_i(t)$ is the output for the i th data sample at the t th iteration.

2.3.2. Advancement of CMAC

Researchers have been active in the advancement of CMAC since Albus [2, 31] first proposed it. Some comparative studies between CMAC and existing models have been performed by Kraft and Campagna [36] and Kambhampati et al. [37]. Some of the current research involves quantization [38-43], hash coding [2, 44, 45], learning convergence [46-49], stability [50-52], learning schemes [53-56], fuzzy CMAC [57-64], new architecture [65-71], application [72-77] and hardware realization [78-81]. Quantization problem and the fuzzification of CMAC network are two major issues considered in our studies. Detailed discussions are given in sections 2.3.2.1 and 2.3.2.2.

2.3.2.1 Quantization Problem

One of the important issues with the CMAC networks is the resolution issue in the input space. The easiest way to resolve this problem is to increase the resolution. However, significant amount of memory is required to accomplish this. For example, in a single layer CMAC (a two-

dimensional CMAC), if the resolution is increased by x times, the memory required will be changed by a factor of x^2 times. To reduce the required memory and make CMAC practical to implement, hash coding is used. There is a probability that distant input vectors are mapped to the same location by the use of hash coding. This effect, referred to as collisions, will produce undesirable generalization between distant input vectors. In 1996, Wang [44] has shown experimental results to conclude that CMAC will lose, and not gain performance by using hash coding. Nevertheless, the mapping collisions are generally ignored, since the probability can be kept small without difficulty. The analysis of collision probability is also discussed in by Albus in [2].

Similar to ANNs, the CMAC-based neural network is constructed with many processing elements in a structured approach. The selection of the CMAC structure is hence important and such decision determines how the CMAC input space is quantized and how the memory is utilized. Therefore, improvement on quantization techniques provides an alternative solution to simply increasing the resolution. The conventional approach, *uniform quantization*, has the input space quantized into equal-size regions. Due to possible uneven degrees of variation of the target function in the problem space, data are not stored efficiently using a uniformly quantized structure. Moody [38] proposed the use of multiple CMACs with different resolution to resolve the quantization problem. Learning starts with CMAC of a low resolution. If the results are unsatisfactory (with a large error value), a CMAC with higher resolution is added. Output of the structure is the summation of outputs from all networks. This expansion process continues until the error is reduced to an acceptable level. For every CMAC added, the memory requirement increases according to the resolution of the additional CMAC. Moody's method ensures a minimum resolution is achieved with acceptable error. However it is still considered as a uniform quantization and the input space is still partitioned equally as shown in Figure 2-6(a).

To resolve the problem in uniform quantization, Kim and Lin [39] investigated an adaptive technique for input space quantization though the use of a mapping function. Learning starts with equal-size quantization, quantization intervals for areas with larger variations are compressed and those with small variations are extended. This improves the accuracy of learned information and

reduces the required memory space. A comparison with conventional CMAC on function approximation and application to control of the *cart-pole balancing* (inverted pendulum) problem are given in their paper. Figure 2-6(b) shows the uneven input quantization in their method. However, the adaptive changing of input quantization requires the use of a *mapping function* as shown in Figure 2-7. The input is converted through the mapping function before it is presented to the CMAC.

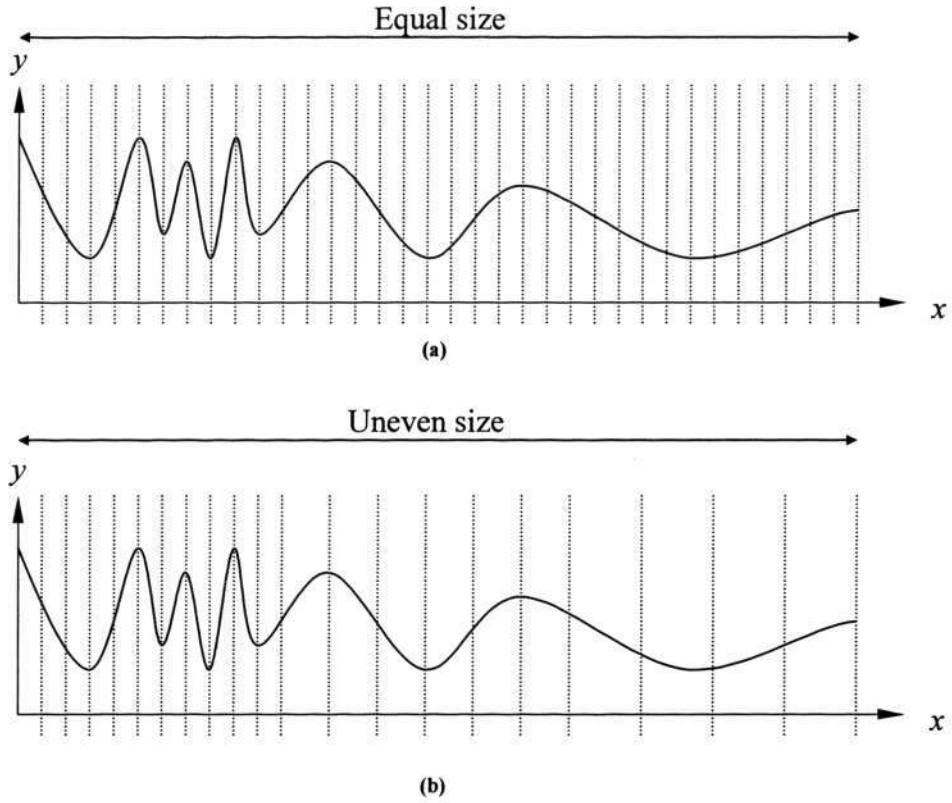


Figure 2-6: (a) Uniform Quantization (Equal Size) and (b) Adaptive Quantization (Uneven Size)

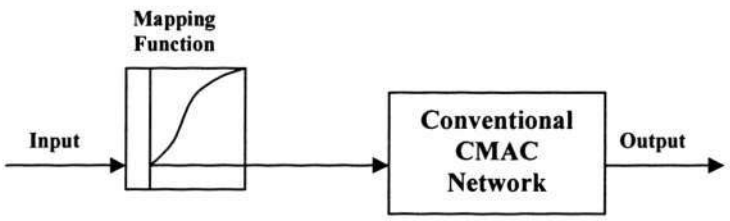


Figure 2-7: Integration of the Mapping Function with Conventional CMAC

The work proposed by González-Serrano et al. [40] modifies the basis functions of the CMAC network which transforms the CMAC input space from *hyper-cubes* to *hyper-parallelepipeds*. It

considers different degrees of generalization for each input. An *adaptive growing method* of the network is also presented. In their paper, it is stated that the requantization suggested by Kim and Lin [39] achieves poor results since the smoothness of the function is unrelated with the variability of the considered input. Kołcz and Allinson [41] have also created the *basis-function models* of CMAC mapping. However, emphasis was placed on the case of uniform quantization. The performance was compared using the benchmark problem of Mackey-Glass chaotic time-series prediction. Shu [43] proposed FMCMAC that handle the quantization problem with a different approach. In FMCMAC, fuzzy quantization is employed in place of linear quantization. The theory is stated as:

"The input space is divided into several regions, and the quantization ratio in that region is proportional to the number of fuzzy sets that cover the region."

The memory space structure for a two-dimensional CMAC (with uniform quantization) and FMCMAC (with fuzzy quantization) are shown in Figure 2-8. However, fuzzy clusters are required prior to the formation of the FMCMAC's structure. The online adaptive ability of the CMAC network is partially impaired by such technique and the quantization problem in CMAC structure remains unsolved.

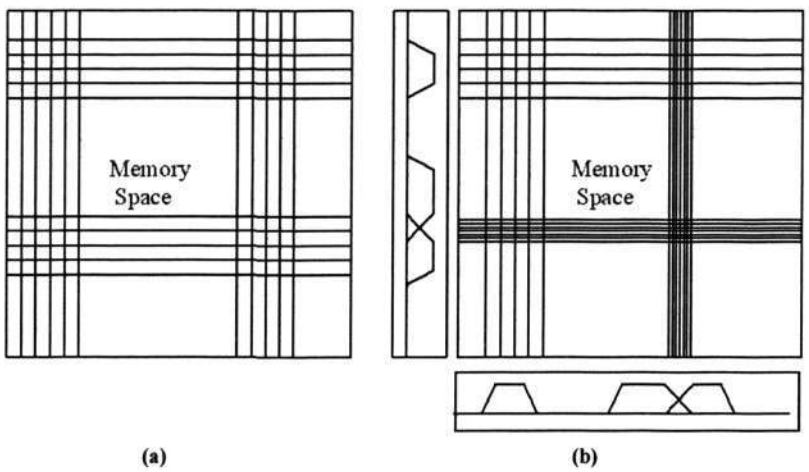


Figure 2-8: Memory Quantization in (a) CMAC (with Uniform Quantization) and (b) FMCMAC (with Fuzzy Quantization)

2.3.2.2 Fuzzy CMAC

Many research efforts attempt to incorporate fuzzy logic into the CMAC network. By doing so, CMAC network is transformed into a white box whereby fuzzy rules could be generated to

interpret the operation of the network. The modified fuzzy CMAC can then be implemented to solve many real world problems with greater semantics and ease. Several existing approaches are described in this section.

In 1992, Jou [60] presented a FCMAC that does not require human operator to identify the fuzzy logical rule. Although *centroid defuzzification* method is employed, no rule-based model and membership function is specified. Jou also argued that the FCMAC could be used as a model-free function estimator. Although comparison between the proposed FMCMAC and CMAC has been made, only conceptual work is proposed and no experimental result is available.

In the same year, Ozawa et al. [57] proposed a more concrete implementation of fuzzy CMAC called *CMAC-fuzzy system* and applied the proposed system to a simulation game of catching the moving object. The proposed fuzzy CMAC employs *triangular membership function* and uses *zero-ordered TSK rule-based model*. The intention of the system is assumed to consist of a global intention and a local intention. The global intention works in the complete input space, and the local intention works in the local input space. The final output of CMAC-fuzzy system is the sum from both intentions. The fuzzy inference rules are constructed from input-output data acquired by communication with the human operator. As a result, the CMAC structure is predefined before the learning phase and, therefore, not adaptive. A two-phase learning scheme is employed. Learning starts by updating the consequent part of the global intention. In the second phase, the consequent of the global intention are fixed while the learning of the local intention starts. The learning of the global intention in the first phase is described as

$$w^{(k).new} = w^{(k).old} + g_{global} w^{(k).old} \left(y_i - y_i^{(k)(global)} \right) \qquad (14)$$

$$\text{for } k = 1, 2, ..., M \; ; \; 0 < g_{global} < 1$$

Where,

g_{global} is the learning factor for global intention;

$y_i^{(k)(global)}$ is the output of the k th layer to the input data x_i for global

intention;

$w^{(k).old}$ is the old weight of k th layer;

$w^{(k).new}$ is the updated weight of k th layer;

M is the total number of layer.

The learning of the consequent part will stop when the condition in Eq (15) is satisfied.

$$Error^{(k)} = \sum_{i=1}^{S} \left(y_i - y_i^{(k)(global)} \right)^2 < \varepsilon \quad \text{for } k = 1, 2, ..., M \tag{15}$$

Where,

ε is the tolerance for the square error;

S is the total number of input data.

After the first phase, the fuzzy inference rules expressing the global intention are fixed. The update of local intention is then performed as Eq (16) in the second phase.

$$w^{(k).new} = w^{(k).old} + g_{Local} w^{(k).old} \frac{1}{M-1} \left(y_i - y_i^{(k)(global)} - y_i^{(k)(local)} \right) \quad \text{for } k = 1, 2, ..., M;\ k \neq M;\ 0 < g_{Local} < 1 \tag{16}$$

Where,

g_{Local} is the learning factor for local intention;

$y_i^{(k)(global)}$ is the global output of the K th layer to the input data x_i;

$y_i^{(k)(local)}$ is the local output of the K th layer to the input data x_i.

The learning of the second phase will stop when

$$Error^{(k)} = \sum_{i=1}^{S} \left(y_i - y_i^{(k)(local)} \right)^2 < \varepsilon_{local} \tag{17}$$

$$\text{for } k = 1, 2, ..., M$$

Where,

ε_{local} is the tolerance of error for local intention;

S is the total number of input data;

The final output of the CMAC-fuzzy system for the input is derived by

$$y_i^{(total)} = y_i^{(global)} + y_i^{(local)} \qquad (18)$$

$$\text{for } i = 1, 2, ..., S$$

Where,

$y_i^{(total)}$ is the total output of the system to input data x_i;

$y_i^{(global)}$ is the global output to input data x_i;

$y_i^{(local)}$ is the local output to input data x_i.

In 1992, Lane et al. [62] proposed a higher-order CMAC using *B-spline receptive field functions* (BCMAC). The proposed network is a multi-layer CMAC network architecture trained with standard error back-propagation learning technique. By employing B-spline receptive field function, the method of updating weights distributes errors among the assigned weights according to the intensity of the B-spline functions. However, B-spline function is known for their complexity in computation and the multi-layer architecture is expected to slow down the network. In their paper, only theoretical works are derived and no application or results are provided.

Nie and Linkens [58] proposed a fuzzified CMAC controller (FCMAC) that is *self-learning*, *self-organizing* in real-time control. They have successfully applied the FCMAC into blood pressure control. FCMAC employs *triangular membership function* and the *Mamdani rule-based model*. The input space is partitioned using the *Kohonen* self-organizing scheme. Although they claimed that FCMAC is self-learning, but a teacher signal v_k^* (the desired output) is still required. Hence, it is still considered as a form of supervised learning. The update rule is given as:

$$\Delta\pi_k^j = \frac{\beta\left[\left(v_k^{\bullet} - v_k(u)\right)a^j\right]}{\sum_{j=1}^{N} a^j} \tag{19}$$

Where,

$v_k^{\bullet}$ is the k th desired output;

u is the input;

$v_k(u)$ is the k th output;

β is the learning rate;

a^j is the matching degree between the j th rule and input u.

In 1997, Ker et al. [63] proposed to use a fuzzy CMAC model (FCMAC) for color reproduction. In their model, recursive *B-spline receptive field functions* employed in [62] were replaced by fuzzy sets with *bell-shaped membership function* and the output weights were fuzzy variables. They use *Mamdani* inference scheme with *center of area* (COA) defuzzification method. In the supervised learning process, the fuzzy CMAC uses *maximum gradient method* to decide the adjustment amount of the mean value of each fuzzy weight while the variance of the bell-shaped membership function remains unchanged. Three different models: conventional CMAC, BMAC [62] and the proposed model were applied to the approximation of sinusoid functions and the results show that the proposed model performs better with simple computation and higher stability. However, in their model, uniform quantization is adopted and the architecture is predefined and not expandable.

Kai and Feng [59] proposed another fuzzy CMAC (FCMAC, same acronym as the fuzzy CMAC proposed by Ker [63]). The mapping algorithm and learning algorithm are also presented. They have shown results for applications on function approximation and prediction. The structure is same as conventional CMAC except that the weights are fuzzified by a *Gaussian function*. Therefore, the structure remains non-adaptive. The gradient decent learning process involves updating the center and variance of the Gaussian function and a real number weights.

In 2002, Kim [61] proposed a CMAC-based fuzzy logic controller (CBFLC) and the architecture is similar to that proposed by Jou [60]. CBFLC uses *bell-shaped membership function* for both the fuzzy sensors and the weights stored in the physical memory. Instead of single layer fuzzy sensors, Kim proposed multiple sensor layers to reduce the quantization problem and to improve the approximation ability of the CMAC. A modified version of *center of gravity* (COG) defuzzification is adopted in CBFLC. The conventional backpropagation learning has also been modified to learn the center and width of the membership functions. CBFLC has been applied to the truck backer-upper control problem to back a truck to a loading dock as quickly and precisely as possible.

In 2006, Nguyen et al. [64] proposed a fuzzy CMAC (FCMAC-BYY) using Bayesian Ying-Yang (BYY) learning [82] to determine the optimal fuzzy sets with truth-value restriction inference scheme. The BYY is motivated from the famous Chinese ancient Ying-Yang philosophy: everything in the universe can be viewed as a product of a constant conflict between opposites—Ying and Yang. A perfect status is reached when Ying and Yang achieve harmony [64]. The parameters in FCMAC-BYY are estimated by maximum likelihood (ML) learning with the expectation-maximization (EM) algorithm.

In 2008, Yu et al. [71] proposes two type of hierarchical FCMACs (HFCMACs) to overcome the memory quantization problem. In addition, a stability analysis on the backpropagation-like learning algorithms has been presented. Although Yu claimed that "generally, if the quantization size is larger, the proposed HFCMAC gives less rules than standard fuzzy systems", no theoretical proof is given to support this statement. In addition, due to the backpropagation-like learning algorithm, the proposed FCMACs are not incremental and cannot perform online training.

A comparison among the fuzzified CMACs discussed above is shown in Table 2-2. As a summary, although conventional CMAC has been successfully transformed into a white box whereby fuzzy rules can now be extracted, the structure for most of the existing fuzzy CMACs remains non-adaptive. They employ uniform quantization in the input space and the CMAC structure is predefined before the learning of the consequent. This observation draws the attention to the needs of developing an adaptive (expandable) online fuzzy CMAC.

Table 2-2: Comparison among Fuzzy CMACs

Model Name [Author, year of publication] [references]	Receptive Field Function	Quantization	Rule-Based Model	Weight Stored	Update Rule	Defuzzification Method	Expand-able	Remarks
Fuzzy CMAC (FCMAC) [Jou, 1992] [60]	Not Specified	NM	NP	NP	Direct error update on weight	Centroid	No	conceptual work only
CMAC-fuzzy [Ozawa et al., 1992] [57]	Triangular membership function	Uniform quantization	TSK	Real number	Direct error update on weight	NA	No	-
Higher-order CMAC (BMACs) [Lane et al., 1992] [62]	B-spline function	Uniform quantization	NM	Real number	Direct error update on weight	NA	No	Multiplayer BMACs
Fuzzified CMAC (FCMAC) [Nie and Linkens, 1993] [58]	Compute matching degree by neighborhood (Euclidean, Hamming, or Maximum) between input and rule pattern	NA	Mamdani	Triangular membership function (uses maximum membership decision scheme)	Direct error update on weight	Center of gravity (COG)	Yes	Self-learning, self-organizing in real time
Fuzzy CMAC (FCMAC) [Ker et al., 1997] [63]	Bell-shaped membership function	Uniform quantization	Mamdani	Bell-shaped membership function	Maximum gradient method to adjust only the mean value for bell-shaped membership function	Center of area (COA)	No	-
Fuzzy CMAC (FCMAC) [Kai and Feng, 2000] [59]	NA	Uniform quantization	NM	Gaussian function	Gradient Descent update for center and variance of the gaussian function	NA	No	-
CMAC-based fuzzy logic controller (CBFLC) [Kim, 2002] [61]	Bell-shaped membership function	Uniform quantization	NM	Bell-shaped membership function	Generalized delta rule and chain-rule of differentiations for center and width of the membership function	Center of gravity (COG)	No	Multiple sensor layers
Fuzzy CMAC (FCMAC-BYY) [Nguyen, 2006] [64]	Gaussian membership function	Bayesian Ying-Yang learning	TSK	Real Number	Parameters estimated with expectation-maximization (EM) algorithm	Center of area (COA)	No	-
Hierarchical Fuzzy CMAC (HFCMAC) [Yu, 2008] [71]	Not Specified	Not Specified	TSK	Real Number	Backpropagation-like learning algorithm	NA	No	Hierarchical structure

NP = no preference; NM = not mentioned; NA = not applicable; center of gravity = center of area = centroid.

2.4. Motivation for Fuzzy CMAC

The *cerebellar model articulation controller* (CMAC), an associative memory proposed by Albus [2, 31, 32] is a powerful and practical tool for nonlinear control that can be easily implemented. It has been found to be applicable in function approximation, pattern recognition and robotic control problem domains. The CMAC network performs localized learning where the serious local minimum problem that exists in multilayer neural network learning does not arise. The learning is incremental and is versatile against changing environment that requires online training capability. However, it suffered from several fundamental problems:

1. The structure of a conventional CMAC is predefined before training samples are presented and hence, not expandable.
2. The quantization problem which affects the structure of the network as well as the utilization of the memory.
3. The operation of the CMAC networks is a black box and it is difficult to interpret the internal operations of the networks.

In addition, human deals with vague concepts with unsharp boundaries while many practical applications involve knowledge that does not have a well-defined boundary. Hence, it becomes difficult to apply CMAC networks in these applications because CMAC only deal with integer value vector.

On the other hand, fuzzy inference systems provide an intuitive channel from a quantitative aspect to qualitative aspect and vice versa. The operations of a fuzzy inference system are perceived as white box operations where expert knowledge could be easily extracted. However, the criticism on the requirement of manual tuning of the system parameters has becoming a major shortcoming for existing fuzzy inference system.

Over the years, similarities between ANNs and fuzzy systems attracted significant attentions from researchers from both fields to explore means to combine their individual advantages. Investigations on literatures reveal various shortcomings and limitations faced by existing techniques. This motivates the synergy between cerebellar model articulation controller and fuzzy

systems and the investigation leads to the development of a neuro-fuzzy system which automatically identifies the parameters in a fuzzy system using neural network techniques.

Chapter 3 TSK^0-FCMAC: Architecture and Learning Algorithm

3.1. Introduction

This chapter presents the architecture and learning algorithm of TSK^0-FCMAC, a localized self-organizing zero-ordered Takagi-Sugeno-Kang fuzzy inference system [18, 19] based on the CMAC structure [2]. Most neuro-fuzzy inference systems employ complex function to represent the antecedent, such as bell-shaped membership function (Ker's FCMAC [63]) and Gaussian membership function (Kim's HyFIS [83] and Jang's ANFIS [84]) to generate smoother approximation. Instead of adopting these complex functions, the antecedents for TSK^0-FCMAC are represented in triangular or trapezoidal membership functions. Thus, accuracy is sacrificed for simplicity and computational efficiency. The consequent can be represented by crisp, fuzzy and functional consequents. Each type of rule consequent has its merits. Neuro-fuzzy inference systems such as GenSoFNN [85], Falcon-ART [86] and Falcon-MART [87] adopt fuzzy consequent to capture imprecise human expertise. ANFIS, another prominent neuro-fuzzy inference system employs functional consequent for accuracy using a small number of rules. Instead of the functional consequent, TSK^0-FCMAC employs crisp consequent (also called *fuzzy singleton*) for efficiency in computation. As a result, TSK^0-FCMAC is equivalent to zero-ordered TSK inference model.

The architecture of the proposed network is presented in section 3.2. This is followed by a detailed illustration of the learning algorithm in sections 3.3 and 3.4. Section 3.5 demonstrates the learning ability of the proposed architecture in addressing problems of three different categories. The application in these problems has the following goals:

1. Demonstrate the adaptive ability of TSK^0-FCMAC in rapid changing environment such as the inverted pendulum problem (in section 3.5.1).
2. Demonstrate the inherent ability of TSK^0-FCMAC to perform rule extractions in a well-known classification problem, the Fisher's Iris classification [88] (in section 3.5.2).

3. Demonstrate the ability of TSK^0-FCMAC in solving regression problem such as the Mackey-Glass time series prediction [89] whose primary objective is to achieve satisfactory precision.

3.2. Architecture

For simplicity of explanation, the structure under consideration in this section has two inputs x_1, x_2 and one output y. Consider the presentation of the s th sample in the t th iteration and assume that the fuzzy inference rule base contains two fuzzy if-then rules of Takagi-Sugeno-Kang's type [18, 19], R_1 and R_2 defined in Eqs (20) and (21):

$$R_1 : IF\ x_1\ is\ A_{1,j}^{(1)}\ and\ x_2\ is\ A_{2,j}^{(1)}, THEN\ y_s^{(1)}(t) = b_{0,s}^{(1)}(t) + b_{1,s}^{(1)}(t)x_1 + b_{2,s}^{(1)}(t)x_2 \qquad (20)$$

$$R_2 : IF\ x_1\ is\ A_{1,j}^{(2)}\ and\ x_2\ is\ A_{2,j}^{(2)}, THEN\ y_s^{(2)}(t) = b_{0,s}^{(2)}(t) + b_{1,s}^{(2)}(t)x_1 + b_{2,s}^{(2)}(t)x_2 \qquad (21)$$

Where,

$A_{i,j}^{(k)}$ is the j th fuzzy set of the i th input that is connected to R_k;

$y_s^{(k)}(t)$ is the output of k th fuzzy if-then rule at the presentation of the s th sample in the t th iteration;

$b_{j,s}^{(k)}(t)$ is the j th data contents of the k th fuzzy if-then rule at the presentation of the s th sample in the t th iteration. $b_{j,s}^{(k)}(t)$ is real-valued parameter.

$A_{1,j}^{(1)}$, $A_{2,j}^{(1)}$, $A_{1,j}^{(2)}$ and $A_{2,j}^{(2)}$ are fuzzy sets and $b_{0,s}^{(1)}(t)$, $b_{1,s}^{(1)}(t)$, $b_{2,s}^{(1)}(t)$, $b_{0,s}^{(2)}(t)$, $b_{1,s}^{(2)}(t)$ and $b_{2,s}^{(2)}(t)$ are real-valued parameter. The total output of the model $y_s(t)$ is given as

$$y_s(t) = \frac{\sum_{k=1}^{N_R} \alpha_s^{(k)} f^{(k)}(x_1, x_2)}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} = \frac{\alpha_s^{(1)}\left(b_{0,s}^{(1)}(t) + b_{1,s}^{(1)}(t)x_1 + b_{2,s}^{(1)}(t)x_2\right) + \alpha_s^{(2)}\left(b_{0,s}^{(2)}(t) + b_{1,s}^{(2)}(t)x_1 + b_{2,s}^{(2)}(t)x_2\right)}{\alpha_s^{(1)} + \alpha_s^{(2)}} \quad (22)$$

Where,

N_R is the total number of rule;

$\alpha_s^{(k)}$ is the degree of matching between the k th fuzzy if-then rule and the s th sample.

In this case, $N_R = 2$ and $\alpha_s^{(k)} = \min\left\{\mu_{A_{1,j}^{(k)}}(x_1^{'}), \mu_{A_{2,j}^{(k)}}(x_2^{'})\right\}$. The inputs to a TSK model are crisp (non-fuzzy) numbers. Therefore, the degree of input $x_1 = x_1^{'}, ..., x_i = x_i^{'}, ..., x_I = x_I^{'}$ that matches the k th rule is typically computed using the min operator:

$$\alpha_s^{(k)} = \min\left\{\mu_{A_{1,j}^{(k)}}(x_1^{'}), \mu_{A_{2,j}^{(k)}}(x_2^{'}), \cdots, \mu_{A_{I,j}^{(k)}}(x_I^{'})\right\} \quad (23)$$

or using product operator:

$$\alpha_s^{(k)} = \mu_{A_{1,j}^{(k)}}(x_1^{'}) \times \mu_{A_{2,j}^{(k)}}(x_2^{'}) \times \cdots \times \mu_{A_{I,j}^{(k)}}(x_I^{'}) \quad (24)$$

Figure 3-1 illustrates the computation process involved in a two-dimensional TSK inference model.

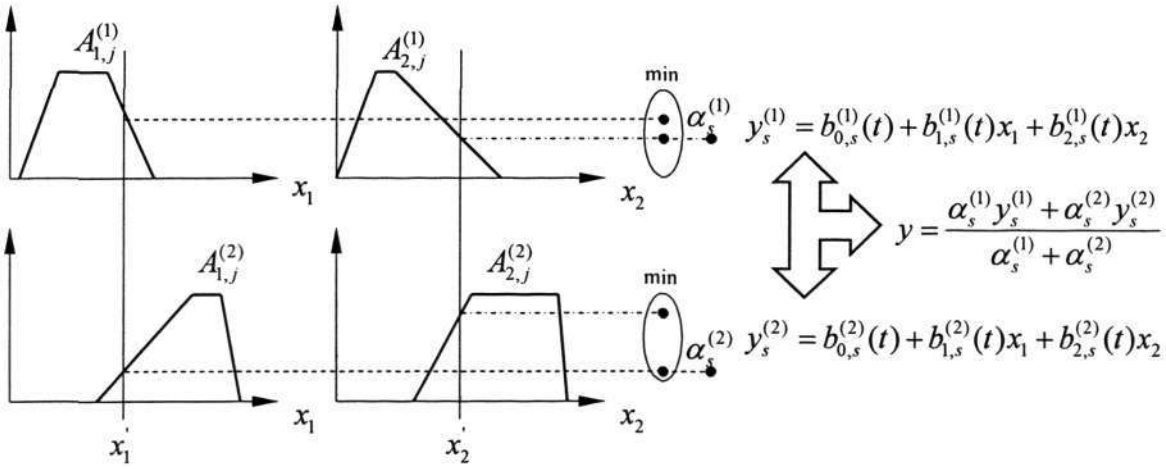


Figure 3-1: TSK Inference Model (two-dimensional)

In TSK^0-FCMAC, the real-valued parameters $b_{1,s}^{(1)}(t)$, $b_{2,s}^{(1)}(t)$, $b_{1,s}^{(2)}(t)$ and $b_{2,s}^{(2)}(t)$ are all set to zero. Figure 3-2 shows a 2-input TSK^0-FCMAC with 16 rules. Four membership functions are associated with each input, so the input space is partitioned into 16 fuzzy subspaces. Each fuzzy subspace is governed by a fuzzy if-then rule. The premise part of a rule defines a fuzzy subspace and the consequent part specifies the output within this fuzzy subspace. In this example, four fuzzy if-then rules are activated for inputs x_1' and x_2' according to their respective matching degree. They are:

$$\begin{aligned}
&R_{13}: IF\ x_1\ is\ A_{1,1}^{(13)}\ and\ x_2\ is\ A_{2,3}^{(13)}, THEN\ y_s^{(13)}(t) = b_{0,s}^{(13)}(t)\\
&R_{14}: IF\ x_1\ is\ A_{1,1}^{(14)}\ and\ x_2\ is\ A_{2,4}^{(14)}, THEN\ y_s^{(14)}(t) = b_{0,s}^{(14)}(t)\\
&R_{23}: IF\ x_1\ is\ A_{1,2}^{(23)}\ and\ x_2\ is\ A_{2,3}^{(23)}, THEN\ y_s^{(23)}(t) = b_{0,s}^{(23)}(t)\\
&R_{24}: IF\ x_1\ is\ A_{1,2}^{(24)}\ and\ x_2\ is\ A_{2,4}^{(24)}, THEN\ y_s^{(24)}(t) = b_{0,s}^{(24)}(t)
\end{aligned} \tag{25}$$

Hence, the final output of the network is:

$$y_s(t) = \frac{\alpha_s^{(13)} b_{0,s}^{(13)}(t) + \alpha_s^{(14)} b_{0,s}^{(14)}(t) + \alpha_s^{(23)} b_{0,s}^{(23)}(t) + \alpha_s^{(24)} b_{0,s}^{(24)}(t)}{\alpha_s^{(13)} + \alpha_s^{(14)} + \alpha_s^{(23)} + \alpha_s^{(24)}} \tag{26}$$

Where the matching degree, $\alpha_s^{(k)} = \min\left\{\mu_{A_{1,j}^{(k)}}(x_1'), \mu_{A_{2,j}^{(k)}}(x_2')\right\}$.

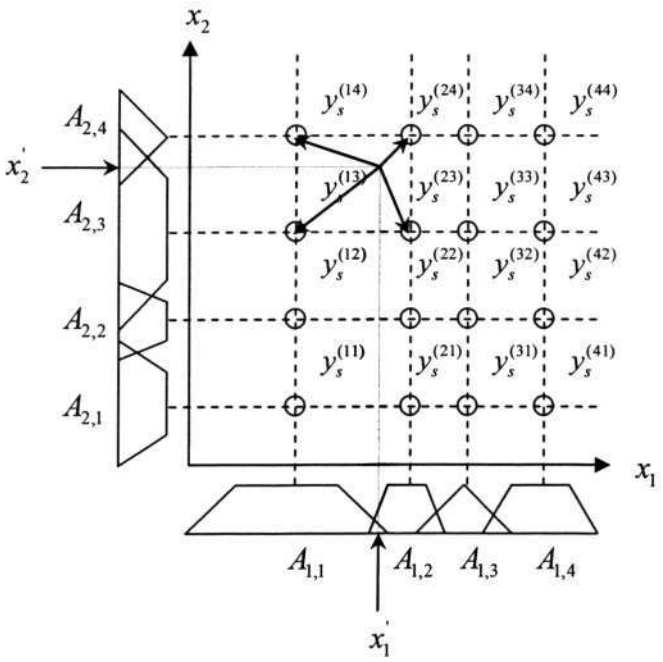


Figure 3-2: Structure of 2-input TSK^0-FCMAC with 16 rules

3.3. Two-Phase Learning Algorithm

TSK^0-FCMAC employs a two-phase training algorithm. During the first phase, the Discrete Incremental Clustering (DIC) technique [90] is performed on newly presented data point. Depending on the degree of matching between the data point and existing clusters, DIC will create a new cluster, expand existing cluster or do nothing. The first phase is illustrated in Figure 3-4. In the second phase, the same data point will activate the fuzzy if-then rules within TSK^0-FCMAC and an output is computed as shown in Eq (26). The crisp consequents, $b_{0,s}^{(k)}(t)$ for every activated fuzzy if-then rule are updated as shown in Eq (27):

$$b_{0,s}^{(k)}(t+1) = b_{0,s}^{(k)}(t) + \lambda \alpha_s^{(k)} \left[\hat{y}_s(t) - y_s(t) \right] \tag{27}$$

Where,

λ is the learning parameter;

$\alpha_s^{(k)}$ is the degree of matching between the k th fuzzy if-then rule and the s th sample;

$y_s(t)$ is the output of the network for the s th sample at the t th

iteration;

$\hat{y}_s(t)$ is the target output of the network for the s th sample at the t th iteration.

The overall learning procedure is depicted as Figure 3-3.

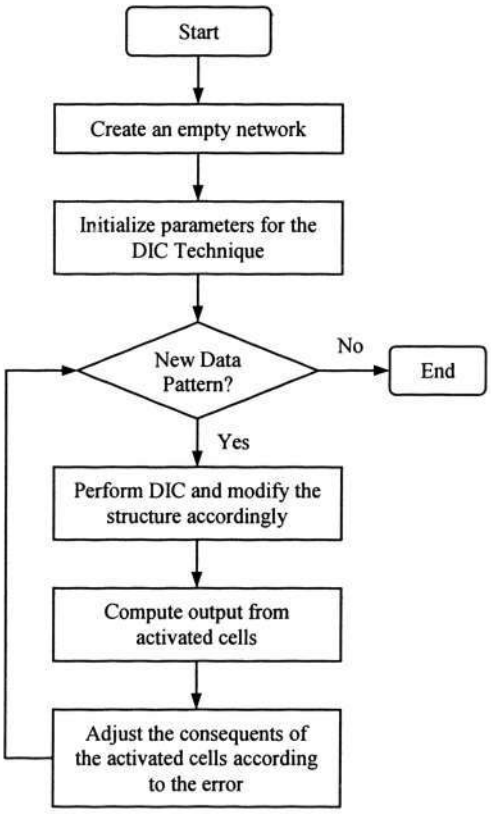


Figure 3-3: Overall learning procedure

The incremental learning process of the proposed architecture is shown in Figure 3-4. Initially, TSK^0-FCMAC starts with an empty structure shown in Figure 3-4(a). When a new data point arrives, TSK^0-FCMAC will create a new cluster to accommodate the new data point, expands an existing cluster to include the data point or do nothing on the existing structure (i.e. the data point falls into an existing cluster) according to the DIC technique, see Figure 3-4(b) and (c). The details of the DIC technique are described in the section 3.4.

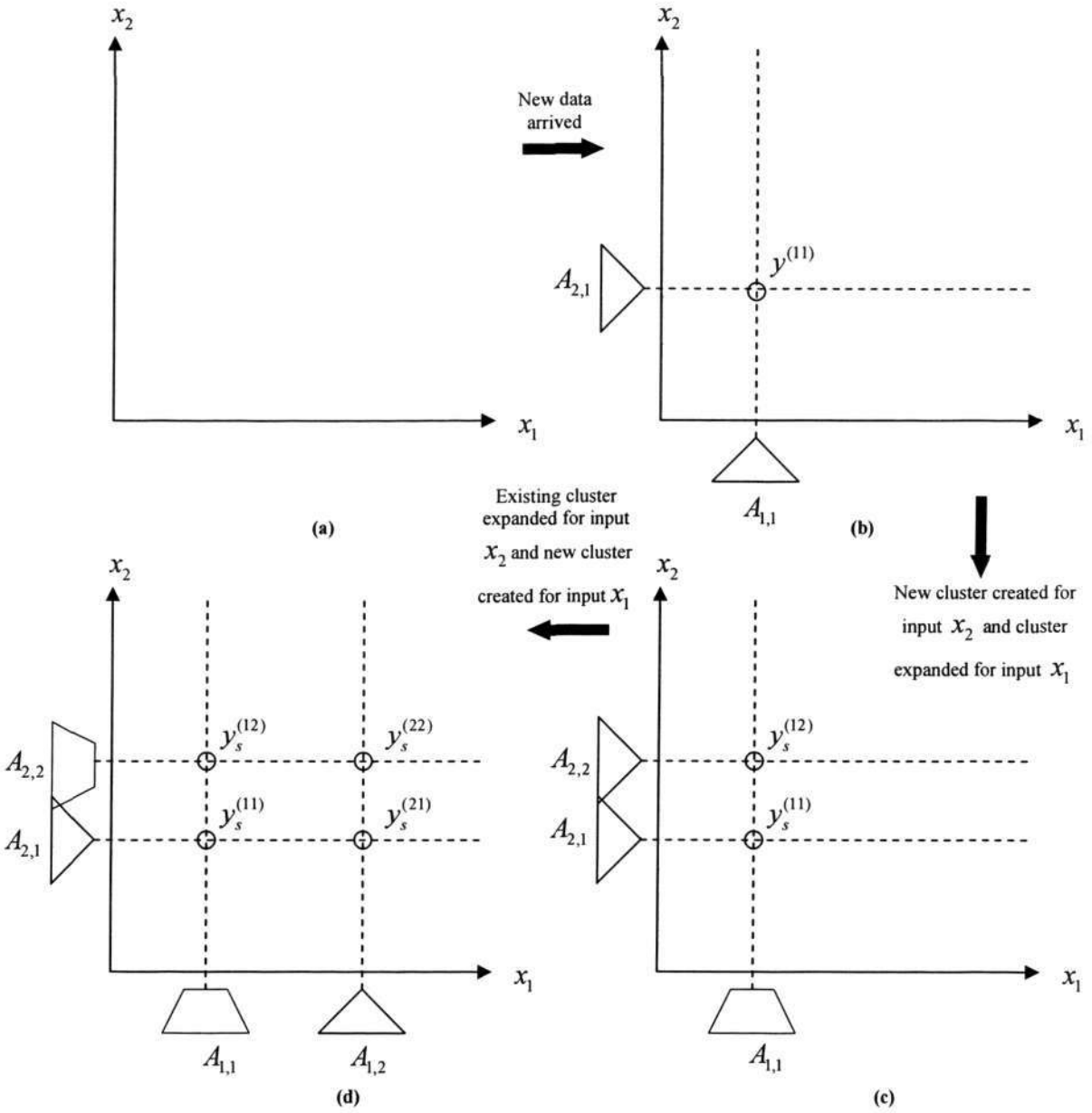


Figure 3-4: TSK^0-FCMAC structure at: (a) Empty; (b) After first data points; (c) After second data point; (d) After third data point

3.4. Discrete Incremental Clustering (DIC) Technique

Traditional clustering techniques suffer from the *stability-plasticity* dilemma [1] where new information cannot be learnt without running the risk of eroding previously learnt but valid knowledge. Therefore, neuro-fuzzy system such as POPFNN [91] violates the networks' ability to self-organize and self-adapt with changing environments. Another shortcoming of these clustering techniques is the requirement of prior knowledge such as the number of classes.

In TSK^0-FCMAC, the formation of input clusters (receptive field function) is governed by a novel Discrete Incremental Clustering (DIC) technique proposed by Tung and Quek [90]. The DIC

technique is not limited by the need to have prior knowledge of the number of clusters C and it preserves the dynamism to learn new knowledge. This novel clustering technique attempts to integrate the merits of fuzzy ART [92] and the LVQ [93] clustering techniques.

The DIC technique has five parameters: a plasticity parameter β, a tendency parameter TD, an input threshold IT, an output threshold OT and a fuzzy set support parameter $SLOPE$. A brief description of DIC is given in this section and full details can be found in [90].

3.4.1. The fuzzy set support parameter *SLOPE*

Each new cluster in DIC begins as a triangular fuzzy set as shown in Figure 3-5(a). The kernel of a new cluster (fuzzy set) takes the value of the data point (x') that triggers its creation and its support is defined by the parameter $SLOPE$ where

$$\begin{aligned} b &= x' \\ a &= x' - SLOPE \times \left(\max(x_i) - \min(x_i)\right) \\ c &= x' + SLOPE \times \left(\max(x_i) - \min(x_i)\right) \end{aligned} \tag{28}$$

As training continues, the cluster "grows" to include more points, but maintains the same amount of buffer regions on both sides of the kernel (Figure 3-5(b)). The trapezoidal fuzzy sets after expansion satisfied the following equations:

$$\begin{aligned} n - m &= b - a \\ p - o &= c - b \end{aligned} \tag{29}$$

A $SLOPE$ value of 0.5 means the distance between a and b (or the distance between b and c) in Figure 3-5(a) will cover fifty percent of the full range of input x. A large value of $SLOPE$ will result in highly overlapped clusters whilst a small value of $SLOPE$ will result in the formation of clearly separable clusters. If the value of $SLOPE$ is set to 0, the resultant clusters will be similar to the adaptive quantization illustrated in Figure 2-6(b).

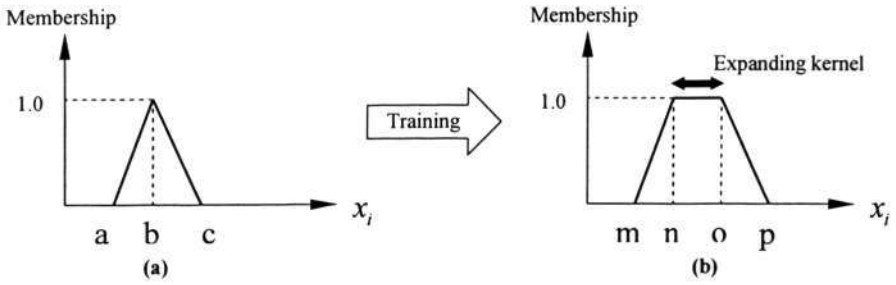


Figure 3-5: (a) A newly created cluster and (b) cluster after training

3.4.2. Plasticity parameter β

A cluster "grows" by expanding its kernel. This expansion is controlled by the plasticity parameter β. A cluster expands its kernel when it is the best-fit cluster (having the highest membership value) to a data point and this data point falls outside its kernel. The plasticity parameter governs how much a cluster (fuzzy set) expands its kernel to include the new data point. A β value of 1 will result in aggressive expansion of existing clusters. That is, the formation of huge clusters but fewer numbers of clusters. A β value of 0 means the cluster will stop expanding. Therefore, to avoid aggressive expansion, the initial value of β for all newly formed clusters is preset at 0.5.

The value of its β parameter decreases as the cluster expands its kernel. The first quadrant of a cosine waveform is used to model the change of β in a cluster. The parameter θ is intuitively interpreted as the maximum expansion a cluster can have and a parameter *STEP* controls the increment of θ from 0 to 1.57 radians (i.e. the resultant value of $\cos(\theta)$ will be in the range of 0 to 1). Hence, the amount of expansion a cluster can adopt decreases with the number of expansions. It is computed as follows:

$$Expansion = \beta \cos(\theta) \times Distance \tag{30}$$

Where *Distance* is the distance between nearest kernel (left kernel or right kernel) and current data point. After each expansion, θ is incremented by *STEP* and β is set to 0 whenever $\theta \geq 1.57$.

3.4.3. Tendency parameter TD

The tendency parameter TD is analogous to a cluster's willingness to "grow" when it is the best-fit cluster to a data point that falls outside its kernel. It complements the use of the plasticity parameter β . Parameter TD maintains the relevance of a cluster and prevents it from incorporating too many points that has low "fitness" or membership values to the cluster. Otherwise, the kernel of a cluster may become overly large and the semantic meaning of the fuzzy label that the cluster represents may be obscured and poorly defined. The initial value of TD of a newly created cluster is preset at 0.5 and the cluster stops expanding its kernel when TD reaches zero. The rate of decrease depends on the "fitness" of the data points that the cluster incorporates as shown in equation (31). With respect to cluster j,

$$TD_j^{new} = TD_j^{old} + \left(A - TD_j^{old}\right) \times \left(1 - \mu_j\left(x_i\right)\right)^2 \tag{31}$$

Where μ_j denotes the membership function of cluster j and $A = -0.5$.

When TD is less than or equal to zero, the cluster stops "growing" and sets its plasticity parameter β to zero. Hence, the less fit the data points (with small membership value) a cluster try to incorporate or absorb, the faster its TD decreases and vice-versa. Thus, TD and β together maintain the integrity of the input clusters and the fuzzy labels they represent.

3.4.4. Thresholds (IT and OT)

The input and output thresholds (IT and OT) specify the minimum "fitness" or membership value an input (output) data point must have before it is considered as relevant to any existing input (output) clusters or fuzzy sets. If the fitness of the input (output) data point to the existing best-fit input (output) cluster falls below the predefined IT (OT), then a new cluster is created. In addition, IT (OT) determines the degree of overlapping of an input (output) cluster with its immediate neighbors. In order to prevent excessive overlapping of the input (output) cluster, IT (OT) is predefined at 0.5.

Figure 3-6 shows an illustration for the influence of parameter IT (OT). In Figure 3-6(a), the parameter IT is set at a high value of 0.8. When new data x_i arrives, the matching degree of x_i

with existing cluster is less than 0.8. Therefore, a second cluster is created. It can be observed that both clusters are closely overlapped. To avoid high-overlapped clusters, the parameter IT can be set to a lower value. As shown in Figure 3-6(b), parameter IT is set to 0.5. Instead of creating a new cluster, existing cluster expand its kernel to "absorb" the new data.

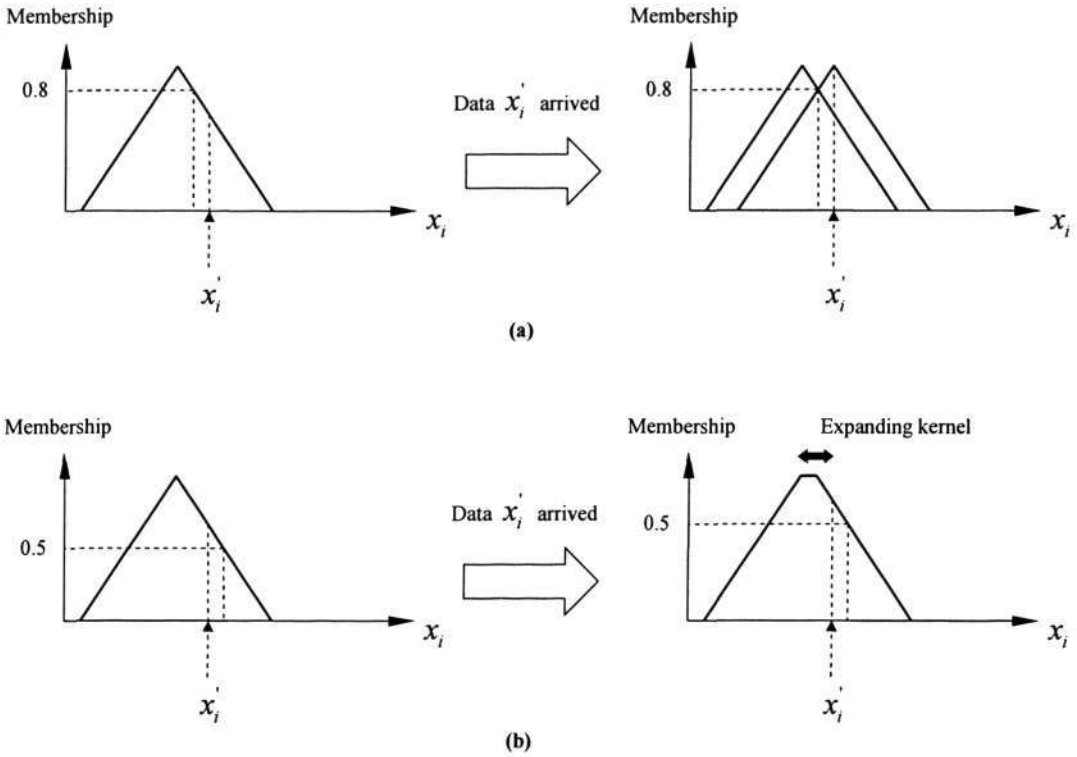


Figure 3-6: (a) Second cluster created for the new data when IT = 0.8 and (b) First cluster expanded to include the new data when IT = 0.5

The algorithm is briefly described as follow:

Algorithm DIC

Assume data set $\overline{X} = \left\{X^{(1)},...,X^{(p)},...,X^{(P)}\right\}$.

Initialize $STEP$, $SLOPE$, IT and TD.

Vector $X^{(p)} = \left\{X_1^{(p)},...,X_i^{(p)},...,X_I^{(p)}\right\}$ represents the p th input training vector to TSK^0-FCMAC.

Variable J_i represents the total number of clusters created for input x_i.

$\forall p \in \{1...P\}$

$\forall i \in \{1...I\}$

When J_i is zero, create a new cluster using $X_i^{(p)}$.

Otherwise,

Determine best-fit cluster *Winner* such that:

$$Winner = \arg \max_{j \in \{1...J_i\}} \left\{\mu_{A_{i,j}}\left(X_i^{(p)}\right)\right\}$$

Update the kernel of the *Winner* if $\mu_{A_{i,Winner}}\left(X_i^{(p)}\right) > IT$

Otherwise, create a new cluster using $X_i^{(p)}$

End DIC.

All parameters in DIC are constant except for the $STEP$ and $SLOPE$ parameters. In the current implementation, the selection of these two parameters is heuristic and varies with different tasks. However, there are several guidelines to assist in the selection of suitable values for these two parameters. A small $STEP$ value results in "fat" fuzzy sets with large kernels and vice versa. On the other hand, a small $SLOPE$ value results in steep slopes (nearly crisp fuzzy sets) and the fuzziness of the fuzzy set (input and output clusters) increases as the value of $SLOPE$ increases.

3.5. Benchmarking Examples

This section demonstrates the ability of the proposed architecture in problems of three different categories; namely: 1) inverted pendulum problem (control), 2) Fisher's Iris classification (classification) and, 3) chaotic time series prediction (regression).

In section 3.5.1, TSK^0-FCMAC is applied to balance an inverted pendulum. The performance of TSK^0-FCMAC is compared with two control schemes; namely: 1) H_∞ based controller by Feng [94] and, 2) robust fuzzy sliding mode (RFSM) controller by Khoo [95]. This experiment aims to demonstrate the ability of the proposed network in control applications. In the experiment, the proposed architecture is shown to be more adaptive than conventional control methodology. In section 3.5.2, TSK^0-FCMAC was applied in a well-known classification problem, the Fisher's Iris classification [88]. This experiment attempts to present the inherent ability of TSK^0-FCMAC to perform rule extractions. The performance of the network is also compared with several fuzzy neural networks, FCMAC-BYY [64], POPFNN-CRI (S) [91] and 3 variations of Falcon network; namely: Falcon-FKP [96], Falcon-ART [86] and Falcon-MART [87]. Lastly, TSK^0-FCMAC was applied in Mackey-Glass time series prediction [89] in section 3.5.3. This experiment attempts to investigate the ability of the proposed network in solving regression problem whose primary objective is to achieve satisfactory precision. The performance of TSK^0-FCMAC is compared with ANFIS, a first-ordered fuzzy inference system proposed by Jang [84].

3.5.1. Inverted Pendulum Problem

An inverted pendulum problem (also called a cart and pole problem) is a classical problem in dynamic control theory. The problem involves continuously applying a horizontal force on a moving cart to maintain the inverted pendulum in its upright position. It has been applied by various researchers from different areas to benchmark their systems [31, 32, 36-44]. Figure 3-7 shows the graphical representation of the problem.

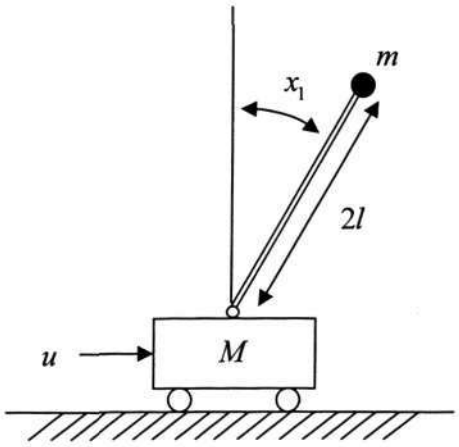


Figure 3-7: Inverted Pendulum Problem

The dynamics of the problem can be described as Eqs (32)-(35):

$$\dot{x}_1(t) = x_2(t) \tag{32}$$

$$\dot{x}_2(t) = \frac{g\sin(x_1(t)) - \frac{1}{2}aml(x_2(t))^2\sin(2x_1(t)) - a\cos(x_1(t))u(t)}{\frac{4}{3}l - aml\left[\cos(x_1(t))\right]^2} + disturbance(t) \tag{33}$$

$$a = \frac{1}{(m+M)} \tag{34}$$

$$disturbance(t) = D\left[2\sin(t) + \sin(200\pi t)\right] \tag{35}$$

Where,

$x_1(t)$ is the angular position of the pendulum from the vertical axis at

time t;

$x_2(t)$ is the angular velocity;

g is the gravitational acceleration, $g = 9.8 m/s^2$;

m is the mass of the pendulum, $m = 2kg$;

M is the mass of the cart, $M = 8kg$;

l is the half-length of the pole, $l = 0.5m$;

$u(t)$ is the horizontal force applied to the moving cart at time t;

$disturbance(t)$ is the disturbance to the dynamic system at time t;

D is the multiplier for the disturbance.

Many existing systems assume the dynamics of the problem to be "known" and design their controller based on these "known" factors. However, in real world applications, uncertainties (disturbance or noise) do exist. To take such uncertainties into consideration, a simple periodic disturbance is added to the dynamic equation to produce a better reflect of a possible real world application.

In this experiment, we compare the performance of TSK^0-FCMAC with two existing controllers; namely: 1) H_∞ based controller by Feng [94] and, 2) robust fuzzy sliding mode (RFSM) controller by Khoo [95]. They belong to two particularly active research fields, the H_∞ control [97-100] and the sliding mode control system [101-104]. Both research areas have been shown to produce interesting results in many control applications.

In the experiments conducted by Feng and Khoo, both controllers use the derivatives of the angular position, x_1 as inputs to their controllers. Similarly, our design uses two inputs derived from the angular position. They are computed as Eqs (36) and (37):

$$Log\_x_1(t) = \begin{cases} 0 & ,if\ |1000x_1(t)| \le 1 \\ sign(1000x_1(t)) \times \ln(abs(1000x_1(t))) & ,\ otherwise \end{cases} \quad (36)$$

and

$$Log\_x_2(t) = \begin{cases} 0 & ,if\ |100x_2(t)| \le 1 \\ sign(100x_2(t)) \times \ln(abs(100x_2(t))) & ,\ otherwise \end{cases} \quad (37)$$

Where,

$sign(.)$ is the function that returns 1 if the number is positive and -1 if the number is negative;

$abs(.)$ is the function that returns the absolute value.

The TSK^0-FCMAC controller is first trained by modeling the response of a H_∞ controller. The training data are collected by simulating the H_∞ controller. The trained network is a two inputs TSK^0-FCMAC network with a size of 168 cells (12×14) and is subsequently applied to balance the inverted pendulum with the following initial conditions:

$x_1(0)$ is the initial angular position, $x_1(0) = 85°$;

T is the total simulation time (sec), $T = 2.0$ and $t \in [0, T]$;

dt is the simulation step (sec), $dt = 0.0005$.

In each step, the TSK^0-FCMAC network is updated based on the error defined in Eq (38):

$$Error_{controller}(t) = (u_{\max} - u_{\min}) \times \delta_{controller} \times \frac{x_1(t) - x_{\text{target}}}{x_{\max} - x_{\min}} \quad (38)$$

Where

$u_{\max}$, $u_{\min}$ are the maximum and minimum output from the controller;

$\delta_{controller}$ is the learning parameter for controller. It is set to $1/60$ to

prevent drastic update on the controller's surface;

$x_1(t)$ is the angular position of the pendulum from the vertical axis at time t;

x_{target} is the target angular position. In this experiment, $x_{target} = 0$;

x_{max}, x_{min} Are the maximum and minimum possible angular position;

In the experiment, mean square error (MSE) is employed as the performance indicator for the three different controllers. The multiplier for disturbance (D) described in Eq (35) is varied to examine the stability of individual controller versus different disturbances. The performances are tabulated in Table 3-1.

The results show that the average MSE of TSK^0-FCMAC is lowest among the three different control schemes. When the disturbance multiplier, D increases from 10 to 300, the MSE of H_∞ and RSFM control schemes differ for 15.35% and 35.06% respectively. In contrast, the MSE of TSK^0-FCMAC remains relatively stable.

Table 3-1: MSE versus Different Disturbances in the Inverted Pendulum Problem

D	H_∞ Control		RSFM Control		TSK^0-FCMAC Control	
	MSE	% w.r.t $D=10$	MSE	% w.r.t $D=10$	MSE	% w.r.t $D=10$
10	0.0821	100.00%	0.0870	100.00%	0.0754	100.00%
50	0.0834	101.58%	0.0888	102.07%	0.0714	94.69%
100	0.0851	103.65%	0.0913	104.94%	0.0715	94.83%
150	0.0871	106.09%	0.0939	107.93%	0.0721	95.62%
200	0.0893	108.77%	0.0971	111.61%	0.0727	96.42%
250	0.0919	111.94%	0.1022	117.47%	0.0735	97.48%
300	0.0947	115.35%	0.1175	135.06%	0.0742	98.41%
Average	0.0877	-	0.0968	-	0.0730	-

D = Disturbance Multiplier; MSE = Mean Square Error.

Figure 3-8, Figure 3-9 and Figure 3-10 show the angular position and controller's output for three different control schemes under disturbance multiplier of 10, 150 and 300 respectively. All three control schemes manage to shift the angular position of the pendulum back to 0° at almost the same time and afterwards, balance the pendulum in its upright position. However, there is a noticeable increase in fluctuations when D increases from 10 to 300. To provide a closer view, Figure 3-11 and Figure 3-12 show the simulation result between 0.4 and 1.5 seconds. In the two mentioned figures, the fluctuations in angular position increase significantly for H_∞ and RSFM control schemes while the angular position of TSK^0-FCMAC control scheme remain fairly consistent.

It is known that both H_∞ and RSFM controllers are designed based on some "known" factors (e.g. the characteristics of the plant) and are non-adaptive to changing environment. Their performances remain good if the "known" factors are indeed accurate. On the other hand, TSK^0-FCMAC is trained based on the same "known" factors, and subsequently performs online learning according to the environment (which consists of possible "unknown" factors). This experiment draw the attention to the shortcoming of having a controller designed based on offline data only. The proposed architecture, TSK^0-FCMAC clearly demonstrates its ability to perform online learning for "unknown" factor and at the same time, achieve reasonable performance.

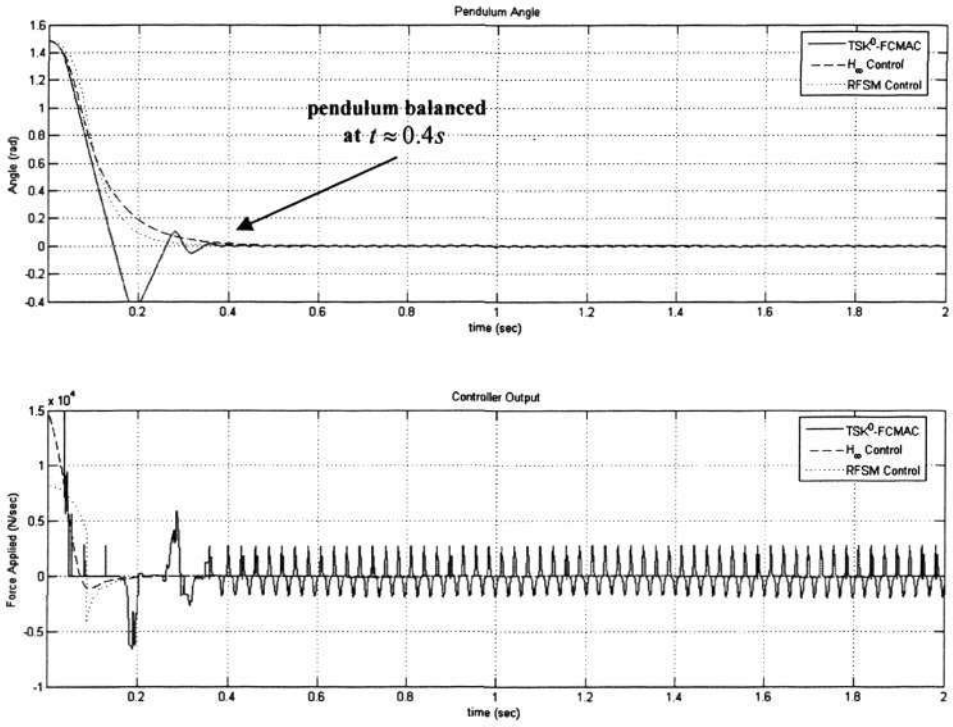


Figure 3-8: Pendulum's Angular Position and Controller's Output ($D = 10$)

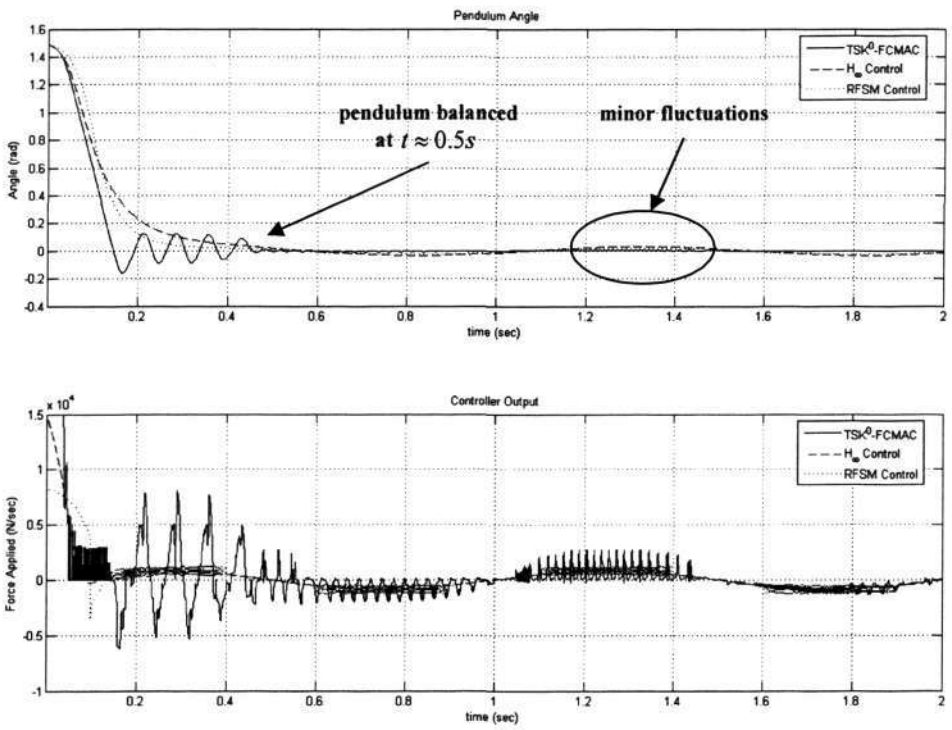


Figure 3-9: Pendulum's Angular Position and Controller's Output ($D = 150$)

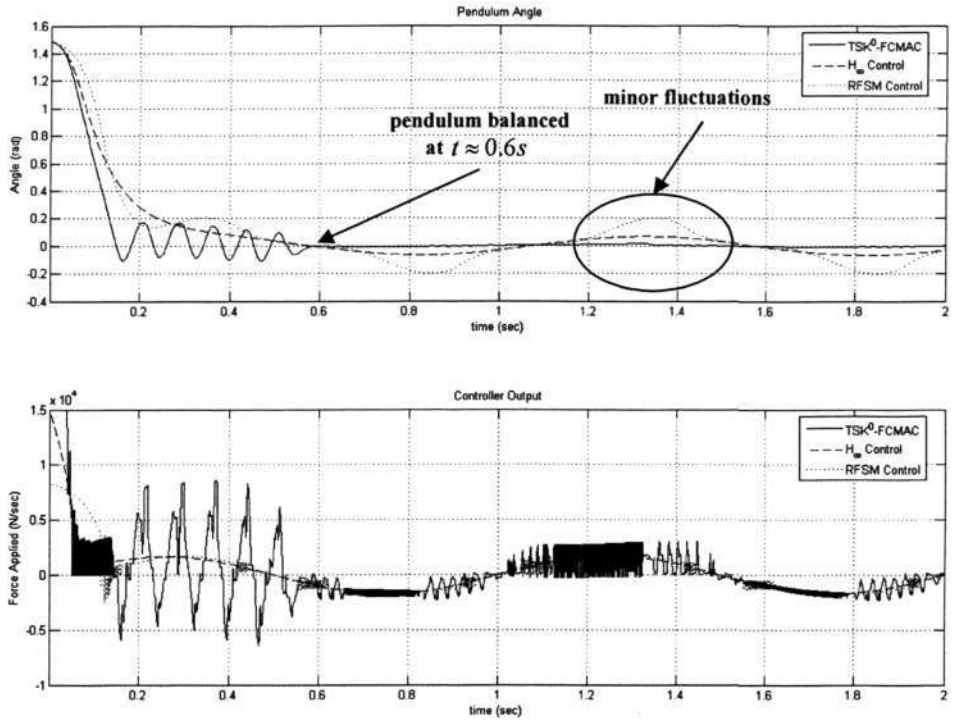


Figure 3-10: Pendulum's Angular Position and Controller's Output ($D = 300$)

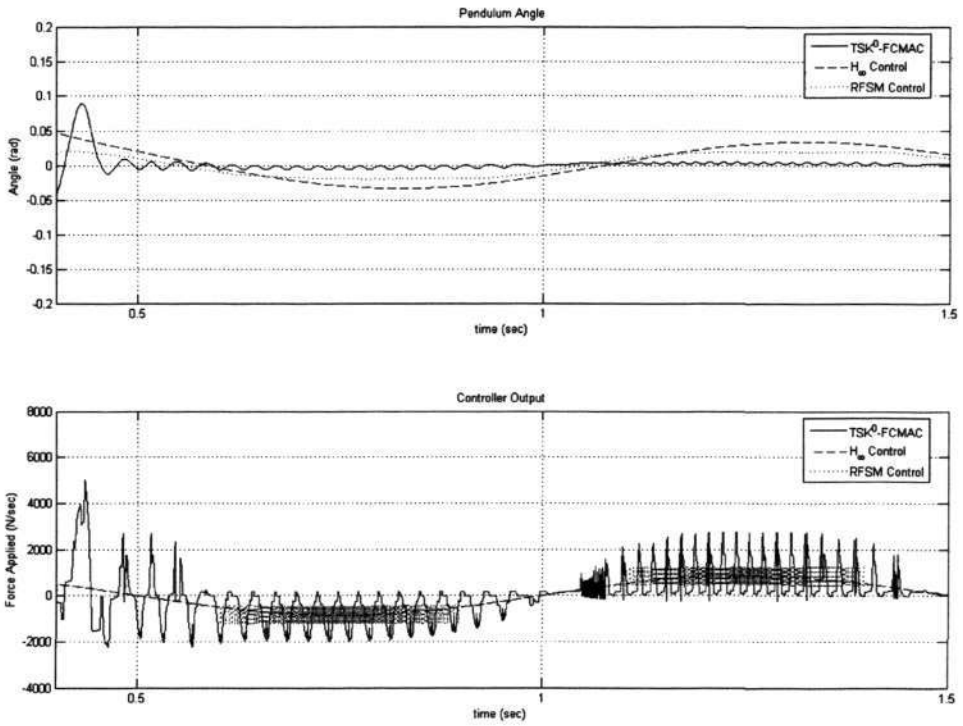


Figure 3-11: Pendulum's Angular Position and Controller's Output ($D = 150$, $t \in [0.4, 1.5]$)

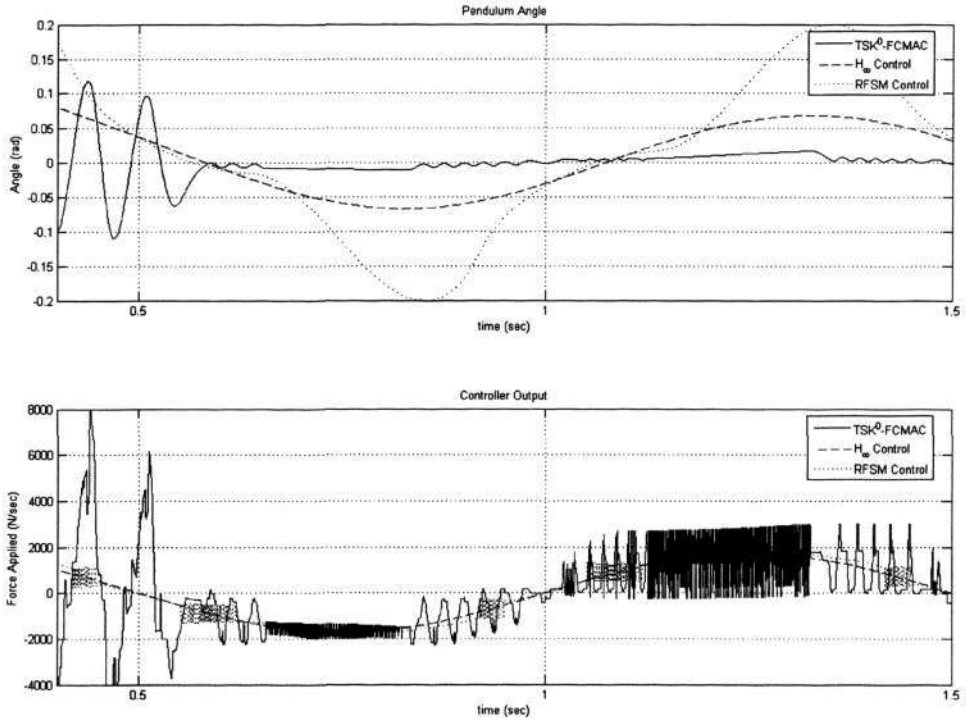


Figure 3-12: Pendulum's Angular Position and Controller's Output ($D = 300$, $t \in [0.4, 1.5]$)

3.5.2. Fisher's Iris Classification

A benchmark classification problem using the Fisher's Iris data set [88] was conducted. There are three classes of irises; namely: Class 1-Setosa, Class 2-Versicolor and Class 3-Virginica. Each class consists of 50 data instances. Figure 3-13 shows the distribution of each of the 3 classes on four numeric attributes. It is clear that Class 1-Setosa is separable from the other 2 classes but classes 2 and 3 are highly overlapped in the sepal length and width dimensions. It can also be noticed that there are some overlapping in the petal length and width for class 2 and 3.

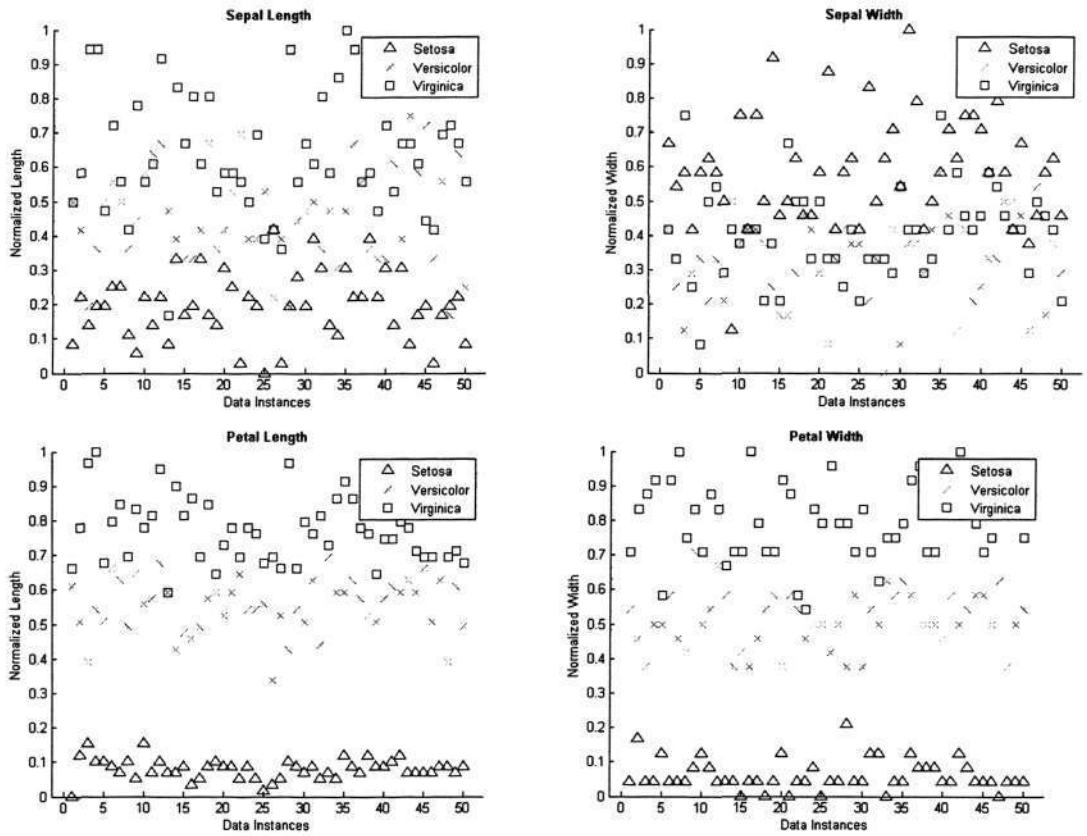


Figure 3-13: Distributions of Fisher's Iris Data

The 150 iris data are separated into 3 groups of equal sizes; each group consists of an equal number for all three classes of irises. By combining the three different groups, seven data sets are generated (one set of 150-data, three sets for 100-data and three sets of 50-data). Hundred permutations are performed on each set of data and a total of 700 data sets are generated. Permutation involves reshuffle of the data instances in random order. The network structure is MISO (multiple inputs single output) and the desired outputs are 0, 0.5 and 1.0 for class 1, class 2 and class 3 respectively.

The experiments were conducted in five different modes; namely:

- *Memory Recall*^{a} : uses all 700 sets of data for both training and testing.
- *Generalization*^{b} : uses 300 sets of 100-data for training and 50-data for testing.
- *Generalization*^{c} : uses 300 sets of 50-data for training and 100-data for testing.
- *Generalization*^{d} : uses 3 sets of 50-data for training and 100-data for testing.
- *Memory Recall*^{e} : uses a single set of 150-data for both training and testing.

The training data for *Memory Recall*^{a} , *Generalization*^{b} and *Generalization*^{c} involve permutation of the original training data. Hence, the mean value and the standard deviations obtained from these three experiments are more representative than those obtained in *Memory Recall*^{e} and *Generalization*^{d} .

Comparisons are conducted with several fuzzy neural networks. They are FCMAC-BYY [64], POPFNN-CRI (S) [91] and 3 variations of Falcon network; namely: Falcon-FKP [96], Falcon-ART [86] and Falcon-MART [87]. FCMAC-BYY [64] is an associative memory neural network using Bayesian Ying-Yang (BYY) learning [82] to determine the optimal fuzzy sets with truth-value restriction inference scheme. POPFNN-CRI (S) (Pseudo-outer product based fuzzy neural network using the compositional rule of inference and singleton fuzzifier) is a two-phase learning fuzzy neural network that is able to effectively construct membership functions and identifies the fuzzy if-then rules. Falcon-MART and Falcon-FKP are both variations of Falcon-ART network proposed by Quek and Tung.

Table 3-2 shows the classification results for all six fuzzy neural networks. For TSK^0-FCMAC, high mean classification rate of 98.43%, 93.75% and 92.48% is achieved for *Memory Recall*^{a} and *Generalization*^{b} and *Generalization*^{c} . The small value in standard deviation shows that the architecture is robust even though different permutation of the same data sets is provided as training data. Hence, TSK^0-FCMAC has a high resistance against influence under different initial seeds.

Table 3-2: Fisher's Iris Classification Rate

	Experiment	Training Epoch	Mean	Std Dev
TSK^0-FCMAC	*Memory Recall*[a]	20	98.43%	1.67%
	Generalization[b]	20	93.75%	3.51%
	Generalization[c]	20	92.48%	3.21%
	Generalization[d]	20	93.33%	3.06%
	Memory Recall[e]	20	98.67%	-
FCMAC-BYY	*Generalization*[d]	10	96.60%	-
Falcon-FKP	*Generalization*[d]	15	90.57%	2.53%
Falcon-ART	*Generalization*[d]	1000	75.76%	7.54%
Falcon-MART	*Generalization*[d]	11	94.95%	4.64%
POPFNN-CRI	*Memory Recall*[e]	15	80.00%	-

Std Dev = Standard Deviation.

The training of TSK^0-FCMAC stopped after 20 training epochs and the MSE after every training epoch is shown in Figure 3-14.

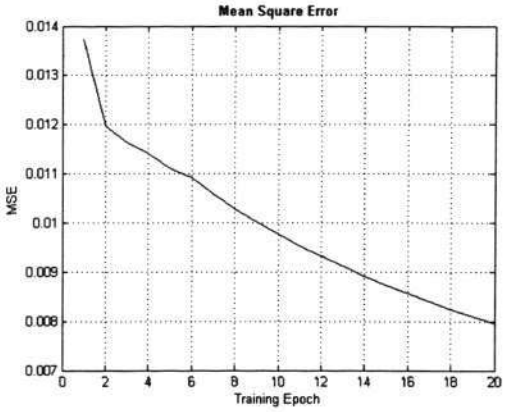


Figure 3-14: Mean Square Error of TSK^0-FCMAC in Fisher's Iris Classification

Figure 3-15 shows the fuzzy sets derived from TSK^0-FCMAC by using a set of 150-data for both training and testing. Among the 150 testing data, 148 have been identified correctly. The total number of possible if-then rules is $4\times4\times4\times3=192$ but only 148 if-then rules are created because the input patterns do not cover the whole input space.

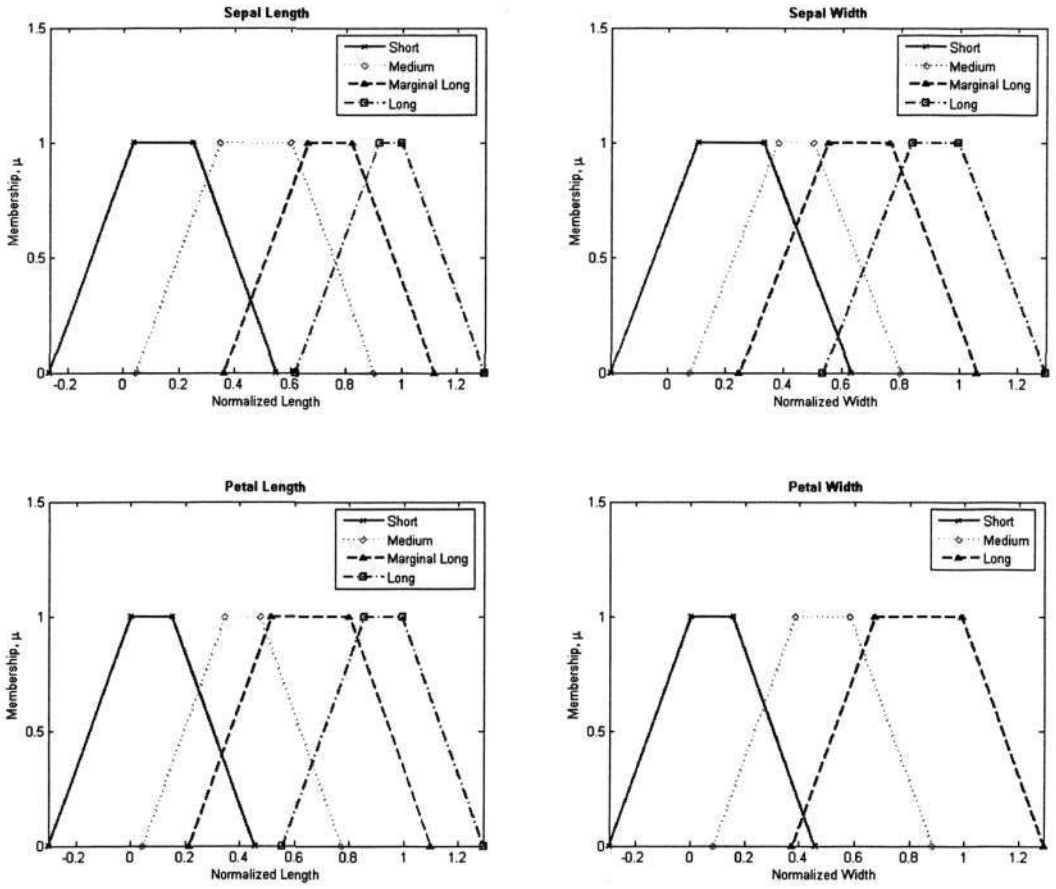


Figure 3-15: Fuzzy Sets Derived for Fisher's Iris Classification using TSK^0-FCMAC

To present a clearer understanding of the fuzzy if-then rules, semantics are assigned to every fuzzy sets of each numeric attributes; namely: Short, Medium, Marginal Long and Long. The fuzzy rules derived for each class of irises are extracted and are listed in Table 3-3. As mentioned above, although classes 2 and 3 have high degree of overlapping for sepal length and width and some overlapping at petal length and width, TSK^0-FCMAC is able to achieve a high classification ratio with only 20 training iterations. The inherent ability of TSK^0-FCMAC to perform rule extractions is also demonstrated in this experiment.

Table 3-3: If-Then Fuzzy Rules Derived for Fisher's Iris Classification

R_1:	*if*							*PL is*	*short*				*then Iris is*	*Setosa*
R_2:	*if*	*SL is*	*short or medium or marginal long*	*and*	*SW is*	*short or medium*	*and*	*PL is*	*medium*	*and*	*PW is*	*short*	*then Iris is*	*Versicolor*
R_3:	*if*	*SL is*	*marginal long or long*	*and*	*SW is*	*marginal long or long*	*and*	*PL is*	*marginal long*	*and*	*PW is*	*medium*	*then Iris is*	*Versicolor*
R_4:	*if*				*SW is*	*medium*	*and*	*PL is*	*long*	*and*	*PW is*	*medium*	*then Iris is*	*Virginica*
R_5:	*if*	*SL is*	*long*				*and*	*PL is*	*long*	*and*	*PW is*	*long*	*then Iris is*	*Virginica*

SL = Sepal Length; SW = Sepal Width; PL = Petal Length; PW = Petal Width.

3.5.3. Chaotic Time Series Prediction

The Mackey-Glass (MG) time-delay differential equation was first investigated by Mackey and Glass [89]. Due to its chaotic nature, it has been used as a benchmark case-study in the neuro-fuzzy community [105-108]. The chaotic differential equation is defined as Eq (39):

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{39}$$

The goal of the task is to use known values of the time series up to the point $x = t$ to predict the value at some point in the future $x = t + \Delta$. The standard method for this type of prediction is to create a mapping from D points of the time series spaced Δ apart, that is, $\big(x(t-(D-1)\Delta), ..., x(t-\Delta), x(t)\big)$, to a predicted future value $x(t+\Delta)$. The value $D = 4$ and $\Delta = 6$ were used to allow comparison with earlier work.

To obtain the time series value at each integer point, the fourth-order Runge-Kutta method [109] was applied to incrementally compute the numerical solution to equation (39). The time step used in the method is 0.1, initial condition $x(0)$ = 1.2, τ = 17 and $x(t)$ is thus derived for $0 \le t \le 1200$. Figure 3-16 shows the time series derived. From the Mackey-Glass time series $x(t)$, 1000 input-output data pairs of the following format were extracted:

$$\big[x(t-18), x(t-12), x(t-6), x(t), x(t+6)\big] \tag{40}$$

Where t = 118 to 1117.

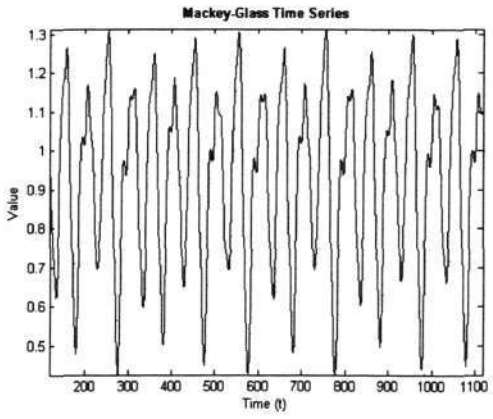


Figure 3-16: Mackey-Glass Time Series

The 1000 data pairs are separated into 3 sets: all 1000 pairs, first 500 pairs and last 500 pairs. A hundred permutations are performed on each group of data and a total of 300 data sets are generated. Permutation involves reshuffling of the data instances in random order. Permutations are performed on the data sets to bestow a more representative meaning to the mean value and the standard deviation of the results. In addition, the experiments will be able to evaluate the robustness of the system if the data are presented in different sequences.

The experiments were conducted in four different modes; namely:

- *Memory Recall*[a] : uses all 300 sets of data, use the same set for both training and testing.
- *Generalization*[b] : uses 1) 100 sets of the first 500 pairs for training, 100 sets of the last 500 pairs for testing and, 2) 100 sets of the last 500 pairs for training, 100 sets of the first 500 pairs for testing.
- *Memory Recall*[c] : uses 1 set of the first 500 pairs for both training and testing.
- *Generalization*[d] : uses 1 set of the first 500 pairs for training and last 500 pairs for testing.

The predicted values in experiment *Memory Recall*[c] are shown in Figure 3-17. In this experiment, TSK[0]-FCMAC generated 4 fuzzy sets for each input and a total of 226 fuzzy if-then rules are generated in 20 epochs. The MSE after every training epoch is shown in Figure 3-18.

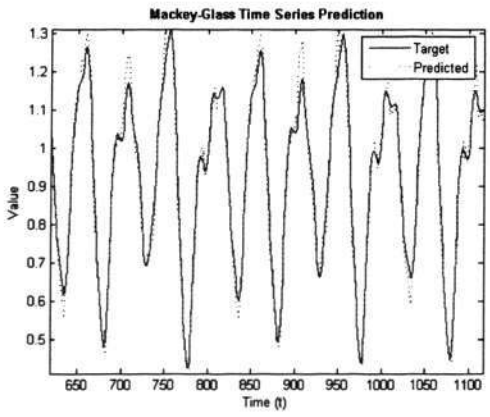


Figure 3-17: Predicted Mackey-Glass Time Series

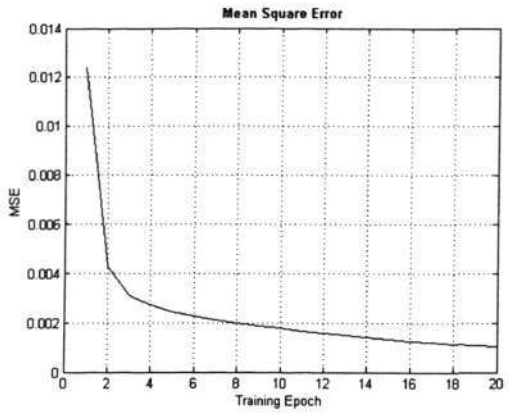


Figure 3-18: Mean Square Error of TSK^0-FCMAC in Mackey-Glass Time Series Prediction

Table 3-4 shows the tabulated results of the four experiments mentioned above. The performance of TSK^0-FCMAC is compared with ANFIS (Adaptive-Network-Based Fuzzy Inference System), a first-ordered TSK model [18, 19] proposed by Jang [84]. The number of membership functions assigned to each input of the ANFIS was arbitrarily set to 2, so the rule number is 16. ANFIS performed 500 epochs and contains 104 fitting parameters, of which 24 are premise parameters and 80 are consequent parameters. 16 fuzzy if-then rules are generated [84]. Excellent results could be obtained with ANFIS at the expense of a large number of training epochs. On the other hand, the requirement of predefined structure is another shortcoming not addressed in the ANFIS architecture.

It can be observed that ANFIS performed better than TSK^0-FCMAC in terms of Mackey-Glass time series prediction as ANFIS implements the first-ordered TSK inference model. In ANFIS, the output for each rule is a linear combination of input variables plus a constant term whereas the output for TSK^0-FCMAC is only a crisp set (equivalent to a constant term). In addition, TSK^0-FCMAC uses triangular or trapezoidal membership function for the premises which is computationally efficient while ANFIS uses the more complex Gaussian function. Nonetheless, the small value in standard deviation of TSK^0-FCMAC demonstrates its robustness although training data is provided in different sequence.

Table 3-4: RMSE for Mackey-Glass Prediction

	Experiment	Training Epoch	Mean of RMSE	Std Dev of RMSE
TSK^0-FCMAC	Memory Recall[a]	20	0.02075	0.00463
	Generalization[b]	20	0.02229	0.00468
	Memory Recall[c]	20	0.03122	-
	Generalization[d]	20	0.03205	-
ANFIS	Memory Recall[c]	500	0.00160	-
ANFIS	Generalization[d]	500	0.00150	-

RMSE = Root Mean Square Error; Std Dev = Standard Deviation.

3.6. Summary

This chapter presents the architecture and learning algorithm of the proposed network, TSK^0-FCMAC. Analogous to conventional CMAC, TSK^0-FCMAC was proposed to be applicable in the domain of control problems. Therefore, complex functions such as the Gaussian or bell-shaped membership function are not adopted. Instead, the antecedents for TSK^0-FCMAC are represented in triangular or trapezoidal membership functions. Thus, accuracy is sacrificed for simplicity and computational efficiency. On the other hand, the consequents of TSK^0-FCMAC are represented by fuzzy singleton for efficiency in computation.

The ability of the proposed network is demonstrated with empirical applications on three different categories; namely: 1) inverted pendulum problem (control), 2) Fisher's Iris classification (classification) and, 3) chaotic time series prediction (regression). In section 3.5.1, TSK^0-FCMAC is first demonstrated to be capable to balance the inverted pendulum. The performance is comparable with two existing controller, the H_∞ controller by Feng [94] and the RSFM controller by Khoo [95]. In additional to comparable performance, the experiment also draws the attention to the advantage of designing a controller that is capable of adapting to "unknown" factors. As a result, the online learning capability of TSK^0-FCMAC proves to be highly advantageous over conventional controllers. In the second experiment, TSK^0-FCMAC was applied in a well-known classification problem, the Fisher's Iris classification [88] in section 3.5.2. The inherent ability of TSK^0-FCMAC to perform rule extractions is demonstrated in the

experiment. At the same time, the proposed network is found to be robust under different permutations of input data (reshuffle of the data instances in random order). In the third experiment, TSK^0-FCMAC was applied in a regression problem, Mackey-Glass time series prediction [89]. Likewise, the proposed network is found to be robust versus permutation of data samples and achieve reasonable results.

Although TSK^0-FCMAC was initially designed to be applied in the domain of control problems, empirical studies present satisfactory performance in problems from other domains. Nonetheless, in terms of solving problem which emphasizes precision as its primary objective, TSK^0-FCMAC may prove to be inadequate. This inspires the implementation of TSK^1-FCMAC, a first-ordered Takagi-Sugeno-Kang fuzzy CMAC, which may theoretically breach the barrier of mediocre precision. The architecture for TSK^1-FCMAC is presented in Chapter 5 in full details.

Chapter 4 Learning Convergence of TSK^0-FCMAC

4.1. Introduction

The cerebellar model articulation controller (CMAC), an associative memory proposed by Albus [2] is capable of performing localized generalization with very fast learning. Since the formulation of the first conventional CMAC network, many different variances have been proposed. Many of them are enhanced by incorporating fuzzy logic into the CMAC networks and hence defined as *Fuzzy CMAC*. Most of them focused on the development of new learning techniques [61, 110-112], new structural learning [64, 83, 113], and novel application [63, 114-116]. However, the mathematical formulation and the convergence characteristics of such *Fuzzy CMAC* are often neglected. The learning convergence of conventional CMAC networks is first investigated by Wong and Sideris [48]. Later, Parks and Militzer [46] provide an in-depth investigation with Lyapunov function, and most recently, Lin and Chiang [49] define the mathematical formulation and have proven that the memory contents of a conventional CMAC structure will converge to a limit cycle providing the learning rate is between zero and two.

In this chapter, a study on the convergence characteristic of the proposed architecture is rigorously undertaken. A mathematical representation of the proposed network is presented together with the investigation of its convergence characteristic. In terms of mathematical formulations, the difference between a conventional CMAC and a fuzzy CMAC is the way a memory cell is activated. In conventional CMAC, individual memory cell is either activated (represented by "1") or non-activated (represented by "0"). However, in fuzzy CMAC, the memory cell could be partially activated (represented by degree of matching, $\alpha \in [0,1]$). Nonetheless, the formulation in the investigation will follow closely to Lin and Chiang's [49] methodology while adapted to our TSK^0-FCMAC architecture. The investigation starts by formulating the mathematical representation of the TSK^0-FCMAC architecture, followed by the proper definition of the difference of memory contents between consecutive iterations. The learning of TSK^0-FCMAC will be proven to converge when this difference approaches zero. An overview for the proof of convergence is presented in Figure 4-1.

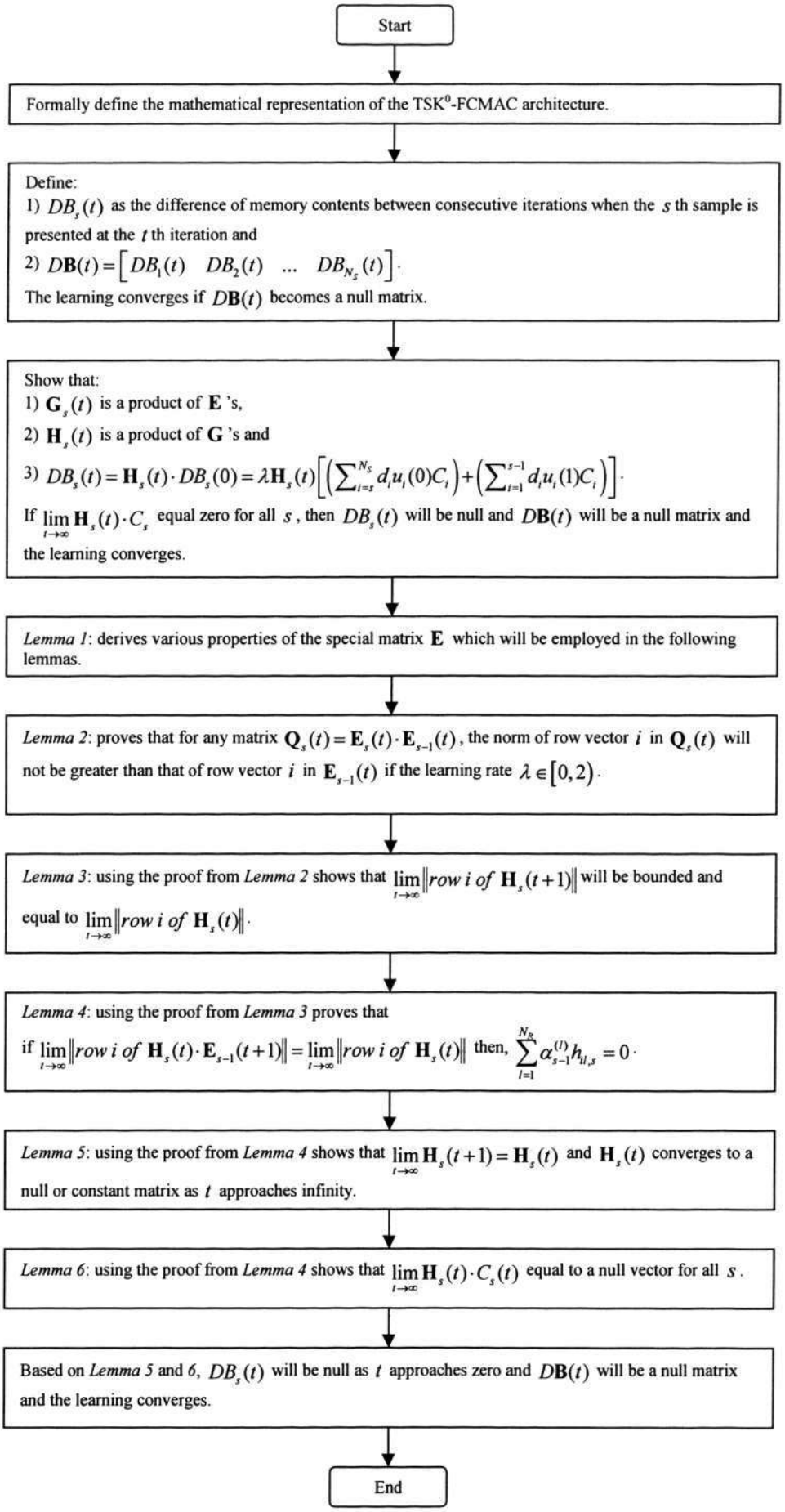


Figure 4-1: An overview for the Proof of Convergence

4.2. Mathematical Formulation

Assuming there are N_R number of rules stored in a TSK^0-FCMAC. For the s th sample, the output can be mathematically expressed as Eq (41).

$$\begin{matrix} & Rule\ 1 & Rule\ 2 & \cdots & Rule\ i & \cdots & Rule\ N_R \end{matrix}$$

$$y_s(t) = C_s^T B = \overbrace{\left[\frac{\alpha_s^{(1)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \quad \frac{\alpha_s^{(2)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \quad \cdots \quad \frac{\alpha_s^{(i)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \quad \cdots \quad \frac{\alpha_s^{(N_R)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \right]}^{C_s^T} \begin{bmatrix} b_{0,s}^{(1)}(t) \\ b_{0,s}^{(2)}(t) \\ \cdots \\ b_{0,s}^{(i)}(t) \\ \cdots \\ b_{0,s}^{(N_R)}(t) \end{bmatrix} \tag{41}$$

$$= \frac{1}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \left[\sum_{k=1}^{N_R} \alpha_s^{(k)} b_{0,s}^{(k)}(t) \right]$$

Where C_s is a memory selection vector computed according to the degree of matching of the s th sample with the respective fuzzy rule. Considering the case with N_S as the number of samples, the output for every sample can be expressed in a matrix form expressed in Eq (42).

$$Y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \cdots \\ y_{N_S}(t) \end{bmatrix} = \begin{bmatrix} C_1^T \\ C_2^T \\ \cdots \\ C_{N_S}^T \end{bmatrix} B_s(t) \tag{42}$$

$$= \mathbf{C} B_s(t)$$

In practice, the training samples are chosen at random among all the training set. However, it will be difficult or even impossible to study the convergence characteristics with samples provided in this nature because the learning will continue adapting to fresh training samples and never converge. Thus, in this study, only the case that a set of N_S training samples is repeatedly presented to the network is considered.

The updating of cell content at the presentation of the $(s-1)$th sample during the t th iteration is expressed as

$$\begin{aligned} B_s(t) &= B_{s-1}(t) + \Delta B_{s-1}(t) \\ &= B_{s-1}(t) + \lambda\left(\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}\right) C_{s-1}\left[\hat{y}_{s-1}(t) - C_{s-1}^T B_{s-1}(t)\right] \end{aligned} \tag{43}$$

Define the sum of the degree of matching between the s th sample and every fuzzy if-then rules as

$$d_s = \sum_{k=1}^{N_R} \alpha_s^{(k)} \tag{44}$$

We have

$$B_s(t) = B_{s-1}(t) + \lambda d_{s-1} C_{s-1}\left[\hat{y}_{s-1}(t) - C_{s-1}^T B_{s-1}(t)\right] \tag{45}$$

Where,

$B_s(t)$ is the cell content at the presentation of the s th sample during the t th iteration;

λ is the learning parameter in the t th iteration, where $\lambda \in [0,1]$;

$d_s C_s$ is the pre-multiplication that distributes the error according to the degree of matching for each fuzzy if-then rule for the s th sample;

$\hat{y}_s(t)$ is the output of the network for the s th sample at the t th iteration.

4.3. Proof of Convergence

Convergence in iterative learning with a set of training data repeatedly presented can be defined in two ways. The first one is that the update on memory content remains the same in different iterations when the same sample is presented. However, from sample to sample, memory contents may still change. The second definition of convergence is such that the variation vanishes after numerous iterations [49]. They are illustrated in Figure 4-2 and Figure 4-3 respectively. In this study, we will first examine the first type of convergence to determine whether the memory

contents at the time when the s th sample is presented can converge. If it converges, by gradually decreasing the learning rate, the second kind of convergence can be achieved.

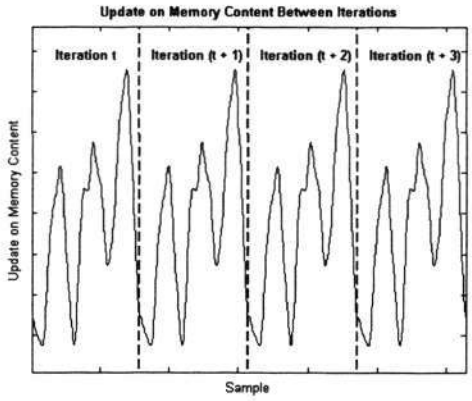


Figure 4-2: First Type of Convergence (update on memory content remains the same in different iterations)

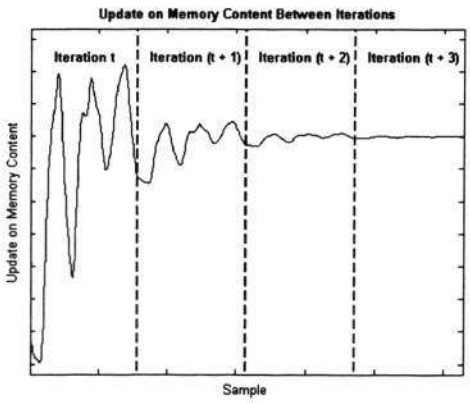


Figure 4-3: Second Type of Convergence (variation between iterations vanishes)

The difference of memory contents between consecutive iterations can be expressed as Eq (46).

$$\begin{aligned} DB_s(t) &= B_s(t+1) - B_s(t) \\ &= \left[B_{s-1}(t+1) + \Delta B_{s-1}(t+1)\right] - \left[B_{s-1}(t) + \Delta B_{s-1}(t)\right] \\ &= DB_{s-1}(t) + \lambda d_{s-1} C_{s-1}\left[\hat{y}_{s-1}(t) - C_{s-1}^T B_{s-1}(t+1)\right] \\ &\quad - \lambda d_{s-1} C_{s-1}\left[\hat{y}_{s-1}(t) - C_{s-1}^T B_{s-1}(t)\right] \\ &= DB_{s-1}(t) - \lambda d_{s-1} C_{s-1} C_{s-1}^T\left[B_{s-1}(t+1) - B_{s-1}(t)\right] \\ &= DB_{s-1}(t) - \lambda d_{s-1} C_{s-1} C_{s-1}^T\left[DB_{s-1}(t)\right] \\ &= \left[I - \lambda d_{s-1} C_{s-1} C_{s-1}^T\right] DB_{s-1}(t) \end{aligned} \tag{46}$$

Defining

$$DB_0(t) \equiv DB_{N_S}(t-1) \tag{47}$$

and

$$C_0 \equiv C_{N_S} \tag{48}$$

will take care of the special case with $s = 1$ in the above equation.

For convenience, define

$$\mathbf{E}_s(t) = I - \lambda d_s C_s C_s^T \tag{49}$$

and

$$D\mathbf{B}(t) = \begin{bmatrix} DB_1(t) & DB_2(t) & \ldots & DB_{N_S}(t) \end{bmatrix} \tag{50}$$

Hence,

$$\begin{aligned} DB_s(t) &= \mathbf{E}_{s-1}(t) DB_{s-1}(t) \\ &= \mathbf{E}_{s-1}(t)\mathbf{E}_{s-2}(t)\ldots\mathbf{E}_1(t)\mathbf{E}_0(t) DB_0(t) \end{aligned} \tag{51}$$

From Eq (47), $DB_s(t)$ can be derived as

$$\begin{aligned} DB_s(t) &= \mathbf{E}_{s-1}(t)\mathbf{E}_{s-2}(t)\ldots\mathbf{E}_1(t)\mathbf{E}_{N_S}(t) DB_{N_S}(t-1) \\ &= \left[\mathbf{E}_{s-1}(t)\mathbf{E}_{s-2}(t)\ldots\mathbf{E}_1(t)\mathbf{E}_{N_S}(t)\mathbf{E}_{N_S-1}(t)\ldots\mathbf{E}_s(t)\right] DB_s(t-1) \end{aligned} \tag{52}$$

For simplicity, define

$$\mathbf{G}_s(t) = \left[\mathbf{E}_{s-1}(t)\mathbf{E}_{s-2}(t)\ldots\mathbf{E}_1(t)\mathbf{E}_{N_S}(t)\mathbf{E}_{N_S-1}(t)\ldots\mathbf{E}_s(t)\right] \tag{53}$$

Hence $DB_s(t)$ can now be expressed as

$$DB_s(t) = \mathbf{G}_s(t) DB_s(t-1) \tag{54}$$

To have the learning converge, $\lim_{T\to\infty} DB_s(t)$ must be a null vector and $\lim_{T\to\infty} D\mathbf{B}(t)$ must be a null matrix. With Eqs (50) and (54), the matrix $D\mathbf{B}(t)$ can be derived as

$$
\begin{aligned}
D\mathbf{B}(t) &= \begin{bmatrix} DB_1(t) & DB_2(t) & \dots & DB_{N_S}(t) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{G}_1(t)DB_1(t-1) & \mathbf{G}_2(t)DB_2(t-1) & \dots & \mathbf{G}_{N_S}(t)DB_{N_S}(t-1) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{G}_1(t)\mathbf{G}_1(t-1)DB_1(t-2) & \mathbf{G}_2(t)\mathbf{G}_2(t-1)DB_2(t-2) & \dots \\ & & \mathbf{G}_{N_S}(t)\mathbf{G}_{N_S}(t-1)DB_{N_S}(t-2) \end{bmatrix} \\
&= \begin{bmatrix} \left(\mathbf{G}_1(t)\mathbf{G}_1(t-1)\dots\mathbf{G}_1(1)\right)DB_1(0) & \left(\mathbf{G}_2(t)\mathbf{G}_2(t-1)\dots\mathbf{G}_2(1)\right)DB_2(0) & \dots \\ & & \left(\mathbf{G}_{N_S}(t)\mathbf{G}_{N_S}(t-1)\dots\mathbf{G}_{N_S}(1)\right)DB_{N_S}(0) \end{bmatrix}
\end{aligned} \tag{55}
$$

For simplicity, define

$$\mathbf{H}_s(t) = \mathbf{G}_s(t)\mathbf{G}_s(t-1)\dots\mathbf{G}_s(1) \tag{56}$$

Hence $D\mathbf{B}(T)$ can now be expressed as Eq (57)

$$D\mathbf{B}(t) = \begin{bmatrix} \mathbf{H}_1(t)DB_1(0) & \mathbf{H}_2(t)DB_2(0) & \dots & \mathbf{H}_{N_S}(t)DB_{N_S}(0) \end{bmatrix} \tag{57}$$

Thus, if $\mathbf{H}_s(t)DB_s(0) = 0$ for $1 \le s \le N_S$ then the learning will converge as $D\mathbf{B}(t)$ becomes a null matrix. The following derivation provides a closer look at the inner product $\mathbf{H}_s(t)DB_s(0)$.

One can derive $DB_s(0)$ as

$$
\begin{aligned}
DB_s(0) &= B_s(1) - B_s(0) \\
&= B_{s-1}(1) + \Delta B_{s-1}(1) - B_s(0) \\
&= B_1(1) + \Delta B_1(1) + \Delta B_2(1) + \dots + \Delta B_{s-1}(1) - B_s(0) \\
&= B_{N_S}(0) + \Delta B_{N_S}(1) + \Delta B_1(1) + \Delta B_2(1) + \dots + \Delta B_{s-1}(1) - B_s(0) \\
&= B_s(0) + \Delta B_s(0) + \Delta B_{s+1}(0) + \dots + \Delta B_{N_S}(0) \\
&\quad + \Delta B_1(1) + \Delta B_2(1) + \dots + \Delta B_{s-1}(1) - B_s(0) \\
&= \Delta B_s(0) + \Delta B_{s+1}(0) + \dots + \Delta B_{N_S}(0) \\
&\quad + \Delta B_1(1) + \Delta B_2(1) + \dots + \Delta B_{s-1}(1)
\end{aligned} \tag{58}
$$

From Eqs (43) and (45), it is noted that

$$\Delta B_s(t) = \lambda d_s C_s \left[\hat{y}_s(t) - C_s^T B_s(t)\right] \tag{59}$$

and $\hat{y}_s(t) - C_s^T B_s(t)$ is a scalar, using $u_s(t)$ to denotes it gives

$$\Delta B_s(t) = \lambda d_s u_s(t) C_s \tag{60}$$

From Eqs (57) and (58), each term $\mathbf{H}_s DB_s(0)$ is given as Eq (61).

$$\mathbf{H}_s(t)\cdot DB_s(0)=\mathbf{H}_s(t)\left[\Delta B_s(0)+\Delta B_{s+1}(0)+\ldots+\Delta B_{N_S}(0)+\Delta B_1(1)+\ldots+\Delta B_{s-1}(1)\right] \quad (61)$$

Substituting Eq (60) into Eq (61) gives

$$\begin{aligned}\mathbf{H}_s(t)\cdot DB_s(0)&=\lambda\mathbf{H}_s(t)\begin{bmatrix}d_s u_s(0)C_s+d_{s+1}u_{s+1}(0)C_{s+1}+\ldots+d_{N_S}u_{N_S}(0)C_{N_S}\\+d_1u_1(1)C_1+\ldots+d_{s-1}u_{s-1}(1)C_{s-1}\end{bmatrix}\\&=\lambda\mathbf{H}_s(t)\left[\left(\sum_{i=s}^{N_S}d_i u_i(0)C_i\right)+\left(\sum_{i=1}^{s-1}d_i u_i(1)C_i\right)\right]\end{aligned} \quad (62)$$

If $\lim_{t\to\infty}\mathbf{H}_s(t)\cdot C_s$ equals zero for all s, the Eq in (62) will be null. This makes $D\mathbf{B}(t)$ in Eq (57) a null matrix and the learning converge. The following derivation will prove the convergence by proving $\lim_{t\to\infty}\mathbf{H}_s(t)\cdot C_s$ equals zero.

Lemma 1: The matrix $\mathbf{E}_s(t)$ defined in (49) has the following properties:

Properties 1: $\mathbf{E}_s(t)$ is a symmetric square matrix with N_R elements in both dimensions.

Proof: $\mathbf{E}_s(t)^T = \left(I - \lambda d_s C_s C_s^T\right)^T = \left(I - \lambda d_s C_s C_s^T\right)$

Properties 2: Let $e_{ii,s}$'s be the diagonal elements of the matrix $\mathbf{E}_s(t)$, $e_{ij,s}$'s be the non-diagonal elements of the matrix $\mathbf{E}_s(t)$ (the element of ith row, jth column of matrix $\mathbf{E}_s(t)$), $f_{i,s}$'s be the elements of the vector C_s and $g_{ij,s}$'s be the elements of the resultant matrix of $C_s \cdot C_s^T$, then

$$f_{i,s} = \frac{\alpha_s^{(i)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \leq 1 \tag{63}$$

$$\begin{aligned} g_{ij,s} &= f_{i,s} f_{j,s} \\ &= \frac{\alpha_s^{(i)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \times \frac{\alpha_s^{(j)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \leq 1 \end{aligned} \tag{64}$$

$$\begin{aligned} d_s g_{ij,s} &= d_s f_{i,s} f_{j,s} \\ &= \sum_{k=1}^{N_R} \alpha_s^{(k)} \times \frac{\alpha_s^{(i)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \times \frac{\alpha_s^{(j)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \\ &= \alpha_s^{(i)} \times \frac{\alpha_s^{(j)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \leq 1 \end{aligned} \tag{65}$$

From Eq (49), $e_{ii,s}$ and $e_{ij,s}$ can now be derived as Eqs (66) and (67).

$$e_{ii,s} = \begin{cases} 1 & if\ \alpha_s^{(i)} = 0 \\ 1 - \lambda d_s g_{ii,s} & if\ \alpha_s^{(i)} > 0 \end{cases} \tag{66}$$

$$e_{ij,s} = \begin{cases} 0 & if\ \alpha_s^{(i)} \alpha_s^{(j)} = 0 \\ 0 - \lambda d_s g_{ij,s} & if\ \alpha_s^{(i)} \alpha_s^{(j)} > 0 \end{cases} \tag{67}$$

Hence, matrix $\mathbf{E}_s(t)$ is also a Hermitian matrix. All Hermitian matrices are normal and have real eigenvalues.

Elements in matrix $\mathbf{E}_s(t)$ form three special terms. These terms will be derived as *properties 3*, *4* and *5* and these properties will be used in the subsequent derivations.

Properties 3:

$$\sum_{j=1}^{N_R} e_{jm,s} e_{jn,s} = \frac{\lambda \alpha_s^{(m)} \alpha_s^{(n)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \left[\lambda \frac{\sum_{j=1}^{N_R} \left(\alpha_s^{(j)}\right)^2}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} - 2 \right] \tag{68}$$

Proof:

$$\begin{aligned}
&\sum_{j=1}^{N_R} e_{jm,s} e_{jn,s} \\
&= \left[\sum_{j=1, j\neq m, j\neq n}^{N_R} \left(-\lambda d_s g_{jm,s}\right)\left(-\lambda d_s g_{jn,s}\right) \right] + \left(1-\lambda d_s g_{mm,s}\right)\left(-\lambda d_s g_{mn,s}\right) \\
&\qquad + \left(-\lambda d_s g_{nm,s}\right)\left(1-\lambda d_s g_{nn,s}\right) \\
&= \left[\left(\lambda d_s\right)^2 \sum_{j=1, j\neq m, j\neq n}^{N_R} g_{jm,s} g_{jn,s} \right] + \left(\lambda d_s\right)^2 \left[g_{mm,s} g_{mn,s} + g_{nm,s} g_{nn,s} \right] \\
&\qquad - \left(\lambda d_s\right)\left[g_{mn,s} + g_{nm,s} \right] \\
&= \left[\left(\lambda d_s\right)^2 \sum_{j=1}^{N_R} g_{jm,s} g_{jn,s} \right] - \left(\lambda d_s\right)\left[g_{mn,s} + g_{nm,s} \right]
\end{aligned} \tag{69}$$

From Eqs (64) and (65), Eq (69) can be derived as Eq (70).

$$\begin{aligned}
&\left[\left(\lambda d_s\right)^2 \sum_{j=1}^{N_R} g_{jm,s} g_{jn,s} \right] - \left(\lambda d_s\right)\left[g_{mn,s} + g_{nm,s} \right] \\
&= \left[\lambda^2 \sum_{j=1}^{N_R} \frac{\alpha_s^{(j)} \alpha_s^{(m)} \alpha_s^{(j)} \alpha_s^{(n)}}{\left(\sum_{k=1}^{N_R} \alpha_s^{(k)} \right)^2} \right] - \lambda \frac{\alpha_s^{(m)} \alpha_s^{(n)} + \alpha_s^{(n)} \alpha_s^{(m)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \\
&= \frac{\lambda \alpha_s^{(m)} \alpha_s^{(n)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \left[\lambda \frac{\sum_{j=1}^{N_R} \left(\alpha_s^{(j)}\right)^2}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} - 2 \right]
\end{aligned} \tag{70}$$

Q.E.D.

Properties 4:

$$\sum_{j=1}^{N_R} e_{ji,s}^2 = 1 + \frac{\lambda\left(\alpha_s^{(i)}\right)^2}{\sum_{k=1}^{N_R}\alpha_s^{(k)}}\left[\lambda\frac{\sum_{j=1}^{N_R}\left(\alpha_s^{(j)}\right)^2}{\sum_{k=1}^{N_R}\alpha_s^{(k)}} - 2\right] \tag{71}$$

Proof:

$$\begin{aligned}
&\sum_{j=1}^{N_R} e_{ji,s}^2 \\
&= \left[\sum_{\substack{j=1\\ j\neq i}}^{N_R}\left(-\lambda d_s g_{ji,s}\right)^2\right] + \left(1-\lambda d_s g_{ii,s}\right)^2 \\
&= \left[\sum_{\substack{j=1\\ j\neq i}}^{N_R}\left(-\lambda d_s g_{ji,s}\right)^2\right] + 1 - 2\lambda d_s g_{ii,s} + \left(\lambda d_s g_{ii,s}\right)^2 \\
&= 1 + \left[\sum_{j=1}^{N_R}\left(-\lambda d_s g_{ji,s}\right)^2\right] - 2\lambda d_s g_{ii,s}
\end{aligned} \tag{72}$$

From Eqs (64) and (65), Eq (72) can be derived as Eq (73).

$$\begin{aligned}
&1 + \left[\sum_{j=1}^{N_R}\left(-\lambda d_s g_{ji,s}\right)^2\right] - 2\lambda d_s g_{ii,s} \\
&= 1 + \left[\sum_{j=1}^{N_R}\lambda^2\left(\frac{\alpha_s^{(j)}\alpha_s^{(i)}}{\sum_{k=1}^{N_R}\alpha_s^{(k)}}\right)^2\right] - 2\lambda\frac{\left(\alpha_s^{(i)}\right)^2}{\sum_{k=1}^{N_R}\alpha_s^{(k)}} \\
&= 1 + \frac{\lambda\left(\alpha_s^{(i)}\right)^2}{\sum_{k=1}^{N_R}\alpha_s^{(k)}}\left[\lambda\frac{\sum_{j=1}^{N_R}\left(\alpha_s^{(j)}\right)^2}{\sum_{k=1}^{N_R}\alpha_s^{(k)}} - 2\right]
\end{aligned} \tag{73}$$

Q.E.D.

Properties 5:

$$\left[\lambda \frac{\sum_{j=1}^{N_R} \left(\alpha_s^{(j)}\right)^2}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} - 2 \right] < 0 \qquad if\ \lambda < 2 \tag{74}$$

Proof: If the degree of matching, $\alpha \in [0,1]$, then

$$\frac{\sum_{j=1}^{N_R} \left(\alpha_s^{(j)}\right)^2}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \in [0,1] \tag{75}$$

If $\lambda < 2$, then the special term mentioned above will be smaller than zero.

Lemma 2: If the learning rate, $\lambda \in [0,2)$; for any matrix $\mathbf{Q}_s(t) = \mathbf{E}_s(t) \cdot \mathbf{E}_{s-1}(t)$, the norm of row vector i in $\mathbf{Q}_s(t)$ will not be greater than that of row vector i in $\mathbf{E}_{s-1}(t)$.

Proof: Let $\mathbf{E}_s(t)$, $\mathbf{E}_{s-1}(t)$ and $\mathbf{Q}_s(t)$ be defined as

$$\mathbf{E}_s(t) = \begin{bmatrix} e_{11,s} & e_{12,s} & \cdots & e_{1N_R,s} \\ e_{21,s} & e_{22,s} & \cdots & e_{2N_R,s} \\ \vdots & & & \vdots \\ e_{N_R1,s} & e_{N_R2,s} & \cdots & e_{N_RN_R,s} \end{bmatrix} \tag{76}$$

$$\mathbf{E}_{s-1}(t) = \begin{bmatrix} e_{11,s-1} & e_{12,s-1} & \cdots & e_{1N_R,s-1} \\ e_{21,s-1} & e_{22,s-1} & \cdots & e_{2N_R,s-1} \\ \vdots & & & \vdots \\ e_{N_R1,s-1} & e_{N_R2,s-1} & \cdots & e_{N_RN_R,s-1} \end{bmatrix} \tag{77}$$

$$\mathbf{Q}_s(t) = \begin{bmatrix} \sum_{j=1}^{N_R} e_{1j,s}e_{j1,s-1} & \sum_{j=1}^{N_R} e_{1j,s}e_{j2,s-1} & \cdots & \sum_{j=1}^{N_R} e_{1j,s}e_{jN_R,s-1} \\ \sum_{j=1}^{N_R} e_{2j,s}e_{j1,s-1} & \sum_{j=1}^{N_R} e_{2j,s}e_{j2,s-1} & \cdots & \sum_{j=1}^{N_R} e_{2j,s}e_{jN_R,s-1} \\ \vdots & & & \vdots \\ \sum_{j=1}^{N_R} e_{N_Rj,s}e_{j1,s-1} & \sum_{j=1}^{N_R} e_{N_Rj,s}e_{j2,s-1} & \cdots & \sum_{j=1}^{N_R} e_{N_Rj,s}e_{jN_R,s-1} \end{bmatrix} \tag{78}$$

The norm of row vector i in $\mathbf{E}_s(t)$ can be derived as

$$\|row\ i\ of\ \mathbf{E}_s(t)\|^2 = \sum_{j=1}^{N_R} e_{ij,s}^2 \tag{79}$$

Let $\mathbf{Q}_s(t) = \mathbf{E}_s(t) \cdot \mathbf{E}_{s-1}(t)$. The norm can be derived as

$$
\begin{aligned}
&\left\| row\ i\ of\ \mathbf{Q}_s(t) \right\|^2 \\
&= \sum_{j=1}^{N_R} q_{ji,s}^2 \\
&= \left(\sum_{j=1}^{N_R} e_{ij,s} e_{j1,s-1} \right)^2 + \left(\sum_{j=1}^{N_R} e_{ij,s} e_{j2,s-1} \right)^2 + \left(\sum_{j=1}^{N_R} e_{ij,s} e_{j3,s-1} \right)^2 + \cdots + \left(\sum_{j=1}^{N_R} e_{ij,s} e_{jN_R,s-1} \right)^2 \\
&= e_{i1,s}^2 e_{11,s-1}^2 + e_{i2,s}^2 e_{21,s-1}^2 + \cdots + e_{iN_R,s}^2 e_{N_R 1,s-1}^2 \\
&\quad + 2 \left(\begin{array}{l} e_{i1,s} e_{11,s-1} e_{i2,s} e_{21,s-1} + e_{i1,s} e_{11,s-1} e_{i3,s} e_{31,s-1} + \cdots + e_{i1,s} e_{11,s-1} e_{iN_R,s} e_{N_R 1,s-1} \\ \qquad + e_{i2,s} e_{21,s-1} e_{i3,s} e_{31,s-1} + \cdots + e_{i2,s} e_{21,s-1} e_{iN_R,s} e_{N_R 1,s-1} \\ \qquad\qquad \vdots \\ \qquad\qquad + e_{i(N_R-1),s} e_{(N_R-1)1,s-1} e_{iN_R,s} e_{N_R 1,s-1} \end{array} \right) \\
&\quad + e_{i1,s}^2 e_{12,s-1}^2 + e_{i2,s}^2 e_{22,s-1}^2 + \cdots + e_{iN_R,s}^2 e_{N_R 2,s-1}^2 \\
&\quad + 2 \left(\begin{array}{l} e_{i1,s} e_{2i,s-1} e_{i2,s} e_{22,s-1} + e_{i1,s} e_{2i,s-1} e_{i3,s} e_{32,s-1} + \cdots + e_{i1,s} e_{12,s-1} e_{iN_R,s} e_{N_R 2,s-1} \\ \qquad + e_{i2,s} e_{22,s-1} e_{i3,s} e_{32,s-1} + \cdots + e_{i2,s} e_{22,s-1} e_{iN_R,s} e_{N_R 2,s-1} \\ \qquad\qquad \vdots \\ \qquad\qquad + e_{i(N_R-1),s} e_{(N_R-1)2,s-1} e_{iN_R,s} e_{N_R 2,s-1} \end{array} \right) \\
&\quad \vdots \\
&\quad + e_{i1,s}^2 e_{1N_R,s-1}^2 + e_{i2,s}^2 e_{2N_R,s-1}^2 + \cdots + e_{iN_R,s}^2 e_{N_R N_R,s-1}^2 \\
&\quad + 2 \left(\begin{array}{l} e_{i1,s} e_{1N_R,s-1} e_{i2,s} e_{2N_R,s-1} + e_{i1,s} e_{1N_R,s-1} e_{i3,s} e_{3N_R,s-1} + \cdots + e_{i1,s} e_{1N_R,s-1} e_{iN_R,s} e_{N_R N_R,s-1} \\ \qquad + e_{i2,s} e_{2N_R,s-1} e_{i3,s} e_{3N_R,s-1} + \cdots + e_{i2,s} e_{2N_R,s-1} e_{iN_R,s} e_{N_R N_R,s-1} \\ \qquad\qquad \vdots \\ \qquad\qquad + e_{i(N_R-1),s} e_{(N_R-1)N_R,s-1} e_{iN_R,s} e_{N_R N_R,s-1} \end{array} \right) \\
&= e_{i1,s}^2 \left(\sum_{j=1}^{N_R} e_{1j,s-1}^2 \right) + e_{i2,s}^2 \left(\sum_{j=1}^{N_R} e_{2j,s-1}^2 \right) + \ldots + e_{iN_R,s}^2 \left(\sum_{j=1}^{N_R} e_{N_R j,s-1}^2 \right) \\
&\quad + 2 \left[\begin{array}{l} e_{i1,s} e_{i2,s} \left(e_{11,s-1} e_{21,s-1} + e_{12,s-1} e_{22,s-1} + \cdots + e_{1N_R,s-1} e_{2N_R,s-1} \right) \\ + e_{i1,s} e_{i3,s} \left(e_{11,s-1} e_{31,s-1} + e_{12,s-1} e_{32,s-1} + \cdots + e_{1N_R,s-1} e_{3N_R,s-1} \right) \\ \qquad + \cdots + e_{i1,s} e_{iN_R,s} \left(e_{11,s-1} e_{N_R 1,s-1} + e_{12,s-1} e_{N_R 2,s-1} + \cdots + e_{1N_R,s-1} e_{N_R N_R,s-1} \right) \\ + e_{i2,s} e_{i3,s} \left(e_{21,s-1} e_{31,s-1} + e_{22,s-1} e_{32,s-1} + \cdots + e_{2N_R,s-1} e_{3N_R,s-1} \right) \\ \qquad + \cdots + e_{i2,s} e_{iN_R,s} \left(e_{21,s-1} e_{N_R 1,s-1} + e_{22,s-1} e_{N_R 2,s-1} + \cdots + e_{2N_R,s-1} e_{N_R N_R,s-1} \right) \\ \qquad\qquad \vdots \\ + e_{i(N_R-1),s} e_{iN_R,s} \left(e_{(N_R-1)1,s-1} e_{N_R 1,s-1} + e_{(N_R-1)2,s-1} e_{N_R 2,s-1} + \cdots + e_{(N_R-1)N_R,s-1} e_{N_R N_R,s-1} \right) \end{array} \right]
\end{aligned}
\tag{80}
$$

$$= \left[\sum_{l=1}^{N_R} e_{il,s}^2 \left(\sum_{j=1}^{N_R} e_{lj,s-1}^2 \right) \right] + \left\{ 2 \sum_{m=1}^{N_R} \left[e_{im,s} \sum_{n=m+1}^{N_R} \left(e_{in,s} \sum_{j=1}^{N_R} e_{mj,s-1} e_{nj,s-1} \right) \right] \right\}$$

According to *Properties 3* and *Properties 4* of *Lemma 1*, the Eq (80) can be expanded as

$$\begin{aligned}
&\left[\sum_{l=1}^{N_R} e_{il,s}^2 \left(\sum_{j=1}^{N_R} e_{lj,s-1}^2 \right) \right] + \left\{ 2 \sum_{m=1}^{N_R} \left[e_{im,s} \sum_{n=m+1}^{N_R} \left(e_{in,s} \sum_{j=1}^{N_R} e_{mj,s-1} e_{nj,s-1} \right) \right] \right\} \\
&= \left(\sum_{l=1}^{N_R} e_{il,s}^2 \left\{ 1 + \lambda \frac{\left(\alpha_{s-1}^{(l)} \right)^2}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} \left[\lambda \frac{\sum_{j=1}^{N_R} \left(\alpha_{s-1}^{(j)} \right)^2}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} - 2 \right] \right\} \right) \\
&\quad + 2 \sum_{m=1}^{N_R} \left(e_{im,s} \sum_{n=m+1}^{N_R} e_{in,s} \left\{ \frac{\lambda \alpha_{s-1}^{(m)} \alpha_{s-1}^{(n)}}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} \left[\lambda \frac{\sum_{j=1}^{N_R} \left(\alpha_{s-1}^{(j)} \right)^2}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} - 2 \right] \right\} \right) \\
&= \left(\sum_{l=1}^{N_R} e_{il,s}^2 \right) + \left(\left\{ \frac{\lambda}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} \left[\lambda \frac{\sum_{j=1}^{N_R} \left(\alpha_{s-1}^{(j)} \right)^2}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} - 2 \right] \right\} \sum_{l=1}^{N_R} \left(\alpha_{s-1}^{(l)} e_{il,s} \right)^2 \right) \\
&\quad + \frac{2\lambda}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} \left[\lambda \frac{\sum_{j=1}^{N_R} \left(\alpha_{s-1}^{(j)} \right)^2}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} - 2 \right] \sum_{m=1}^{N_R} \left(\alpha_{s-1}^{(m)} e_{im,s} \sum_{n=m+1}^{N_R} \alpha_{s-1}^{(n)} e_{in,s} \right) \\
&= \left\| row\ i\ of\ \mathbf{E}_s(t) \right\|^2 \\
&\quad + \frac{\lambda}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} \left[\lambda \frac{\sum_{j=1}^{N_R} \left(\alpha_{s-1}^{(j)} \right)^2}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} - 2 \right] \left\{ \left[\sum_{l=1}^{N_R} \left(\alpha_{s-1}^{(l)} e_{il,s} \right)^2 \right] + 2 \sum_{m=1}^{N_R} \left(\alpha_{s-1}^{(m)} e_{im,s} \sum_{n=m+1}^{N_R} \alpha_{s-1}^{(n)} e_{in,s} \right) \right\} \\
&= \left\| row\ i\ of\ \mathbf{E}_s(t) \right\|^2 + \frac{\lambda}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} \left(\sum_{l=1}^{N_R} \alpha_{s-1}^{(l)} e_{il,s} \right)^2 \left[\lambda \frac{\sum_{j=1}^{N_R} \left(\alpha_{s-1}^{(j)} \right)^2}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} - 2 \right]
\end{aligned} \tag{81}$$

It is known that the degree of matching, $\alpha \in [0,1]$ and the learning parameter, $\lambda \in [0,1]$. Hence, the following term will be

$$\frac{\lambda}{\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}} \left(\sum_{l=1}^{N_R} \alpha_{s-1}^{(l)} e_{il,s} \right)^2 \geq 0$$

According to *Properties 5* of *Lemma 1*,

$$\left[\lambda \frac{\sum_{j=1}^{N_R} \left(\alpha_s^{(j)} \right)^2}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} - 2 \right] < 0 \qquad if\ \lambda < 2$$

We conclude that if the learning rate, $\lambda \in [0,2)$, $\|row\ i\ of\ \mathbf{Q}_s(t)\|^2 \leq \|row\ i\ of\ \mathbf{E}_{s-1}(t)\|^2$ for all i.

As shown in Eq (53), $\mathbf{G}_s(t)$ is defined as the product of $\mathbf{E}$'s and according to *Lemma 2*, the multiplication of any vector by $\mathbf{G}_s(t)$ will result in a vector with a smaller or equal norm. This induces that $\mathbf{G}_s(t)$ has no eigenvalue with its absolute value greater than one.

Lemma 3: $\lim_{t\to\infty}\|row\ i\ of\ \mathbf{H}_s(t+1)\|$ will be bounded and equal to $\lim_{t\to\infty}\|row\ i\ of\ \mathbf{H}_s(t)\|$

Proof: $\mathbf{G}_s(t)$ is a product of $\mathbf{E}$'s as shown in Eq (53). From *Lemma 2*,

$$\begin{aligned}
&\|row\ i\ of\ \mathbf{H}_s(t)\| \\
&\quad\geq \|row\ i\ of\ \mathbf{H}_s(t)\cdot\mathbf{E}_{s-1}(t+1)\| \\
&\quad\geq \|row\ i\ of\ \mathbf{H}_s(t)\cdot\mathbf{E}_{s-1}(t+1)\cdot\mathbf{E}_{s-2}(t+1)\| \\
&\quad\vdots \\
&\quad\geq \left\|\begin{array}{r} row\ i\ of\ \mathbf{H}_s(t-1)\cdot\mathbf{E}_{s-1}(t+1)\cdot\mathbf{E}_{s-2}(t+1)...\mathbf{E}_1(t+1) \\ \cdot\mathbf{E}_{N_S}(t+1)\cdot\mathbf{E}_{N_S-1}(t+1)...\mathbf{E}_s(t+1) \end{array}\right\| \\
&\quad= \|row\ i\ of\ \mathbf{H}_s(t)\cdot\mathbf{G}_s(t+1)\|
\end{aligned} \tag{82}$$

From Eq (56),

$$\|row\ i\ of\ \mathbf{H}_s(t)\| \geq \|row\ i\ of\ \mathbf{H}_s(t+1)\| \tag{83}$$

Eq (83) shows that when t increases, the norm of any row in $\mathbf{H}_s(t)$ will never increase and is bounded. Due to the fact that norm must be greater or equal to zero, we conclude that

$$\lim_{t\to\infty}\|row\ i\ of\ \mathbf{H}_s(t+1)\| = \lim_{t\to\infty}\|row\ i\ of\ \mathbf{H}_s(t)\| \tag{84}$$

Lemma 4: if $\lim_{t\to\infty}\|row\ i\ of\ \mathbf{H}_s(t)\cdot\mathbf{E}_{s-1}(t+1)\| = \lim_{t\to\infty}\|row\ i\ of\ \mathbf{H}_s(t)\|$, then $\sum_{l=1}^{N_R}\alpha_{s-1}^{(l)}h_{il,s}=0$.

Proof: According to Eq (82), equality $\lim_{t\to\infty}\|row\ i\ \mathbf{H}_s(t)\cdot\mathbf{E}_{s-1}(t+1)\| = \lim_{t\to\infty}\|row\ i\ of\ \mathbf{H}_s(t)\|$ can be derived. Referring to Eq (80), by replacing $e_{ij,s}$ with $h_{ij,s}$, this equality can be further expanded as

$$
\begin{aligned}
&\lim_{t\to\infty}\|row\ i\ \mathbf{H}_s(t)\cdot\mathbf{E}_{s-1}(t+1)\| = \lim_{t\to\infty}\|row\ i\ of\ \mathbf{H}_s(t)\| \\
&\Leftrightarrow \lim_{t\to\infty}\|row\ i\ \mathbf{H}_s(t)\cdot\mathbf{E}_{s-1}(t+1)\|^2 = \lim_{t\to\infty}\|row\ i\ of\ \mathbf{H}_s(t)\|^2 \\
&\Leftrightarrow \left[\sum_{l=1}^{N_R}h_{il,s}^2\left(\sum_{j=1}^{N_R}e_{lj,s-1}^2\right)\right]+\left\{2\sum_{m=1}^{N_R}\left[h_{im,s}\sum_{n=m+1}^{N_R}\left(h_{in,s}\sum_{j=1}^{N_R}e_{mj,s-1}e_{nj,s-1}\right)\right]\right\}=\sum_{l=1}^{N_R}h_{il,s}^2 \\
&\Leftrightarrow \left[\begin{array}{l}\left(\sum_{l=1}^{N_R}h_{il,s}^2\left\{1+\lambda\frac{\left(\alpha_{s-1}^{(l)}\right)^2}{\sum_{k=1}^{N_R}\alpha_{s-1}^{(k)}}\left[\lambda\frac{\sum_{j=1}^{N_R}\left(\alpha_{s-1}^{(j)}\right)^2}{\sum_{k=1}^{N_R}\alpha_{s-1}^{(k)}}-2\right]\right\}\right) \\ +2\sum_{m=1}^{N_R}\left(h_{im,s}\sum_{n=m+1}^{N_R}\left\{h_{in,s}\lambda\frac{\alpha_{s-1}^{(m)}\alpha_{s-1}^{(n)}}{\sum_{k=1}^{N_R}\alpha_{s-1}^{(k)}}\left[\lambda\frac{\sum_{j=1}^{N_R}\left(\alpha_{s-1}^{(j)}\right)^2}{\sum_{k=1}^{N_R}\alpha_{s-1}^{(k)}}-2\right]\right\}\right)\end{array}\right]=\sum_{l=1}^{N_R}h_{il,s}^2 \\
&\Leftrightarrow \left[\begin{array}{l}\left\{\sum_{l=1}^{N_R}h_{il,s}^2\lambda\frac{\left(\alpha_{s-1}^{(l)}\right)^2}{\sum_{k=1}^{N_R}\alpha_{s-1}^{(k)}}\left[\lambda\frac{\sum_{j=1}^{N_R}\left(\alpha_{s-1}^{(j)}\right)^2}{\sum_{k=1}^{N_R}\alpha_{s-1}^{(k)}}-2\right]\right\} \\ +2\sum_{m=1}^{N_R}\left(h_{im,s}\sum_{n=m+1}^{N_R}\left\{h_{in,s}\lambda\frac{\alpha_{s-1}^{(m)}\alpha_{s-1}^{(n)}}{\sum_{k=1}^{N_R}\alpha_{s-1}^{(k)}}\left[\lambda\frac{\sum_{j=1}^{N_R}\left(\alpha_{s-1}^{(j)}\right)^2}{\sum_{k=1}^{N_R}\alpha_{s-1}^{(k)}}-2\right]\right\}\right)\end{array}\right]=0 \\
&\Leftrightarrow \left[\sum_{l=1}^{N_R}\left(\alpha_{s-1}^{(l)}h_{il,s}\right)^2\right]+\left\{2\sum_{m=1}^{N_R}\left[\alpha_{s-1}^{(m)}h_{im,s}\sum_{n=m+1}^{N_R}\left(\alpha_{s-1}^{(n)}h_{in,s}\right)\right]\right\}=0
\end{aligned}
\tag{85}
$$

$$\Leftrightarrow \left[\begin{array}{l} \alpha_{s-1}^2 h_{i1,s}^2 + \alpha_{s-1}^2 h_{i2,s}^2 + \cdots + \alpha_{s-1}^2 h_{iN_R,s}^2 \\ +2 \left(\begin{array}{r} \alpha_{s-1}^{(1)} h_{i1,s} \alpha_{s-1}^{(2)} h_{i2,s} + \alpha_{s-1}^{(1)} h_{i1,s} \alpha_{s-1}^{(3)} h_{i3,s} + \cdots + \alpha_{s-1}^{(1)} h_{i1,s} \alpha_{s-1}^{(N_R)} h_{iN_R,s} \\ + \alpha_{s-1}^{(2)} h_{i2,s} \alpha_{s-1}^{(3)} h_{i3,s} + \cdots + \alpha_{s-1}^{(2)} h_{i2,s} \alpha_{s-1}^{(N_R)} h_{iN_R,s} \\ \vdots \\ + \alpha_{s-1}^{(N_R-1)} h_{i(N_R-1),s} \alpha_{s-1}^{(N_R)} h_{iN_R,s} \end{array} \right) \end{array} \right] = 0$$

$$\Leftrightarrow \left(\sum_{l=1}^{N_R} \alpha_{s-1}^{(l)} h_{il,s} \right)^2 = 0$$

$$\Leftrightarrow \sum_{l=1}^{N_R} \alpha_{s-1}^{(l)} h_{il,s} = 0$$

Q.E.D.

Lemma 5: $\lim_{t\to\infty} \mathbf{H}_s(t+1) = \mathbf{H}_s(t)$ and $\mathbf{H}_s(t)$ converges to a null or constant matrix as t approaches infinity.

Proof: To prove this, $\mathbf{H}_s(t)\mathbf{E}_{s-1}(t+1)$ is first examined and shown in Eq (86).

$$\mathbf{H}_s(t)\mathbf{E}_{s-1}(t+1) = \begin{bmatrix} h_{11,s} & h_{12,s} & \cdots & h_{1N_R,s} \\ h_{21,s} & h_{22,s} & \cdots & h_{2N_R,s} \\ \vdots & & & \vdots \\ h_{N_R1,s} & h_{N_R2,s} & \cdots & h_{N_RN_R,s} \end{bmatrix} \times \begin{bmatrix} e_{11,s-1} & e_{12,s-1} & \cdots & e_{1N_R,s-1} \\ e_{21,s-1} & e_{22,s-1} & \cdots & e_{2N_R,s-1} \\ \vdots & & & \vdots \\ e_{N_R1,s-1} & e_{N_R2,s-1} & \cdots & e_{N_RN_R,s-1} \end{bmatrix} \tag{86}$$

The n th row of $\mathbf{H}_s(t)\mathbf{E}_{s-1}(t+1)$ can be expressed as

$$\begin{bmatrix} \sum_{k=1}^{N_R} h_{nk,s}e_{k1,s-1} & \sum_{k=1}^{N_R} h_{nk,s}e_{k2,s-1} & \cdots & \sum_{k=1}^{N_R} h_{nk,s}e_{kN_R,s-1} \end{bmatrix} \tag{87}$$

And the m th element of the n th row of $\mathbf{H}_s(t)\mathbf{E}_{s-1}(t+1)$ can be expressed as

$$\begin{aligned} &\sum_{k=1}^{N_R} h_{nk,s}e_{km,s-1} \\ &= h_{nm,s}e_{mm,s-1} + \sum_{\substack{k=1 \\ k\neq m}}^{N_R} h_{nk,s}e_{km,s-1} \\ &= h_{nm,s}\left[1-\lambda d_{s-1}g_{mm,s-1}\right] + \sum_{\substack{k=1 \\ k\neq m}}^{N_R} h_{nk,s}\left[-\lambda d_{s-1}g_{km,s-1}\right] \\ &= h_{nm,s} - \sum_{k=1}^{N_R} h_{nk,s}\left[\lambda d_{s-1}g_{km,s-1}\right] \\ &= h_{nm,s} - \frac{\lambda\alpha_{s-1}^{(m)}}{\sum_{k=1}^{N_R}\alpha_{s-1}^{(k)}} \sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}h_{nk,s} \end{aligned} \tag{88}$$

According to *Lemma 4*, $\sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}h_{nk,s} = 0$ as t approaches infinity. Eq (88) can be derived as

$$\sum_{k=1}^{N_R} h_{nk,s}e_{km,s-1} = h_{nm,s} - \frac{\lambda\alpha_{s-1}^{(m)}}{\sum_{k=1}^{N_R}\alpha_{s-1}^{(k)}} \sum_{k=1}^{N_R} \alpha_{s-1}^{(k)}h_{nk,s} = h_{nm,s} \tag{89}$$

As a result, the m th element of the n th row of $\mathbf{H}_s(t)\mathbf{E}_{s-1}(t+1)$ is equal to that of $\mathbf{H}_s(t)$ as t approaches infinity.

The investigation for $\mathbf{H}_s(t)\cdot\mathbf{E}_{s-1}(t+1)\cdot\mathbf{E}_{s-2}(t+1)=\mathbf{H}_s(t)\cdot\mathbf{E}_{s-1}(t+1)$ as t approaches infinity can be shown by similar proving, and, by continuing this procedure, we can conclude that

$$\lim_{T\to\infty}\mathbf{H}_s(t)\cdot\mathbf{G}_s(t+1)=\lim_{T\to\infty}\mathbf{H}_s(t) \tag{90}$$

And $\lim_{T\to\infty}\mathbf{H}_s(t)$ exists and equal to a null or constant matrix.

Lemma 6: $\lim_{t\to\infty} \mathbf{H}_s(t)\cdot C_s(t)$ equals to a null vector for all s.

Proof: $\mathbf{H}_s(t)\cdot C_s(t)$ can be expressed as

$$
\begin{aligned}
&\mathbf{H}_s(t)\cdot C_s(t) \\
&= \begin{bmatrix} h_{11,s} & h_{12,s} & \cdots & h_{1N_R,s} \\ h_{21,s} & h_{22,s} & \cdots & h_{2N_R,s} \\ \vdots & & & \vdots \\ h_{N_R1,s} & h_{N_R2,s} & \cdots & h_{N_RN_R,s} \end{bmatrix} \times \left(\frac{1}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \right) \begin{bmatrix} \alpha_s^{(1)} \\ \alpha_s^{(2)} \\ \vdots \\ \alpha_s^{(N_R)} \end{bmatrix} \\
&= \left(\frac{1}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \right) \left[\sum_{k=1}^{N_R} \alpha_s^{(k)} h_{1k,s} \quad \sum_{k=1}^{N_R} \alpha_s^{(k)} h_{2k,s} \quad \cdots \quad \sum_{k=1}^{N_R} \alpha_s^{(k)} h_{N_Rk,s} \right]^T \\
&= \left(\frac{1}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \right) \left[0 \quad 0 \quad \cdots \quad 0 \right]^T
\end{aligned}
\tag{91}
$$

Thus, $\mathbf{H}_s(t)\cdot DB_s(0)$ in Eq (62), and $D\mathbf{B}(t)$ in Eq (57) will be null as t approaches infinity and the learning of TSK^0-FCMAC converges.

4.4. Summary

This chapter presents the study on the convergence characteristic of the proposed architecture. The convergence of TSK^0-FCMAC has been proven to hold if the learning parameter, λ, is between zero and two. The mathematical formulation in this study provides a strong foundation for further investigation on the convergence property of fuzzy inference systems with similar characteristic.

Chapter 5 TSK[1]-FCMAC: Architecture and Learning Algorithm

5.1. Introduction

The role of computer science is two-fold: first, in training, we need efficient algorithms to solve the optimization problem, as well as to store and process the massive amount of data we generally have. Second, once a model is learned, its representation and algorithmic solution for inference needs to be efficient as well. In certain applications, the efficiency of the learning or inference algorithm, namely, its space and time complexity, may be as important as its predictive accuracy [5]. The *Principle of incompatibility* suggested by Professor Lotfi A. Zadeh presents a genuine synopsis on the dilemma between precision and cost. The *Principle of incompatibility* stated informally that: "*as the complexity of a system increases, our ability to make precise yet significant descriptions about its behavior diminishes until a threshold is reached beyond which precision and significance (or relevance) become almost mutually exclusive characteristics*" [10]. In other words, one has to pay a cost for high precision. An example often used by Professor Zadeh to illustrate this trade-off is the problem of parking a car. Usually it takes a driver less than half a minute to do a parallel parking. However, if one were asked to park a car in a parking space such that the outside wheels are within 0.01 degree from a specified angle, how long would one take to park the car? In fact, most people will probably give up. The point is that the cost (i.e. the time required to park a car) increases as the precision of the car parking task increases. This trade-off between precision and cost exists not only in car parking but also in other control, modeling, decision making, and almost any kind of problems [9].

The first proposed architecture, TSK[0]-FCMAC was initially designed to be applied in the domain of control problems. Besides achieving excellent performance in the control domain, empirical studies in Chapter 3 also show that the network is able to achieve satisfactory performance in other problem domains such as classification and regression. Nevertheless, the precision accomplished by TSK[0]-FCMAC may not be agreeable for applications where precision is emphasized. This inspires the implementation of a higher ordered Takagi-Sugeno-Kang fuzzy inference system to breach the barrier of lesser precision at the expense of higher complexity. The

product of such implementation is an extension of TSK^0-FCMAC, the TSK^1-FCMAC architecture, a localized self-organizing first-ordered Takagi-Sugeno-Kang fuzzy inference system based on the CMAC structure.

This chapter first presents the architecture and learning algorithm of the proposed network, TSK^1-FCMAC. In contrast to its predecessor, TSK^1-FCMAC is designed to give emphasis to achieving a higher performance in terms of precision. Instead of using a single crisp value, the consequent of TSK^1-FCMAC is represented by a linear function of input parameters. Therefore, computational complexity is anticipated to increase. The ability of the proposed network is then demonstrated with empirical applications to three different problems; namely: 1) sinusoidal function/surface approximation, 2) automobile MPG (miles per gallon) prediction and, 3) two-spiral problems.

5.2. Architecture and Learning Algorithm

TSK^1-FCMAC adopts the two-phase learning algorithm as described in section 3.3. The antecedents of TSK^1-FCMAC are similar to its predecessor, the TSK^0-FCMAC architecture. However, as TSK^1-FCMAC implements the first-ordered Takagi-Sugeno-Kang fuzzy inference model, the consequents are no longer represented by crisp value but a linear function of input parameters. In this section, a different learning algorithm for the consequent part is proposed and will be described in details.

Consider that the structure of the network has N_{IP} inputs $x_1, x_2, \ldots, x_{N_{IP}}$ and one output y. Assuming the fuzzy inference rule base contains N_R fuzzy if-then rules of Takagi-Sugeno-Kang's type [18, 19]. The fuzzy if-then rules $R_1, R_1, \cdots R_{N_R}$ are defined in Eq (92).

$$
\begin{aligned}
&R_1 : IF\ x_{1,s}\ is\ A_{1,j}^{(1)}\ and\ x_{2,s}\ is\ A_{2,j}^{(1)}\ and \cdots and\ x_{N_{IP},s}\ is\ A_{N_{IP},j}^{(1)},\\
&\qquad THEN\ y_s^{(1)}(t) = b_{0,s}^{(1)}(t) + b_{1,s}^{(1)} x_{1,s} + \cdots + b_{N_{IP},s}^{(1)} x_{N_{IP},s}\\
&R_2 : IF\ x_{1,s}\ is\ A_{1,j}^{(2)}\ and\ x_{2,s}\ is\ A_{2,j}^{(2)}\ and \cdots and\ x_{N_{IP},s}\ is\ A_{N_{IP},j}^{(2)},\\
&\qquad THEN\ y_s^{(2)}(t) = b_{0,s}^{(2)}(t) + b_{1,s}^{(2)} x_{1,s} + \cdots + b_{N_{IP},s}^{(2)} x_{N_{IP},s} \qquad (92)\\
&\qquad \vdots\\
&R_{N_R} : IF\ x_{1,s}\ is\ A_{1,j}^{(N_R)}\ and\ x_{2,s}\ is\ A_{2,j}^{(N_R)}\ and \cdots and\ x_{N_{IP},s}\ is\ A_{N_{IP},j}^{(N_R)},\\
&\qquad THEN\ y_s^{(N_R)}(t) = b_{0,s}^{(N_R)}(t) + b_{1,s}^{(N_R)} x_{1,s} + \cdots + b_{N_{IP},s}^{(N_R)} x_{N_{IP},s}
\end{aligned}
$$

Where,

N_R is the total number of rules;

N_{IP} is the total number of inputs;

R_k is the k th fuzzy if-then rule;

$x_{i,s}$ is the i th input of the s th sample;

$A_{i,j}^{(k)}$ is the j th fuzzy set of the i th input that is connected to R_k;

$y_s^{(k)}(t)$ is the output of k th fuzzy if-then rule at the presentation of the s th sample in the t th iteration;

$b_{j,s}^{(k)}(t)$ is the j th data content of the k th fuzzy if-then rule at the presentation of the s th sample in the t th iteration. $b_{j,s}^{(k)}(t)$ is real-valued parameter.

The total output of the model $y_s(t)$ at the presentation of the s th sample in the t th iteration is given as Eq (93).

$$
\begin{aligned}
y_s(t) &= \frac{1}{\sum_{k=1}^{N_R}\alpha_s^{(k)}}\sum_{k=1}^{N_R}\alpha_s^{(k)} f^{(k)}\left(x_{1,s}, x_{2,s}, \cdots, x_{N_{IP},s}\right) \\
&= \frac{1}{\sum_{k=1}^{N_R}\alpha_s^{(k)}}\begin{bmatrix} \alpha_s^{(1)}\left(b_{0,s}^{(1)}(t) + b_{1,s}^{(1)}x_{1,s} + \cdots + b_{N_{IP},s}^{(1)}x_{N_{IP},s}\right) \\ \quad + \alpha_s^{(2)}\left(b_{0,s}^{(2)}(t) + b_{1,s}^{(2)}x_{1,s} + \cdots + b_{N_{IP},s}^{(2)}x_{N_{IP},s}\right) \\ \quad \cdots \\ \quad + \alpha_s^{(N_R)}\left(b_{0,s}^{(N_R)}(t) + b_{1,s}^{(N_R)}x_{1,s} + \cdots + b_{N_{IP},s}^{(N_R)}x_{N_{IP},s}\right) \end{bmatrix} \\
&= \frac{1}{\sum_{k=1}^{N_R}\alpha_s^{(k)}}\begin{bmatrix} \alpha_s^{(1)} & \alpha_s^{(2)} & \cdots & \alpha_s^{(N_R)} \end{bmatrix} \times \begin{bmatrix} b_{0,s}^{(1)}(t) + b_{1,s}^{(1)}(t)x_{1,s} + \cdots + b_{N_{IP},s}^{(1)}(t)x_{N_{IP},s} \\ b_{0,s}^{(2)}(t) + b_{1,s}^{(2)}(t)x_{1,s} + \cdots + b_{N_{IP},s}^{(2)}(t)x_{N_{IP},s} \\ \cdots \\ b_{0,s}^{(N_R)}(t) + b_{1,s}^{(N_R)}(t)x_{1,s} + \cdots + b_{N_{IP},s}^{(N_R)}(t)x_{N_{IP},s} \end{bmatrix} \\
&= \frac{1}{\sum_{k=1}^{N_R}\alpha_s^{(k)}}\begin{bmatrix} \alpha_s^{(1)} & \alpha_s^{(1)}x_{1,s} & \cdots & \alpha_s^{(1)}x_{N_{IP},s} \\ & \alpha_s^{(2)} & \alpha_s^{(2)}x_{1,s} & \cdots & \alpha_s^{(2)}x_{N_{IP},s} \\ & & \cdots & & \\ & & \alpha_s^{(N_R)} & \alpha_s^{(N_R)}x_{1,s} & \cdots & \alpha_s^{(N_R)}x_{N_{IP},s} \end{bmatrix} \times \begin{bmatrix} b_{0,s}^{(1)}(t) \\ b_{1,s}^{(1)}(t) \\ \vdots \\ b_{N_{IP},s}^{(1)}(t) \\ b_{0,s}^{(2)}(t) \\ b_{1,s}^{(2)}(t) \\ \vdots \\ b_{N_{IP},s}^{(2)}(t) \\ \vdots \\ b_{0,s}^{(N_R)}(t) \\ b_{1,s}^{(N_R)}(t) \\ \vdots \\ b_{N_{IP},s}^{(N_R)}(t) \end{bmatrix}
\end{aligned}
\tag{93}
$$

Where,

$\alpha_s^{(k)}$ is the degree of matching between the k th fuzzy if-then rule and the s th sample.

Consider a batch learning process where the data content $b_{j,s}^{(k)}(t)$ is updated after all data samples are presented in the same iteration, the data content $b_{j,s}^{(k)}(t)$ will then remain unchanged in the same iteration. This can be mathematically expressed as Eq (94).

$$b_{j,1}^{(k)}(t) = b_{j,2}^{(k)}(t) = \cdots = b_{j,N_S}^{(k)}(t) = b_j^{(k)}(t) \tag{94}$$

For simplicity, define

$$\overline{\alpha_s^{(i)}} = \frac{\alpha_s^{(i)}}{\sum_{k=1}^{N_R} \alpha_s^{(k)}} \tag{95}$$

The total output of the model $y_s(t)$ can now be expressed as Eq (96).

$$y_s(t) = \begin{bmatrix} \overline{\alpha_s^{(1)}} & \overline{\alpha_s^{(1)}}x_{1,s} & \cdots & \overline{\alpha_s^{(1)}}x_{N_{IP},s} \\ & \overline{\alpha_s^{(2)}} & \overline{\alpha_s^{(2)}}x_{1,s} & \cdots & \overline{\alpha_s^{(2)}}x_{N_{IP},s} \\ & & \cdots & & \\ & & \overline{\alpha_s^{(N_R)}} & \overline{\alpha_s^{(N_R)}}x_{1,s} & \cdots & \overline{\alpha_s^{(N_R)}}x_{N_{IP},s} \end{bmatrix} \times \begin{bmatrix} b_0^{(1)}(t) \\ b_1^{(1)}(t) \\ \vdots \\ b_{N_{IP}}^{(1)}(t) \\ b_0^{(2)}(t) \\ b_1^{(2)}(t) \\ \vdots \\ b_{N_{IP}}^{(2)}(t) \\ \vdots \\ b_0^{(N_R)}(t) \\ b_1^{(N_R)}(t) \\ \vdots \\ b_{N_{IP}}^{(N_R)}(t) \end{bmatrix} \tag{96}$$

Consider there are N_S training data in the form of $\{X_s, y_{desired,s}\}$, where $X_s = \{x_{1,s}, x_{2,s}, \cdots, x_{N_{IP},s}\}$. There will be a total number of N_S linear equations to be solved simultaneously. The linear equations can be mathematically expressed as Eq (97).

$$\begin{bmatrix} y_{desired,1} \\ y_{desired,2} \\ \vdots \\ y_{desired,N_S} \end{bmatrix} = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_{N_S}(t) \end{bmatrix} + \begin{bmatrix} \varepsilon_1(t) \\ \varepsilon_2(t) \\ \vdots \\ \varepsilon_{N_S}(t) \end{bmatrix}$$

$$\begin{bmatrix} y_{desired,1} \\ y_{desired,2} \\ \vdots \\ y_{desired,N_S} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{\sum_{k=1}^{N_R} \alpha_1^{(k)}} \sum_{k=1}^{N_R} \alpha_1^{(k)} f^{(k)}\left(x_{1,1}, x_{2,1}, \cdots, x_{N_{IP},1}\right) \\ \dfrac{1}{\sum_{k=1}^{N_R} \alpha_2^{(k)}} \sum_{k=1}^{N_R} \alpha_2^{(k)} f^{(k)}\left(x_{1,2}, x_{2,2}, \cdots, x_{N_{IP},2}\right) \\ \vdots \\ \dfrac{1}{\sum_{k=1}^{N_R} \alpha_3^{(k)}} \sum_{k=1}^{N_R} \alpha_{N_S}^{(k)} f^{(k)}\left(x_{1,N_S}, x_{2,N_S}, \cdots, x_{N_{IP},N_S}\right) \end{bmatrix} + \begin{bmatrix} \varepsilon_1(t) \\ \varepsilon_2(t) \\ \vdots \\ \varepsilon_{N_S}(t) \end{bmatrix} \quad (97)$$

Where,

N_S is the total number of samples;

$X_s = \left\{x_{1,s}, x_{2,s}, \cdots, x_{N_{IP},s}\right\}$ is the input of the s th sample;

$y_{desired,s}$ is the desired output of the s th sample;

$\varepsilon_i(t)$ is the difference between $y_{desired,s}$ and $y_s(t)$ of the s th sample in the t th iteration.

Substituting Eq (96) into Eq (97) gives

$$\begin{bmatrix} y_{desired,1} \\ y_{desired,2} \\ \vdots \\ y_{desired,N_S} \end{bmatrix} = \begin{bmatrix} \overbrace{\begin{matrix} \overline{\alpha_1^{(1)}} & \overline{\alpha_1^{(1)}}x_{1,1} & \cdots & \overline{\alpha_1^{(1)}}x_{N_{IP},1} \\ \overline{\alpha_1^{(2)}} & \overline{\alpha_1^{(2)}}x_{1,1} & \cdots & \overline{\alpha_1^{(2)}}x_{N_{IP},1} \\ & & \cdots & \\ \overline{\alpha_1^{(N_R)}} & \overline{\alpha_1^{(N_R)}}x_{1,1} & \cdots & \overline{\alpha_1^{(N_R)}}x_{N_{IP},1} \end{matrix}}^{for\ y_1(t)} \\ \overbrace{\begin{matrix} \overline{\alpha_2^{(1)}} & \overline{\alpha_2^{(1)}}x_{1,2} & \cdots & \overline{\alpha_2^{(1)}}x_{N_{IP},2} \\ \overline{\alpha_2^{(2)}} & \overline{\alpha_2^{(2)}}x_{1,2} & \cdots & \overline{\alpha_2^{(2)}}x_{N_{IP},2} \\ & & \cdots & \\ \overline{\alpha_2^{(N_R)}} & \overline{\alpha_2^{(N_R)}}x_{1,2} & \cdots & \overline{\alpha_2^{(N_R)}}x_{N_{IP},2} \end{matrix}}^{for\ y_2(t)} \\ \vdots \\ \overbrace{\begin{matrix} \overline{\alpha_{N_S}^{(1)}} & \overline{\alpha_{N_S}^{(1)}}x_{1,N_S} & \cdots & \overline{\alpha_{N_S}^{(1)}}x_{N_{IP},N_S} \\ \overline{\alpha_{N_S}^{(2)}} & \overline{\alpha_{N_S}^{(2)}}x_{1,N_S} & \cdots & \overline{\alpha_{N_S}^{(2)}}x_{N_{IP},N_S} \\ & & \cdots & \\ \overline{\alpha_{N_S}^{(N_R)}} & \overline{\alpha_{N_S}^{(N_R)}}x_{1,N_S} & \cdots & \overline{\alpha_{N_S}^{(N_R)}}x_{N_{IP},N_S} \end{matrix}}^{for\ y_{N_S}(t)} \end{bmatrix} \times \begin{bmatrix} b_0^{(1)}(t) \\ b_1^{(1)}(t) \\ \vdots \\ b_{N_{IP}}^{(1)}(t) \\ b_0^{(2)}(t) \\ b_1^{(2)}(t) \\ \vdots \\ b_{N_{IP}}^{(2)}(t) \\ \vdots \\ b_0^{(N_R)}(t) \\ b_1^{(N_R)}(t) \\ \vdots \\ b_{N_{IP}}^{(N_R)}(t) \end{bmatrix} + \begin{bmatrix} \varepsilon_1(t) \\ \varepsilon_2(t) \\ \vdots \\ \varepsilon_{N_S}(t) \end{bmatrix} \tag{98}$$

For convenience, define

$$U_s = \begin{bmatrix} \overline{\alpha_s^{(1)}} & \overline{\alpha_s^{(1)}}x_{1,s} & \cdots & \overline{\alpha_s^{(1)}}x_{N_{IP},s} \\ \overline{\alpha_s^{(2)}} & \overline{\alpha_s^{(2)}}x_{1,s} & \cdots & \overline{\alpha_s^{(2)}}x_{N_{IP},s} \\ & & \cdots & \\ \overline{\alpha_s^{(N_R)}} & \overline{\alpha_s^{(N_R)}}x_{1,s} & \cdots & \overline{\alpha_s^{(N_R)}}x_{N_{IP},s} \end{bmatrix}^T \tag{99}$$

The linear equations in Eq (98) can now be expressed in matrix form as Eq (100).

$$\begin{bmatrix} y_{desired,1} \\ y_{desired,2} \\ \vdots \\ y_{desired,N_S} \end{bmatrix} = \begin{bmatrix} U_1^T \\ U_2^T \\ \vdots \\ U_{N_S}^T \end{bmatrix} B(t) + \begin{bmatrix} \varepsilon_1(t) \\ \varepsilon_2(t) \\ \vdots \\ \varepsilon_{N_S}(t) \end{bmatrix} \tag{100}$$

$$Y_{desired,N_S} = \mathbf{U}_{N_S} B(t) + \boldsymbol{\varepsilon}_{N_S}(t)$$

Where,

$Y_{desired,N_S} = \begin{bmatrix} y_{desired,1} & y_{desired,2} & \cdots & y_{desired,N_S} \end{bmatrix}^T$ is the vector formed by N_S elements of

$y_{desired,s}$;

$\mathbf{U}_{N_S} = \begin{bmatrix} U_1^T & U_2^T & \cdots & U_{N_S}^T \end{bmatrix}^T$ is the matrix formed by N_S elements of U_s;

$$B(t) = \begin{bmatrix} b_0^{(1)}(t) & b_1^{(1)}(t) & \cdots & b_{N_{IP}}^{(1)}(t) \\ b_0^{(2)}(t) & b_1^{(2)}(t) & \cdots & b_{N_{IP}}^{(2)}(t) \\ & \cdots & & \\ b_0^{(N_R)}(t) & b_1^{(N_R)}(t) & \cdots & b_{N_{IP}}^{(N_R)}(t) \end{bmatrix}^T$$

is the vector formed by $(N_{IP}+1)\times N_R$ elements of $b_j^{(k)}(t)$.

$\boldsymbol{\varepsilon}_{N_S}(t) = \begin{bmatrix} \varepsilon_1(t) & \varepsilon_2(t) & \cdots & \varepsilon_{N_S}(t) \end{bmatrix}^T$ is the vector formed by N_S elements of $\varepsilon_s(t)$.

One of the efficient ways to solve the simultaneous linear equations in Eq (100) is the *Pseudoinverse* technique [117, 118],

$$\hat{P} = \left(\mathbf{U}_{N_S}^T \mathbf{U}_{N_S}\right)^{-1} \mathbf{U}_{N_S}^T Y_{desired,N_S} \tag{101}$$

Where,

$\hat{P}$ is the vector containing all the estimated data contents (real-valued parameters); $B(t)$ in Eq (100) is equal to $\hat{P}$ when the training is completed.

To assist the derivation of the learning technique, define

$$\mathbf{M}_s = \mathbf{U}_s^T \mathbf{U}_s \tag{102}$$

and

$$\mathbf{N}_s = \mathbf{M}_s^{-1} \tag{103}$$

The $i+1$ th element can be derived as

$$\mathbf{M}_{s+1} = \mathbf{M}_s + U_{s+1} U_{s+1}^T \tag{104}$$

and

$$\mathbf{N}_{s+1} = \mathbf{M}_{s+1}^{-1} \tag{105}$$

According to Sherman-Morrison Identity [119-123], given three matrices $\mathbf{A} \in \Re^{p\times p}$, $\mathbf{B} \in \Re^{p\times n}$ and $\mathbf{C} \in \Re^{n\times p}$, the inverse of $(\mathbf{A}+\mathbf{BC})$ can be expressed as

$$(\mathbf{A}+\mathbf{BC})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(I+\mathbf{CA}^{-1}\mathbf{B}\right)^{-1}\mathbf{CA}^{-1} \tag{106}$$

Where,

I is the identity matrix;

Let $\mathbf{A} = \mathbf{M}_s$, $\mathbf{B} = U_{s+1}$ and $\mathbf{C} = U_{s+1}^T$, the inverse of Eq (104) can be expressed as

$$\left(\mathbf{M}_s + U_{s+1}U_{s+1}^T\right)^{-1} = \mathbf{M}_s^{-1} - \mathbf{M}_s^{-1}U_{s+1}\left(I+U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1} \tag{107}$$

The equality in Eq (107) can be further expanded as

$$\begin{aligned}&\left(\mathbf{M}_s + U_{s+1}U_{s+1}^T\right)^{-1} = \mathbf{M}_s^{-1} - \mathbf{M}_s^{-1}U_{s+1}\left(I+U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1}\\ &\Leftrightarrow I = \left(\mathbf{M}_s + U_{s+1}U_{s+1}^T\right)\left[\mathbf{M}_s^{-1} - \mathbf{M}_s^{-1}U_{s+1}\left(I+U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1}\right]\end{aligned} \tag{108}$$

The equality in Eq (107) will be proven if the RHS of Eq (108) can be reduced to an identity matrix.

Proof: The RHS of Eq (108) can be expanded as

$$\begin{aligned}&\left(\mathbf{M}_s + U_{s+1}U_{s+1}^T\right)\left[\mathbf{M}_s^{-1} - \mathbf{M}_s^{-1}U_{s+1}\left(I+U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1}\right]\\ &= \mathbf{M}_s\mathbf{M}_s^{-1} + U_{s+1}U_{s+1}^T\mathbf{M}_s^{-1} - \mathbf{M}_s\mathbf{M}_s^{-1}U_{s+1}\left(I+U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1}\\ &\quad - U_{s+1}U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\left(I+U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1}\\ &= I + U_{s+1}U_{s+1}^T\mathbf{M}_s^{-1} - U_{s+1}\left(I+U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1}\\ &\quad - U_{s+1}U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\left(I+U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1}\end{aligned} \tag{109}$$

Note that U_{s+1}^T and U_{s+1} are row vector and column vector with $\left[\left(N_{IP}+1\right)\times N_R\right]$ elements while $\mathbf{M}_s^{-1}$ is a $\left[\left(N_{IP}+1\right)\times N_R\right] \times \left[\left(N_{IP}+1\right)\times N_R\right]$ matrix. The expression $U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}$ is a

1×1 matrix (i.e. a scalar). Therefore, the expression $\left(I + U_{s+1}^T \mathbf{M}_s^{-1} U_{s+1}\right)$ is also a scalar. Thus, we can rewrite Eq (109) as

$$\begin{aligned}
&I + U_{s+1}U_{s+1}^T\mathbf{M}_s^{-1} - U_{s+1}\left(I + U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1} \\
&\quad - U_{s+1}U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\left(I + U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1} \\
&= I + U_{s+1}U_{s+1}^T\mathbf{M}_s^{-1} - U_{s+1}\left(I + U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}\left(I + U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)U_{s+1}^T\mathbf{M}_s^{-1} \\
&= I + U_{s+1}U_{s+1}^T\mathbf{M}_s^{-1} - U_{s+1}U_{s+1}^T\mathbf{M}_s^{-1} \\
&= I
\end{aligned} \tag{110}$$

Q.E.D.

Hence, substituting the equality in Eq (107) into Eq (105), $\mathbf{N}_{s+1}$ can be expressed as

$$\begin{aligned}
\mathbf{N}_{s+1} &= \mathbf{M}_{s+1}^{-1} \\
&= \left(\mathbf{M}_s + U_{s+1}U_{s+1}^T\right)^{-1} \\
&= \mathbf{M}_s^{-1} - \mathbf{M}_s^{-1}U_{s+1}\left(I + U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}\right)^{-1}U_{s+1}^T\mathbf{M}_s^{-1} \\
&= \mathbf{M}_s^{-1} - \frac{\mathbf{M}_s^{-1}U_{s+1}U_{s+1}^T\mathbf{M}_s^{-1}}{1 + U_{s+1}^T\mathbf{M}_s^{-1}U_{s+1}} \\
&= \mathbf{N}_s - \frac{\mathbf{N}_s U_{s+1}U_{s+1}^T\mathbf{N}_s}{1 + U_{s+1}^T\mathbf{N}_s U_{s+1}}
\end{aligned} \tag{111}$$

Vector $\hat{P}$ can be derived by computing P_s iteratively and

$$\hat{P} = P_{N_S} \tag{112}$$

Where,

N_S is the total number of samples;

Consider the equality

$$\mathbf{U}_s P_s = Y_{desired,s} \tag{113}$$

We have

$$\begin{aligned}
&\mathbf{U}_s P_s = Y_{desired,s} \\
&\Leftrightarrow \mathbf{U}_s^T \mathbf{U}_s P_s = \mathbf{U}_s^T Y_{desired,s} \\
&\Leftrightarrow \mathbf{U}_s^T Y_{desired,s} = \mathbf{M}_s P_s \\
&\Leftrightarrow \mathbf{U}_s^T Y_{desired,s} = \mathbf{N}_s^{-1} P_s
\end{aligned} \tag{114}$$

When the next sample ($s+1$th sample) is presented, we have

$$\begin{aligned}
&\begin{bmatrix} \mathbf{U}_s \\ U_{s+1}^T \end{bmatrix} P_{s+1} = \begin{bmatrix} Y_{desired,s} \\ y_{desired,s+1} \end{bmatrix} \\
&\Leftrightarrow \begin{bmatrix} \mathbf{U}_s \\ U_{s+1}^T \end{bmatrix}^T \begin{bmatrix} \mathbf{U}_s \\ U_{s+1}^T \end{bmatrix} P_{s+1} = \begin{bmatrix} \mathbf{U}_s \\ U_{s+1}^T \end{bmatrix}^T \begin{bmatrix} Y_{desired,s} \\ y_{desired,s+1} \end{bmatrix} \\
&\Leftrightarrow \left(\mathbf{U}_s^T \mathbf{U}_s + U_{s+1} U_{s+1}^T \right) P_{s+1} = \left(\mathbf{U}_s^T Y_{desired,s} + y_{desired,s+1} U_{s+1} \right) \\
&\Leftrightarrow \mathbf{M}_{s+1} P_{s+1} = \left(\mathbf{U}_s^T Y_{desired,s} + y_{desired,s+1} U_{s+1} \right) \\
&\Leftrightarrow P_{s+1} = \mathbf{N}_{s+1} \left(\mathbf{U}_s^T Y_{desired,s} + y_{desired,s+1} U_{s+1} \right)
\end{aligned} \tag{115}$$

Substitute Eqs (111) and (114) into Eq (115) gives

$$\begin{aligned}
P_{s+1} &= \mathbf{N}_{s+1} \left(\mathbf{U}_s^T Y_{desired,s} + y_{desired,s+1} U_{s+1} \right) \\
&= \mathbf{N}_{s+1} \mathbf{U}_s^T Y_{desired,s} + y_{desired,s+1} \mathbf{N}_{s+1} U_{s+1} \\
&= \left(\mathbf{N}_s - \frac{\mathbf{N}_s U_{s+1} U_{s+1}^T \mathbf{N}_s}{1 + U_{s+1}^T \mathbf{N}_s U_{s+1}} \right) \mathbf{N}_s^{-1} P_s + y_{desired,s+1} \mathbf{N}_{s+1} U_{s+1} \\
&= \mathbf{N}_s \mathbf{N}_s^{-1} P_s - \frac{\mathbf{N}_s U_{s+1} U_{s+1}^T}{1 + U_{s+1}^T \mathbf{N}_s U_{s+1}} P_s + y_{desired,s+1} \mathbf{N}_{s+1} U_{s+1} \\
&= P_s - \frac{\mathbf{N}_s U_{s+1}}{1 + U_{s+1}^T \mathbf{N}_s U_{s+1}} U_{s+1}^T P_s + y_{desired,s+1} \mathbf{N}_{s+1} U_{s+1} \\
&= P_s - \left[\mathbf{N}_s U_{s+1} - \frac{\mathbf{N}_s U_{s+1} \left(1 + U_{s+1}^T \mathbf{N}_s U_{s+1} \right) - \mathbf{N}_s U_{s+1}}{1 + U_{s+1}^T \mathbf{N}_s U_{s+1}} \right] U_{s+1}^T P_s + y_{desired,s+1} \mathbf{N}_{s+1} U_{s+1} \\
&= P_s - \left(\mathbf{N}_s U_{s+1} - \frac{\mathbf{N}_s U_{s+1} U_{s+1}^T \mathbf{N}_s U_{s+1}}{1 + U_{s+1}^T \mathbf{N}_s U_{s+1}} \right) U_{s+1}^T P_s + y_{desired,s+1} \mathbf{N}_{s+1} U_{s+1} \\
&= P_s - \left(\mathbf{N}_s - \frac{\mathbf{N}_s U_{s+1} U_{s+1}^T \mathbf{N}_s}{1 + U_{s+1}^T \mathbf{N}_s U_{s+1}} \right) U_{s+1} U_{s+1}^T P_s + y_{desired,s+1} \mathbf{N}_{s+1} U_{s+1} \\
&= P_s - \mathbf{N}_{s+1} U_{s+1} U_{s+1}^T P_s + y_{desired,s+1} \mathbf{N}_{s+1} U_{s+1} \\
&= P_s + \mathbf{N}_{s+1} U_{s+1} \left(y_{desired,s+1} - U_{s+1}^T P_s \right)
\end{aligned} \tag{116}$$

Where $\mathbf{N}_{s+1}$, U_{s+1}, $y_{desired,s+1}$ and P_s are all known during the presentation of $s+1$th sample.

Thus, the learning for the consequents of TSK[1]-FCMAC can be summarized as the estimation of vector $\hat{P}$ which can be derived by computing P_s iteratively using Eqs (111), (112) and (116) with the initial conditions of

$$P_0 = 0 \tag{117}$$

and

$$\mathbf{N}_0 = \gamma I \tag{118}$$

Where,

γ is a positive large number;

I is the identity matrix of dimension $\left[\left(N_{IP}+1\right)\times N_R\right] \times \left[\left(N_{IP}+1\right)\times N_R\right]$.

The learning algorithm of TSK[1]-FCMAC involves solving simultaneous linear equations using the *pseudoinverse* technique. It is a well-known technique to compute the least squares solution and is also referred to as the *Kalman filter*. Kalman filter estimates the system state vector $\mathbf{x}$ from a set of noisy measurement $\mathbf{z}$ with linear recursive technique. It is optimal in the mean squared error sense given certain assumptions and is the best possible of all filters (including nonlinear filters) [1]. This suggests that the learning algorithm will inherently converge. In addition, the stability and convergence of Kalman filter based algorithm has already been established by Guo [124]. Therefore, the study on the convergence for TSK[1]-FCMAC is not pursued.

5.3. Benchmarking Examples

In this section, we demonstrate the ability of the proposed architecture on the following problem categories; namely: 1) sinusoidal function/surface approximation (function approximation), 2) automobile MPG prediction (regression) and, 3) two-spiral problem (classification).

In section 5.3.1, the performance of TSK[1]-FCMAC was evaluated by approximating a sinusoidal function and surface. This experiment attempts to assess the improvement of the architecture from the aspect of precision. The proposed architectures were also appraised on the basic of the training time requirement. In section 5.3.2, TSK[1]-FCMAC was applied to predict the fuel consumption for different type of automobiles in the MPG (miles per gallon) prediction [125]. A simple feature selection method were conducted with the proposed architectures and compared to existing feature selection methods proposed by Quah [126] and Jang [127]. Lastly, the proposed architectures were applied on the two-spiral classification problem in section 5.3.3. This problem is highly segmented in the input space and the experiment aims to evaluate the capability of TSK[1]-FCMAC in dealing with such problems.

5.3.1. Sinusoidal Function/Surface Approximation

Sinusoidal function is one of the most commonly used functions. The goal is to approximate this function and evaluate the performance by root mean square error (RMSE). As mentioned previously, the inspiration for implementing a higher ordered Takagi-Sugeno-Kang fuzzy inference system is to improve the performance of TSK[0]-FCMAC from the aspect of precision.

This experiment aims to compare the performance between TSK[1]-FCMAC and its predecessor, TSK[0]-FCMAC. To appraise the improvement over its predecessor, both architectures were evaluated under same network size (i.e. same number of fuzzy rules). Two experiments are conducted; namely: 1) the sinusoidal function approximation and, 2) the sinusoidal surface approximation. The sinusoidal function is defined in Eq (119).

$$y = \sin\left(\frac{\pi x}{180}\right) \tag{119}$$

A set of 360 data (where $x \in \{0,1,2,\ldots,359\}$) was used for both training and testing.

The sinusoidal surface is defined in Eq (120).

$$z = \sin\left(\frac{\pi x}{180}\right)\sin\left(\frac{\pi y}{180}\right) \tag{120}$$

A set of 1296 data (where $x, y \in \{0,10,20,\ldots,350\}$) was used for both training and testing. The sinusoidal function and surface are shown in Figure 5-1.

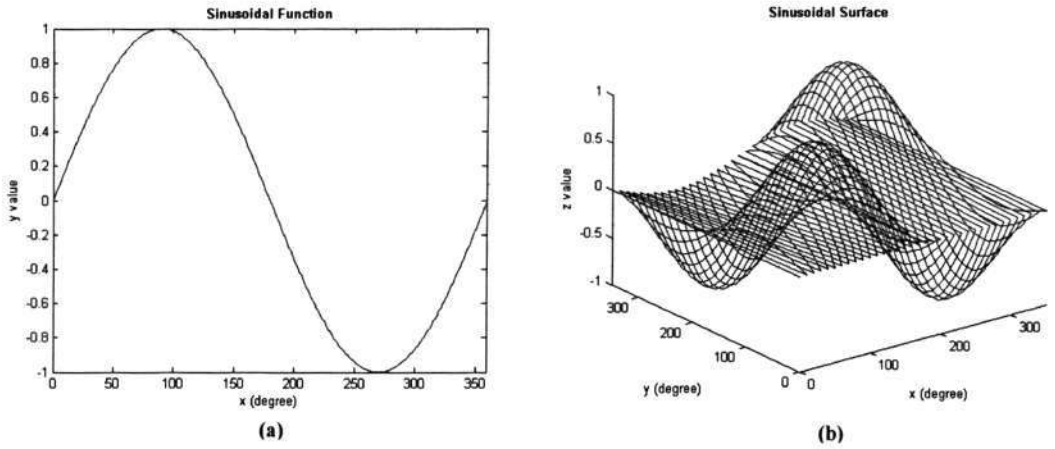


Figure 5-1: (a) Sinusoidal Function; (b) Sinusoidal Surface

To have an unbiased comparison, the performances of both architectures are evaluated with the same network size. As described in section 5.2, TSK^0-FCMAC and TSK^1-FCMAC employ the same structure in the antecedents. The size of the network is controlled by the parameter *SLOPE* (described in section 3.4.1). The comparisons for RMSE sinusoidal function and surface are shown in Table 5-1 and Table 5-2 respectively. From both tables, it can be observed that the RMSE of TSK^1-FCMAC is lower than that of TSK^0-FCMAC in all cases.

Table 5-1: Comparison of RMSE between TSK^0-FCMAC and TSK^1-FCMAC (Sinusoidal Function)

SLOPE	Network Size	$RMSE_{TSK^0-FCMAC}$	$RMSE_{TSK^1-FCMAC}$	$\frac{RMSE_{TSK^1-FCMAC}}{RMSE_{TSK^0-FCMAC}}$
0.5	4	0.36642	0.01561	4.26%
0.4	4	0.14876	0.02909	19.56%
0.3	4	0.08352	0.01907	22.83%
0.2	7	0.04970	0.00809	16.27%
0.1	11	0.02842	0.00567	19.94%

Table 5-2: Comparison of RMSE between TSK[0]-FCMAC and TSK[1]-FCMAC (Sinusoidal Surface)

SLOPE	Network Size	$RMSE_{TSK^0-FCMAC}$	$RMSE_{TSK^1-FCMAC}$	$\frac{RMSE_{TSK^1-FCMAC}}{RMSE_{TSK^0-FCMAC}}$
0.5	2 x 2	0.39276	0.23153	58.95%
0.4	4 x 3	0.31885	0.09819	30.79%
0.3	4 x 4	0.21300	0.06308	29.61%
0.2	7 x 6	0.10509	0.02470	23.50%
0.1	13 x 13	0.03069	0.00601	19.57%

To highlight the improvement in precision of TSK[1]-FCMAC over its predecessor, Figure 5-2 shows the sinusoidal function approximated by both architectures. The details are listed as follow:

- Figure 5-2(a): sinusoidal function approximated by TSK[0]-FCMAC ($SLOPE = 0.1$, Network Size = 11 (11 fuzzy rules), RMSE = 0.02842)
- Figure 5-2(b): sinusoidal function approximated by TSK[1]-FCMAC ($SLOPE = 0.5$, Network Size = 4 (4 fuzzy rules), RMSE = 0.01561)

TSK[1]-FCMAC achieved RSME of 0.01561 with 4 fuzzy if-then rules while TSK[0]-FCMAC requires 11 fuzzy if-then rules to achieved RMSE of 0.02842. This result shows that TSK[1]-FCMAC is able to better approximate the sinusoidal function with fewer fuzzy rules (equivalent to network size) and yet, result in a smaller RMSE.

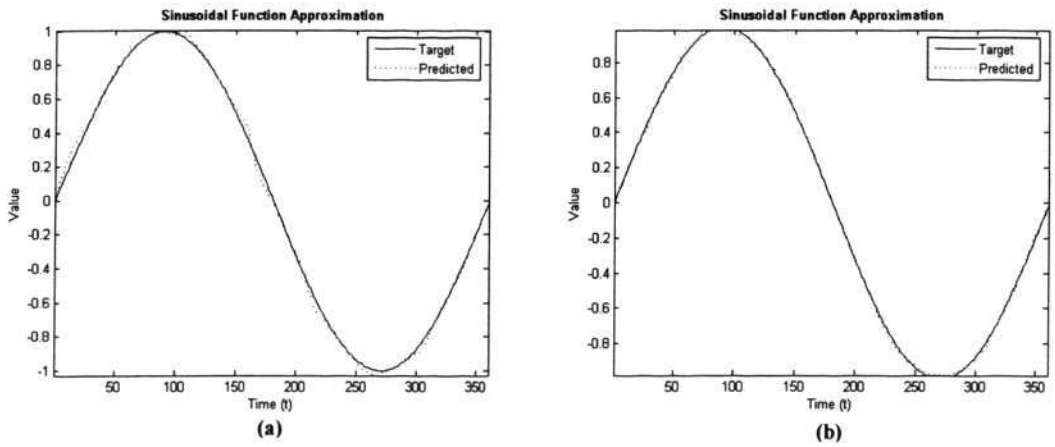


Figure 5-2: Sinusoidal Function Approximated by: (a) TSK[0]-FCMAC ($SLOPE = 0.1$, Network Size = 11); (b) TSK[1]-FCMAC ($SLOPE = 0.5$, Network Size = 4)

Another example in the case of sinusoidal surface approximation is shown in Figure 5-3 and Figure 5-4. The details of both architectures are listed as follow:

- Figure 5-3: sinusoidal surface approximated by TSK^0-FCMAC ($SLOPE$ = 0.1, Network Size = 13 x 13 (169 fuzzy rules), RMSE = 0.03069)
- Figure 5-4: sinusoidal surface approximated by TSK^1-FCMAC ($SLOPE$ = 0.2, Network Size = 7 x 6 (42 fuzzy rules), RMSE = 0.02470)

This result shows that TSK^1-FCMAC is able to achieve superior RMSE with 42 fuzzy rules whereas TSK^0-FCMAC requires 169 fuzzy rules.

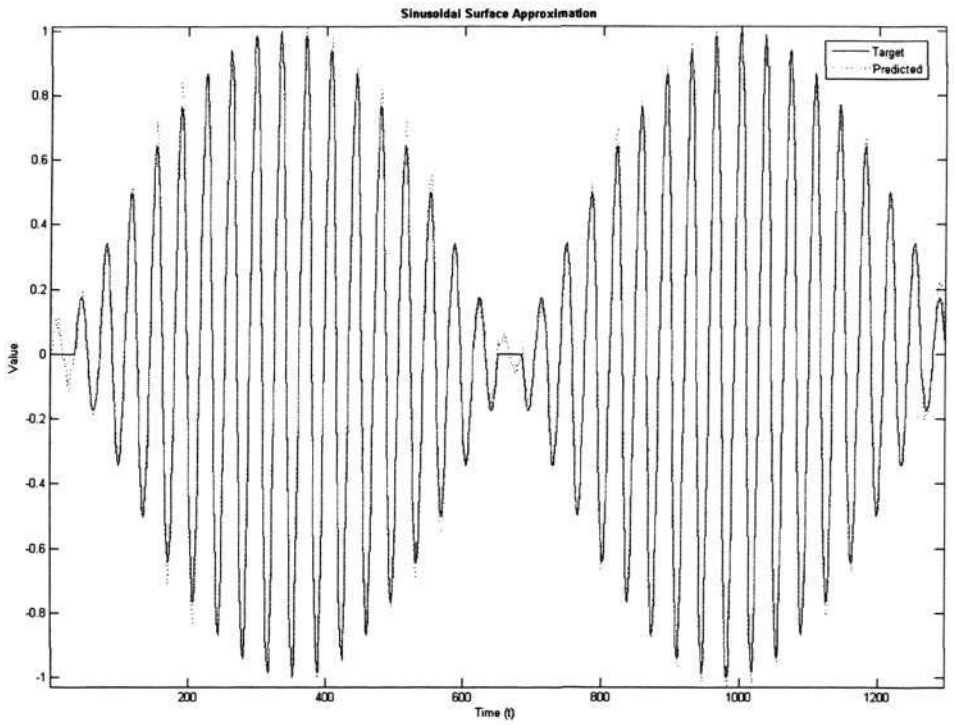


Figure 5-3: Sinusoidal Surface Approximated by TSK^0-FCMAC ($SLOPE$ = 0.1, Network Size = 13 x 13)

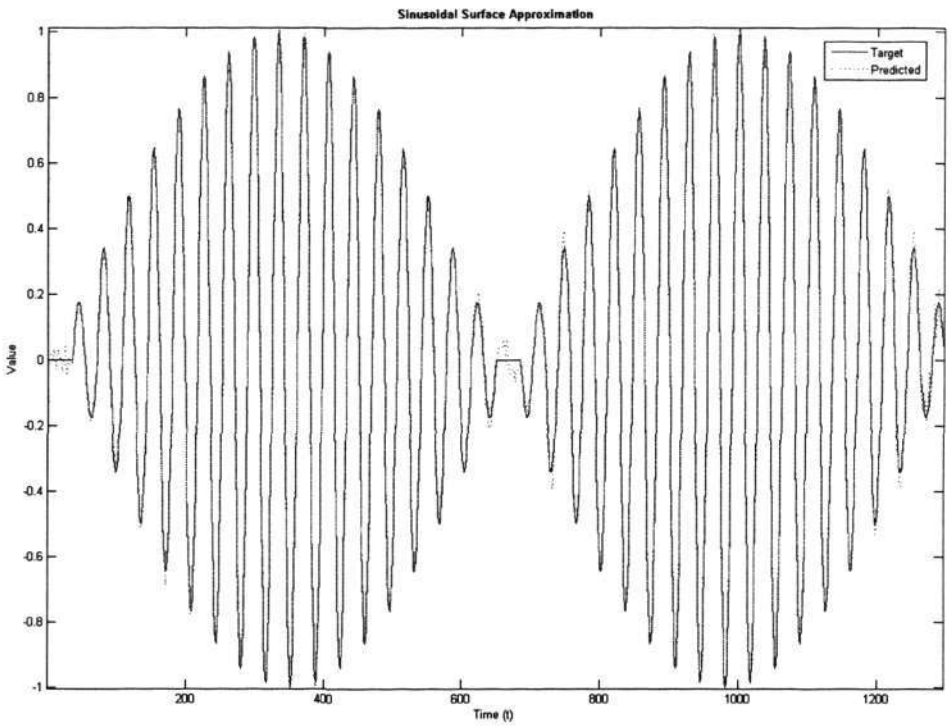


Figure 5-4: Sinusoidal Surface Approximated by TSK[1]-FCMAC (*SLOPE* = 0.2, Network Size = 7 x 6)

To highlight the improvement from the perspective of the total number of parameters required in individual architectures, a comparison is presented in Table 5-3. The table presents a clear illustration on the influence of having a smaller network size. It can be observed that the total number of parameters required by TSK[1]-FCMAC for sinusoidal function approximation is only 43.64% and that for sinusoidal surface approximation is 34.49% with respect to the number of parameters required by TSK[0]-FCMAC to achieve similar precision.

Table 5-3: Comparison of No. of Parameters between TSK[0]-FCMAC and TSK[1]-FCMAC

Experiment	Network Type	Network Size	Parameters in Antecedents	Parameters in Consequents	Total Parameters	Ratio
Sinusoidal Function	TSK[0]-FCMAC	11	44	11	55	24/55 (43.64%)
	TSK[1]-FCMAC	4	16	8	24	
Sinusoidal Surface	TSK[0]-FCMAC	13 x 13	676	169	845	294/845 (34.49%)
	TSK[1]-FCMAC	7 x 6	168	126	294	

No. of parameters for antecedents of both architectures = No. of fuzzy rules x 4 (i.e. No. of parameters to represent a trapezoidal function);
No. of parameters for consequents of TSK[0]-FCMAC = No. of fuzzy rules;
No. of parameters for consequents of TSK[1]-FCMAC = No. of fuzzy rules x (No. of input features + 1).

As described in section 5.2, even though there is a decrease in the number of parameter required, the learning algorithm of TSK[1]-FCMAC is more complex and is anticipated to incur higher computational power than that of TSK[0]-FCMAC. To substantiate such belief, further investigations are conducted on the training time requirements of individual architectures. Table 5-4 and Table 5-5 show the training time taken to approximate sinusoidal function and surface by both architectures under 100 training epochs. The specifications for the system to conduct this experiment are listed as follow:

- Intel Core ™ Duo CPU, E6750 @ 2.66 GHz
- 2.66 GHz, 3.00 GB of RAM
- Window XP Professional version 2002, service pack 2
- Microsoft Visual C++ version 6.0

From Table 5-4 and Table 5-5, it can be observed that the training time of the network increases exponentially when the network size is increased. This result provides further support to the theoretical expectation on the increment of computational complexity. Nevertheless, as stated in the principle of incompatibility [10] by Professor Lotfi A. Zadeh regarding the dilemma between precision versus cost, *one has to pay a cost for high precision*. The solution for such a dilemma is to find a balance between the two which fulfills the requirements of the problems at a reasonable cost.

Table 5-4: Comparison of Training Time between TSK[0]-FCMAC and TSK[1]-FCMAC (Sinusoidal Function)

SLOPE	Network Size	$T_{Training,TSK^0-FCMAC}$ (Sec)	$T_{Training,TSK^1-FCMAC}$ (Sec)	$\frac{T_{Training,TSK^1-FCMAC}}{T_{Training,TSK^0-FCMAC}}$
0.5	4	0.407	0.408	100.25%
0.4	4	0.485	0.454	93.61%
0.3	4	0.407	0.407	100.00%
0.2	7	0.345	0.704	204.06%
0.1	11	0.344	1.236	359.30%

Table 5-5: Comparison of Training Time between TSK^0-FCMAC and TSK^1-FCMAC (Sinusoidal Surface)

SLOPE	Network Size	$T_{Training,TSK^0-FCMAC}$ (Sec)	$T_{Training,TSK^1-FCMAC}$ (Sec)	$\frac{T_{Training,TSK^1-FCMAC}}{T_{Training,TSK^0-FCMAC}}$
0.5	2 x 2	1.608	2.409	149.81%
0.4	4 x 3	2.015	10.302	511.27%
0.3	4 x 4	1.892	16.951	895.93%
0.2	7 x 6	2.243	105.938	4723.05%
0.1	13 x 13	2.344	1851.490	78988.48%

5.3.2. Automobile MPG Prediction

The section performs a case study of the automobile MPG (miles per gallon), in which an automobile's fuel consumption in terms of MPG is predicted based on six other characteristics. The characteristics are listed as follows:

- c : "no. of cylinders"
- p : "horsepower"
- y : "model year"
- w : "weight"
- d : "displacement"
- a : "acceleration"

The six features are either discrete value ("no. of cylinders" and "model year") or continuous value ("displacement", "horsepower", "weight" and "acceleration"). The data set was obtained from the UCI Machine Learning Repository [125]. There are 398 data instances in the data set where 6 instances with unknown "horsepower" are discarded. The remaining 392 data instances are randomly divided into two data sets of equal size. Fifteen data sets are generated from each of them by using different combinations from two of the inputs characteristics (i.e. use "no. of cylinders" and "model year" as one combination of input). Subsequently, the thirty two-input data sets are permutated a hundred times and a total of 3000 data sets are generated. Permutation involves reshuffling of the data instances in random order. Network trained with permutation from one set will use permutation from another set as testing data.

Jang proposed an input selection method with ANFIS [127] and the best two features derived are ("weight" + "model year"). Another feature selection method proposed by Quah, the *Monte Carlo*

Evaluative Selection (MCES) [126] conducted similar experiment and selected ("displacement" + "model year") as the best two features.

In our experiment, both TSK^0-FCMAC and TSK^1-FCMAC are applied to predict the fuel consumption. The RMSE (mean value over 100 different permutations) using 15 different characteristics is shown in Figure 5-5. As highlighted by A_1 in Figure 5-5, it can be observed that input characteristics ("displacement" + "model year") and ("weight" + "model year") produce the smallest RMSE value in most cases. This observation is consistent with the features selected by both ANFIS and MCES.

It can also be noted that the mean RMSE of TSK^1-FCMAC is generally smaller than that of TSK^0-FCMAC. Using ("weight" + "model year") as input characteristics, the mean RMSE of TSK^1-FCMAC under hundred permutations is 3.08. Using the same input characteristics, the result reported in Jang's ANFIS [127] achieves a RMSE of 2.98. This experiment shows that TSK^1-FCMAC is able to achieve comparable results with ANFIS, another first-ordered TSK fuzzy inference system.

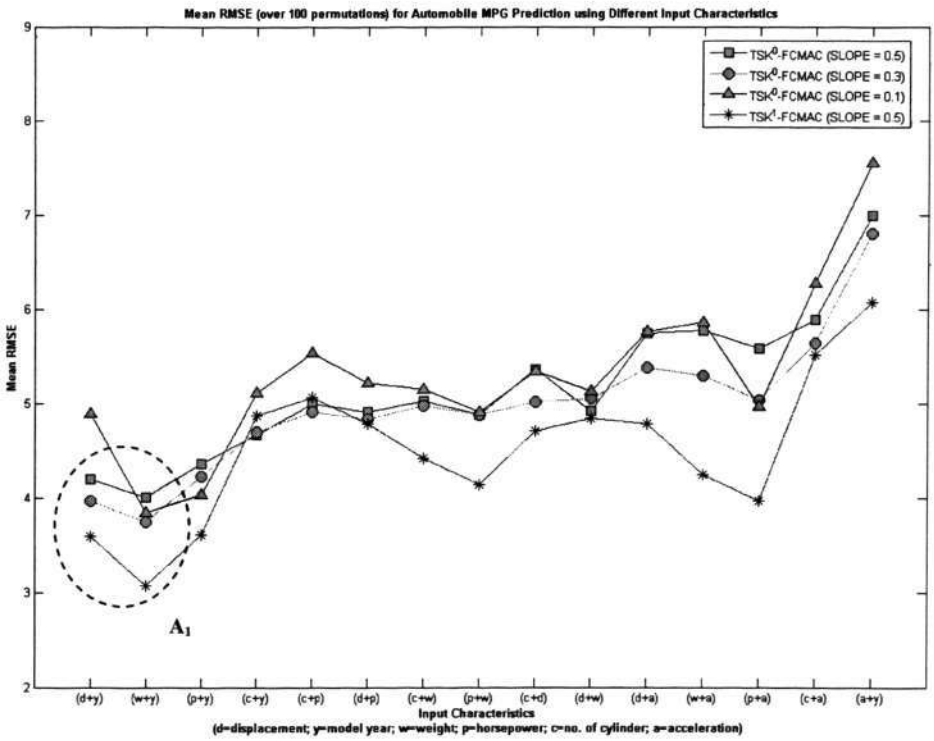


Figure 5-5: RMSE comparison between TSK^0-FCMAC and TSK^1-FCMAC in Automobile MPG Prediction

Figure 5-6 and Figure 5-7 show the predicted MPG (miles per gallon) and the training RMSE using the two selected feature sets, ("displacement" + "model year") and ("weight" + "model year"). In both cases, only 4 fuzzy if-then rules are created to achieve the RMSE of 3.45 and 2.99 respectively. In terms of TSK[1]-FCMAC architecture, four fuzzy if-then rules will result in $4 \times 4 = 16$ linear parameters in the antecedent (4 points to represent a trapezoidal membership function) and $4 \times (2+1) = 12$ linear parameters in the consequent (for a two-input network).

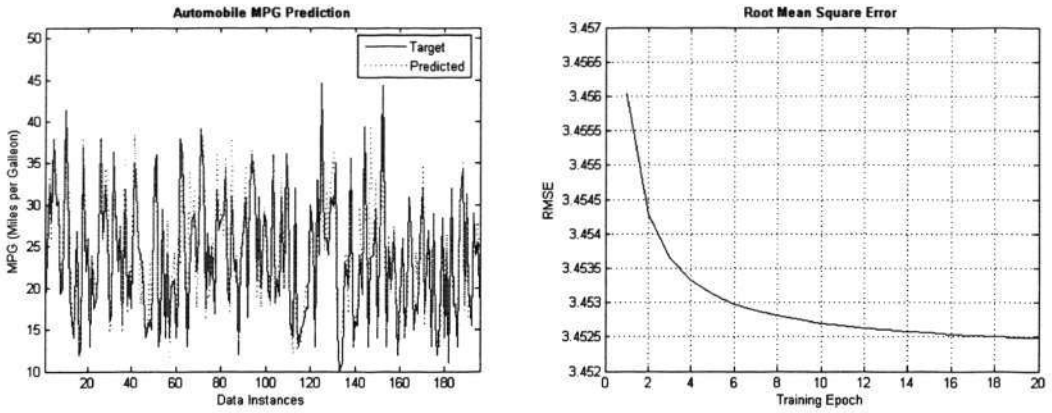


Figure 5-6: (a) Predicted Automobile MPG; (b) MSE of 20 Training Epochs (with TSK[1]-FCMAC, *SLOPE* = 0.5, Network Size = 2 x 2) ("displacement" + "model year")

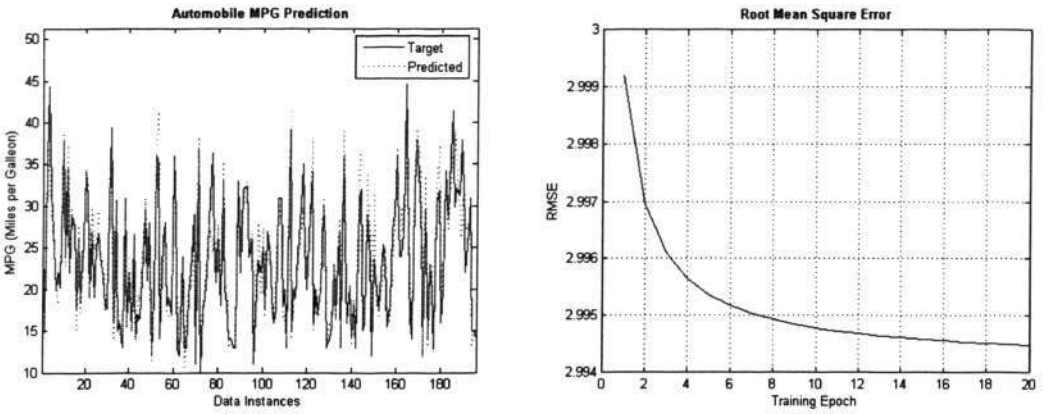


Figure 5-7: (a) Predicted Automobile MPG; (b) MSE of 20 Training Epochs (with TSK[1]-FCMAC, *SLOPE* = 0.5, Network Size = 2 x 2) ("weight" + "model year")

5.3.3. Two-spiral Problem

The two-spiral problem was proposed by Alexis P. Wieland on the connectionist mailing list as an interesting benchmark task for neural networks. The problem involves the correct classification of two intertwined spirals.

The formulation for the generation of the two-spiral training data is given as:

$$spiral\ 1: \begin{cases} x = \gamma \cos\theta \\ y = \gamma \sin\theta \end{cases} \tag{121}$$

$$spiral\ 2: \begin{cases} x = -\gamma \cos\theta \\ y = -\gamma \sin\theta \end{cases} \tag{122}$$

Where

$$\gamma = \frac{1}{\pi}\left(\theta + \frac{2}{\pi}\right) \tag{123}$$

$$\theta = \frac{k\pi}{16}, k = 0, 1, 2, ..., 96 \tag{124}$$

Two spiral data sets are generated, they are:

- A standard spiral set (sparse spiral): consists of 194 points, with 97 points for each spiral;
- A detailed spiral set (dense spiral): consists of 770 points, with 385 points for each spiral.

The sparse and dense spiral sets are shown in Figure 5-8(a) and (b), respectively.

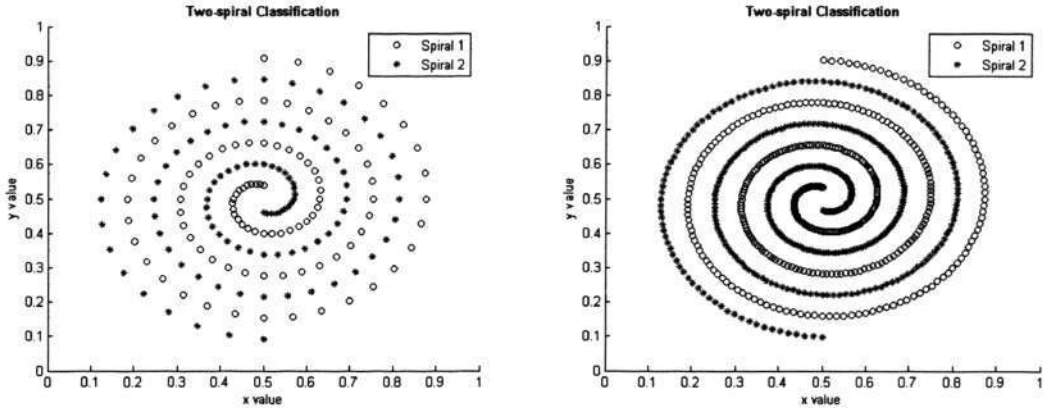


Figure 5-8: Two-spiral Classification. (a) Standard Spiral Set (Sparse Spiral); (b) Detailed Spiral Set (Dense Spiral)

Lang and Witbrock [128] reported that a conventional feed-forward back-propagation neural network could not solve this problem. Hence, they employed back-propagation neural networks with shortcut connections as pathways for providing information to all parts of the network to eliminate the problem of attenuated error signals when back-propagating through the layers. The proposed network is a special network with a 2-5-5-5-1 structure that has shortcut connections,

with each node being connected to all nodes in all subsequent layers. With one additional bias weight for each node, it has 138 trainable weights. When the weights of the specialized network are updated using vanilla back-propagation, training required an average of 20,000 training epochs.

Carpenter et al. [129] also evaluated the fuzzy ARTMAP system with 2-spiral data set. They used the most stringent criteria (i.e. creating new ART category for every unseen pattern) to train the fuzzy ARTMAP to obtain 100% classification for the dense spiral. As a result, the fuzzy ARTMAP creates 194 ART categories for the standard 2-spiral data set that contains 194 points, which essentially degenerates to a pure one-one memory recall.

Tung and Quek [85] proposed a generic self-organizing fuzzy neural network, GenSoFNN, and reported that it is able to achieve 100% classification for both standard and dense spirals with 23 fuzzy sets in each of the two input dimensions. A total of 156 fuzzy rules are created.

Nguyen et al. [64] proposed a fuzzy CMAC based on the Bayesian Ying-Yang learning (FCMAC-BYY) and achieve 100% classification for both standard and dense spirals with 20 fuzzy sets in each of the two input dimensions.

To evaluate the performance between TSK^0-FCMAC and TSK^1-FCMAC as well as benchmarking with existing techniques, two experiments are conducted, they are:

- Experiment A: uses standard spiral set as training data and both the standard and detailed spiral set as testing data;
- Experiment B: uses detailed spiral as training and testing data.

The experiment results between different methods are shown in Table 5-6. As shown in Table 5-6, both TSK^0-FCMAC and TSK^1-FCMAC are able to achieve 100% classification rate for both experiments. In Experiment A, TSK^0-FCMAC is able to achieve 100% classification with 15 fuzzy sets in dimension x and 17 fuzzy sets in dimension y. Although the complete input space can accommodate a maximum of $15 \times 17 = 255$ fuzzy rules, only 224 fuzzy rules are created. Due to the full connection between antecedent and consequent, TSK^0-FCMAC created more fuzzy rules although there were fewer fuzzy sets in both dimensions as compared to GenSoFNN.

As anticipated, the full connectionist strategy provides a simple architecture for the TSK^0-FCMAC network at the expense of constructing a larger number of fuzzy rules. Nonetheless, an input space efficiency of 87.84% is achieved in TSK^0-FCMAC. The training completed in 20 training epochs and the mean square error is shown in Figure 5-9.

Table 5-6: Classification Results in Two-spiral Problem

Architecture	Experiment A				Experiment B		
	Training Set (194 points)				Training Set (770 points)		
	Network Size	Fuzzy Rules	Classification Rate		Network Size	Fuzzy Rules	Classification Rate
			Testing Set (194 points)	Testing Set (770 points)			Testing Set (770 points)
Lang's 2-5-5-5-1 structure	N.A.	N.A.	100%	92.80%	N.A.	N.A.	N.A.
Fuzzy ARTMAP	N.A.	N.A.	100%	100%	N.A.	N.A.	N.A.
GenSoFNN	23 x 23	156	100%	100%	N.A.	N.A.	N.A.
FCMAC-BYY	20 x 20	N.M.	100%	100%	N.A.	N.A.	N.A.
TSK^0-FCMAC	15 x 17	224	100%	100%	18 x 20	342	100%
TSK^1-FCMAC	21 x 21	375	100%	100%	8 x 8	61	100%

N.A. = Not Applicable; N.M. = Not Mentioned.

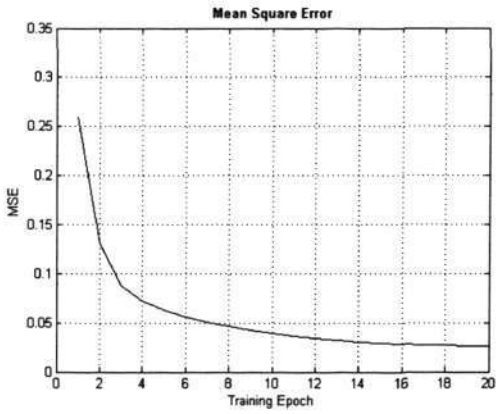


Figure 5-9: Mean Square Error for TSK^0-FCMAC in Two-spiral Classification (For Experiment A)

On the other hand, TSK^1-FCMAC requires 21 fuzzy sets in both dimension x and dimension y to achieve classification rate of 100%. A total of 375 fuzzy rules are created which is more than the 224 fuzzy rules required by TSK^0-FCMAC. The results of Experiment A suggest that the learning of TSK^1-FCMAC is more susceptible to outlier or unseen data. To further investigate this

assumption, a second experiment, Experiment B has been conducted. The proposed architectures were trained and tested with all 770 data points. From Table 5-6, it can be observed that the fuzzy rules required by TSK[1]-FCMAC are significantly reduced to 61 fuzzy rules as compared to the 342 fuzzy rules required by TSK[0]-FCMAC. Results from Experiment B show that if adequate training data is provided, TSK[1]-FCMAC could model a function with higher precision using fewer numbers of if-then fuzzy rules.

5.4. Summary

This chapter presents the architecture and learning algorithm of TSK[1]-FCMAC (a localized self-organizing first-ordered Takagi-Sugeno-Kang fuzzy inference system based on the CMAC structure), an extension of the architecture proposed in Chapter 3. It aims to enhance the performance of its predecessor in terms of precision.

The ability of the proposed network is demonstrated with empirical applications to three different problems; namely: 1) sinusoidal function/surface approximation, 2) automobile MPG (miles per gallon) prediction and, 3) two-spiral problems. TSK[1]-FCMAC is used to approximate sinusoidal function and surface. Empirical results show that TSK[1]-FCMAC is able to achieve similar RMSE value as compared to TSK[0]-FCMAC with fewer numbers of fuzzy if-then rules in both cases. A further experiment to investigate the influence of higher computational complexity in terms of training time has also been conducted. TSK[1]-FCMAC was subsequently applied to predict the fuel consumption for different types of automobile in the MPG (miles per gallon) prediction [125]. A simple feature selection method has been undertaken to select the best feature from the original six input characteristics. The features selected are consistent with two existing input selection methods proposed by Quah [126] and Jang [127] and the results of TSK[1]-FCMAC are comparable in terms of RMSE with Jang's ANFIS, another first-ordered TSK fuzzy inference system. In the third experiment, TSK[1]-FCMAC was applied in a classification problem, the two-spiral classification. Experimental results show that TSK[1]-FCMAC is competent in classification problem that is highly segmented in the input space. Given adequate training samples, TSK[1]-FCMAC could model a problem with higher precision using fewer numbers of fuzzy if-then rules as compared to TSK[0]-FCMAC.

In summary, the experiments conducted in Chapter 3 and Chapter 5 show that TSK^0-FCMAC is better suits for application that requires online training capability and minimum training time while its successor, TSK^1-FCMAC is applicable to problem that requires higher precision.

Chapter 6 Medical Case Study: Personalized Drug Delivery System – A Diabetic Treatment without Meal Announcement

6.1. Introduction

Type I or insulin-dependent diabetes mellitus is a chronic disease that occurs when the pancreas does not produce sufficient insulin. Prolonged period of hyperglycemia, or raised blood sugar, is a common effect of uncontrolled diabetes which may lead to serious damage to many of the body's systems, especially the nerves, eye and kidney. On the other hand, Hypoglycemia, or low blood sugar, may cause unconsciousness, brain damage or even death in some severe cases. The World Health Organization (WHO) estimates that more than 180 million people worldwide have diabetes and this number is likely to more than double by 2030. In 2005, an estimated 1.1 million people died from diabetes [130].

As diabetic cases are expected to increase in future, blood glucose regulation has been an active research field over the years. Despite various attempts by researchers, the constraints of having limited information on current blood glucose level restricted superior results in this field. With the emergence of new electro-enzymatic manufacturing technique [131, 132], the glucose concentration can now be estimated through measuring the catalyzed production of hydrogen peroxide. As a result, a spin-off commercial product, MiniMed [133] enables subcutaneous glucose measurement at five minutes intervals. This advancement in sensor technology dramatically changes the research effort and direction in glucose-insulin dynamics. A significant amount of research has been carried out since then with this newly available information in blood glucose regulation.

Early research efforts focused on mathematical modeling of glucose-insulin dynamics [134-138]. Based on these mathematical models, some educational simulators are developed [139, 140]. Many other researchers tried to handle blood glucose regulation with expert systems [141], model-based predictive control algorithm [142, 143], PID control systems [144], H^{∞} optimal control [145, 146], parametric programming [147], neural networks [148-151] or fuzzy-based controller [152]. However, several important constraints are neglected; namely:

- Disturbance (e.g. food intake): Most previous works assume the diabetic patient to have regular food intake at fixed timing. Some of these works may even employ the amount of food intake, meal time or time of the day as part of the input to their system. However, the accuracy of information such as food intake is highly dependent on the self-discipline of a patient. They are also susceptible to human mistakes, especially for adolescent patients.

- Personalized glucose-insulin dynamics: Instead of adapting to glucose-insulin dynamics of individual, a set of parameters are identified by defining various assumptions. Thus, the response of the system is bounded at the expense of optimal result. These systems do not perform online training and hence are non-adaptive to vibrant environment such as human's endocrine system.

In this chapter, we propose a novel blood glucose regulation using TSK^0-FCMAC, a brain inspired cerebellar like fuzzy association memory based on the zero-ordered Takagi-Sugeno-Kang fuzzy inference scheme [18, 19]. The proposed system is capable of performing localized online training with an effective fuzzy inference scheme that could respond swiftly to changing environment such as human's endocrine system. Without prior knowledge of disturbance (i.e. meal announcement), the proposed fuzzy CMAC is able to capture the glucose-insulin dynamics of individuals under different dietary profiles. Preliminary simulation results show that the proposed system is able to effectively adapt to both intra and inter-patient variations. The design of the proposed system follows closely to what is available in real life and is suitable for animal and clinical pilot testing in the near future.

6.2. System Description

A closed-loop blood glucose control system consists of four components; namely: 1) Patient model, 2) Glucose sensor, 3) Insulin infusion pump with controller, and 4) Reference model. The block diagram for the proposed system is shown in Figure 6-1. In the proposed system, the patient model is simulated by a web-based educational simulation package, GlucoSim [139, 153]. The reference model provides reference blood glucose level for the controller of the insulin pump. Both the reference model and the controller are modeled by a TSK^0-FCMAC network.

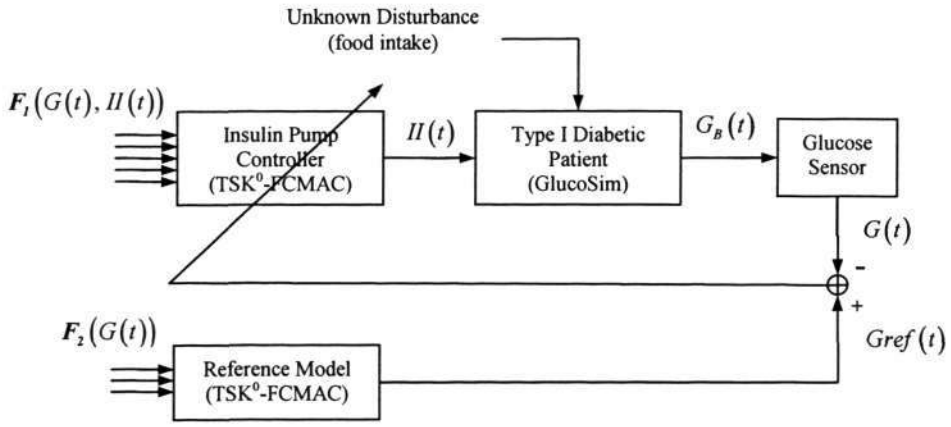


Figure 6-1: Block Diagram for Closed-loop Blood Glucose Control System

6.2.1. GlucoSim

GlucoSim, a web-based educational simulation package for glucose-insulin levels in the human body from Illinois Institute of Technology is employed in this study [139, 153]. Based on the mathematical models by Sorensen [136] and Puckett [138], the simulator can provide virtual experiments to simulate blood glucose and insulin dynamics in healthy individuals and Type I diabetes patients under different dietary profile.

The main disadvantage of the simulated model is that the personal variations in physiological parameters are not taken into account. Therefore, the outputs are merely average values [153]. Thus, body weight is the only parameter that can be adjusted to simulate individuals of different profiles with GlucoSim. In our experiments, GlucoSim is used to simulate a detailed model based on Puckett's work [138].

6.2.2. Controller's Error for TSK^0-FCMAC

In the learning algorithm for TSK^0-FCMAC, the network is updated based on the difference between the desired output and the actual network output. However, in the context of blood glucose regulation, no specific desired output is available as the regulation of blood glucose in response to disturbance of food intake is constrained by the desired upper and lower bounds. As a result, a different error is devised for its learning. The statement below describes the phenomenon in blood glucose regulation:

"Insulin delivered at time t reduces the blood glucose level at time $(t+1)$, $(t+2)$, ... $(t+n)$"

Therefore, to realize the statement above, updating cells that are activated during the current iterations alone is insufficient. Instead, the controller in our system will also update the cells that are activated previously. The error, $Error_{controller}(t)$ for these updates are defined in Eqs (9) and (10):

$$Error_{controller}(t) = (II_{\max} - II_{\min}) \times \delta_{controller} \times \frac{G(t) - Gref(t)}{Gref_{\max} - Gref_{\min}} \times decay\ rate \tag{125}$$

$$decay\ rate = -\log_{10}(0.1 + \omega \times hist) \tag{126}$$

Where

$II_{\max}$, $II_{\min}$ are the maximum and minimum infusion rate for insulin pump;

$\delta_{controller}$ is the learning parameter for controller. It is set to 0.2 to prevent drastic update on the controller's surface;

$G(t)$ is the output from the glucose sensor at time t;

$Gref(t)$ is output from the reference model at time t;

$Gref_{\max}$, $Gref_{\min}$ are the maximum and minimum allowable output from the reference model;

ω is the decay constant for difference between iterations;

$hist$ is the iteration difference between current iterations and the instance when the updating cells are activated previously.

$hist \in \{0, 1, 2, ...\}$

6.3. Experiments

This section describes how the experiments are conducted.

6.3.1. Assumptions

Many researchers developed system that requires information on food intake. The accuracy of such information is highly dependent on the self-discipline of a patient and is susceptible to mistakes, especially for adolescent patients. The assumptions in our proposed system eliminate such needs. With the goal of possible human/animal pilot testing in mind, the following assumptions have been made prior to the conduct of the experiments:

- The blood glucose level is measured at 5 minutes interval, which, is achievable with existing glucose monitoring devices [133].
- No measurement is available for blood insulin level.
- Dietary profile of individual is unknown.

6.3.2. Controller's Objective and Performance Indicator

The Diabetes Control and Complications Trial (DCCT) [154], a clinical study conducted by National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK) [155] showed that keeping blood glucose levels as close to normal as possible slows the onset and progression of eye, kidney and nerve diseases caused by diabetes. As a result, the controller's objective is to maintain the blood glucose level at euglycemia (60 ~ 110 mg/dL) before meals and <180 mg/dL after meals.

Two performance indicators are used to evaluate the performance of the controller. These performance indicators are defined to measure the occurrences of prolonged low and high blood glucose levels which may lead to hyperglycemia and hypoglycemia. They describe the average duration (in terms of minutes) of blood glucose level that exceeds the condition for every diet taken by the person. They are depicted in Eqs (127) and (128).

$$\text{Average duration of hyperglycemia, } Ind_{G(t)>180} = \frac{N_{G(t)>180}}{N_{food\ intake}} \times T_{sampling} \qquad (127)$$

and

$$\text{Average duration of hypoglycemia, } Ind_{G(t)<60} = \frac{N_{G(t)<60}}{N_{food\ intake}} \times T_{sampling} \quad (128)$$

Where

$N_{G(t)<60}$ is the total number of occurrences that the blood glucose level is below 60 mg/dL;

$N_{G(t)>180}$ is the total number of occurrences that the blood glucose level is above 180 mg/dL;

$N_{food\ intake}$ is the total number of food intakes;

$T_{sampling}$ is the duration of one iteration in the simulation. It is equal to the sampling rate of blood glucose level (5 minutes).

6.3.3. Simulation Setup

In order to conduct the simulation, the patient's profile and its dietary habit must be properly defined. A Type I diabetic patient's profile (P_A) is formulated in our simulations and is shown in Table 6-1. The patient is assumed to be a middle-aged Asian office worker whose job is clerical in nature. The BMI (Body Mass Index) for the patient is set at 23 and the daily calorie needed are computed based on Harris-Benedict equation [156]. The computation for the corresponding daily carbohydrate intake follows the assumptions below:

- The ratio of calorie obtained from diet is: carbohydrate (57%), fats (30%) and protein (13%)
- The energy yield per grams is: carbohydrate (4 kcal), fats (9 kcal) and protein (4 kcal)

Table 6-1: Patient A's Profile

Sex	Male
Age	40 years old
Weight	70 kg
Height	174.5 cm
Body Mass Index	23
Activity Factor	Light exercise or sports 1-3 days a week
Daily Calorie Needed	1957 kcal
Corresponding Daily Carbohydrate Intake	279 grams

The Dietary profile for the patient is defined from two perspectives; namely: carbohydrate intake (CI) and eating habit (EH). The nomenclature is defined as follows:

P_A	Type I diabetic patient A;
HP_A	Healthy person A. HP_A is the same as P_A except HP_A is able to regulate his own blood glucose level;
$diet_{P_A, CI, EH}$	Dietary profile of P_A with carbohydrate intake CI and eating habit EH, where $CI \in \{normal, overeat, undereat, mixed\}$ and $EH \in \{regular, irregular\}$;
$diet_{P_A, normal, EH}$	Daily carbohydrates intake equals to the amount computed based on Harris-Benedict equation [156] for P_A;
$diet_{P_A, overeat, EH}$	Daily carbohydrates intake equals to 150% of $diet_{P_A, normal, EH}$;
$diet_{P_A, undereat, EH}$	Daily carbohydrates intake equals to 50% of $diet_{P_A, normal, EH}$;
$diet_{P_A, mixed, EH}$	Daily carbohydrates intake switches among $diet_{P_A, normal, EH}$, $diet_{P_A, overeat, EH}$ and $diet_{P_A, undereat, EH}$;
$diet_{P_A, CI, regular}$	Carbohydrates intake and meal time is consistent every meal;
$diet_{P_A, CI, irregular}$	Carbohydrates intake varies between $\pm 50\%$ of the original value. Meal time varies between ± 1 hour;

With respect to the definitions above, the dietary profile for regular and irregular eating habit are shown as Table 6-2 and Table 6-3.

Table 6-2: Regular Dietary Profile

Meal Type	Meal Time	% of Carbohyrate w.r.t. Total Carbohydrate Intake
Breakfast	0800 hrs	20%
Lunch	1200 hrs	30%
Tea break	1530 hrs	10%
Dinner	1930 hrs	40%

Table 6-3: Irregular Dietary Profile

Meal Type	Meal Time	% of Carbohyrate w.r.t. Total Carbohydrate Intake
Breakfast	0700 hrs ~ 0900 hrs	10% ~ 30%
Lunch	1100 hrs ~ 1300 hrs	15% ~ 45%
Tea break	1430 hrs ~ 1630 hrs	5% ~ 15%
Dinner	1830 hrs ~ 2030 hrs	20% ~ 60%

The block diagram for a healthy person (HP_A) and Type I diabetic patient (P_A) under insulin infusion pump are shown in Figure 6-2 and Figure 6-3 respectively. It illustrates a simplified glucose-insulin model by highlighting the gastrointestinal tract, liver, pancreas and subcutaneous tissue only. The detailed glucose-insulin model can be found in [139, 153]. In Figure 6-3, the pancreas of the diabetic patient is replaced by subcutaneous tissue where external insulin source is supplied.

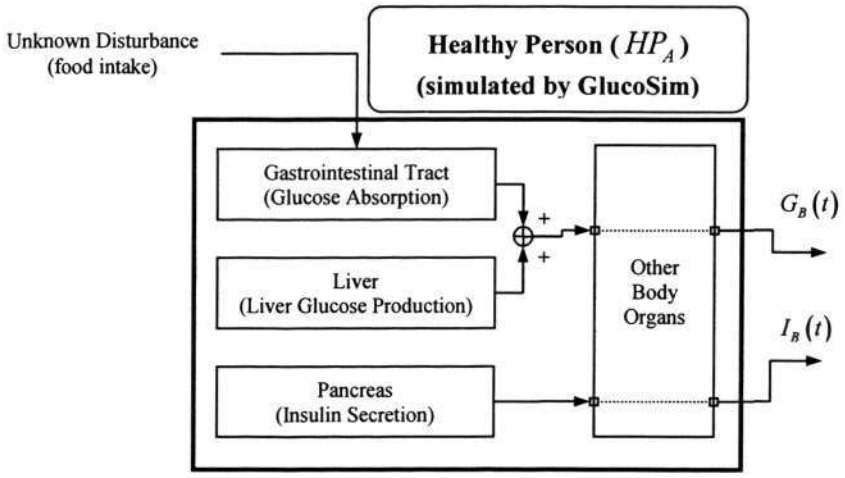


Figure 6-2: Block Diagram for Healthy Person (HP_A)

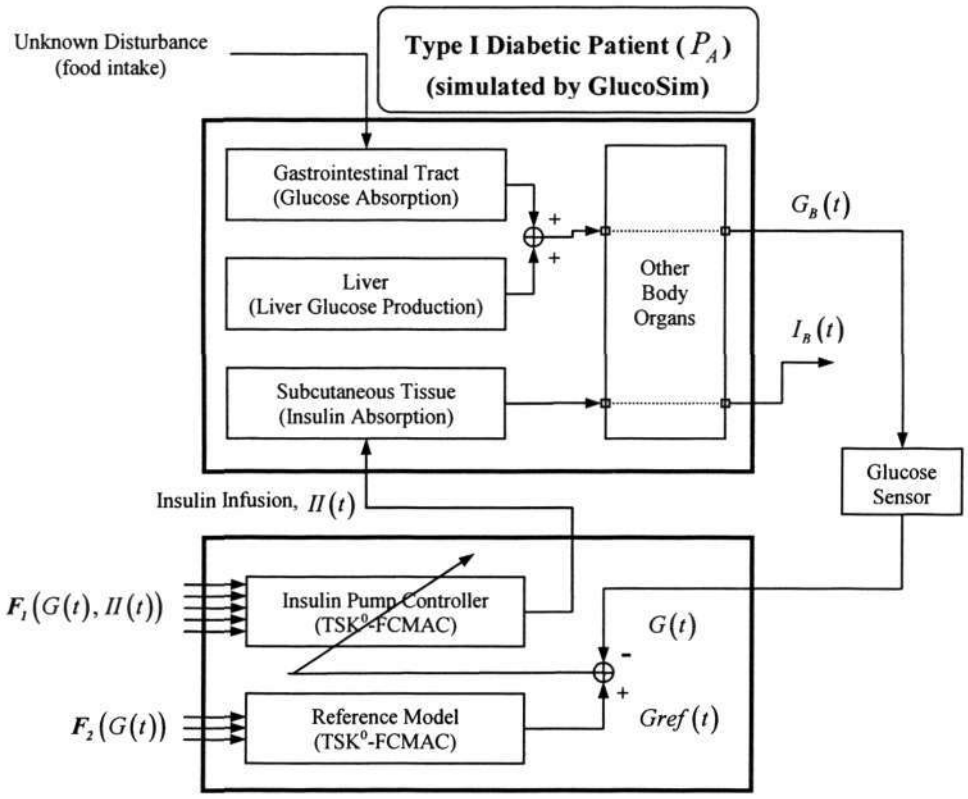


Figure 6-3: Block Diagram for Type I Diabetic Patients (P_A) under closed-loop Blood Glucose Control System

6.4. Results and Discussion

To reduce the risk of complications from diabetic mellitus, the life style of a Type I diabetic patient must be under continuous scrutiny. Much of these require human interventions and are susceptible to mistakes. As a result, the proposed closed-loop control system must be able to operate with minimum human intervention. Information such as food intake should not be included as part of the input to the system. With such limitations, our goal is to implement a closed-loop blood glucose control system that is capable of handling: 1) intra-personal variations, and 2) inter-personal variations.

The first TSK^0-FCMAC network (referred as the reference model) was trained by modeling the simulated response of a healthy person (HP_A). The training data are collected by simulating a healthy person under irregular dietary profile with GlucoSim. It is responsible to provide reference blood glucose level for the controller of the insulin infusion pump. The trained reference model is a three inputs TSK^0-FCMAC network with the size of 18 cells ($3\times2\times3$). The inputs of

the network are derived from historical reading of the glucose sensor ($G(t)$). The three inputs are represented in the form of $\{x_1(t), x_2(t), x_3(t)\}$ and are computed as Eqs (129)-(131).

$$x_1(t) = G_1(t) \tag{129}$$

$$x_2(t) = \begin{cases} sign\left(G_{1,4}(t)\right) \times \ln\left(\left|G_{1,4}(t)\right|\right) & ,if \left|G_{1,4}(t)\right| \le 1 \\ 0 & , \ otherwise \end{cases} \tag{130}$$

$$x_3(t) = \begin{cases} sign\left(H_{1,4}(t)\right) \times \ln\left(\left|H_{1,4}(t)\right|\right) & ,if \left|H_{1,4}(t)\right| \le 1 \\ 0 & , \ otherwise \end{cases} \tag{131}$$

$$G_i(t) = G\left(t - i \times T_{sampling}\right) \tag{132}$$

$$G_{i,j}(t) = G_i(t) - G_j(t) \tag{133}$$

$$H_{i,j}(t) = G_{i,j}(t) - G_{i,j}\left(t - T_{sampling}\right) \tag{134}$$

Where

$G(t)$ is the output from the glucose sensor at time t;

$T_{sampling}$ is the duration of one iteration in the simulation. It is equal to the sampling rate of blood glucose level (5 minutes);

$sign(.)$ is the function that return the sign value.

A second TSK^0-FCMAC network (referred as the controller), was subsequently trained to control the infusion rate for insulin infusion pump. As shown in Figure 6-3, the reference model and the controller were attached to a Type I diabetic patient (P_A). The controller started as an empty TSK^0-FCMAC network. As shown in Figure 3-4, input clusters were formed and trained online as new data arrived. Irregular diet was supplied to the patient P_A to cover the whole spectrum of his eating habit. The trained controller is a five input fuzzy CMAC network with the size of 13440 cells ($8 \times 8 \times 7 \times 6 \times 5$), which is equivalent to 13440 fuzzy if-then rules of Takagi-Sugeno-Kang's

type [18, 19]. Out of these rules, a total of 1761 rules are trained. The utilization ratio of the TSK^0-FCMAC network is hence equal to 1761/13440 = 13.1%. This utilization ratio is deemed as passable considering that the controller is trained to cover the whole spectrum of a patient's eating habit. The inputs of the network are derived from historical reading of the glucose sensor ($G(t)$) and the previous insulin infusion rate ($II(t)$). The five inputs are represented in the form of $\{x_1(t), x_2(t), x_3(t), x_4(t), x_5(t)\}$. The first three inputs are same as the inputs employed by the reference model (i.e. the first TSK^0-FCMAC network) and the last two inputs are computed as Eqs (135) and (136).

$$x_4(t) = \frac{\sum_{i=1}^{12} G_i(t)}{12} \tag{135}$$

$$x_5(t) = \frac{\sum_{i=1}^{12} II_i(t)}{12} \tag{136}$$

$$II_i(t) = II(t - i \times T_{sampling}) \tag{137}$$

Where

$II(t)$ is the infusion rate for insulin pump at time t.

With the trained controller, experimental cases are formulated to examine the controller's ability in handling intra and inter-personal variations. Each experimental case consists of 40 diets within the 10 days simulation. The same diet are simulated on both a healthy person (HP_A) and a Type I diabetic patient (P_A) for comparison. The block diagrams for HP_A and P_A under closed-loop blood glucose control system are shown in Figure 6-2 and Figure 6-3 respectively.

6.4.1. Intra-personal Variation

Individual tends to change his dietary habit. A rigid system that assumes a fixed regular diet may not be able to handle such situations. As a result, the first two experimental cases are designed to examine the ability of the proposed system in handling intra-personal variations. The same controller is used to control the insulin infusion rate of the same patient under different dietary profile. The experimental cases are listed as follows:

- Case 1, EC_1 : Apply the trained controller on P_A with regular eating habit, $diet_{P_A, CI, regular}$.
- Case 2, EC_2 : Apply the trained controller on P_A with irregular eating habit, $diet_{P_A, CI, irregular}$.

Where $CI \in \{normal, overeat, undereat, mixed\}$.

Figure 6-4 presents the results for EC_1 and EC_2. From the bar charts, it can be noticed that the controller is able to achieve a significantly shorter period of hyperglycemia ($Ind_{G(t)>180}$) for all dietary profiles (as shown in Figure 6-4(a) and (c)). On the other hand, there is only a slightly longer period of hypoglycemia ($Ind_{G(t)<60}$) when the patient is under dietary profile of under eating (as highlighted by A_1 and A_2 in Figure 6-4(b) and (d)).

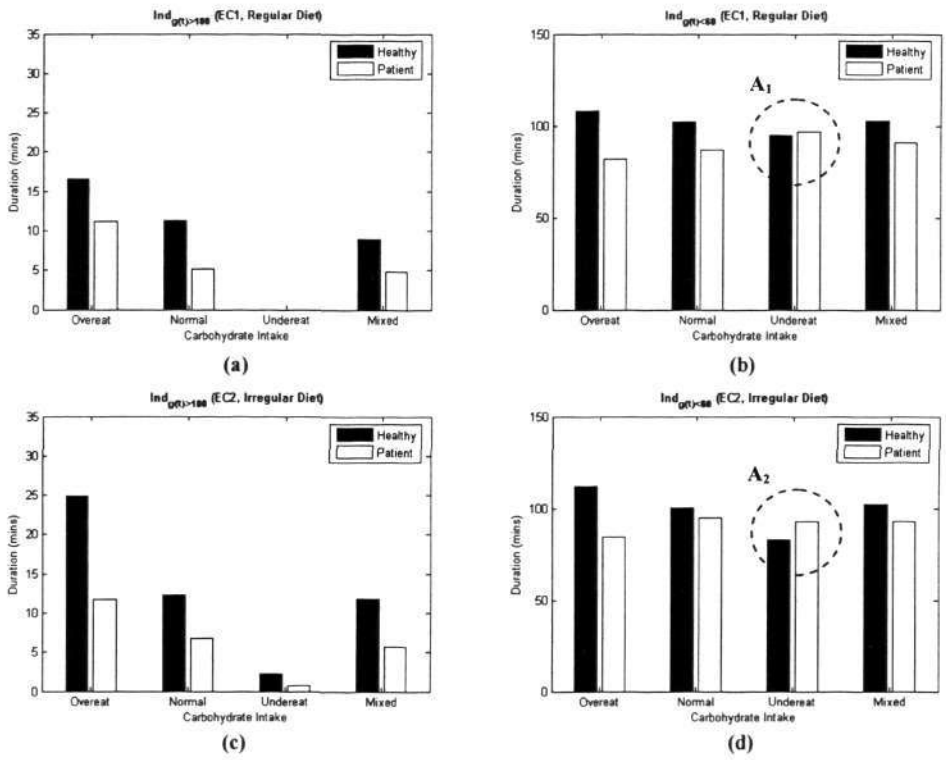


Figure 6-4: Results for EC_1 and EC_2 (70 kg)

Figure 6-5 shows the 10 days simulated results for HP_A and P_A under the same dietary profile, $diet_{P_A, mixed, irregular}$. Figure 6-5(a) shows their carbohydrates intake. Figure 6-5(b) and (c) show the respective simulated response of HP_A and P_A under the same carbohydrates intake. The simulated blood glucose level is represented by solid line and the simulated blood insulin level is represented by dotted line. The two horizontal lines indicate the upper acceptable limit (180 mg/dL) and lower acceptable limit (60 mg/dL) for the blood glucose level. The insulin infusion rate for P_A are shown in Figure 6-5(d). Figure 6-5 shows that the simulated response for P_A under insulin infusion pump is comparable with that of a healthy person. The blood glucose level is bounded within its limit under different carbohydrate intake and meal time.

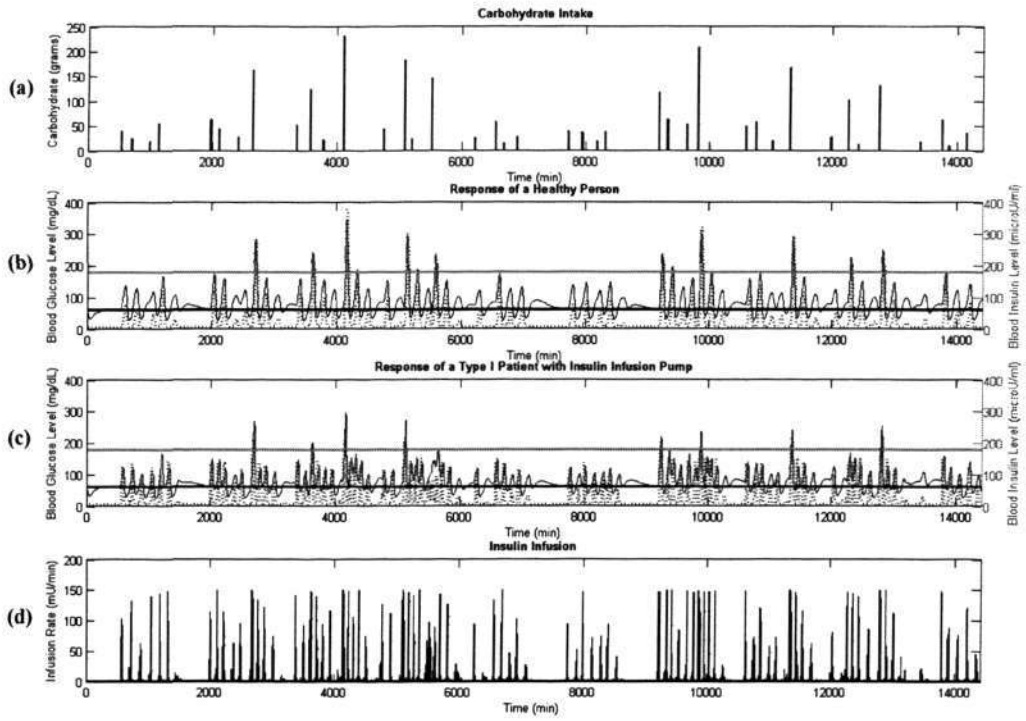


Figure 6-5: Ten Days Response of HP_A and P_A under Dietary Profile $diet_{P_A,mixed,irregular}$

For further analysis, the simulated response of day 3 is shown in Figure 6-6. Day 3 is chosen because one of its meals contains the highest carbohydrate intake over the 10 days period. The simulated blood glucose level exceeded the upper limit of 180 mg/dL in two instances due to huge amount of carbohydrate intake. As indicated in the charts, the peak of P_A (201.40 & 297.34 mg/dL) is significantly lower than that of the healthy person (242.79 & 346.26 mg/dL). The shorter duration of hyperglycemia is mainly because of the early detection of possible rising glucose level. As shown in the simulated response for P_A, the controller is able to anticipate the increase of the insulin infusion rate before the rising of blood glucose level. It can also be observed that the duration of hypoglycemia in P_A is slightly shorter than that in HP_A.

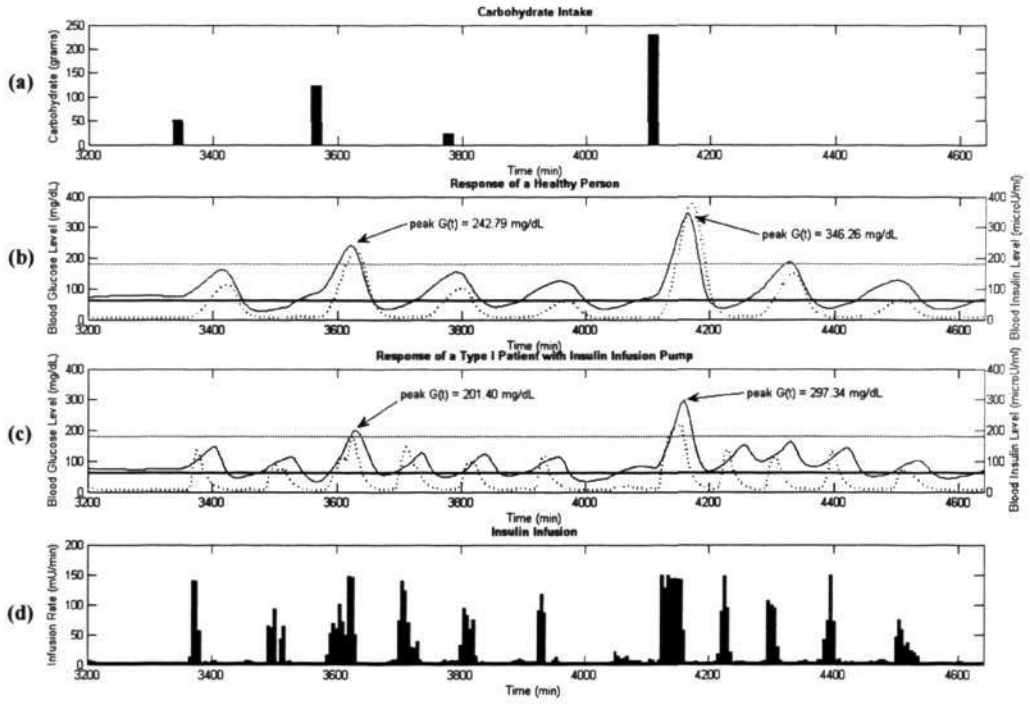


Figure 6-6: Response of Day 3 for HP_A and P_A under Dietary Profile $diet_{P_A,\,mixed,\,irregular}$

As mentioned previously, the duration of hypoglycemia ($Ind_{G(t)<60}$) for patient under dietary profile of $diet_{P_A,\,undereat,\,EH}$ is slightly longer than that of a healthy person (refer to Figure 6-4(b) and (d)). As part of the analysis, the 10 days simulated response under dietary profile of $diet_{P_A,\,undereat,\,irregular}$ is shown in Figure 6-7. Closer observation on a single day response (refer to Figure 6-8) suggests that the response of P_A is acceptable. The reason of higher $Ind_{G(t)<60}$ is mainly because of additional occurrence (per diet) of having the glucose level decreases beyond the lower limit of 60 mg/dL. Nevertheless, the duration of each occurrence is comparable with that of a healthy person and no prolonged hypoglycemic condition is detected.

As discussed previously, EC_1 and EC_2 are designed to examine the ability of the proposed system in handling intra-personal variations. Simulated results show that the proposed system is able to handle the glucose-insulin dynamics of the same patient under all different dietary profiles.

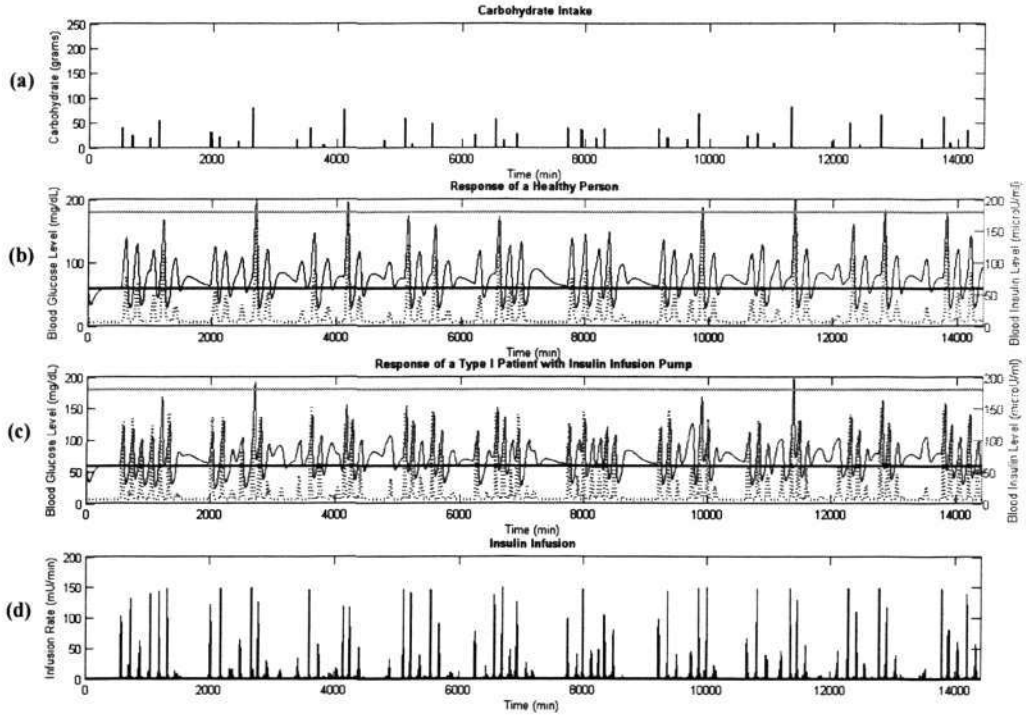


Figure 6-7: Ten Days Response of HP_A and P_A under Dietary Profile $diet_{P_A, undereat, irregular}$

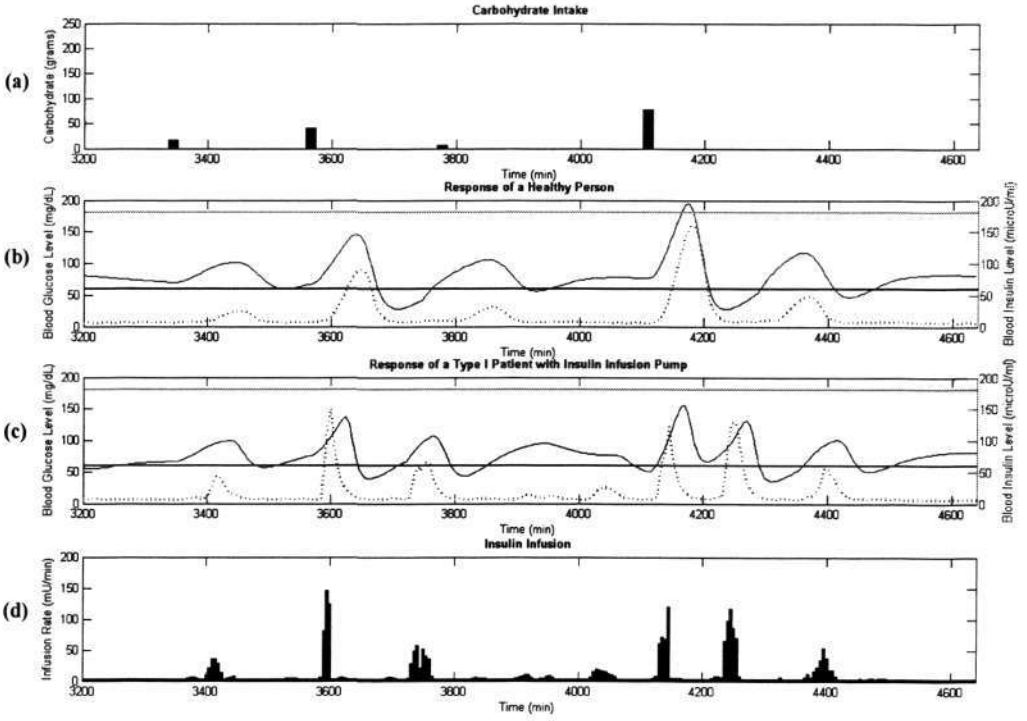


Figure 6-8: Response of Day 3 for HP_A and P_A under Dietary Profile $diet_{P_A, undereat, irregular}$

6.4.2. Inter-personal Variation

The glucose-insulin dynamics simulated by GlucoSim [153] are average values based on individual body weight. To study inter-personal variations under the proposed control regime, two different patients, P_B and P_C are formulated and are shown in Table 6-4. They are the same as P_A except having different body weight ($\pm 5\%$).

Table 6-4: Patient B and Patient C's Profile

Patient	P_B	P_C
Weight	73.5 kg	66.5 kg
Body Mass Index	24.14	21.84
Daily Calorie Needed	2014 kcal	1899 kcal
Corresponding Daily Carbohydrate Intake	287 grams	270 grams

Four additional experimental cases are designed to examine the ability of the controller (trained for P_A) to adapt to the glucose-insulin dynamics of different patients (P_B and P_C). They are listed as follows:

- Case 3, EC_3 : Apply the trained controller on P_B with regular eating habit, $diet_{P_B, CI, regular}$.
- Case 4, EC_4 : Apply the trained controller on P_B with irregular eating habit, $diet_{P_B, CI, irregular}$.
- Case 5, EC_5 : Apply the trained controller on P_C with regular eating habit, $diet_{P_C, CI, regular}$.
- Case 6, EC_6 : Apply the trained controller on P_C with irregular eating habit, $diet_{P_C, CI, irregular}$.

Where $CI \in \{normal, overeat, undereat, mixed\}$.

Figure 6-9 and Figure 6-10 show the results of applying the controller trained for P_A on different patients, P_B and P_C. The results are analogous to that of EC_1 and EC_2 in Figure 6-4. The

duration of hyperglycemia ($Ind_{G(t)>180}$) for diabetic patients is shorter than that of the healthy person for all dietary profiles. However, there are two cases where the duration of $Ind_{G(t)<60}$ for patient is notably longer than that of a healthy person. They are:

- P_B (112.00 minutes) under dietary profile of $diet_{P_B, undereat, regular}$ versus HP_B (89.75 minutes) in EC_3 (as highlighted by A_3 of Figure 6-9(b)) and
- P_C (118.00 minutes) under dietary profile of $diet_{P_C, undereat, irregular}$ versus HP_C (92.50 minutes) in EC_6 (as highlighted by A_4 of Figure 6-10(d)).

For further investigation, the simulated responses of P_B and P_C over the 10 days period are presented in Figure 6-11 and Figure 6-12 respectively. Likewise, the higher $Ind_{G(t)<60}$ is mainly due to the additional occurrence (per diet) of having the glucose level decreases beyond the lower limit. An example for such observation can be shown by comparing the region highlighted by B_1 and B_2 in Figure 6-11. There are five occurrences of having the glucose level decreases beyond the lower acceptable limit (60 mg/dL) in region B_1. However, there are six occurrences in region B_2 even though they are under the same carbohydrate intake. Similar observation is highlighted by region B_3 (4 occurrences) and B_4 (6 occurrences) in Figure 6-12. Nevertheless, no prolonged hypoglycemic condition is detected in both cases. These results show no evidence of instability in the system even though the controller is applied on patients with different profile ($\pm 5\%$ in body weight).

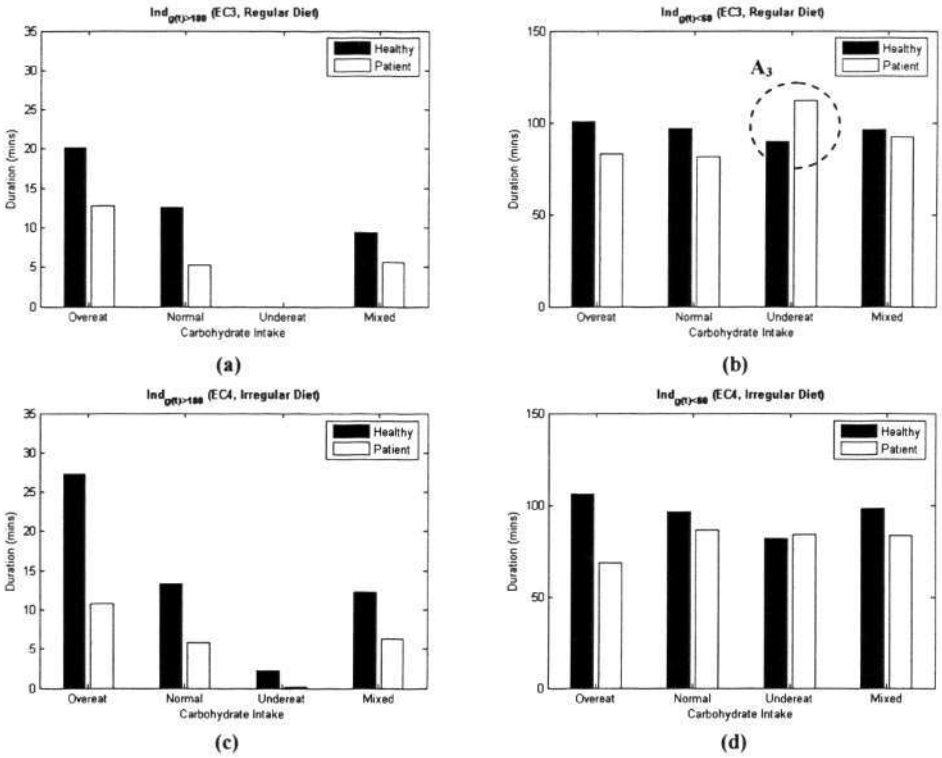


Figure 6-9: Results for EC_3 and EC_4 (73.5 kg)

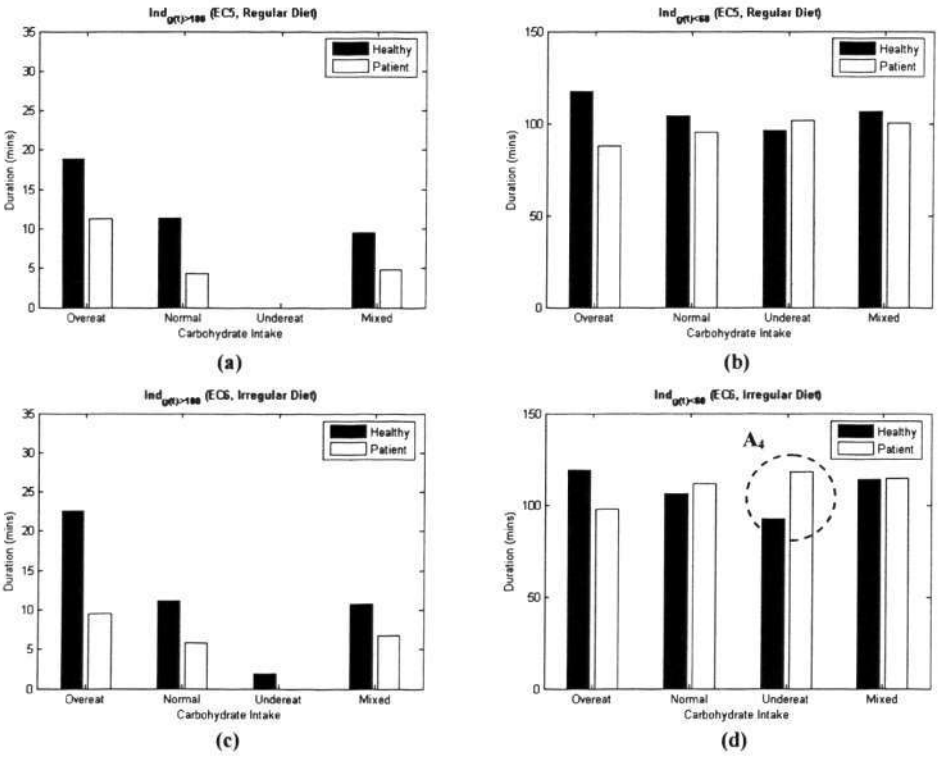


Figure 6-10: Results for EC_5 and EC_6 (66.5 kg)

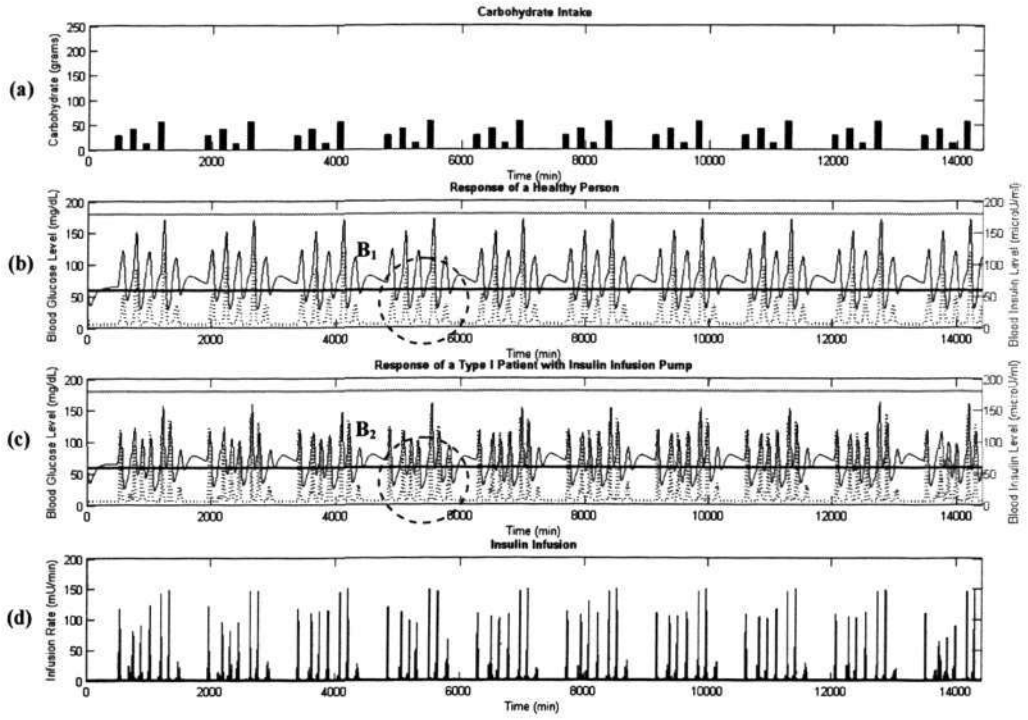


Figure 6-11: Ten Days Response of HP_B and P_B under Dietary Profile $diet_{P_B, undereat, regular}$

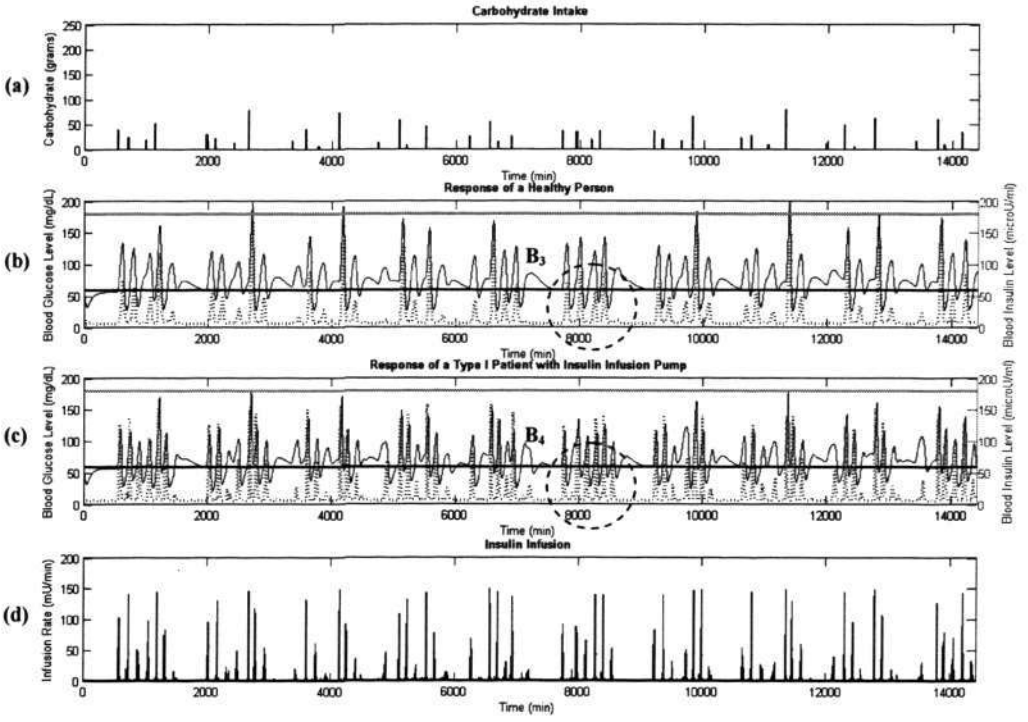


Figure 6-12: Ten Days Response of HP_C and P_C under Dietary Profile $diet_{P_C, undereat, irregular}$

6.5. Summary

A novel blood glucose regulation for Type I diabetes patient using a fuzzy CMAC based controller is presented. Without prior knowledge of disturbance (e.g. food intake), the proposed fuzzy CMAC is able to capture the glucose-insulin dynamics of individuals with different dietary profiles. The trained controller is a five input fuzzy CMAC network with a size of 13440 cells ($8\times8\times7\times6\times5$), which is equivalent to 13440 fuzzy if-then rules of Takagi-Sugeno-Kang's type [18, 19]. Out of these rules, a total of 1761 rules are trained. The utilization ratio of the TSK^0-FCMAC network is hence equal to 1761/13440 = 13.1%. This utilization ratio is deemed as passable considering that the controller is trained to cover the whole spectrum of a patient's eating habit. Nevertheless, preliminary simulations show that the proposed system is capable of handling both intra and inter-personal variations. The design of the proposed system follows closely to what is available in real life and hence suitable for animal and clinical pilot testing in near future.

Chapter 7 Hydroinformatics Case Study: Rainfall Runoff

7.1. Introduction

The process of analyzing an unsteady flow event through a sewer system is referred to as "routing the flow". The National Engineering Handbook [157] defines routing as "computing the flood at a downstream point from the flood at an upstream point, taking storage into account". The flow of water through the soil and stream channels of a watershed is a distributed process because the flow rate, velocity, and depth vary in space throughout the watershed. Estimates of the flow rate or water level at important locations in the channel system can be obtained using a distributed flow routing model [158]. Distributed flow routing model can be employed to describe the transformation of storm rainfall into runoff over a channel. Such a model can be utilized to generate a flow hydrograph that could be considered as inflow at the upstream end of another channel and route it to the downstream end. The velocity in a channel varies in three perspectives; namely: 1) along the channel, 2) across the channel and, 3) from the water surface to the channel's bottom. For practical purposes, only the variation along the channel is considered in the routing model. The routing of an unsteady open channel flow is well described by a constitutive relationship, the Saint-Venant equations [159]. However, the Saint-Venant equations cannot be solved analytically. The kinematic wave model (KWM), a simplified model derived from the Saint-Venant equations, offers an alternate well-established numerical solution for modeling the flow of an open channel.

However, modeling techniques such as the kinematic wave model require accurate information on the physical aspect of the channel, such as the Manning's roughness coefficient, the width and slope of the channel and, the height of the flow. On the other hand, many artificial intelligence (AI) based techniques have also been applied in hydrological modeling. In contrast to modeling technique derived from constitutive relationship such as the Saint-Venant equations, AI techniques predict the hydrograph of a channel based on monitored data from past experience, which can be easily obtained by measurement equipments. In the recent years, applications on rainfall runoff with AI techniques from various fields have been proposed; namely: 1) neural

networks [160-163], 2) Bayesian networks [164, 165], 3) fuzzy systems [166-169] and, 4) genetic algorithm [170-174].

This chapter will first describe the fundamental model in open channel routing, the Saint-Venant equations, the Manning's equation, the kinematic wave model and its numerical solutions. In the experiments, the proposed fuzzy associative memory architecture, the TSK^0-FCMAC network was applied to predict the hydrograph at the end of the channel. The experimental data are collected by Wong [175] from an outdoor experimental plot set up at Nanyang Technological University, Singapore. It consists of the hydrograph measured at the end of the channel and the rainfall intensity measured at 15 seconds intervals for ten different storm events. The objective is to predict the hydrograph at the end of the channel according to the measured rainfall intensity. The hydrograph at the end of the channel is also modeled by the kinematic wave model. The numerical method to solve the kinematic wave model is presented in section 7.1.4 in details. The performances of the proposed architecture are evaluated with the prediction results from the kinematic wave model.

7.1.1. Saint-Venant Equations

The "principle of conservation of mass" states that matter is neither created nor destroyed. The mass entering a system is equal to the mass leaving that system, plus or minus the accumulation of mass (that is, storage) within the system. Without doubt, this principle is the single most important concept that must be applied in hydrologic and hydraulic engineering [176]. The Saint-Venant equations [159] proposed by Barre de Saint-Venant in 1871 describe the unsteady open channel flow in one-dimension, that is, along the direction of flow. The Saint-Venant equations are summarized as Eqs (138) and (139) in [158].

The continuity equation,

$$\frac{\partial Q}{\partial d}+\frac{\partial A}{\partial t}=0 \qquad (138)$$

and the momentum equation,

$$\underbrace{\frac{1}{A}\frac{\partial Q}{\partial t}}_{\text{Local inertia term}} + \underbrace{\frac{1}{A}\frac{\partial}{\partial t}\left(\frac{Q^2}{A}\right)}_{\text{Convective inertia term}} + \underbrace{g\frac{\partial y}{\partial d}}_{\text{Pressure force term}} - \underbrace{g\left(S_o - S_f\right)}_{\text{Gravity force term}} \underbrace{= 0}_{\text{Friction force term}} \quad (139)$$

Where

Q is the discharge rate (cfs, m^3/s);

d is the distance along the longitudinal axis of the channel (ft, m);

A is the wetted cross-sectional area of flow (ft^2, m^2);

t is the time (s);

g is gravitational acceleration constant (ft^2/s, m^2/s);

y is the depth of the flow (ft, m);

S_o is the channel slope (ft/ft, m/m);

S_f is the energy gradient. It is equal to the channel slope S_o in the case of uniform flow.

However, the Saint-Venant equations cannot be solved analytically and as a result, a variety of numerical solutions have been developed over the years [176]. One of the numerical solutions is the Muskingum routing proposed by McCarthy in 1938 [177, 178]. The computation of the Muskingum routing method is based on the continuity equation and the storage-discharge relation [179]. In 1969, Cunge [180] extended the Muskingum routing method into a finite-difference scheme called the Muskingum-Cunge method [181, 182]. In 1972, Mockus and Styner proposed another routing model, the Convex routing [183] which uses only flows from the previous time step to determine the outflow.

Among these numerical solutions, the kinematic wave approach derives from the one-dimensional continuity and momentum equations (the Saint-Venant equations) describes the physical processes of the movement of water. The concept of kinematic wave is well established among

the existing methods to solve unsteady, one-dimensional, gradually varied open channel flow problems [184]. Manual solutions of kinematic wave formulations in relation to hydrologic design are impractical. Nevertheless, they have been incorporated into the Penn State Runoff Model [185], HEC-1 [186], as an option for routing overland flow. The kinematic wave model is described in details in the following section.

7.1.2. Kinematic Wave Model (KWM)

In modeling an unsteady flow in open channel, two simplified models are derived from the Saint-Venant equations. The first type simplifies the equations by neglecting the local inertia, convective inertia and the pressure force terms [187]. A second, less restrictive type is formulated by neglecting the local inertia and convective inertia terms only [188]. The first type is referred to as kinematic wave model and the second type is referred to as diffusion wave [189]. Kinematic waves were originally used for describing river flows [190]. As the inertia and pressure terms in the momentum equation (of the Saint-Venant equations) are considered negligible, the wave motion is described principally by the equation of continuity. The name kinematic is thus applicable, as kinematics refers to the study of motion exclusive of the influence of mass and force [158]. The kinematic wave model is defined as Eqs (140) and (141).

The continuity equation,

$$\frac{\partial Q}{\partial d} + \frac{\partial A}{\partial t} = q \tag{140}$$

and the momentum equation,

$$S_o = S_f \tag{141}$$

Where

Q is the discharge rate (cfs, m^3/s);

d is the distance along the longitudinal axis of the channel (ft, m);

A is the cross-sectional area of flow (ft^2, m^2);

t is the time (s);

q is the lateral inflow or outflow (ft^2/s, m^2/s);

S_o is the channel slope (ft/ft, m/m);

S_f is the energy gradient. It is equal to the channel slope S_o in the case of uniform flow.

The momentum equation can also be expressed in the form of

$$A = \alpha Q^{\beta} \tag{142}$$

where parameters α and β are determined by the formula employed to estimate the discharge rate of the channel. For gradually varied flow in an open channel, the depth and velocity of a steady flow change along the length of the channel. As the name implies, the rates of change of depth and velocity are gradual and there are no abrupt changes such as hydraulic jumps. Because of this, the constitutive relationships developed for uniform flow, such as the Cheźy's formula or the Manning equation, can be assumed to be valid for analyses of gradually varied flow [176].

The Cheźy's formula was derived by Antoine Cheźy while designing an improvement for the water system in Paris, France in the period of 1768-1775 [191]. The formula relates the uniform velocity of flow to the hydraulic radius and the longitudinal slope of the channel. The Manning equation [192] is an empirically derived expression obtained from observations of flows in laboratory channels by an Irish engineer Robert Manning in 1889. It was developed as a formula for uniform flow but has also been successfully applied to analyze gradually and spatially varied flows. The Manning's equation is one of the most widely used of all constitutive relationships for open-channel-flow analysis [176]. In this section, the discussion will focus on the Manning's equation.

7.1.3. Manning's Equation

The Manning's equation can be expressed as Eq (143):

$$V = \frac{\phi}{n} R_h^{2/3} \sqrt{S_f} \tag{143}$$

Where

V is the flow velocity (ft/s, m/s);

ϕ is the unit conversion factor (1.49 if English units are used; 1.00 if SI units are used);

n is the Manning coefficient of roughness;

R_h is the hydraulic radius (ft, m);

S_o is the channel slope (ft/ft, m/m);

S_f is the energy gradient. It is equal to the channel slope S_o in the case of uniform flow.

For rectangular channel, the hydraulic radius R_h can be expressed as

$$R_h = \frac{A}{P} \tag{144}$$

and

$$P = b + 2y \tag{145}$$

Where

A is the cross-sectional area of flow (ft^2, m^2);

P is the wetted perimeter (ft, m);

b is the width of the channel (ft, m);

y is the depth of the flow (ft, m).

The discharge rate Q is defined as

$$Q = VA \tag{146}$$

Substituting Eqs (141), (144) and (146) into Eq (143), the Manning's equation can be further derived as

$$\begin{aligned} \frac{Q}{A} &= \frac{\phi}{n}\left(\frac{A}{P}\right)^{2/3}\sqrt{S_o} \\ \Leftrightarrow A &= \left(\frac{nP^{2/3}}{\phi\sqrt{S_o}}\right)^{3/5} Q^{3/5} \end{aligned} \tag{147}$$

Referring to Eq (142), the momentum equation can now be expressed as Eq (147) where

$$\alpha = \left(\frac{nP^{2/3}}{\phi\sqrt{S_o}}\right)^{3/5} \tag{148}$$

and

$$\beta = 3/5 = 0.6 \tag{149}$$

Eq (142) consists of two dependent variables, A and Q. Differentiating Eq (142) gives

$$\frac{\partial A}{\partial t} = \alpha\beta Q^{\beta-1}\frac{\partial Q}{\partial t} \tag{150}$$

Substitute Eq (150) into Eq (140) gives

$$\frac{\partial Q}{\partial d} + \alpha\beta Q^{\beta-1}\frac{\partial Q}{\partial t} = q \tag{151}$$

where Q is now the only dependent variable.

7.1.4. Numerical Solution for Kinematic Wave Model (KWM)

Combined with the Manning's equation, the kinematic wave can now be represented with Q as the only dependent variable as shown in Eq (151) . Given the upstream's hydrograph, the goal is to solve Eq (151) and determine the downstream's hydrograph. That is, to determine $Q_{d+1}(t+1)$ from $Q_d(t+1)$ and $Q_{d+1}(t)$. A graphical representation is shown in Figure 7-1.

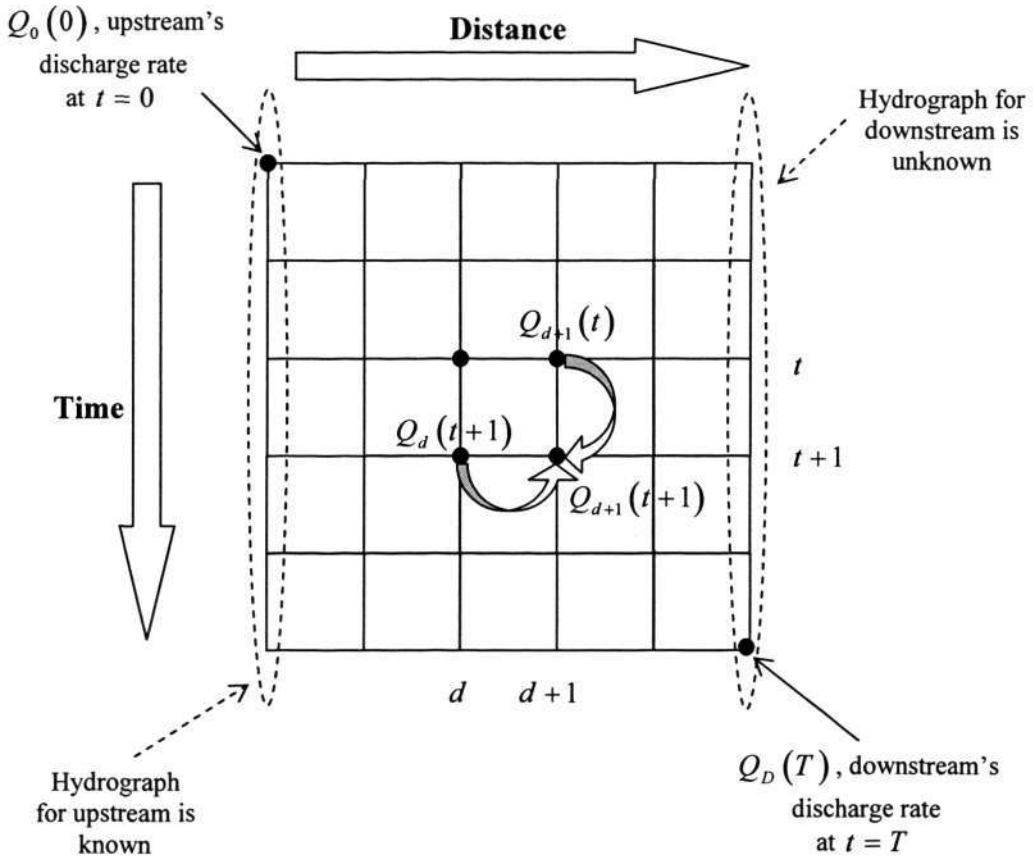


Figure 7-1: A Graphical Representation for Kinematic Wave Model

Given the channel parameters α and β, the lateral inflow $q(t)$, and the initial condition, a linear numerical solution can be derived to estimate the outflow hydrograph. The outflow discharge rate Q is estimated by averaging the values across the diagonal grid as shown in Figure 7-1. This can be expressed as Eqs (152)-(154).

$$\frac{\partial Q}{\partial d} \approx \frac{Q_{d+1}(t+1) - Q_d(t+1)}{\Delta d} \tag{152}$$

$$\frac{\partial Q}{\partial t} \approx \frac{Q_{d+1}(t+1) - Q_{d+1}(t)}{\Delta t} \tag{153}$$

$$Q \approx \frac{Q_d(t+1) + Q_{d+1}(t)}{2} \tag{154}$$

Where

$Q_d(t)$ is the discharge rate (cfs, m^3/s) of point d at time t;

Δd is the length of the equal sized grid (m);

Δt is the time difference between iterations (s).

The value of lateral inflow q can also be derived as

$$q \approx \frac{q_d(t+1)+q_{d+1}(t)}{2} \tag{155}$$

Where

$q_d(t)$ is the lateral inflow or outflow (ft^2/s, m^2/s) of point d at time t.

Substitute Eqs (152)-(155) into Eq (151) gives

$$\begin{aligned}
&\frac{\partial Q}{\partial d}+\alpha\beta Q^{\beta-1}\frac{\partial Q}{\partial t}=q \\
&\Leftrightarrow \frac{Q_{d+1}(t+1)-Q_d(t+1)}{\Delta d}+\alpha\beta\left(\frac{Q_d(t+1)+Q_{d+1}(t)}{2}\right)^{\beta-1}\frac{Q_{d+1}(t+1)-Q_{d+1}(t)}{\Delta t} \\
&\quad =\frac{q_d(t+1)+q_{d+1}(t)}{2} \\
&\Leftrightarrow \frac{Q_{d+1}(t+1)}{\Delta d}+\alpha\beta\left(\frac{Q_d(t+1)+Q_{d+1}(t)}{2}\right)^{\beta-1}\frac{Q_{d+1}(t+1)}{\Delta t} \\
&\quad =\frac{Q_d(t+1)}{\Delta d}+\alpha\beta\left(\frac{Q_d(t+1)+Q_{d+1}(t)}{2}\right)^{\beta-1}\frac{Q_{d+1}(t)}{\Delta t}+\frac{q_d(t+1)+q_{d+1}(t)}{2} \\
&\Leftrightarrow Q_{d+1}(t+1) \\
&\quad =\frac{\frac{\Delta t}{\Delta d}Q_d(t+1)+\alpha\beta\left(\frac{Q_d(t+1)+Q_{d+1}(t)}{2}\right)^{\beta-1}Q_{d+1}(t)+\Delta t\left(\frac{q_d(t+1)+q_{d+1}(t)}{2}\right)}{\left[\frac{\Delta t}{\Delta d}+\alpha\beta\left(\frac{Q_d(t+1)+Q_{d+1}(t)}{2}\right)^{\beta-1}\right]}
\end{aligned} \tag{156}$$

Where the α and β are channel parameters defined in Eqs (148) and (149) and Eq (156) represents the numerical method to determine $Q_{d+1}(t+1)$ from $Q_d(t+1)$ and $Q_{d+1}(t)$.

7.2. Experiments

7.2.1. Experimental Data Set

The experimental data are collected by Wong [175] from an outdoor experimental plot set up at Nanyang Technological University, Singapore. Figure 7-2 shows the snapshots for the outdoor experimental plot and the details of the experimental plot are described in [193] The experimental plot has four 25 meters x 1 meter open channels as shown in Figure 7-3. The data collected are based on the concrete channel with a bottom slope of 2% surrounded by 1 meter high concrete wall along the channel (i.e. Bay 1). The collected experimental data consists of the hydrographs measured at the end of the channel and the rainfall intensities measured at 15 seconds intervals for ten different storm events. The objective is to predict the hydrograph at the end of the channel according to the measured rainfall intensity along the channel. Table 7-1 shows a summary of the ten storm events recorded by Wong, a more detailed descriptions can be found in [175].

Table 7-1: Summary of Rainfall and Runoff Data for Ten Storm Events

Event	Event Date	Event Duration (min)	Rainfall			Runoff from Bay 1 (2% Concrete Plane)		
			Peak Intensity (mm/hr)	Average Intensity (mm/hr)	Total Depth (mm)	Peak Discharge (L/s)	Average Discharge (L/s)	Total Depth (mm)
1	6-Oct-2002	40	144.0	33.4	22.27	0.641	0.194	18.77
2	14-Oct-2002	80	144.0	19.9	26.56	0.758	0.118	22.69
3	27-Oct-2002	40	216.0	43.2	28.78	0.912	0.241	23.29
4	3-Nov-2002	120	144.0	17.5	35.00	0.745	0.105	30.24
5	13-Nov-2002	30	144.0	32.6	16.3	0.601	0.182	13.22
6	17-Nov-2002	100	144.0	25.7	42.89	0.730	0.158	38.07
7	18-Nov-2002	120	96.0	8.4	16.81	0.495	0.054	15.72
8	22-Nov-2002	90	120.0	19.0	28.57	0.601	0.121	26.24
9	5-Dec-2002	100	144.0	16.6	27.6	0.832	0.111	26.76
10	7-Dec-2002	100	72.0	11.9	19.88	0.352	0.075	17.96

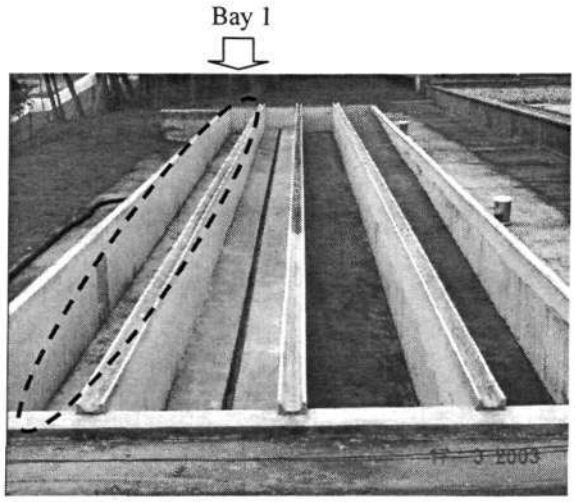


Figure 7-2: Outdoor Experimental Plot in Nanyang Technological University, Singapore

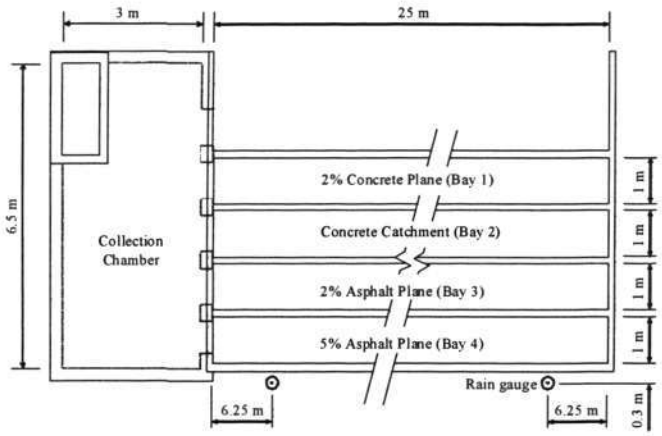


Figure 7-3: Plan view of experimental plot

As described in section 7.1.4, the numerical method of kinematic wave model involves the computation of the next discharge rate, $Q_{d+1}(t+1)$ from the discharge rate of the previous grid, $Q_d(t+1)$, the discharge rate of the previous time step, $Q_{d+1}(t)$ and the lateral inflow, $q_d(t)$. In this experiment, the only source for lateral inflow is the rain water. However, to predict the discharge rate with ANNs, features must be extracted from historical rainfall data. In this experiment, three features are selected as inputs to the proposed architecture to predict the discharge rate. The training data samples are in the form of

$$\left[x_1(t), x_2(t), x_3(t) \rightarrow Q(t+1)\right] \tag{157}$$

and

$$x_1(t) = \frac{\sum_{k=1}^{6} I(t-k)}{6} \tag{158}$$

$$x_2(t) = I(t-3) \tag{159}$$

$$x_3(t) = I(t-4) \tag{160}$$

Where

$I(t)$ is the rainfall (mm/hr) at time t and $\Delta t = 15$ seconds;

7.2.2. Parameters for Kinematic Wave Model (KWM)

The goal of the experiment is to predict the hydrograph at the end of the channel according to the measured rainfall intensity by the proposed architecture, TSK^0-FCMAC. The hydrograph at the end of the channel is also modeled by the kinematic wave model (KWM). The results from the proposed architecture are evaluated against the prediction results from the kinematic wave model. The kinematic wave model is computed by the numerical method presented in section 7.1.4. The channel is segmented into equal sized grids in the length of Δd. In terms of the time axis, iterations are Δt apart. In the experiments, the kinematic wave model adopted Δd of one meter and Δt of one second. For the selection of the roughness coefficient of concrete channel (n), it is usually based on "best engineering judgment" or on values prescribed by municipal design ordinances. Several tables are available in the general literature for the selection of Manning's roughness coefficient of a particular open channel [179]. The most common values recommended for concrete open channel are between the range of 0.010 to 0.015 [158, 176, 178, 179, 194, 195]. However, as recommended by Engman [194], the Manning's roughness coefficient of 0.011 is suitable for concrete surface in experimental plot.

The parameters adopted for the kinematic wave model are:

- Channel width, b = 1 meter
- Channel full length = 25 meters

- Channel slope, $S_o = 0.02$ m/m
- Manning's roughness coefficient, $n = 0.011$
- Computational subsegment, $\Delta d = 1$ meter
- Computational time step, $\Delta t = 1$ seconds

7.2.3. Loss Model for Kinematic Wave Model

In the work investigated by Wong [175], several loss models are proposed. These loss models aim to compensate the losses of effective rain water of various reasons: the interception by the concrete walls, the "time lag" and the "initial loss" [175]. In this experiment, two types of kinematic wave model are adopted:

- Type I KWM: kinematic wave model with no loss model (zero loss rate) and,
- Type II KWM: kinematic wave model with "upperbound loss model" presented by Wong.

In [175], Wong shows that the hydrographs simulated by Type II KWM are closer to the measured hydrographs. However, one criticism on employing the upperbound loss model in the computation of kinematic wave model is the need of calibration of the loss rate. Even though for the same channel, the loss rate calibrated varies among different storm events. Nevertheless, taking loss rate into consideration, the effective rainfall is computed as Eq (161).

$$I_{Effective}(t) = \max\left\{\left[I_{Observed}(t) - I_{Loss\,Rate}\right], 0\right\} \tag{161}$$

and the upperbound loss rate $I_{Loss\,Rate}$ is computed with the following equality:

$$\sum\left\{\left[I_{Observed}(t) - I_{Loss\,Rate}\right] \times \Delta t\right\} = Depth_{runoff} \tag{162}$$

Where

$I_{Effective}(t)$ is the effective rainfall (mm/hr) at time t;

$I_{Observed}(t)$ is the observed (measured) rainfall (mm/hr) at time t;

$I_{Loss\ Rate}$ is the loss rate (mm/hr);

$\max(x_1, x_2)$ is the function that return the maximum between x_1 and x_2;

$Depth_{runoff}$ is the runoff depth (mm).

7.2.4. Performance Indicators

The performances of the prediction are evaluated based on three performance indicators; namely: 1) RMSE value, 2) Pearson's correlation and, 3) R^2 function. The RMSE values and Pearson's correlation coefficient are the most commonly used performance indicators in the computational intelligence community. However, due to the fact that the discharge rates for individual event are different in terms of magnitude, the RMSE values may not be a good performance indicator. On the other hand, the R^2 function, as described by Nash [196], provides a better measurement on the performance of the prediction. The R^2 function is computed using Eq (163). Throughout the experiment, the R^2 function will be employed as the primary performance indicator. The RMSE value and Pearson's correlation will be employed to provide supporting statistics.

$$R^2 = 1 - \frac{\sum_{t=1}^{T}\left[Q_o(t) - Q_p(t)\right]^2}{\sum_{t=1}^{T}\left[Q_o(t) - Q_m\right]^2} \tag{163}$$

Where

$Q_o(t)$ is the observed (measured) discharge rate at time t;

$Q_p(t)$ is the predicted discharge rate at time t;

Q_m is the mean discharge rate of the event.

Two sets of experiments are conducted:

- *Memory Recall*^{a} : using the rainfall data extracted from Event X for both training and testing;

- *Generalization*^{b}: using the rainfall data extracted from Events 3 and 6 for training and the rainfall data extracted from Event X for testing;

and $X = 1, 2, ..., 10$.

For the TSK^0-FCMAC network to generalize, the training samples extracted from a single event will not be sufficient. Therefore, the training samples for *Generalization*^{b} are extracted from two events: Events 3 and 6. Event 3 is selected because it consists of the highest rainfall intensity while Event 6 consists of multiple peaks of different rainfall intensities (refer to Figure 7-4(c) and (f)).

7.3. Results and Discussion

As described in section 7.2.1, the collected experimental data consists of the hydrographs for ten different storm events. The goal of the experiment is to predict the hydrograph at the end of the channel according to the measured rainfall intensity along the channel. In *Memory Recall*^{a}, TSK^0-FCMAC was applied to predict the discharge rate of the next time step according to historical rainfall intensities. This was achieved by conducting memory recall operations with TSK^0-FCMAC, that is, the training and testing data samples are extracted from the same storm event. The data presented to TSK^0-FCMAC are in the form of $\left[x_1(t), x_2(t), x_3(t) \rightarrow Q(t+1)\right]$ (as described in section 7.2.1).

To evaluate the performance of TSK^0-FCMAC, the discharge rates are also computed using a conventional numerical method in hydroinformatics, the kinematic wave model. The results predicted by both TSK^0-FCMAC and kinematic wave model (Type I and Type II KWM) are presented in Table 7-2. From the last row of Table 7-2, it can be observed that the average RMSE value from TSK^0-FCMAC is smaller than that of Type I and Type II KWM (0.0244 versus 0.0401 and 0.0266). In addition, TSK^0-FCMAC is able to achieve higher average Pearson's correlation coefficient (0.9900 versus 0.9885 and 0.9887).

With regards to the primary performance indicator, the R^2 value, Type I KWM is able to achieve an average R^2 value of 0.9346. After the introduction of the upperbound loss model (Type II

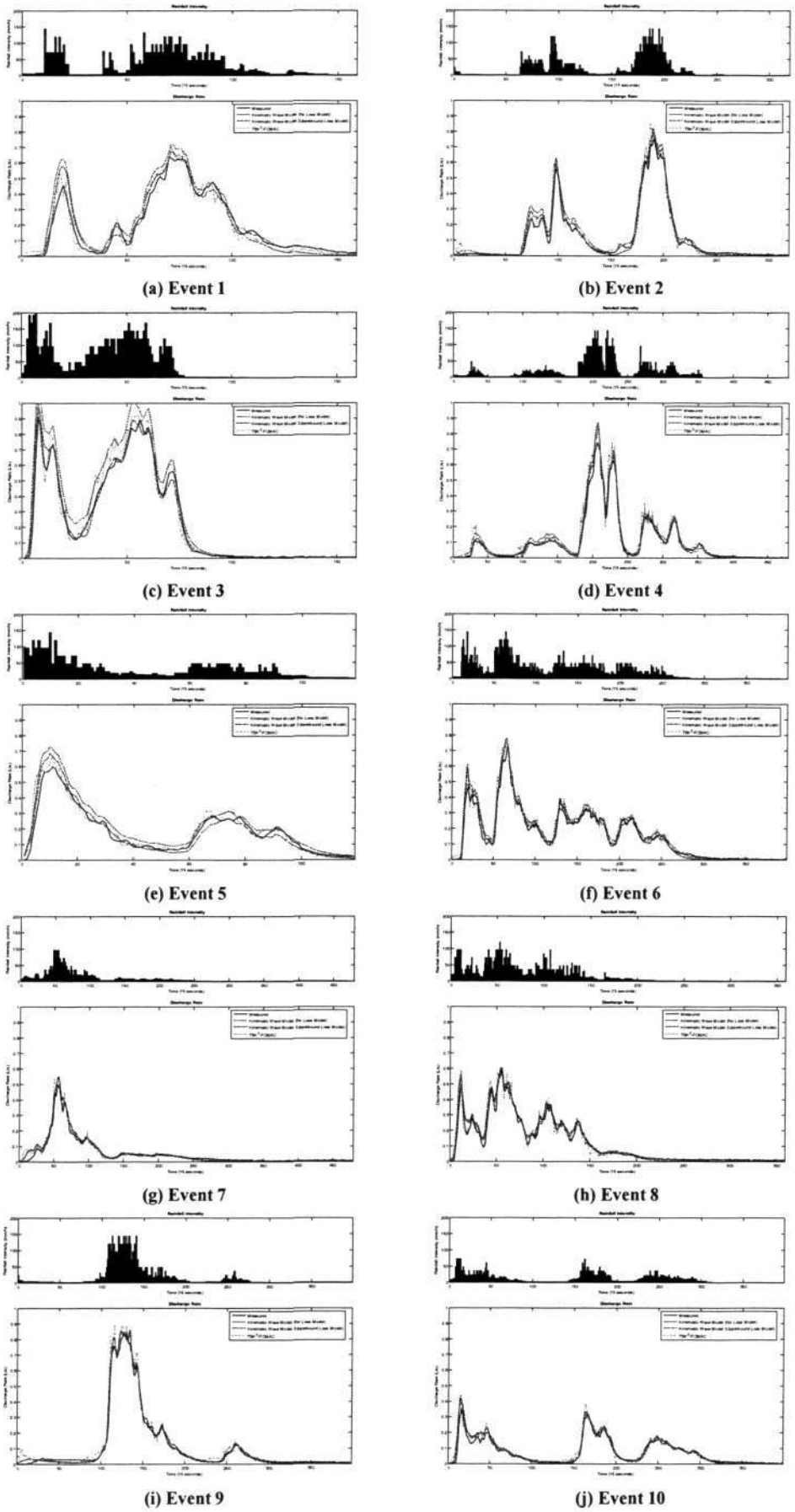
KWM), there is an improvement of 3.49% (0.9672 versus 0.9346) in the average R^2 value. In contrast, TSK^0-FCMAC outperformed Type I KWM with an average improvement of 4.49%. On the other hand, although the R^2 value of TSK^0-FCMAC in event 4 and 9 is slightly lesser than the R^2 value of Type II KWM (99.33% and 99.45% with respect to the R^2 value of Type II KWM), TSK^0-FCMAC outperformed Type II KWM in all other storm events with an average improvement of 1.23%. This improvement in the primary performance indicator presents clear evidence that TSK^0-FCMAC is capable of predicting the discharge rate of an open channel.

The predicted discharge rates from TSK^0-FCMAC and kinematic wave model (Type I and Type II) for all ten events are presented in Figure 7-4. In Figure 7-4(a), the bar chart shows the rainfall intensity. The bottom line chart shows the measured and predicted discharge rate by KWM and TSK^0-FCMAC. The measured discharge rate is represented by solid line. The discharge rate computed by Type I KWM (No Loss Model) is represented by dashed line. The discharge rate computed by Type II KWM (Upperbound Loss Model) is represented by dash-dot line. The discharge rate predicted by TSK^0-FCMAC is represented by dotted line.

In event 5, the R^2 value of TSK^0-FCMAC is significantly higher than the R^2 value of Type I and Type II KWM (12.61% and 4.39%). To have a closer look, the predicted discharge rates for event 5 are presented in Figure 7-5. In Figure 7-5, it can be observed that the difference between the discharge rates predicted by TSK^0-FCMAC and the measured discharge rates is notably smaller than that of Type I and Type II KWM. In event 9, the R^2 value of TSK^0-FCMAC is slightly lesser than the R^2 value of Type I and Type II KWM (99.50% and 99.45% with respect to the R^2 value of Type I and Type II KWM). Figure 7-6 shows the predicted discharge rates for event 9 and it can be observed that the discharge rates predicted by TSK^0-FCMAC follow closely to the measured discharge rates. This observation suggests that the slightly lesser R^2 value mentioned previously is fairly negligible.

Table 7-2: Memory Recall Performance KWM (Type I and Type II) and of TSK^0-FCMAC

Training and Testing Event	Kinematic Wave Model								TSK^0-FCMAC				
	Type I KWM (No Loss Model)				Type II KWM (Upperbound Loss Model)								
	Loss Rate (mm/hr)	RMSE Value	Pearson's Correlation	R^2 Function	Loss Rate (mm/hr)	RMSE Value	Pearson's Correlation	R^2 Function	RMSE Value	Pearson's Correlation	R^2 Function	Improvement of R^2 Ratio over Type I KWM	Improvement of R^2 Ratio over Type II KWM
1	0.00	0.0624	0.9776	0.8918	6.35	0.0452	0.9783	0.9433	0.0332	0.9877	0.9751	8.33%	3.18%
2	0.00	0.0375	0.9916	0.9561	4.82	0.0246	0.9928	0.9810	0.0256	0.9920	0.9835	2.74%	0.25%
3	0.00	0.0899	0.9928	0.9065	16.75	0.0337	0.9939	0.9869	0.0313	0.9954	0.9908	8.43%	0.39%
4	0.00	0.0306	0.9941	0.9621	3.48	0.0212	0.9947	0.9820	0.0284	0.9879	0.9753	1.32%	-0.67%
5	0.00	0.0548	0.9876	0.8602	6.74	0.0352	0.9841	0.9424	0.0190	0.9932	0.9863	12.61%	4.39%
6	0.00	0.0365	0.9901	0.9432	3.98	0.0246	0.9905	0.9741	0.0267	0.9872	0.9747	3.15%	0.06%
7	0.00	0.0174	0.9910	0.9614	0.66	0.0165	0.9912	0.9652	0.0106	0.9942	0.9884	2.70%	2.32%
8	0.00	0.0288	0.9860	0.9590	1.82	0.0256	0.9865	0.9677	0.0243	0.9878	0.9756	1.66%	0.79%
9	0.00	0.0208	0.9964	0.9892	0.51	0.0204	0.9964	0.9897	0.0276	0.9941	0.9842	-0.50%	-0.55%
10	0.00	0.0226	0.9780	0.9164	1.53	0.0192	0.9786	0.9395	0.0172	0.9805	0.9606	4.42%	2.11%
Average	-	0.0401	0.9885	0.9346	-	0.0266	0.9887	0.9672	0.0244	0.9900	0.9794	4.49%	1.23%



(a) Event 1 (b) Event 2

(c) Event 3 (d) Event 4

(e) Event 5 (f) Event 6

(g) Event 7 (h) Event 8

(i) Event 9 (j) Event 10

Figure 7-4: Measured and Predicted Discharge Rate for all Ten Events (Memory Recall), (a) Event 1 ~ (j) Event 10

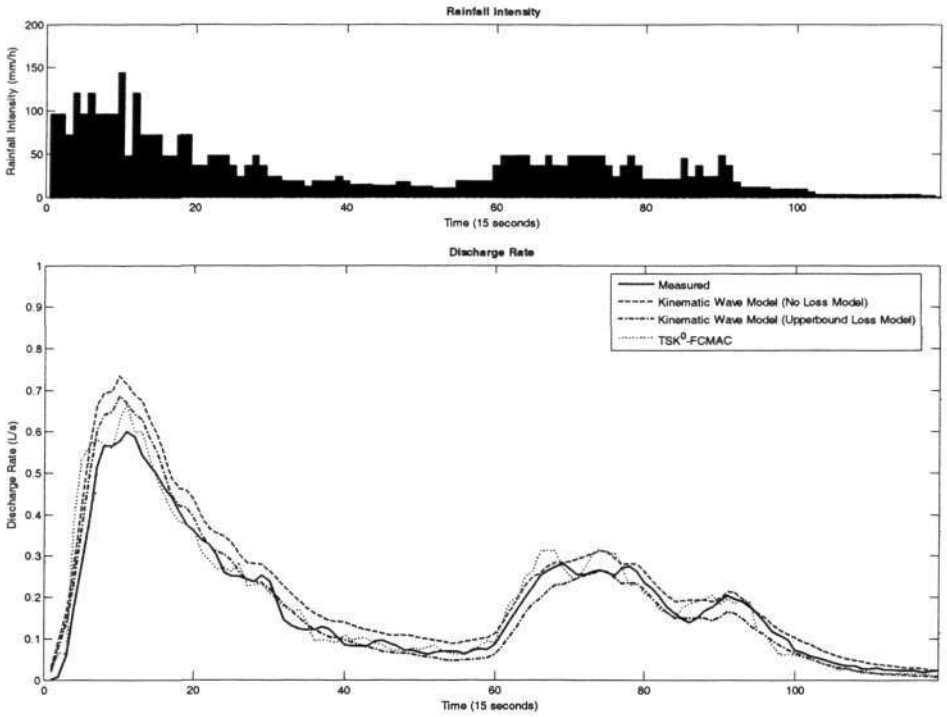


Figure 7-5: Measured and Predicted Discharge Rate for Event 5 (Memory Recall)

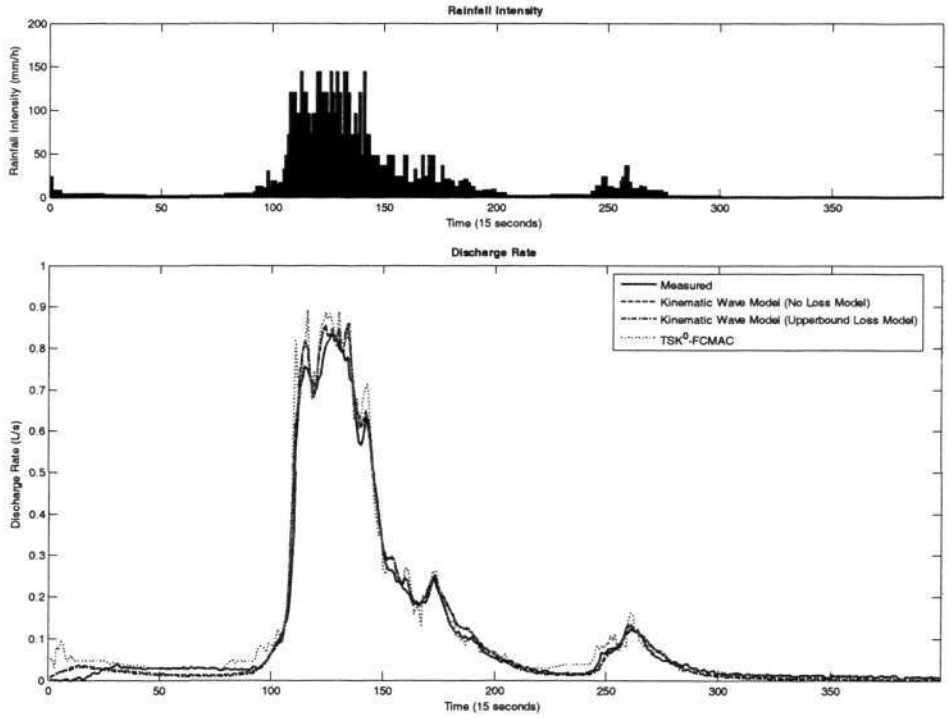
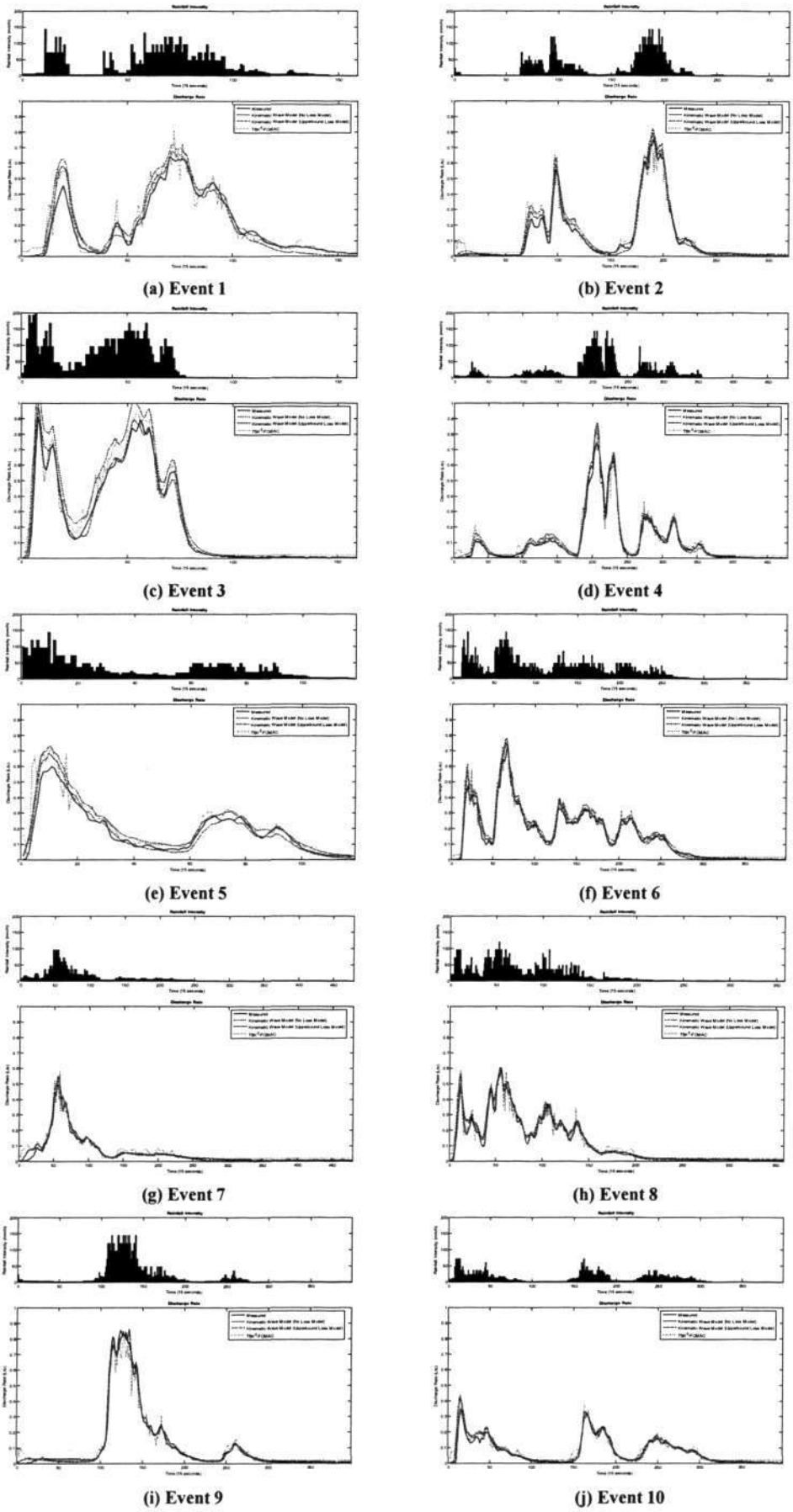


Figure 7-6: Measured and Predicted Discharge Rate for Event 9 (Memory Recall)

As mentioned in section 7.2.3, the shortcoming of employing the upperbound loss model in kinematic wave model (i.e. Type II KWM) is the need of calibration for the loss rate. According to the 6th column of Table 7-2, the calibrated loss rates between different storm events vary to great extent (0.51 mm/hr to 16.75 mm/hr). In contrast, given sufficient training samples, TSK^0-FCMAC network can be employed to predict the discharge rate of different storms event along the same channel without the need of calibration. Such investigation will demonstrate the generalization ability of the TSK^0-FCMAC network. However, for TSK^0-FCMAC to generalize, the training samples extracted from a single event will not be sufficient. Therefore, in the second experiments, the training samples for *Generalization*^{b} are extracted from two events: Events 3 and 6. Event 3 is selected because it consists of the highest rainfall intensity while Event 6 consists of multiple peaks of different rainfall intensities (refer to Figure 7-4(c) and (f)). The generalization performances of TSK^0-FCMAC are presented in Table 7-3. It can be observed that the average R^2 value is degraded to 0.9365 as compared to 0.9672 of the Type II KWM and 0.9794 of the memory recall by TSK^0-FCMAC (refer to the last row of Table 7-2). The trained TSK^0-FCMAC network is a three-input network with 648 fuzzy if-then rules ($8\times9\times9$). For further investigation, the predicted discharge rates for Event 5, the lowest R^2 value (0.8963) among the ten storm events is shown in Figure 7-8. Although the training samples employed to train the TSK^0-FCMAC network are not extracted from event 5, TSK^0-FCMAC is still capable of predicting the general trend of the discharge rate without difficulty. In addition, the predicted discharge rates for event 4 and 7 are also presented in Figure 7-9 and Figure 7-10 respectively. Both figures suggest that TSK^0-FCMAC can achieve comparable results even though the physical aspect of the concrete plane is not provided to TSK^0-FCMAC.

Table 7-3: Generalization Performance of TSK^0-FCMAC using Events 3 & 6 as Training Events

Training Event	Testing Event	RMSE Value	Pearson's Correlation	R^2 Function
(3, 6)	1	0.0640	0.9566	0.9073
(3, 6)	2	0.0576	0.9622	0.9164
(3, 6)	3	0.0468	0.9914	0.9794
(3, 6)	4	0.0419	0.9780	0.9462
(3, 6)	5	0.0522	0.9638	0.8963
(3, 6)	6	0.0333	0.9803	0.9604
(3, 6)	7	0.0247	0.9745	0.9375
(3, 6)	8	0.0378	0.9725	0.9412
(3, 6)	9	0.0435	0.9894	0.9607
(3, 6)	10	0.0247	0.9632	0.9193
Average		0.0426	0.9732	0.9365



(a) Event 1 (b) Event 2

(c) Event 3 (d) Event 4

(e) Event 5 (f) Event 6

(g) Event 7 (h) Event 8

(i) Event 9 (j) Event 10

Figure 7-7: Measured and Predicted Discharge Rate for all Ten Events (Generalization), (a) Event 1 ~ (j) Event 10

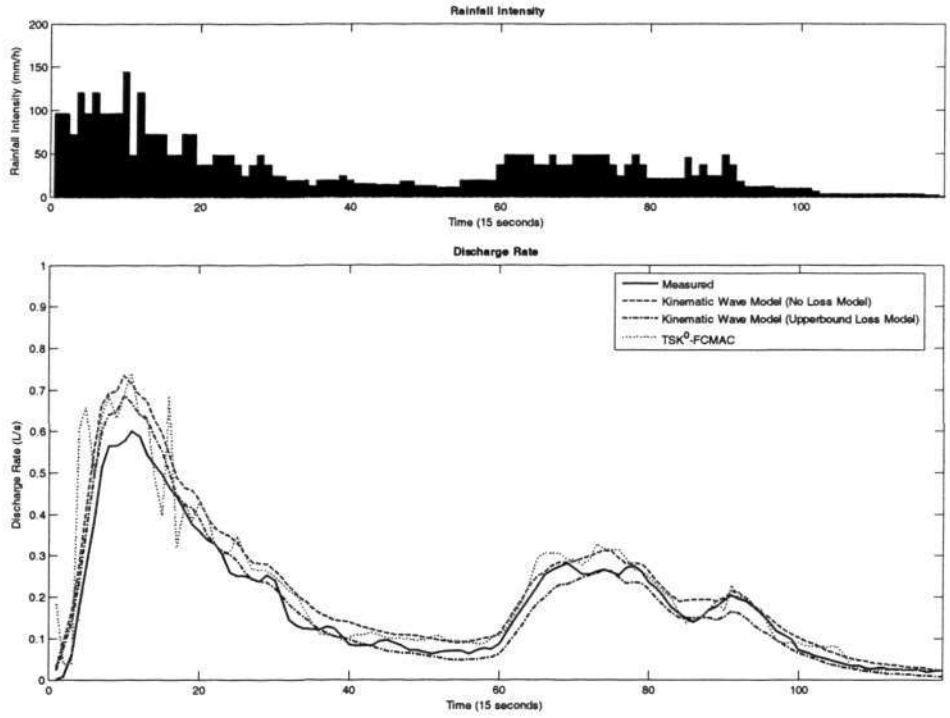


Figure 7-8: Measured and Predicted Discharge Rate for Event 5 (Generalization)

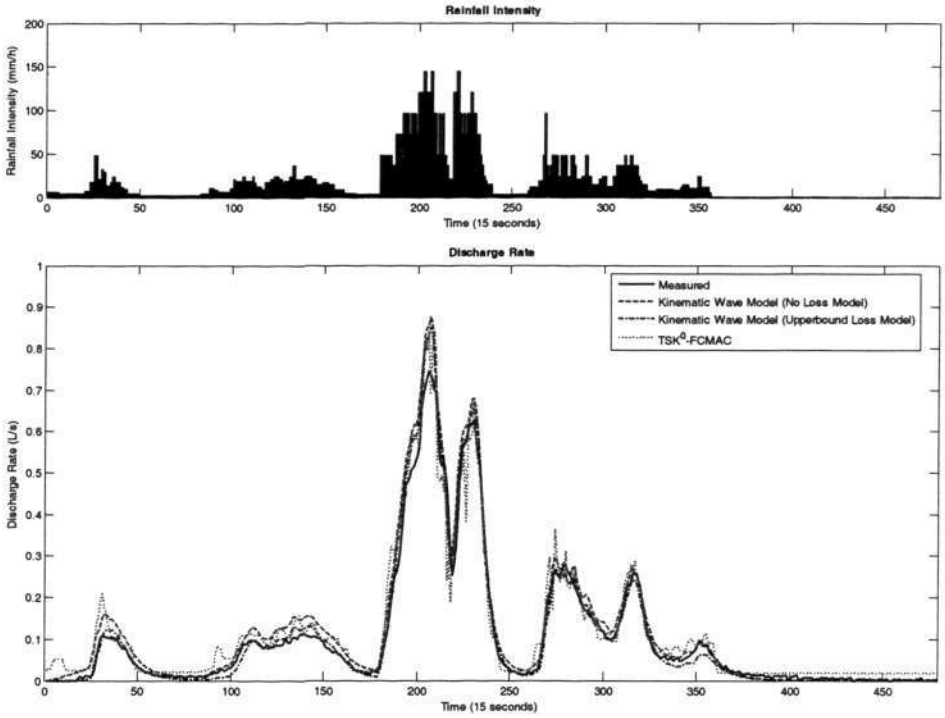


Figure 7-9: Measured and Predicted Discharge Rate for Event 4 (Generalization)

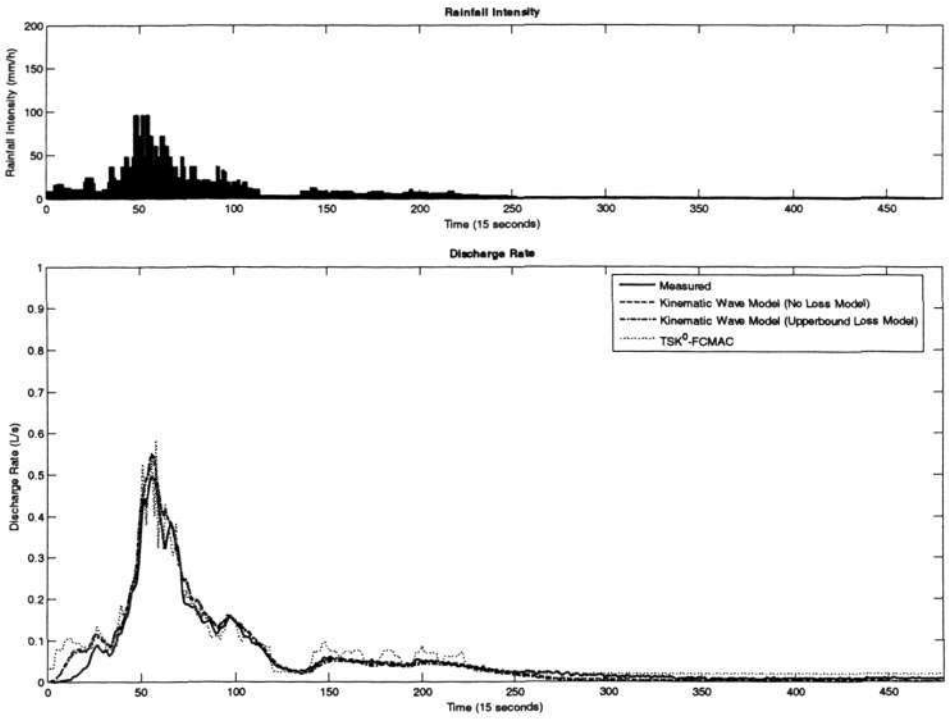


Figure 7-10: Measured and Predicted Discharge Rate for Event 7 (Generalization)

7.4. Summary

This chapter investigates the application of TSK^0-FCMAC on the prediction of rainfall runoff. The performances of TSK^0-FCMAC are evaluated by comparing the results with conventional method in hydroinformatics, the kinematic wave model (KWM). Modeling the runoff characteristics of a channel using kinematic wave model requires accurate information on the physical aspect of the channel, such as the Manning's roughness coefficient, the width and slope of the channel and, the height of the flow. In contrast, the proposed TSK^0-FCMAC network could model the runoff characteristics of a channel with:

- no knowledge of the physical aspect of the concrete plane,
- no knowledge of the observed outflow hydrograph Q, and
- features extracted from historical rainfall data alone.

In real life, the flow of water navigates through irregular-shaped waterway. The estimation of the runoff in an irregular-shaped waterway involves complex computation with conventional mathematical models. Moreover, such mathematical models often require details information on

the physical aspect of the waterway to give accurate estimation. On the other hand, this chapter presents a viable approach for rainfall runoff modeling using neuro-fuzzy approach and alleviates such requirements. Preliminary experiments show that the proposed TSK^0-FCMAC network is able to achieve a higher R^2 value in memory recall (an average improvement of 4.49% over Type I KWM and 1.23% over Type II KWM). The generalization ability of the network has also been exploited. With a three-input TSK^0-FCMAC network ($8\times9\times9$ cells, 648 fuzzy if-then rules), an average R^2 value of 0.9365 has been achieved and the results are comparable with the kinematic wave model.

Chapter 8 Engineering Case Study: Audio Detection Problem

8.1. Introduction

The audio detection problem originates from an application on embedded audio detection system proposed by Ting [197]. It aims to implement a miniature device that can monitor and reliably classify ambient audio signals. Such implementation will act as an assistive device for the hearing impaired that will inform its user of emergency signals such as fire alarms and sirens through an LCD display or vibration alert. The implementation involves the development of a real-time embedded audio detection system using general soft computing techniques to analyze and classify a stream of audio input samples in real-time.

There are two main approaches in performing such pattern classification; namely: implementing a rule-based system, or training an artificial neural network (ANN) [6, 7] to undertake the classification task. Classification by a rule-based system involves a detailed analysis of all types of audio signal to be classified, followed by an algorithmic description of their detection. Whilst it is easy to verbally classify a sound as, for example, a siren, rule based classification is problematic since there are countless variants of siren. The rules must encompass all of these, and be built through subjective human decisions based on analysis of the set of these. ANN, on the other hand, can learn on the basis of available data, and provide an objective view of the problem. It boils down to collecting an audio recording of each supported siren type and train a suitable network.

There has been growing interest in the use of ANNs for audio recognition in the past few years [198-202]. In Ting's work [197], three types of ANN are evaluated; they are the common Multilayer Perceptron Network (MLP) [21], the Gaussian Radial Basis Function Network (GRBF) [203] and the Falcon-MART Fuzzy Neural Network [87]. Nonetheless, it is obvious that the crucial consideration in real time embedded algorithms is the computational complexity. Many complex neural networks are unsuitable for real time implementation due to the extensive use of computations such as division, sine, cosine, square root and so on. In contrast, the proposed TSK^0-FCMAC and TSK^1-FCMAC appear to meet such criteria by employing crisp value or a

linear combination of input variables in the consequents. An investigation on the possibility of applying the proposed architectures in embedded audio detection system has been conducted. Together with the three ANNs reported in [197], comparisons on the performance in terms of classification ratio, training time and recall time are presented in this chapter.

Figure 8-1 shows a block diagram overview of the whole system. The basic hardware consists of a microprocessor (a 32-bits Intel SA1110 operating at 220 MHz and running Linux kernel 2.4.18), a microphone with ADC, an LCD display and a miniature vibration device. Captured audio is passed into the microprocessor in pulse coded modulation (PCM) format. This signal vector is then segmented and features are extracted and passed into an ANN for classification. The classification output informs a decision-making unit which averages results over time, determines information to be displayed on the LCD, and trigger the vibration device to alert the user.

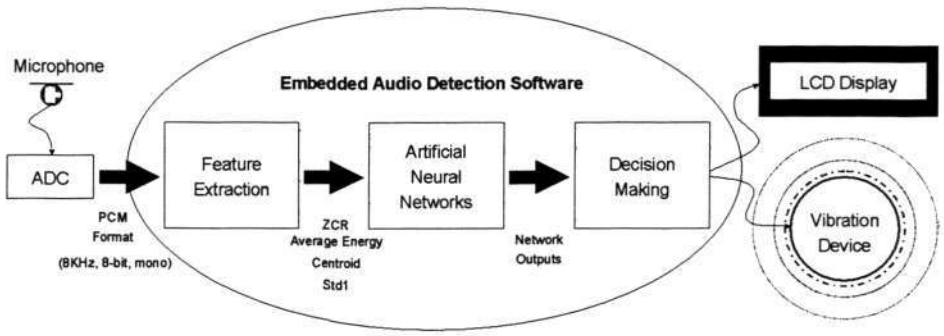


Figure 8-1: System Overview for Embedded Audio Classification

8.2. Experiments

8.2.1. Audio Format

The audio data set for both training and testing consists of 120 files representing six audio categories (fire alarm, glass breaking, horn, phone ring, siren and speech). These offline data comprises approximately 38 minutes of audio. They are represented in single-channel 8-bit PCM format. For real-time classification, preprocessing and feature extraction using simple arithmetic are utilized in the experiment to reduce storage requirements and processing time. A representative group of audio features are also identified from a larger set of tested methods.

The selection of a minimum number of significant attributes from the audio signals is critical for real time applications to minimize processing. Feature extraction transforms an N-entry measurement to an M-vector of features where M < N. It thereby:

1. Reduces storage requirements.
2. Reduces data bandwidth and thus processing time.
3. Provides a reduced form which is more representative of the items of interest; thus improving the generalization process.

As a result, algorithms for features extraction must not be too computationally complex. Operations such as square root, logarithm, sine, cosine, tangent and exponential should be minimized or preferably completely eliminated. Obviously the wanted information should be preserved during the feature extraction process since a successful classification depends strongly on the ability to identify key features for each audio type to be classified.

On the other hand, to support real time classification of audio signals, classification decision must be available within a short period, preferably comparable to the human response speed. For block-based audio features (i.e. ones that provide a single output score for a frame of fixed input audio size), feature output time is the collection time plus processing time. Collection time is needed to fill an analysis frame (a block of fixed-length audio signal to undergo feature extraction), and processing time depends upon the time taken to perform feature extraction plus the ANN recall speed. In order to capture important audio features that may straddle the boundary of successive frames, the analysis frames are overlapped by 50%, and frame size is set to 128 samples (a power of two is chosen to aid in the Fourier transform efficiency [204] that will be used as the basis for several candidate features).

8.2.2. Feature Selection

As mentioned in section 8.2.1, the audio signal is first segmented into fixed size overlapping frames. Each frame is then processed and analyzed in one of several ways to provide a vector of output features describing each frame. The full candidate vector consists of ten different measures of the content of the audio in the frame as shown in the following:

1. ZCR	3. Average Energy	5. Peak1	7. Peak3	9. Std1
2. AMDF	4. Variance	6. Peak2	8. Centroid	10. Std2

Ting [197] proposed a feature selection method and selected four of the features: ZCR, Average Energy, Centroid and Std1. Quah [126] conducted an extensive feature selection on the same data set with several feature selection method; namely: 1) FOCUS [205], 2) Relief [206, 207], 3) Correlation-based feature selection (CFS) [208, 209], 4) RReliefF [210] and, 5) MCES [126]. Nonetheless, in this experiment, features selected in [197] will be employed.

8.2.3. Evaluation Terms

A given testing pattern is considered successfully classified when the output value of the desired entity (e.g. neuron in the output layer) is the maximum among its peers. The classification ratio is defined as the ratio of successful classification against the total number of testing patterns.

The training of neural networks can be classified into three ways:

- Positive Training: providing only the positive patterns (patterns that the networks are trained to accept) into the network during the training process. Hence, the network will only be trained to recognize positive patterns without consideration of negative patterns. For example, a single output network may be trained to output "1" for pattern $\mathbf{A} = \{A_1, A_2, \ldots, A_N\}$.

- Negative Training: negative patterns that the networks are trained to reject. For example, a single output network may be trained to output "0" for pattern $\mathbf{B} = \{B_1, B_2, \ldots, B_N\}$.

- Positive and Negative Training: providing both the positive and negative patterns into the network during the training process. In this training mode, the network is trained to recognize both types of patterns and to give the correct output accordingly. For example, a single output network is trained to output "1" for pattern $\mathbf{A} = \{A_1, A_2, \ldots, A_N\}$ and output "0" for pattern $\mathbf{B} = \{B_1, B_2, \ldots, B_N\}$.

The ability of trained networks can be assessed in two ways:

- Memory Recall: the ability of a network to memorize what has been trained previously. The data sets used for training and testing will be identical. During testing, the network must simply respond to test patterns that it has already been trained to recognize.
- Generalization: the ability of the network to generalize a larger amount of data from a smaller training set. In this case the testing data set differs from the training data set, with the assumption that the training data is at least representative of the testing data.

Once training of a network is completed, the results generated from either a memory recall or generalization test can be classified into four categories:

- TA: true acceptance measures the ability of a network to output a correct result when a positive pattern is passed into the network.
- TR: true rejection measures the ability of a network to output a correct negative result, thus rejecting a negative pattern which is passed into the network.
- FA: false acceptance measures the error of the network in failing to reject negative patterns.
- FR: false rejection measures the error of the network in rejecting positive patterns.

FR and FA are defined as type I and type II error rates respectively in [211], with the relation among the four categories being given as follow:

$$\begin{aligned} TA + FR &= 100\% \\ TR + FA &= 100\% \end{aligned} \tag{164}$$

8.3. Results and Discussion

The performances of the individual networks will be evaluated from two aspects, the classification ratio and computational time requirement. The results are presented in the following section.

8.3.1. Classification Ratio

This section evaluates the classification ratio on five different networks, MLP, GRBF and Falcon-MART, TSK^0-FCMAC and TSK^1-FCMAC. The data sets used in the evaluation are based on the

four selected features (ZCR, Average Energy, Centroid and Std1). Figure 8-2 shows two arrangements of neural network used in this section. The networks are trained in three modes; namely:

- Model I: Single output network (refer to Fig 8-2(a)) with positive training. This uses 90 sets of experimental data, each consisting of 400 positive training patterns and 400 testing patterns.
- Model II: Single output network (refer to Fig 8-2(a)) with both positive and negative training. This uses 50 sets of experimental data, each consisting of 400 positive patterns, 400 negative patterns and 400 testing patterns.
- Model III: Double output network (refer to Fig 8-2(b)) with both positive and negative training. This uses 100 sets of experimental data, each consisting of 400 positive patterns, 400 negative patterns and 400 testing patterns, in a similar arrangement to model II.

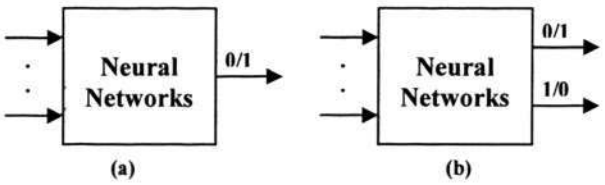


Figure 8-2: A Simple Representation of ANN. (a) Single Output, (b) Double Outputs

Results from the experiment, in terms of classification ability for both true acceptance (TA) and true rejection (TR) are shown in Table 8-1 (for memory recall) and Table 8-2 (for generalization). Results from Model I show that all networks perform very well TA under positive training. However, the TR for all networks are below 20%, showing that by only using positive training, the network is susceptible to misclassification of negative patterns. Take note that Model I is not supported by the Falcon-MART architecture, which requires both positive and negative training. In Model II and III, although the performances in TA for Falcon-MART are the lowest among all networks, it is compensated by its superior results in TR which is highest among all networks in all cases. In the experiments for Model II, TR for all networks increase significantly from the range of 0-20% to 56.14-70.38%. This increment shows that negative training plays an important role in the training process of the network.

It can also be noted that the results for MLP, GRBF, TSK^0-FCMAC and TSK^1-FCMAC are comparable, with MLP and TSK^1-FCMAC perform slightly better than GRBF and TSK^0-FCMAC in terms of TA. In term of memory recall, TSK^1-FCMAC shows exceptional well performance in all cases by achieving classification ratio of more than 97% in both TA and TR (as tabulated in the last row of Table 8-1). Both the proposed architectures, TSK^0-FCMAC and TSK^1-FCMAC perform well in all cases and the results are comparable with conventional neural networks in terms of memory recall and generalization ability.

Table 8-1: Tabulated Classification Ratio (For Memory Recall)

Network Type	Model I		Model II		Model III	
	TA (%)	TR (%)	TA (%)	TR (%)	TA (%)	TR (%)
MLP	100.00	-	94.57	97.51	94.67	97.70
GRBF	100.00	-	91.97	97.46	90.42	95.77
Falcon-MART	-	-	70.30	100.00	70.30	100.00
TSK^0-FCMAC	100.00	-	93.92	98.60	93.97	98.60
TSK^1-FCMAC	100.00	-	97.21	97.61	97.36	97.36

Table 8-2: Tabulated Classification Ratio (For Generalization)

Network Type	Model I		Model II		Model III	
	TA (%)	TR (%)	TA (%)	TR (%)	TA (%)	TR (%)
MLP	97.78	5.00	76.93	56.14	76.40	58.84
GRBF	82.22	20.00	68.15	67.43	71.99	55.37
Falcon-MART	-	-	61.36	70.38	61.36	70.22
TSK^0-FCMAC	98.42	0.17	64.11	62.04	67.61	55.66
TSK^1-FCMAC	100.00	0.00	70.62	65.45	72.43	63.97

8.3.2. Computational Time Requirement

The experiments discussed in this section attempt to investigate the computational time requirements for all five networks. The networks are evaluated from two perspectives; namely: training time and recall time. The maximum iteration for both MLP and GRBF networks was fixed at 2000 iterations with 4 input layer neurons, 20 hidden layer neurons. This does not apply

to Falcon-MART which creates new nodes during the training process. For Falcon-MART, different training patterns will lead to a different number of nodes after the training process. It is important to take note that the ratio shown in the following analysis will vary when the maximum number of iterations for MLP and GRBF is changed.

The specifications for the system to conduct this experiment are listed as follows:

- Intel Core ™ Duo CPU, E6750 @ 2.66 GHz
- 2.66 GHz, 3.00 GB of RAM
- Window XP Professional version 2002, service pack 2
- Microsoft Visual C++ version 6.0

The training data consists of 100 sets; each set has 4017 training patterns from 2 different categories of audio types. The values of average, maximum, minimum and standard deviation of the training time requirements are tabulated in Table 8-3. It can be observed that both Falcon-MART and TSK^0-FCMAC require minimal amount of training time, while the TSK^1-FCMAC requires the most training time due to its complex learning algorithm.

Table 8-4 shows the tabulated results for recall time for all five networks. The recall time is defined as the time required to perform a single recall. GRBF requires a recall time of 179.2μs which is quite high compared to other networks. MLP's recall time is the lowest due to the straightforward computation in the activation function for each neuron. An interesting observation shown in Table 8-4 is that the recall time of a more complex TSK^1-FCMAC is in fact lower than that of TSK^0-FCMAC. It is known that a TSK^1-FCMAC network employs linear combination of input parameters in its consequents and a TSK^0-FCMAC network employs a much simpler crisp value. The main reason of a lower recall time is that although TSK^1-FCMAC is more complex than its predecessor in the consequents, it requires a lesser number of fuzzy rules in the antecedents to achieve the same precision. In the experiment mentioned in 8.3.1, TSK^1-FCMAC created two to three membership functions for each input while TSK^0-FCMAC created four to six membership functions to achieve comparable classification ratio. Considering a four inputs

network, this difference could contribute to different network size (equivalent to number of fuzzy rules created) as high as 81 versus 1296 (3^4 versus 6^4).

Table 8-3: Tabulated Training Time (Average, Maximum, Minimum and Standard Deviation)

Network Type	Training Time (s)			
	Ave	Max	Min	Stdev
MLP	209.26	233.35	179.63	18.67
GRBF	47.17	50.22	43.06	1.61
Falcon-MART	6.05	9.22	3.36	1.76
TSK^0-FCMAC	6.22	9.28	4.00	1.08
TSK^1-FCMAC	626.66	2526.47	119.96	494.02

Table 8-4: Tabulated Recall Time (Average, Maximum, Minimum and Standard Deviation)

Network Type	Recall Time (μs)			
	Ave	Max	Min	Stdev
MLP	24.67	31.12	19.42	2.37
GRBF	179.20	225.54	108.79	19.85
Falcon-MART	35.21	42.82	31.12	1.50
TSK^0-FCMAC	62.50	81.65	50.54	6.12
TSK^1-FCMAC	51.58	73.94	38.83	6.79

8.4. Summary

This chapter investigates the possible application of the proposed architectures on embedded audio detection system. The system could be downloaded to a miniature device that could be used as an assistive device for the hearing impaired and inform its user of emergency signals such as fire alarms and sirens through LCD display or vibration alert. The proposed architectures are evaluated in terms of classification ratio and computational time requirement together with three other conventional networks; namely: MLP, GRBF and Falcon-MART.

In terms of classification ratio, the results from TSK^0-FCMAC and TSK^1-FCMAC are comparable with the conventional networks while TSK^1-FCMAC shows exceptional better performance in memory recall. In terms of computational time requirement, TSK^0-FCMAC and TSK^1-FCMAC achieve satisfactory recall time at an average of 62.5μs and 51.58μs respectively. However, the training time required by the TSK^1-FCMAC network is high compared to the other networks. Nonetheless, after taking into consideration that the embedded device does not require online training capability, TSK^1-FCMAC appears to provide excellent classification ratio with reasonable amount of recall time and is suitable for implementation of embedded audio detection software.

Chapter 9 Conclusions and Recommendations

9.1. Conclusions

This thesis focused on the development of a synergy between artificial neural networks and fuzzy systems. It aims to combine their individual advantages and at the same time, complement the shortcomings of one another. This thesis can be summarized as:

- A new class of CMAC, referred to as TSK^0-FCMAC (a localized self-organizing zero-ordered Takagi-Sugeno-Kang fuzzy inference system based on the CMAC structure) is proposed in Chapter 3. The structures and learning algorithm of the proposed architecture are described in details. The quantization problems faced by numerous CMAC networks are partially resolved by the use of a novel clustering technique, Discrete Incremental Clustering (DIC) [90]. TSK^0-FCMAC employs a two-phase training algorithm. The rigid memory structure of conventional CMAC network is replaced with DIC technique. This technique enables the CMAC network to grow without prior knowledge of the number of clusters. In addition, fuzzy inference model has been successfully incorporated into the CMAC networks whereby fuzzy rules could be generated to interpret the operation of the network. TSK^0-FCMAC can now be implemented to model many real world problems with greater semantics and ease. The ability of TSK^0-FCMAC has been demonstrated through applications on three benchmarking case studies. In section 3.5.1, the adaptive ability of TSK^0-FCMAC has been highlighted through application in the inverted pendulum problem. Moreover, the inherent ability of the TSK^0-FCMAC to perform rule extraction has been demonstrated in section 3.5.2 with Fisher's Iris classification [88]. In section 3.5.3, TSK^0-FCMAC has been applied to solve a well-known regression problem; namely the Mackey-Glass time series prediction [89]. In general, TSK^0-FCMAC has demonstrated outstanding performance in solving problem from different areas.

- In Chapter 4, an investigation into the convergence characteristic of TSK^0-FCMAC has been rigorously undertaken. The investigation starts by formulating the mathematical representation of the TSK^0-FCMAC architecture, followed by the proper definition of the

difference of memory contents between consecutive iterations. This difference has been proven to approach zero if the learning parameter, λ, is limited between zero and two. This leads to the conclusion that the convergence characteristic of TSK^0-FCMAC holds when the learning parameter is between zero and two. The mathematical formulation presented in this chapter provides a strong foundation for further investigation on the convergence characteristic of fuzzy CMACs with similar fuzzy inference scheme.

- An extension of the proposed CMAC, TSK^1-FCMAC (based on the first-ordered Takagi-Sugeno-Kang fuzzy inference system), is proposed in Chapter 5. This extension aims to improve the precision of its predecessor. The architecture and learning algorithm of TSK^1-FCMAC are presented in section 5.2. In contrast to its predecessor, TSK^1-FCMAC is designed to achieve a higher performance in terms of precision. Instead of using a single crisp value, the consequent of TSK^1-FCMAC is represented by a linear function of input parameters. In section 5.3.1, TSK^1-FCMAC has been applied to perform function approximation for sinusoidal function and surface. The improvement of TSK^1-FCMAC over its predecessor has been evaluated from the aspect of precision. An investigation over memory requirement and training time requirement for both architectures has also been appraised. TSK^1-FCMAC has been shown to produce excellent precision at the expense of a higher computational complexity. In section 5.3.2, TSK^1-FCMAC has been applied to predict the fuel consumption for different type of automobile in the MPG (miles per gallon) prediction [125]. A simple feature selection method has been conducted with the proposed architectures. The results has been found to be comparable to existing feature selection methods proposed by Quah [126] and Jang [127]. In section 5.3.3, TSK^1-FCMAC has been applied to the two-spiral classification problem [128]. This problem is highly segmented in the input space. Given adequate training samples, TSK^1-FCMAC has been shown to model the two-spiral classification problem with higher precision and fewer numbers of fuzzy if-then rules as compared to TSK^0-FCMAC. As a summary, the case studies conducted in section 5.3 have shown that TSK^1-FCMAC is an excellent alternative over its predecessor in solving problem that emphasize on precision.

- Chapter 6 presents a case study for diabetic treatment using TSK^0-FCMAC. The proposed system, a personalized drug delivery system, is able to capture the glucose-insulin dynamics of individuals under different dietary profiles. Without prior knowledge of disturbance (i.e. meal announcement), preliminary simulation results show that the proposed system is able to effectively adapt to both intra and inter-patient variations. Moreover, the design of the proposed system follows closely to what is available in real life and is suitable for animal and clinical pilot testing in the near future.

- Chapter 7 presents a case study on the open channel routing of an outdoor experimental plot set up at Nanyang Technological University, Singapore. TSK^0-FCMAC has been successfully applied in predicting the hydrographs at the end of the channel through monitored rainfall events. The performance of the proposed architecture is evaluated with the prediction results from the kinematic wave model, a conventional approach in hydroinformatics. Experiment in section 7.3 shows that the TSK^0-FCMAC network is able to achieve outstanding results as compared to that produced by the kinematic wave model.

- Chapter 8 investigates the possible application of both TSK^0-FCMAC and TSK^1-FCMAC on embedded audio detection. The proposed architectures are evaluated in terms of classification ratio and computational time requirement together with three other conventional networks; namely: MLP, GRBF and Falcon-MART. After taking into consideration that the embedded device does not require online training capability, TSK^1-FCMAC has been shown to provide excellent classification ratio with reasonable amount of recall time and is suitable for the implementation of an embedded audio detection software.

The research efforts and achievements of this thesis are tabulated in Table 9-1.

Table 9-1: A Summary of Research Efforts and Achievements in this Thesis

Contributions	Part	Existing Systems	Issues/Problems	Proposed Models/Algorithms	Presented in Section
Structural Improvements	(a)	MLP/CMAC	Black box operation	TSK fuzzy inference scheme	Section 3.2
	(b)	Fuzzy systems	Manual tuning of system parameters	Two-phase learning algorithm based on the CMAC structure	Section 3.3
	(c)	MLP	Globalized learning, offline training		
	(d)	CMAC	Predefined structure, quantization problem, memory utilization problem	Discrete Incremental Clustering technique	Section 3.4
Theoretical Contributions	(e)	TSK^0-FCMAC	Convergence characteristic	Mathematical formulation and theoretical proof	Section 4.3
	(f)	TSK^0-FCMAC	Requirement for higher precision	Higher order TSK fuzzy inference scheme	Section 5.2
Application Case Studies	(g)	-	Personalized drug delivery system – a diabetic treatment without meal announcement	TSK^0-FCMAC	Section 6
	(h)	-	Rainfall runoff prediction	TSK^0-FCMAC	Section 7
	(i)	-	Audio detection problem	TSK^0-FCMAC & TSK^1-FCMAC	Section 8

The research efforts and achievements are highlighted from three aspects:

- *Structural Improvements*: In part (a) (refer to 2nd column in Table 9-1), the conventional CMAC network has been successfully transformed into a white box. This is achieved by incorporating the Takagi-Sugeno-Kang (TSK) fuzzy inference system into the CMAC structure. In parts (b) and (c), a two-phase learning algorithm is proposed to tune the system parameters systematically. Moreover, the online training is localized based on the CMAC structure. In part (d), the rigid memory structure of conventional CMAC network is replaced with Discrete Incremental Clustering (DIC) technique. In addition, the quantization and memory utilization problems are partially resolved with DIC technique.

- *Theoretical Contributions*: In part (e), the convergence characteristic of the proposed TSK^0-FCMAC is established and it provides a strong foundation for further investigation on the convergence characteristic of fuzzy CMAC with similar fuzzy inference scheme. In part (f), an implementation of higher order TSK fuzzy inference scheme, the TSK^1-

FCMAC architecture is proposed. This extension emphasizes on achieving higher precision and provides an excellent alternative over its predecessor.

- *Application Case Studies*: In parts (g)-(i), the proposed neuro-fuzzy inference systems have been successfully applied in real life applications of three different areas; namely: 1) personalized drug delivery system (control), 2) rainfall runoff (regression) and, 3) embedded audio detection (classification).

To conclude, the research achievements in this thesis concur with the research objectives highlighted in Figure 1-1.

9.2. Recommendations

This work presents the development of a neuro-fuzzy system. Although both theoretical foundation and empirical case-studies for the proposed architectures have shown promising results, some further investigations can still be pursued:

- *Multilayer Structure*: The organization of the proposed TSK^0-FCMAC and TSK^1-FCMAC are based on the structure of a single-layer CMAC. The fuzzification of the single-layer CMAC minimizes the requirement on the resolution in the input space. However, for problems that require higher precision, increasing the resolution of the input space alone may not be a favorable solution. Therefore, a structure with multiple layers provides an alternative solution. An example for a two-layer structure TSK^0-FCMAC is shown in Figure 9-1. In this example, eight fuzzy if-then rules are activated for input x' and y'. This will effectively improve the generalization ability by distributing the activation effort to eight fuzzy if-then rules than the original four in the case of single-layer structure (refer to Figure 3-2 of Chapter 3). Nevertheless, an in-depth study is needed to investigate the influence to the existing learning algorithm.

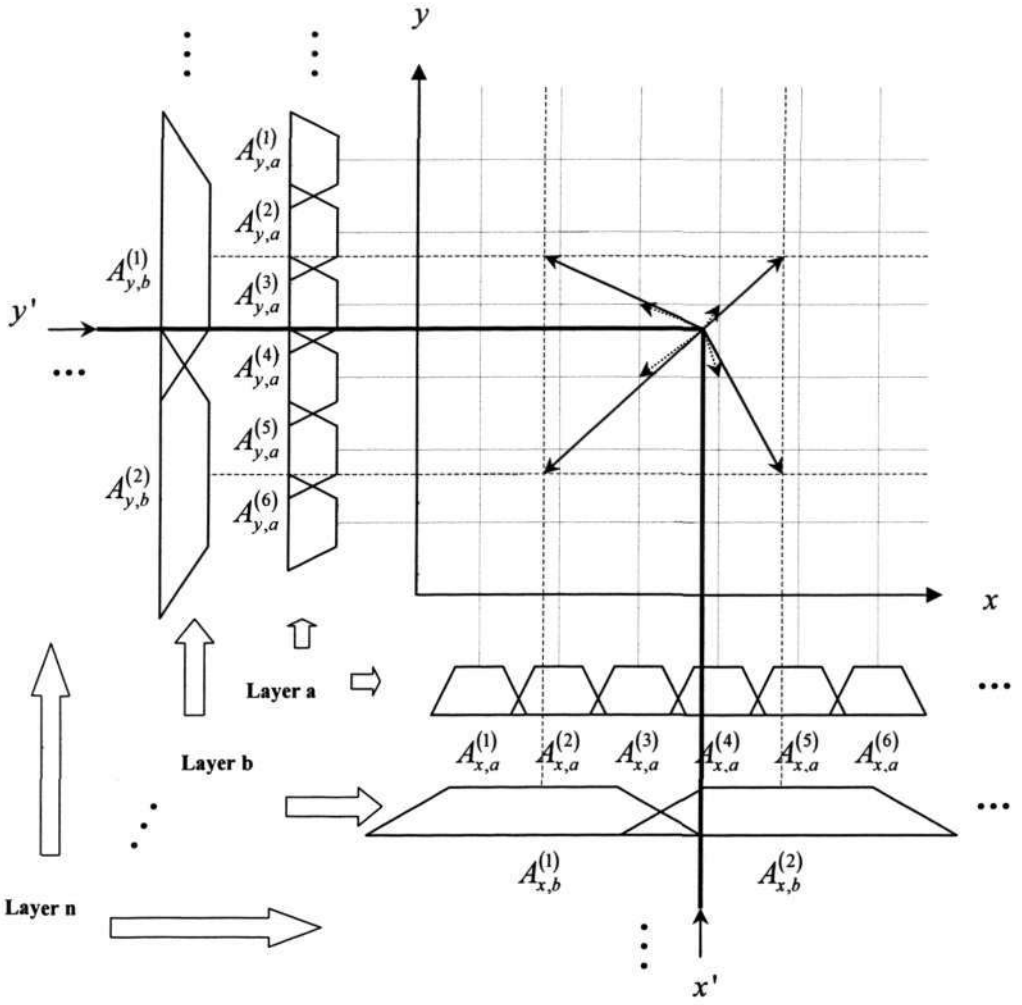


Figure 9-1: Structure of a 2-input TSK^0-FCMAC with Multiple Layers

- *Complex Membership Function*: The antecedents in the proposed architectures employed triangular and trapezoidal membership functions for simplicity and computational efficiency. It is known that Gaussian or bell-shaped membership functions produce a smoother approximation in the input space. However, higher computational power is required. With the advancement in technology, computational speed of computer has been significantly improved. The use of complex membership function will not have great impact on the time complexity. Therefore, a feasibility study on the implementation of complex membership function to generate smoother approximation will be a potential direction.

- *Momentum Learning*: Results from case-studies in section 3.5 show that the proposed architecture requires a small number of training epochs. To further improve the rate of

learning, an *inertia* or *momentum* term can be included in the learning algorithm. This will involve adding a fraction of the previous weight change to the current weight change to speed up the learning process.

- *Neighborhood Learning*: A CMAC network can be described in general terms as "functions with similar outputs for similar inputs" [34] or "input vectors that are close in the input space will give outputs that are close" [35]. This enthuse a feasible investigation on the use of neighborhood learning. This modification on the learning process will effectively distribute the activation effort into neighboring cells and enhance the generalization ability of the network.

- *Other applications*: The proposed architectures have been successfully applied in real-life case studies from three domains. They are: 1) personalized drug delivery system (control), 2) rainfall runoff (regression) and, 3) embedded audio detection (classification). In the domain of diabetic treatment, application could be further extended to life animal subject treatment as a second step. The goal will eventually lead to an alternative diabetic treatment for human subject. In additional, results from the case-studies (Chapter 6 to Chapter 8) suggest that the proposed architectures are also applicable for solving problems in different domains. Some possible applications include: financial analysis, actuarial risk analysis or autonomous vehicle control. Researchers are actively involved in applying neuro-fuzzy system into these applications and the proposed architectures may provide a plausible alternative.

Bibliography

[1] C.-T. Lin and C. S. Lee, *Neural Fuzzy Systems: a Neuro-Fuzzy Synergism on Intelligent System*. Upper Saddle River, NJ: Prentice-Hall PTR, 1996.

[2] J. S. Albus, "A new approach to manipulator control: the cerebellar model articulation controller (CMAC)," *Journal of Dynamic Systems, Measurement and Control, Transactions of ASME*, vol. 97, pp. 220-227, 1975.

[3] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338-359, 1965.

[4] T. M. Mitchell, *Machine Learning*: McGraw Hill, 1997.

[5] E. Alpaydin, *Introduction to Machine Learning*. Cambridge: MIT Press, 2004.

[6] J. M. Zurada, *Introduction to Artificial Neural Systems*. Boston: PWS Publishing Company, 1992.

[7] S. Haykin, *Neural Networks: A Comprehensive Foundation*. NY: Macmillan, 1994.

[8] L. A. Zadeh, "A theory of approximate reasoning," in *Machine Intelligence*, vol. 9, J. Hayes, Ed. New York: Halstead Press, 1979, pp. 149-194.

[9] J. Yen and R. Langari, *Fuzzy Logic: Intelligence, Control and Information*. New Jersey: Prentice Hall, 1998.

[10] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-3, pp. 28-44, 1973.

[11] L. A. Zadeh, "Fuzzy logic," *Computer*, vol. 1, pp. 88-93, 1988.

[12] H. J. Zimmermann, *Fuzzy Set Theory and Its Applications*. Boston: Kluwer Academic, 1988.

[13] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*. New York: Academic press, 1980.

[14] A. Kaufmann, *Introduction to the Theory of Fuzzy Subsets*. London: Academic Press, 1975.

[15] A. Kaufmann and M. M. Gupta, *Introduction to Fuzzy Arithmetic: Theory and Applications*. New York: Van Nostrand Reinhold Company, 1985.

[16] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. NJ: Prentice-Hall, 1991.

[17] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Machine Studies*, vol. 7, 1975.

[18] T. Takagi and M. Sugeno, "Derivation of fuzzy control rules from human operator's control actions," in *Proc. IFAC Symp. on Fuzzy Inform., Knowledge Representation and Decision Analysis*, 1983, pp. 55-60.

[19] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems*, vol. 28, pp. 15-33, 1988.

[20] B. Kosko, *Fuzzy Engineering*. NY: Prentice-Hall, 1997.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533-536, 1986.

[22] B. Widrow and J. M.E. Hoff, "Adaptive switching circuits," presented at 1960 IRE Western Electric Show and Conv. Rec., Part 4, 1960.

[23] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition," in *IEEE Computer Magazine*, vol. 21, 1988, pp. 25-39.

[24] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.

[25] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251-257, 1991.

[26] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational properties like those of two-state neurons," in *Proc. of the Nat. Academy of Sci. of the USA*, vol. 79, 1982, pp. 2554-2558.

[27] J. J. Hopfield, "Neurons with graded responses have collective computational properties like those of two-state neurons," in *Proc. of the Nat. Academy of Sci. of the USA*, vol. 81. USA, 1984, pp. 3088-3092.

[28] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, pp. 147-169, 1985.

[29] B. Kosko, "Adaptive bidirectional associative memories," *Appl. Opt.*, vol. 26, pp. 4947-4959, 1987.

[30] B. Kosko, "Bidirectional associative memories," *IEEE Trans. Syst. Man Cybern.*, vol. 18, pp. 49-60, 1988.

[31] J. S. Albus, "Theoretical and experimental aspects of a cerebellar model," Ph.D. dissertation, University of Maryland, 1972.

[32] W. T. Miller, III and F. H. Glanz, "Cerebellar model arithmetic computer," in *Fuzzy logic and neural network handbook*, C. H. Chen, Ed. New York: McGraw-Hill, 1996, pp. 26.1-26.31.

[33] J. S. Albus, "Data storage in the cerebellar model articulation controller (CMAC)," *Journal of Dynamic Systems, Measurement and Control, Transactions of ASME*, pp. 228-233, 1975.

[34] J. S. Albus, "Mechanisms of planning and problem solving in the brain," *Mathematical Biosciences*, vol. 45, pp. 247-293, 1979.

[35] W. T. Miller, III, F. H. Glanz, and L. G. Kraft, III, "CMAC: an associative neural network alternative to backpropagation," in *Proc. of the IEEE*, vol. 78, F. H. Glanz, Ed., 1990, pp. 1561-1567.

[36] L. G. Kraft and D. P. Campagna, "A comparison between CMAC neural network control and two traditional adaptive control systems," *Control Systems Magazine, IEEE*, vol. 10, pp. 36-43, 1990.

[37] C. Kambhampati, K. Warwick, and C. S. Berger, "A comparative study of multilayered and single layered neural network based predictive controllers," in *Proc. First Int. Conf. on Intell. Syst. Eng.*, vol. 360, 1992, pp. 293-298.

[38] J. Moody, "Fast learning in multi-resolution hierarchies," in *Advances in Neural Information Processing Systems*, 1 ed. D. S. Touretzky: Morgan Kaufmann Publishers, 1989, pp. 29-38.

[39] H. Kim and C. S. Lin, "Use of adaptive resolution for better CMAC learning," in *Proc. Int. Joint Conf. on Neural Networks, IJCNN*, vol. 1, 1992, pp. 517-522.

[40] F. J. Gonzalez-Serrano, A. R. Figueiras-Vidal, and A. Artes-Rodriguez, "Generalizing CMAC architecture and training," *IEEE Trans. Neural Netw.*, vol. 9, pp. 1509-1514, 1998.

[41] A. Kolcz and N. M. Allinson, "Basis function models of the CMAC network," *Neural Networks*, vol. 12, pp. 107-126, 1999.

[42] M. Brown, C. J. Harris, and P. C. Parks, "The interpolation capabilities of the binary CMAC," *Neural Networks*, vol. 6, pp. 429-440, 1993.

[43] Z. Shu, "Fuzzy associative memory for the automatic control of a continuous variable transmission control in an automobile," rep. on honours year project, Nanyang Technological University 2002.

[44] Z.-Q. Wang, J. L. Schiano, and M. Ginsberg, "Hash-coding in CMAC neural networks," in *Proc. Int. Conf. on Neural Netw.*, vol. 3, 1996, pp. 1698-1703.

[45] Z. Luo, Z. Zhao, and C. Zhu, "The unfavourable effects of hash coding on CMAC convergence and compensatory measure," in *IEEE Int. Conf. Intell. Process. Syst., ICIPS '97*, vol. 1, 1997, pp. 419-422.

[46] P. C. Parks and J. Militzer, "Convergence properties of associative memory storage for learning control system," *Automat. Remote Contr.*, vol. 50, pp. 254-286, 1989.

[47] P. C. Parks and J. Militzer, "A comparison of five algorithms for the training of CMAC memories for learning control systems," *Automatica*, vol. 28, pp. 1027-1035, 1992.

[48] Y. Wong and A. Sideris, "Learning convergence in the cerebellar model articulation controller," *IEEE Trans. Neural Netw.*, vol. 3, pp. 115-121, 1992.

[49] C.-S. Lin and C.-T. Chiang, "Learning convergence of CMAC technique," *IEEE Trans. Neural Netw.*, vol. 8, pp. 1281-1292, 1997.

[50] F.-C. Chen and C.-H. Chang, "Practical stability issues in CMAC neural network control systems," *IEEE Trans. Control Syst. Technol.*, vol. 4, pp. 86-91, 1996.

[51] C. Shang, D. Reay, and B. Williams, "Adapting CMAC neural networks with constrained LMS algorithm for efficient torque ripple reduction in switched reluctance motors," *IEEE Trans. Control Syst. Technol.*, vol. 7, pp. 401-413, 1999.

[52] C. Quek and P. W. Ng, "Realisation of neural network controllers in integrated process supervision," *Artificial Intelligence in Engineering*, vol. 10, pp. 135-142, 1996.

[53] D. E. Thompson and K. Sunggyu, "Neighborhood sequential and random training techniques for CMAC," *IEEE Trans. Neural Netw.*, vol. 6, pp. 196-202, 1995.

[54] C. Quek and A. Wahab, "Real-time integrated process supervision," *Engineering Applications of Artificial Intelligence*, vol. 13, pp. 645-658, 2000.

[55] K. K. Ang and C. Quek, "Improved MCMAC with momentum, neighborhood, and averaged trapezoidal output," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 30, pp. 491-500, 2000.

[56] K. K. Ang, C. Quek, and A. Wahab, "MCMAC-CVT: a novel on-line associative memory based CVT transmission control system," *Neural Networks*, vol. 15, pp. 219-236, 2002.

[57] J. Ozawa, I. Hayashi, and N. Wakami, "Formulation of CMAC-fuzzy system," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1992, pp. 1179-1186.

[58] J. Nie and D. A. Linkens, "A fuzzified CMAC self-learning controller," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1993, pp. 500-505.

[59] Z. Kai and Q. Feng, "Fuzzy CMAC and its application," in *Proc. World Congr. Intell. Control and Automation*, vol. 2, 2000, pp. 944-947.

[60] C.-C. Jou, "A fuzzy cerebellar model articulation controller," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1992, pp. 1171-1178.

[61] D. Kim, "A design of CMAC-based fuzzy logic controller with fast learning and accurate approximation," *Fuzzy Sets and Systems*, vol. 125, pp. 93-104, 2002.

[62] S. H. Lane, D. A. Handelman, and J. J. Gelfand, "Theory and development of higher-order CMAC neural networks," *IEEE Control. Syst. Mag.*, vol. 12, pp. 23-30, 1992.

[63] J.-S. Ker, C.-C. Hsu, Y.-H. Kuo, and B.-D. Liu, "A fuzzy CMAC model for color reproduction," *Fuzzy Sets and Systems*, vol. 91, pp. 53-68, 1997.

[64] M. N. Nguyen, D. Shi, and C. Quek, "FCMAC-BYY: fuzzy CMAC using bayesian ying-yang learning," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 36, pp. 1180-1190, 2006.

[65] C.-T. Chiang and C.-S. Lin, "CMAC with general basis functions," *Neural Networks*, vol. 9, pp. 1199-1211, 1996.

[66] S.-L. Hung and J. C. Jan, "MS_CMAC neural network learning model in structural engineering," *Journal of Computing in Civil Engineering*, vol. 13, pp. 1-11, 1999.

[67] J. C. Jan and H. Shih-Lin, "High-order MS CMAC neural network," *IEEE Trans. Neural Netw.*, vol. 12, pp. 598-603, 2001.

[68] C.-S. Lin and C.-K. Li, "A new neural network structure composed of small CMACs," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 3, 1996, pp. 1777-1783.

[69] C.-M. Chen and C.-M. Hong, "A weighted grey CMAC neural network with output differentiability," in *Proc. Int. Conf. IFSA World Congr. and 20th NAFIPS*, vol. 2, H. Chin-Ming, Ed., 2001, pp. 1009-1014.

[70] C. S. Berger, "Linear splines with adaptive mesh sizes for modelling nonlinear dynamic systems," *IEE Proc. Control Theory and Applications*, vol. 141, pp. 277-284, 1994.

[71] W. Yu, F. O. Rodriguez, and M. A. Moreno-Armendariz, "Hierarchical Fuzzy CMAC for Nonlinear Systems Modeling," *IEEE Trans. on Fuzzy Syst.*, vol. 16, pp. 1302-1314, 2008.

[72] C. S. Lin and H. Kim, "CMAC-based adaptive critic self-learning control," *IEEE Trans. Neural Netw.*, vol. 2, pp. 530-533, 1991.

[73] S. Alcozer, J. P. Segovia, and D. Sbarbaro, "Using higher order CMAC to improve the performance of control valves," *ISA Transactions*, vol. 36, pp. 65-70, 1997.

[74] F. J. González-Serrano, A. R. Figueiras-Vidal, and A. Artés-Rodriguez, "Fourier analysis of the generalized CMAC neural network," *Neural Networks*, vol. 11, pp. 391-396, 1998.

[75] G. Cembrano, G. Wells, J. Sardá, and A. Ruggeri, "Dynamic control of a robot arm using CMAC neural networks," *Control Engineering Practice*, vol. 5, pp. 485-492, 1997.

[76] Y. Iiguni, "Hierarchical image coding via cerebellar model arithmetic computers," *IEEE Trans. Image Process.*, vol. 5, pp. 1393-1401, 1996.

[77] Y. H. Kim and F. L. Lewis, "Optimal design of CMAC neural-network controller for robot manipulators," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 30, pp. 22-31, 2000.

[78] W. T. Miller, III, "Real-time neural network control of a biped walking robot," *IEEE Control. Syst. Mag.*, vol. 14, pp. 41-48, 1994.

[79] J. S. Ker, Y. H. Kuo, and B. D. Liu, "Systolic implementation of higher-order CMAC and its application in colour calibration," *IEE Proc. Circuits, Devices and Syst.*, vol. 144, pp. 129-137, 1997.

[80] J.-S. Ker, Y.-H. Kuo, R.-C. Wen, and B.-D. Liu, "Hardware implementation of CMAC neural network with reduced storage requirement," *IEEE Trans. Neural Netw.*, vol. 8, pp. 1545-1556, 1997.

[81] H. Shiraishi, S. L. Ipri, and D. I. D. Cho, "CMAC neural network controller for fuel-injection systems," *IEEE Trans. Control Syst. Technol.*, vol. 3, pp. 32-38, 1995.

[82] L. Xu, "Advances on BYY harmony learning: information theoretic perspective, generalized projection geometry, and independent factor autodetermination," *IEEE Trans. Neural Netw.*, vol. 15, pp. 885-902, 2004.

[83] J. Kim and N. Kasabov, "HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems," *Neural Networks*, vol. 12, pp. 1301-1319, 1999.

[84] J. S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man Cybern.*, vol. 23, pp. 665-685, 1993.

[85] W. L. Tung and C. Quek, "GenSoFNN: a generic self-organizing fuzzy neural network," *IEEE Trans. Neural Netw.*, vol. 13, pp. 1075-1086, 2002.

[86] C.-J. Lin and C.-T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 477-496, 1997.

[87] C. Quek and W. L. Tung, "A novel approach to the derivation of fuzzy membership functions using the Falcon-MART architecture," *Pattern Recognition Letters*, vol. 22, pp. 941-958, 2001.

[88] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. of Eugenics*, vol. 7, pp. 179-188, 1936.

[89] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, pp. 287-289, 1977.

[90] W. L. Tung and C. Quek, "DIC: a novel discrete incremental clustering technique for the derivation of fuzzy membership functions," presented at Proc. of 7th Pacific Rim Int. Conf. on Artificial Intell., Tokyo, 2002.

[91] K. K. Ang, C. Quek, and M. Pasquier, "POPFNN-CRI(S): pseudo outer product based fuzzy neural network using the compositional rule of inference and singleton fuzzifier," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 33, pp. 838-849, 2003.

[92] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759-771, 1991.

[93] T. K. Kohonen, "Self-organized formation of topologically correct feature maps," *Bio. Cybern.*, vol. 43, pp. 59-69, 1982.

[94] G. Feng, S. G. Cao, and N. W. Rees, "An approach to H-infinity control of a class of nonlinear systems," *Automatica*, vol. 32, pp. 1469-1474, 1996.

[95] S. Khoo, "Variable structure control with applications to T-S fuzzy systems," Ph.D. dissertation, School of Computer Engineering, Nanyang Technological University, 2007.

[96] W. L. Tung and C. Quek, "Falcon: neural fuzzy control and decision systems using FKP and PFKP clustering algorithms," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 34, pp. 686-695, 2004.

[97] W. Assawinchaichote, S. K. Nguang, and P. Shi, "H-Infinity output feedback control design for uncertain fuzzy singularly perturbed systems: an LMI approach," *Automatica*, vol. 40, pp. 2147-2152, 2004.

[98] S. Garg, "Robust integrated flight/propulsion control design for a STOVL aircraft using H-infinity control design techniques," *Automatica*, vol. 29, pp. 129-145, 1993.

[99] T. Iwasaki and R. E. Skelton, "All fixed-order H-Infinity controllers: observer-based structure and covariance bounds," *IEEE Trans. Autom. Control*, vol. 40, pp. 512-516, 1995.

[100] R. A. Nichols, R. T. Reichert, and W. J. Rugh, "Gain scheduling for H-infinity controllers: a flight control example," *IEEE Trans. Control Syst. Technol.*, vol. 1, pp. 69-79, 1993.

[101] M.-S. Chen, C.-H. Chen, and F.-Y. Yang, "An LTR-observer-based dynamic sliding mode control for chattering reduction," *Automatica*, vol. 43, pp. 1111-1116, 2007.

[102] S. C. Chung and C.-L. Lin, "A general class of sliding surface for sliding mode control," *IEEE Trans. Autom. Control*, vol. 43, pp. 115-119, 1998.

[103] C. K. Lin, "Nonsingular terminal sliding mode control of robot manipulators using fuzzy wavelet networks," *IEEE Trans. Fuzzy Syst.*, vol. 14, pp. 849-859, 2006.

[104] Y. B. Shtessel, I. A. Shkolnikov, and A. Levant, "Smooth second-order sliding modes: Missile guidance application," *Automatica*, vol. 43, pp. 1470-1476, 2007.

[105] J. Nie, "Nonlinear time-series forecasting: A fuzzy-neural approach," *Neurocomputing*, vol. 16, pp. 63-76, 1997.

[106] F. J. de Souza, M. M. R. Vellasco, and M. A. C. Pacheco, "Hierarchical neuro-fuzzy quadtree models," *Fuzzy Sets and Systems*, vol. 130, pp. 189-205, 2002.

[107] I. Tokuda, R. Tokunaga, and K. Aihara, "Back-propagation learning of infinite-dimensional dynamical systems," *Neural Networks*, vol. 16, pp. 1179-1193, 2003.
[108] C.-J. Lin and Y.-J. Xu, "A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications," *Fuzzy Sets and Systems*, vol. 157, pp. 1036-1056, 2006.
[109] F. Cellier and E. Kofman, *Continuous System Simulation*: Springer Verlag, 2006.
[110] J. Sim, W. L. Tung, and C. Quek, "FCMAC-Yager: a novel Yager-inference-scheme-based fuzzy CMAC," *IEEE Trans. Neural Netw.*, vol. 17, pp. 1394-1410, 2006.
[111] S.-F. Su, Z.-J. Lee, and Y.-P. Wang, "Robust and fast learning for fuzzy cerebellar model articulation controllers," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 36, pp. 203-208, 2006.
[112] Z. Tang and X. a. Yan, "Fuzzy CMAC model predictive control," in *Proc. Int. Conf. Fuzzy Syst. and Knowledge Discovery, FSKD*, vol. 2, X. a. Yan, Ed., 2007, pp. 566-569.
[113] F. O. Rodriguez, W. Yu, and M. A. Moreno-Armendariz, "Recurrent fuzzy CMAC in hierarchical form for dynamic system identification," in *Proc. Amer. Control Conf., ACC '07*, W. Yu, Ed., 2007, pp. 5706-5711.
[114] G. S. Reis and P. E. M. Almeida, "Modified fuzzy-CMAC networks with clustering-based structure," presented at Proc. Int. Joint Conf. Neural Netw., IJCNN '06, 2006.
[115] Z. Shen, C. Guo, and H. Li, "General fuzzified CMAC based model reference adaptive control for ship steering," in *Proc. IEEE Int. Symp. Mediterrean Conf. on Control and Autom.*, C. Guo, Ed., 2005, pp. 1257-1262.
[116] G. Wang and W. Ning, "Application of fuzzy CMAC to filament tension control," in *Proc. Int. Conf. Fuzzy Syst. and Knowledge Discovery*, vol. 4, W. Ning, Ed., 2007, pp. 164-167.
[117] G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control*: Prentice-Hall, 1984.
[118] P. Strobach, *Linear Prediction Theory : A Mathematical Basis for Adaptive Systems*. New York: Springer-Verlag, 1990.
[119] M. S. Bartlett, "An inverse matrix adjustment arising in discriminant analysis," *The Annals of Mathematical Statistics*, vol. 22, pp. 107-111, 1951.
[120] D. Goldfarb, "Modification methods for inverting matrices and solving systems of linear algebraic equations," *Mathematics of Computation*, vol. 26, pp. 829-852, 1972.
[121] H. V. Henderson and S. R. Searle, "On deriving the inverse of a sum of matrices," *SIAM Review*, vol. 23, pp. 53-60, 1981.
[122] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in the elements of a given column or a given row of the original matrix," *The Annals of Mathematical Statistics*, vol. 20, pp. 621, 1949.
[123] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *The Annals of Mathematical Statistics*, vol. 21, pp. 124-127, 1950.
[124] L. Guo, "Estimating time-varying parameters by the Kalman filter based algorithm: stability and convergence," *IEEE Trans. Autom. Control*, vol. 35, pp. 141-147, 1990.
[125] "UCI machine learning repository," [Online]. Available: http://archive.ics.uci.edu/ml/. [Accessed: May 1, 2008].
[126] K. H. Quah and C. Quek, "MCES: a novel monte carlo evaluative selection approach for objective feature selections," *IEEE Trans. Neural Netw.*, vol. 18, pp. 431-448, 2007.

[127] J. S. R. Jang, "Input selection for ANFIS learning," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, vol. 2, 1996, pp. 1493-1499.

[128] K. J. Lang and M. J. Witbrock, "Learning to tell two spirals apart," in *Proc. Connectionist Models Summer School*, 1988, pp. 52-59.

[129] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, pp. 698-713, 1992.

[130] Z. Ying, F. Huajing, and H. O. Wang, "Takagi-sugeno fuzzy-model-based fault detection for networked control systems with Markov delays," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 36, pp. 924-929, 2006.

[131] R. J. Morff, et al., "Microfabrication of reproducible, economical, electroenzymatic glucose sensors," in *Proc. 12th IEEE Annu. Int. Conf. Eng. in Medicine and Biology Soc.*, 1990, pp. 483-484.

[132] J. J. Mastrototaro, et al., "Preliminary clinical results from an electroenzymatic glucose sensor implanted in subcutaneous tissue," in *Proc. 14th IEEE Annu. Int. Conf. Eng. in Medicine and Biology Soc.*, 1992, pp. 153-154.

[133] J. A. Tamada, M. Lesho, and M. J. Tierney, "Keeping watch on glucose," *IEEE Spectr.*, vol. 39, pp. 52-57, 2002.

[134] R. Bergman, L. Phillips, and C. Cobelli, "Physiologic evaluation of factors controlling glucose tolerance in man," *J. Clin. Invest.*, vol. 68, pp. 1456-1467, 1981.

[135] J. R. Guyton, et al., "A model of glucose-insulin homeostasis in man that incorporates the heterogenous fast pool theory of pancreatic insulin release," *Diabetes Care*, vol. 27, pp. 1027-1042, 1978.

[136] J. T. Sorensen, "A physiologic model of glucose metabolism in man and its use to design and assess improved insulin therapies for diabetes," Ph.D. dissertation, Dept. of Chemical Eng., Massachusetts Inst. of Technology, 1985.

[137] M. Berger and D. Rodbard, "Computer simulation of plasma insulin and glucose dynamics after subcutaneous insulin injection," *Diabetes Care*, vol. 12, pp. 725-736, 1989.

[138] W. R. Puckett, "Dynamic modeling of diabetes mellitus," Ph.D. dissertation, Dept. of Chemical Eng., Wisconsin-Madison University, 1992.

[139] J. Seul, C. Hyun-Taek, and T. C. Hsia, "Neural Network Control for Position Tracking of a Two-Axis Inverted Pendulum System: Experimental Studies," *Neural Networks, IEEE Transactions on*, vol. 18, pp. 1042-1048, 2007.

[140] C.-J. Lin and C.-T. Lin, "Corrections to "Reinforcement Learning for an ART-Based Fuzzy Adaptive Learning Control Network"," *Neural Networks, IEEE Transactions on*, vol. 7, pp. 1315, 1996.

[141] F. Chee, T. Fernando, and P. V. van Heerden, "Closed-loop glucose control in critically ill patients using continuous glucose monitoring system (CGMS) in real time," *IEEE Trans. Inf. Technol. Biomed.*, vol. 7, pp. 43-53, 2003.

[142] R. S. Parker, F. J. Doyle, III, and N. A. Peppas, "A model-based algorithm for blood glucose control in Type I diabetic patients," *IEEE Trans. Biomed. Eng.*, vol. 46, pp. 148-157, 1999.

[143] A. Roy and R. S. Parker, "Mixed meal modeling and disturbance rejection in type I diabetic patients," in *Proc. 28th IEEE Annu. Int. Conf. Eng. in Medicine and Biology Soc.*, 2006, pp. 323-326.

[144] F. Chee, T. L. Fernando, A. V. Savkin, and V. van Heeden, "Expert PID control system for blood glucose control in critically ill patients," *IEEE Trans. Inf. Technol. Biomed.*, vol. 7, pp. 419-425, 2003.

[145] F. Chee, A. V. Savkin, T. L. Fernando, and S. Nahavandi, "Optimal H-infinity insulin injection control for blood glucose regulation in diabetic patients," *IEEE Trans. Biomed. Eng.*, vol. 52, pp. 1625-1631, 2005.

[146] K. H. Kienitz and T. Yoneyama, "A robust controller for insulin pumps based on H-infinity theory," *IEEE Trans. Biomed. Eng.*, vol. 40, pp. 1133-1137, 1993.

[147] P. Dua, F. J. Doyle, and E. N. Pistikopoulos, "Model-based blood glucose control for type 1 diabetes via parametric programming," *IEEE Trans. Biomed. Eng.*, vol. 53, pp. 1478-1491, 2006.

[148] V. Tresp, T. Briegel, and J. Moody, "Neural-network models for the blood glucose metabolism of a diabetic," *IEEE Trans. Neural Netw.*, vol. 10, pp. 1204-1213, 1999.

[149] F. Andrianasy and M. Milgram, "Applying neural networks to adjust insulin-pump doses," in *Proc. IEEE Workshop Neural Networks for Signal Process.*, 1997, pp. 182-188.

[150] M. F. Alamaireh, "A predictive neural network control approach in diabetes management by insulin administration," in *Proc. Int. Conf. Inform. and Commun. Technologies*, 2006, pp. 1618-1623.

[151] S. G. Mougiakakou, K. Prountzou, and K. S. Nikita, "A real time simulation model of glucose-insulin metabolism for type 1 diabetes patients," in *Proc. 27th IEEE Annu. Int. Conf. Eng. in Medicine and Biology Soc.*, 2005, pp. 298-301.

[152] D. U. Campos-Delgado, M. Hernandez-Ordonez, R. Femat, and A. Gordillo-Moscoso, "Fuzzy-based controller for glucose regulation in type-1 diabetic patients by subcutaneous route," *IEEE Trans. Biomed. Eng.*, vol. 53, pp. 2201-2210, 2006.

[153] F. C. Erzen, G. Birol, and A. Cinar, "Glucosim: a simulator for education on the dynamics of diabetes mellitus," in *Proc. 23th IEEE Annu. Int. Conf. Eng. in Medicine and Biology Soc.*, 2001, pp. 3163-3166.

[154] "The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus," *N. Engl. J. Med.*, vol. 329, pp. 977-986, 1993.

[155] H. K. Lam, F. H. Leung, and P. K. S. Tam, "Design and stability analysis of fuzzy model-based nonlinear controller for nonlinear systems using genetic algorithm," *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, vol. 33, pp. 250-257, 2003.

[156] J. A. Harris and F. G. Benedict, *A Biometric Study of Basal Metabolism in Man.* Washington DC: Carnegie Inst. of Washington, 1919.

[157] V. Morkus, "Estimation of direct runoff from storm rainfall," in *National Engineering Handbook, Section 4: Hydrology, Chapter 11*: Washington D.C: Soil Conservation Service (SCS), U.S. Department of Agriculture, 1972.

[158] V. T. Chow, D. R. Maidment, and L. W. Mays, *Applied Hydrology*: McGraw-Hill, 1988.

[159] B. d. Saint-Venant, "Theory of unsteady water flow, with application to river floods and to propagation of tides in river channels," *Frence Academy of Science*, vol. 73, pp. 148-154, 1871.

[160] G. Tayfur and V. P. Singh, "ANN and fuzzy logic models for simulating event-based rainfall-runoff," *Journal of Hydraulic Engineering*, vol. 132, pp. 1321-1330, 2006.

[161] A. S. Tokar and M. Momcilo, "Precipitation-runoff modeling using artificial neural networks and conceptual models," *Journal of Hydrologic Engineering*, vol. 5, pp. 156-161, 2000.

[162] M. P. Rajurkar, U. C. Kothyari, and U. C. Chaube, "Modeling of the daily rainfall-runoff relationship with artificial neural network," *Journal of Hydrology*, vol. 285, pp. 96-113, 2004.

[163] M. R. Gautam, K. Watanabe, and H. Saegusa, "Runoff analysis in humid forest catchment with artificial neural network," *Journal of Hydrology*, vol. 235, pp. 117-136, 2000.

[164] M. Aqil, I. Kita, A. Yano, and S. Nishiyama, "A comparative study of artificial neural networks and neuro-fuzzy in continuous modeling of the daily and hourly behaviour of runoff," *Journal of Hydrology*, vol. 337, pp. 22-34, 2007.

[165] G. Kuczera, D. Kavetski, S. Franks, and M. Thyer, "Towards a Bayesian total error analysis of conceptual rainfall-runoff models: characterising model error using storm-dependent parameters," *Journal of Hydrology*, vol. 331, pp. 161-177, 2006.

[166] E. C. Özelkan and L. Duckstein, "Fuzzy conceptual rainfall-runoff models," *Journal of Hydrology*, vol. 253, pp. 41-68, 2001.

[167] L. Xiong, A. Y. Shamseldin, and K. M. O'Connor, "A non-linear combination of the forecasts of rainfall-runoff models by the first-order Takagi-Sugeno fuzzy system," *Journal of Hydrology*, vol. 245, pp. 196-217, 2001.

[168] F.-J. Chang and Y.-C. Chen, "A counterpropagation fuzzy-neural network modeling approach to real time streamflow prediction," *Journal of Hydrology*, vol. 245, pp. 153-164, 2001.

[169] A. P. Jacquin and A. Y. Shamseldin, "Development of rainfall-runoff models using Takagi-Sugeno fuzzy inference systems," *Journal of Hydrology*, vol. 329, pp. 154-173, 2006.

[170] S.-Y. Liong, S.-T. Khu, and W.-T. Chan, "Derivation of pareto front with genetic algorithm and neural network," *Journal of Hydrologic Engineering*, vol. 6, pp. 52-61, 2001.

[171] C. T. Cheng, C. P. Ou, and K. W. Chau, "Combining a fuzzy optimal model with a genetic algorithm to solve multi-objective rainfall-runoff model calibration," *Journal of Hydrology*, vol. 268, pp. 72-86, 2002.

[172] F. Anctil, N. Lauzon, V. Andréassian, L. Oudin, and C. Perrin, "Improvement of rainfall-runoff forecasts through mean areal rainfall optimization," *Journal of Hydrology*, vol. 328, pp. 717-725, 2006.

[173] C.-T. Cheng, M.-Y. Zhao, K. W. Chau, and X.-Y. Wu, "Using genetic algorithm and TOPSIS for Xinanjiang model calibration with a single procedure," *Journal of Hydrology*, vol. 316, pp. 129-140, 2006.

[174] G.-F. Lin and C.-M. Wang, "A nonlinear rainfall-runoff model embedded with an automated calibration method - part 2: the automated calibration method," *Journal of Hydrology*, vol. 341, pp. 196-206, 2007.

[175] T. S. W. Wong and C. K. Lim, "Effect of loss model on evaluation of Manning roughness coefficient of experimental concrete catchment," *Journal of Hydrology*, vol. 331, pp. 205-218, 2006.

[176] H. Methods and S. R. Durrans, *Stormwater convenyance modeling and design*. Waterbury, CT USA: Haestad Press, 2003.

[177] G. T. McCarthy, "The unit hydrograph and flood routing," presented at Conf. of the North Atlantic Div., United States Army Corps of Engineers, 1938.

[178] V. T. Chow, *Open Channel Hydraulics*. New York: McGraw-Hill, 1959.

[179] P. B. Bedient, C. H. Wayne, and B. E. Vieux, *Hydrology and Floodplain Analysis*, 4th ed: Prentice Hall, 2008.

[180] K. A. Cunge, "On the subject of a flood propagation method (Muskingum Method)," *Journal of Hydraulic Research*, vol. 7, pp. 205-230, 1969.

[181] V. M. Ponce, R. M. Li, and D. B. Simons, "Applicability of kinematic and diffusion models," *Journal of Hyraulic Division, ASCE*, vol. 104, pp. 353-360, 1978.

[182] B. C. Yen, "Hydraulics of storm sewer systems," in *Stormwater Collection Systems Design Handbook*. New York: McGraw-Hill, 2001.

[183] V. Mockus and W. Steiner, "Flood routing," in *National Engineering Handbook, Section 4 Hydrology, Chapter 17*. Washington D.C: Soil Conservation Service (SCS), U.S. Department of Agriculture, 1972.

[184] V. M. Ponce, "Kinematic Wave Controversy," *Journal of Hydraulic Engineering*, vol. 117, pp. 511-525, 1991.

[185] G. Aron, *Penn State Runoff Model for IBM PC: User Manual*. Pennsylvania: Dept. of Civil Engineering and Institute for Research on Land and Water Resources, Pennsylvania State University, 1987.

[186] "HEC-1, flood hydrograph package." U.S. Army Corps of Engineers, Hydrologic Engineering Center, 1990.

[187] M. J. Lighthill and G. B. Whitham, "On kinematic waves. I: flood movement in long rivers," in *Proc. Royal Society*, vol. A229. London, England, 1955, pp. 281-316.

[188] S. Hayami, "On the propagation of flood waves," in *Bulletin of the Disaster Prevention Research Institute, Disaster Prevention Research Institute*, vol. 1, 1951, pp. 1-16.

[189] V. M. Ponce and D. B. Simons, "Shallow wave propagation in open channel flow," *Journal of Hyraulic Division, ASCE*, vol. 103, pp. 1461-1476, 1977.

[190] J. A. Seddon, "River hydraulics," *Trans., ASCE*, vol. 43, pp. 179–229, 1900.

[191] E. Levi, *The Science of Water: The Foundation of Modern Hydraulics*. New York: ASCE Press, translated from Spanish by D. E. Medina, 1995.

[192] R. Manning, "On the flow of water in open channels and pipes," *Transactions of the Institution of Civil Engineers of Ireland*, vol. 20, pp. 161-207, 1891.

[193] T. S. W. Wong, "Optimum rainfall interval and Manning's roughness coefficient for runoff simulation," *Journal of Hydrologic Engineering*, vol. 13, pp. 1097-1102, 2008.

[194] E. T. Engman, "Roughness coefficients for routing surface runoff," *Journal of Irrigation and Drainage Engineering*, vol. 112, pp. 39-53, 1986.

[195] L. W. Mays, *Stormwater Collection Systems Design Handbook*. NY: McGraw-Hill, 2001.

[196] J. E. Nash and J. V. Sutcliffe, "River flow forecasting through conceptual models part I -- A discussion of principles," *Journal of Hydrology*, vol. 10, pp. 282-290, 1970.

[197] C. W. Ting, "Embedded audio detection software," Honours Year Rep., School of Computer Engineering, Nanyang Technological University, Singapore 2001.

[198] W.-Y. Chen, S.-H. Chen, and C.-J. Lin, "A speech recognition method based on the sequential multi-layer perceptrons," *Neural Networks*, vol. 9, pp. 655-669, 1996.

[199] K. Yamauchi, M. Fukuda, and K. Fukushima, "Speed invariant speech recognition using variable velocity delay lines," *Neural Networks*, vol. 8, pp. 167-177, 1995.

[200] K. Yamauchi, M. Oota, and N. Ishii, "A self-supervised learning system for pattern recognition by sensory integration," *Neural Networks*, vol. 12, pp. 1347-1358, 1999.

[201] S. Albrecht, J. Busch, M. Kloppenburg, F. Metze, and P. Tavan, "Generalized radial basis function networks for classification and novelty detection: self-organization of optimal Bayesian decision," *Neural Networks*, vol. 13, pp. 1075-1093, 2000.

[202] M. Kermit and Å. J. Eide, "Audio signal identification via pattern capture and template matching," *Pattern Recognition Letters*, vol. 21, pp. 269-275, 2000.

[203] J. Moody and C. Darken, "Fast Learning in Networks of Locally Tuned Processing Units," *Neural Computations*, vol. 1, pp. 281-294, 1989.

[204] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed: MIT Press and McGraw-Hill, 2001.

[205] H. Almuallim and T. G. Dietterich, "Learning Boolean concepts in the presence of many irrelevant features," *Artificial Intelligence*, vol. 69, pp. 279-305, 1994.

[206] K. Kira and L. A. Rendell, "The feature selection problem: traditional method and a new algorithm," presented at 10th Nat. Conf. Artif. Intell., 1992.

[207] K. Kira and L. A. Rendell, "A practical approach to feature selection," presented at 9th Int. Conf. Machi. Learn., 1992.

[208] M. A. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 15, pp. 1437-1447, 2003.

[209] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," presented at 17th Conf. Mach. Learn., 2000.

[210] M. Robnik-Sikonja and I. Kononenko, "An adaptatio of relief for attribute estimation in regression," presented at 14th Int. Conf. Mach. Learn., 1997.

[211] R. Plamondon and G. Lorette, "Automatic signature verification and writer identification -- the state of the art," *Pattern Recognition*, vol. 22, pp. 107-131, 1989.

Author's Publications

C.W. Ting and C. Quek, "A novel blood glucose regulation using TSK^0-FCMAC: a fuzzy cmac based on the zero-ordered TSK fuzzy inference scheme", *IEEE Trans. Neural Networks*, vol. 20, pp. 856-871, 2009.

C.W. Ting and C. Quek, "Learning convergence of TSK^0-FCMAC: a localized self-organizing TSK fuzzy inference system based on CMAC structure", submitted for publication, 2008.

C.W. Ting and C. Quek, "Rainfall runoff modeling for concrete plane with TSK^0-FCMAC: a localized self-organizing TSK fuzzy inference system based on CMAC structure", paper under preparation.

C.W. Ting and C. Quek, "Glucose-insulin dynamics modeling in healthy human subject", paper under preparation.

C.W. Ting, C. Quek and Ian McLoughlin, "A real-time embedded sound recognition system based on soft computing technique", paper under preparation.