DESIGN AND ANALYSIS OF REAL INTELLIGENT MAPPING SYSTEMS WITH APPLICATIONS TO SYSTEMS AND CONTROL

By

Yeung Wai Leung

A Thesis

submitted in partial fulfillment of the requirements

for the Degree of Master of Philosophy

Division of Information Engineering

The Chinese University of Hong Kong

June 1995

Acknowledgement

I would like to express my acknowledgement to my supervisor, Dr. C. P. Kwong, for his supervision throughout these two years. I am grateful to him for his valuable advice and guidances.

I would also like to thank Prof. Z. B. Xu, Mr. T. K. Koo, Ms Z. Y. Chen, Mr. John Sum - for their helpful discussions on various topics.

Abstract

The recently proposed *Real Intelligent Mapping* (RIM) [1] argues that transformation of the real input data to the fuzzy domain and then back to the real domain again in the fuzzy inference approach is essentially unnecessary in many applications. Instead, the fuzzy inference procedure is replaced by a mapping operating directly on real data. In the thesis, we introduce the use of *Dirichlet Tessellation* (DT) for the implementation of RIM systems, which is general enough to attack approximation problems regardless of data distribution. Roughly speaking, the method partitions the state-space data points into simplexes to form the input-output relation and then represents each simplex by a linear equation. The piecewise-linear property of the DT-based RIM systems makes qualitative analysis possible. As an important example, we analyze the stability property of DT-based RIM control systems by using Liapunov's direct and indirect methods. The proposed method for designing RIM systems can handle different types of expert knowledge and it has been used successfully to solve serveral well-known problems such as balancing an inverted pendulum, inverted pendulum with cart, truck backing-up, and chaotic time series prediction. Also, an interactive CAD platform has been developed to enhance the RIM systems design based on the DT method.

Contents

Chapter 1

Introduction

1.1 Fuzzy Inference and Real Intelligent Mapping

In ordinary non-fuzzy approximation problems, the choice of using which particular approximation method is usually problem dependent. In other words, a method that works well for one problem may not be so efficient for another problem. To choose a suitable method to solve a particular approximation problem, one first need to get some knowledge of the *geometric distribution*, the *quantity* and the *dimension* of the data to be approximated. To achieve this, some statistical methods may be used. However, this is not always possible especially when there are numerous data points to be approximated or when the dimension of the problem is high.

Fuzzy inference (FI) can be considered as one of the many existing function approximation methods and it has been proved to be an *universal approximator* [20][21]. The power of FI is based on the fact that it is a *general* method of

solving approximation problem, in the sense that there is no need to consider the characteristics of the geometric distribution of the data (in terms of linguistic fuzzy if-then rules) in the space during the approximation process. The reason is that FI is a piecewise approximation technique when viewed in the real domain. Provided that the data is distributed in a regular manner (i.e. the regular distribution of the fuzzy if-then rules in the fuzzy space), the overall system can be piecewisely divided into simple subsystems regardless of the dimension and complexity of the system. Thus, the quality of fuzzy approximation is to some extend assured by choosing reasonable membership functions and a good inference mechanism.

It is generally claimed that fuzzy inference is best applied to complex system design in which no mathematical model exists but experienced human operators are available for providing expert knowledge in terms of linguistic fuzzy if-then rules. While this approach has emerged as an alternative solution to several system design problems, we pose the following fundamental question: Is the use of linguistic fuzzy if-then rules a good way of representing the human expert knowledge? Our study shows that the answer is essentially negative. Firstly, the design of fuzzy inference systems is usually performed in an ad hoc manner; it is hard to justify the effect of system parameters' change on the final input-output relation because of the very nonlinear fuzzy inference mechanism acting as a barrier between the input and output variables. Secondly, fuzzy systems require considerable computation time when they are used for real time inference since every input has to pass through the following three stages to obtain the desired output: (1) Fuzzify the real input data via their respective membership functions; (2) Perform inference in the fuzzy domain following a

set of if-then rules; (3) Return a real inference result by defuzzifying the fuzzy consequent. Furthermore, fuzzy inference is based on linguistic fuzzy if-then rules distributed in a regular manner. Its ability to approximate data which is not regularly distributed (e.g. sampled input-output data points from successful control) is limited. Last but not least, due to the very nonlinear fuzzy inference mechanisms, qualitative analysis of fuzzy inference systems is very difficult if not impossible. This hinders the further development of fuzzy inference to handle more sophisticated problems.

In fact, stability analysis of fuzzy system has been studied for a period of time. Braae and Rutherford [27][28] proposed a linguistic phase plane trajectory to analyze and improve the stability of fuzzy systems by exchanging the control rules. Kickert and Mamdani [26] use the describing function method to evaluate the stability of fuzzy control systems. B. Kiszka [29] introduced the energetistic stability of fuzzy dynamic systems and developed an entropy for fuzzy system. Besides, De Glas [24] and A. Kania [23] use the concept of α stability for analyzing fuzzy systems, in which the distinction between stability and instability is removed and a real value between 0 and 1 is used to describe the "degree of stability" of a fuzzy system. However, all the above stated methods are in fact quite restrictive and have only limited applicability.

On the other hand, the recently proposed *Real Intelligent Mapping* (RIM) [1] argues that transformation of the real input data to the fuzzy domain and then back to the real domain again in the fuzzy inference approach is essentially unnecessary in many applications. Instead, the fuzzy inference procedure is replaced by a mapping operating directly on real data.

RIM has already been used to solve the well-known truck backing-up problem by using linear regression [1]. However, linear regression is only one of the many approximation methods that can be used to design a RIM system. To make the RIM systems design more problem independent, we introduce the use of Dirichlet Tessellation for the implementation of RIM systems, which is general enough to attack approximation problems regardless of data distribution. Roughly speaking, the method partitions the state-space data points into simplexes to form the input-output relation and then represents each simplex by a linear equation. Thus, the resulting approximation is piecewise-linear. Real time computation time is saved when compared to the conventional fuzzy inference since the input has to pass through only one stage to get the desired output, namely, substitute the input value to the appropriate *real* function. Furthermore, the piecewise-linear property of the DT-based RIM systems makes qualitative analysis possible. As an important example, we analyze the stability property of DT-based RIM control systems by using Liapunov's direct and indirect methods. We first analyze the local stability property of them by Liapunov's linearization method. We then analyze the globally stability property of them by a method based on Liapunov's direct method and show that their design can be based on linear control theory.

The proposed method for designing RIM systems can handle different types of expert knowledge and it has been used successfully to solve several well-known problems such as balancing an inverted pendulum, inverted pendulum with cart, truck backing-up, and chaotic time series prediction. Also, an interactive CAD platform has been developed to enhance the RIM design based on the DT method.

1.2 Organization of the thesis

In Chapter 2, we first introduce the basic idea of fuzzy inference. Fuzzy set theory is reviewed and the fuzzy inference mechanism is introduced with an example.

In Chapter 3, we try to state out the weaknesses of fuzzy inference which initiate our study of the newly proposed Real Intelligent Mapping (RIM) approach.

In Chapter 4, we introduce the general idea of Real Intelligent Mapping.

In Chapter 5, we introduce the method of using *Dirichlet Tessellation* (DT) in the design of Real Intelligent Mapping systems. We first introduce our proposed method of function approximation using Dirichlet Tessellation. Afterward, we introduce a means to limit the number of generated functions when the number of input data points is large. This method ensures that the resulting approximation is optimal in the sense of *Least-Squares*. Finally, we apply DT based RIM to solve some well known problems, including (1) Balancing an inverted pendulum; (2) Inverted pendulum with cart; (3) Truck backing up; and (4) Chaotic time series prediction. We also introduce the interactive CAD platform that has been developed to enhance the RIM design.

In Chapter 6, we try to analyze the stability property of RIM systems by Liapunov's direct and indirect method. We first analyze the local stability property of a DT based RIM system by Liapunov's linearization method with the inverted pendulum problem and the truck backing-up problem used as examples. Then we analyze the globally stability property of DT based RIM systems by a method based on Liapunov's direct method and show that their design

can be based on linear system theory. Finally, we elaborate the stability analysis method to piecewise-nonlinear systems and show that the method can be applied to a class of conventional fuzzy system by establishing an equivalent piecewise-polynomial representation of it in real domain.

In Chapter 7, we conclude the thesis by giving a brief summary on what have been done and their significance.

Chapter 2

Fuzzy Logic inference

In this chapter, we introduce the basic idea of conventional fuzzy inference (CFI). Fuzzy set theory is reviewed and the fuzzy inference mechanism is introduced with an example. Some of the materials in this section are obtained from [45].

2.1 Fuzzy logic

2.1.1 Fuzzy sets

The concept of fuzzy set was originally introduced by L. A. Zadeh [6] as a generalization of the idea of an ordinary or crisp set. A fuzzy set can be seen as a predicate whose truth values are drawn from the unit interval, $I = [0, 1]$, rather than the set $\{0, 1\}$ as in the case of an ordinary set. Thus the fuzzy set has as its underlying logic a multivalued logic. The fuzzy set allows for the description of concepts in which the boundary between having a property and not having a property is not sharp. A binary valued characteristic function (membership function) $\mu_A(u)$ can be used to represent whether the object u ($u \in U$ where U

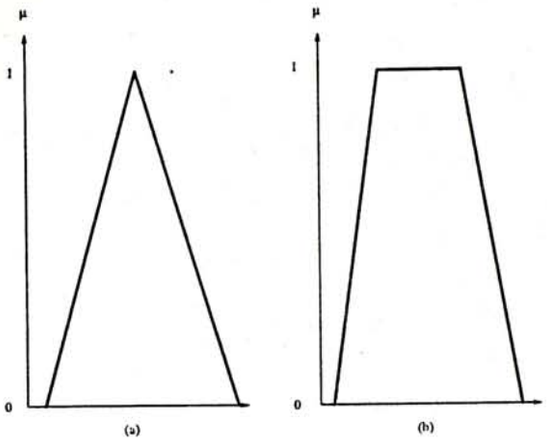

Figure 2.1: Fuzzy Sets: (a) Triangular (b) Trapezoidal

is the *universe of discourse*) belongs to the set A or not.

$$\mu_A : U \rightarrow 0, 1 \tag{2.1}$$

$$\mu_A(u) = \begin{cases} 1 & if\ u \in A \\ 0 & otherwise \end{cases}$$

Definition 2.1 Fuzzy set

A **fuzzy set** *in a universe of discourse U is characterized by the membership function μ_A, which takes values in the interval [0, 1] namely $\mu_A : U \rightarrow [0, 1]$. A fuzzy set A in U may be represented as a set of ordered pairs of a generic element u and its grade of membership μ_A as:*

$$A = \{(u, \mu_A(u)) | u \in U\} \tag{2.2}$$

i.e. the fuzzy variables u take on fuzzy values $\mu_A(u)$.

□

When the membership space contains only two points 0 and 1, A is a non-fuzzy (crisp) set, and $\mu_A(u)$ is identical to the characteristic function of a non-fuzzy set. Elements with zero degree of membership are not usually listed. When

A is a discrete, finite, fuzzy set it may be expressed as

$$A = \mu_A(u_1)/u_1 + \cdots + \mu_A(u_n)/u_n \tag{2.3}$$

$$= \sum_{i=1}^{n} \mu_A(u_i)/u_i \tag{2.4}$$

where "+" denotes the set theory union operator rather than arithmetic sum. The oblique line "/" does not denote division, instead it denotes a particular membership function to a value on the universe of discourse. If u is continuous then the fuzzy set may be written as

$$A = \int_u \mu_A(u)/u \tag{2.5}$$

Thus the grade of membership, $\mu_A(U)$ can be characterized by either a set of discrete values or a function. In the following, we will introduce some definitions used for describing the characteristics of a fuzzy set.

Definition 2.2 Normal and subnormal fuzzy set

A fuzzy subset A of U is called **normal** *if there exists at least one element* $u \in U$ *such that* $\mu_A(u) = 1$. *A fuzzy set that is not normal is called* **subnormal.**

□

Definition 2.3 Height of fuzzy set *The* **height** *of a fuzzy set A is the largest membership grade of any element in A. It is denoted* $Height(A)$. *Hence* $Height(A) = Max_u \mu_A(u)$. *A normal fuzzy set can thus be defined as one with height equal to one.*

□

Definition 2.4 Fuzzy support *Assume A is a fuzzy subset of U; the* **support** *of A, denoted* $Supp(A)$, *is the crisp subset of U whose elements all have nonzero*

membership grades in A.

$$Supp(A) = \{u | \mu_A(u) > 0 \ and \ u \in U\} \tag{2.6}$$

□

Definition 2.5 Fuzzy core *Assume A is a fuzzy subset of U; the* **core** *of A, denoted Core(A), is the crisp subset of U consisting of all elements with membership grade one.*

$$Core(A) = \{u | \mu_A(u) = 1 \ and \ u \in U\} \tag{2.7}$$

□

2.1.2 Operations on fuzzy sets

The following are the definitions of the most widely accepted operations on fuzzy sets.

Definition 2.6 Union *Assume A and B are two fuzzy subsets of U. Their* **union** *is a fuzzy subset C of U, denoted $C = A \cup B$, such that for each $u \in U$*

$$\mu_C(u) = Max[\mu_A(u), \mu_B(u)] = \mu_A(u) \vee \mu_B(u) \tag{2.8}$$

□

It is common practice in the fuzzy set literature to use $\vee$ as the Max operator.

Definition 2.7 Intersection *Assume A and B are two subsets of U. Their* **intersection** *is a fuzzy subset D of U, denoted $D = A \cap B$, such that for each $u \in U$*

$$\mu_D(u) = Min[\mu_A(u), \mu_B(u)] = \mu_A(u) \wedge \mu_B(u) \tag{2.9}$$

□

Definition 2.8 Relative complement *Assume A and B are two fuzzy subsets of U. The* **relative complement** *of B with respect to A, denoted $E = A - B$, is defined as the fuzzy subset E of U where for each $u \in U$*

$$\mu_E(u) = Max[0, \mu_A(u) - \mu_B(u)] \tag{2.10}$$

□

Definition 2.9 Complement or negation *Assume A is a fuzzy subset of U. The* **complement** *or* **negative** *of A, denoted $\bar{A}$, is defined as the fuzzy subset $\bar{A} = U - A$; hence for each $u \in A$*

$$\mu_{\bar{A}}(u) = 1 - \mu_A(u) \tag{2.11}$$

□

Thus the negation is the complement of A with respect to the whole space U.

2.2 Fuzzy Inference

The fuzzy inference process can be viewed as a complex functional relationship between relevant variables. In addition, the form used to represent this relationship involves the use of if-then fuzzy rules involving vague predicates. The collection of these rules is called the knowledge base. We shall use the term **reasoning** or **inference** to indicate the process in which we are given the values of the inputs, the antecedent variables, and use these in conjunction with our

knowledge base to obtain the value of the consequent. Formally, we represent the rule-base in the following format:

IF U_1 is B_{11} AND U_2 is B_{12} THEN V is D_1

ALSO

$\vdots$

ALSO

IF U_1 is B_{m1} AND U_2 is B_{m2} THEN V is D_m

U_1, U_2, and V are fuzzy variables, U_1 and U_2 are the input variables, and V is the output variable. B_{i1}, B_{i2}, and D_i are linguistic values (labels) represented as fuzzy subsets of the respective universes of discourse X_1, X_2, and Y. The membership functions of these linguistic values are denoted $B_{i1}(x_1)$, $B_{i2}(x_2)$, and $D_i(y)$. If the inputs to the system are the values $U_1 = x_1^*$ and $U_2 = x_2^*$, then we are faced with the problem of determining the appropriate value of the output variable V. This is the problem we called fuzzy *reasoning* or *inference* which consists of the following steps:

1. Find the firing level of each of the rules.

2. Find the output of each of the rules.

3. Aggregate the individual rule outputs to obtain the overall system output.

Let us consider these steps in turn. We first consider the determination of the firing level of the individual rules. The firing level of a rule is determined by the satisfaction of each of the antecedent components. The *level of matching* between the linguistic label B_{i1} and the input value x_1^* is determined as the membership grade of x_1^* in the fuzzy set representing B_{i1}, hence we get $B_{i1}(x_1^*)$

as the level of matching for the first antecedent. Similarly the level of matching of the second antecedent is $B_{i2}(x_2^*)$. We must now combine these two values to obtain the firing level of the rule.

Noting the conjunctive connecting AND between the inputs U_1 and U_2 in the antecedent part of each individual rule

$$\text{IF } U_1 \text{ is } B_{i1} \text{ AND } U_2 \text{ is } B_{i2} \text{ THEN } V \text{ is } D_i$$

we combine these two levels of matching using an AND aggregation. In particular, our initial choice is to use the Min ($\wedge$)

$$\tau_i = B_{i1}(x_1^*) \wedge B_{i2}(x_2^*) \tag{2.12}$$

where τ_i is called the *degree of firing* (DOF) of the i^{th} rule with respect to the input values $U_1 = x_1^*$ and $U_2 = x_2^*$.

The DOF τ_i takes values from the unit interval; it characterizes the truthfulness (relevance) of the antecedent part of the i^{th} rule, different values of τ_i that are related to different levels of relevance between the measured value x_1^* and x_2^*, and the conditions associated with the linguistic labels B_{i1} and B_{i2}.

If the input variables take fuzzy values, that is, $U_1 = A_1$ and $U_2 = A_2$, where A_1 and A_2 are fuzzy subsets of the universes X_1 and X_2, then the level of matching between input fuzzy value A_1 and the linguistic label B_{i1} is obtained from the conditional possibility $Poss(B_{i1}|A_1) = Max_{i1}[B_{i2}(x_2) \wedge A_2(x_2)]$

The DOF of the i^{th} rule in this case is:

$$\tau_i = Poss(B_{i1}|A_1) \wedge Poss(B_{i2}|A_2) \tag{2.13}$$

The process is illustrated in **Fig. 2.2.**

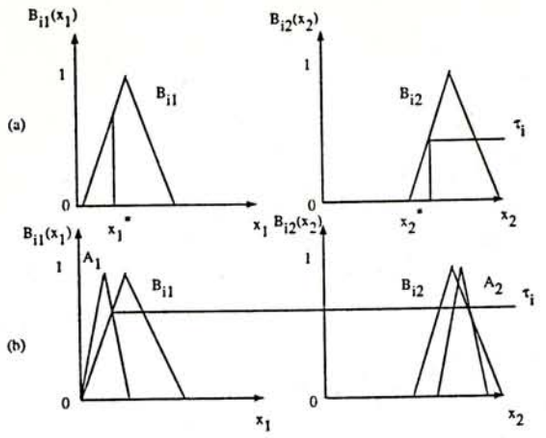

Figure 2.2: Calculation of DOF τ_i of the i^{th} rule: (a) The input variables of the FLC take crisp values $U_1 = x_1^*$ and $U_2 = x_2^*$; (b) The input variables of the FLC take fuzzy values $U_1 = A_1$ and $U_2 = A_2$.

The next step in the process is the determination of the individual rule output, which we shall denote as F_i. The degree of firing τ_i of a rule interacts with its consequent D_i to provide the output of the rule, F_i, a fuzzy subset over the output universe Y. The formulation used to determine how the τ_i and fuzzy set D_i interact to form the rule output is called a *fuzzy implication.* The most commonly used method for inferring the rule output is the so-called Mamdani method. In the Mamdani method, the output fuzzy set F_i is obtained by an AND of the DOF τ_i and the consequent fuzzy set D_i:

$$F_i(y) = \tau_i \wedge D_i(y) \tag{2.14}$$

The third step in the process is the aggregation of the individual rule outputs to obtain the overall system output, F, also a fuzzy subset over Y. The individual rule outputs are aggregated using a disjunctive connective ALSO. Thus the fuzzy output F inferred by the rule-base is

$$F(y) = \vee_i F_i(y) = \vee_i(\tau_i \wedge D_i(y)) \tag{2.15}$$

For use in the fuzzy control environment a fourth step must be added. We need a crisp single value to be the input to the controlled system. The output fuzzy set F inferred by the rule-base cannot be used directly as input to the controlled deterministic system. In order to obtain a crisp value from the output of the Fuzzy logic controller (FLC) we are faced with the problem of selecting one element y^* from the universe Y to represent the value to implement. This process of selecting one representative crisp element based upon the knowledge that the fuzzy value of the output variable, V is F, is called *defuzzification.*

Two often-used methods of defuzzification are the Center of Area (COA) method and the Mean of Maxima (MOM) method. The COA method defines the defuzzified value of a fuzzy set F as its fuzzy centroid:

$$y^* = \frac{\int_Y yF(y)dy}{\int_Y F(y)dy} \tag{2.16}$$

The calculation of the COA defuzzified value is simplified if we consider finite universe of discourse Y and thus a discrete membership function $F(y)$:

$$y^* = \frac{\sum_{j=1}^{n} F(y_j)y_j}{\sum_{j=1}^{n} F(y_j)} \tag{2.17}$$

The MOM method determines the defuzzified value, as a mean of all values of the universe of discourse, having maximal membership grades:

$$y^* = \frac{1}{q}\sum_{j \in J^*} y_j \tag{2.18}$$

where J^* is the set of elements of the universe Y which attain the maximum value of $F(y)$ and q is the cardinality of J^*.

Chapter 3

Weaknesses of fuzzy inference

In this chapter, we try to state out the weaknesses of fuzzy inference which initiates our study of the newly proposed Real Intelligent Mapping approach. The main drawbacks and doubtnesses of using fuzzy inference are discussed in the following subsections.

3.1 Is the use of linguistic fuzzy if-then rules and membership functions a good means of representing human expert knowledge?

The original method of constructing fuzzy models is based upon what we shall call the *direct* approach. In this direct approach the system is first described linguistically using terms from natural language and then translated into the formal structure of a fuzzy model with the aid of the representational power of the theory of approximate reasoning. The linguistic description is constructed

subjectively on the basis of the a priori knowledge about the system. Thus the source for deriving the linguistic rules is the expert's direct knowledge of the system. It is this knowledge that is expressed in the form of logical rules. This method can be seen as a qualitative version of the traditional model building used in system science [35]. This direct approach to fuzzy modeling, based solely upon the use of experts' description of the functioning of the system, has some inherent limitations.

Human controller is a very complicated learning machine. The knowledge of human controller is stored in huge amount of interconnected neurons. The data storage area (brain) is connected to sensor organs (sensors) and muscles (actuators) to form a complete feedback controller. When the human is viewed as an controller, we can imagine that in fact numerous input-output relations (control surfaces) are stored in it and each of them is responsible for a particular control task. For instance, when an human being is learning how to back-up a truck. At the first stage he is not yet an expert in performing that particular task. The data (or the control surface) in this brain at this instance is inadequate for him to back-up the truck well. During the learning process, he percept real input data (e.g. speed, position and direction of the truck) and use the premature data stored in his brain to give an output (i.e. steering angle). From the error (difference between the ideal state and the actual state of the truck), the human controller adaptively update the control surface in his brain by changing the weights stored in the neurons. After a number of trial-and-errors, the control surface is well established and the human becomes an expert in backing-up a truck. Put in other words, from the experience and using intuition, people build *mental models* in their minds. If a human is an expert in a particular task,

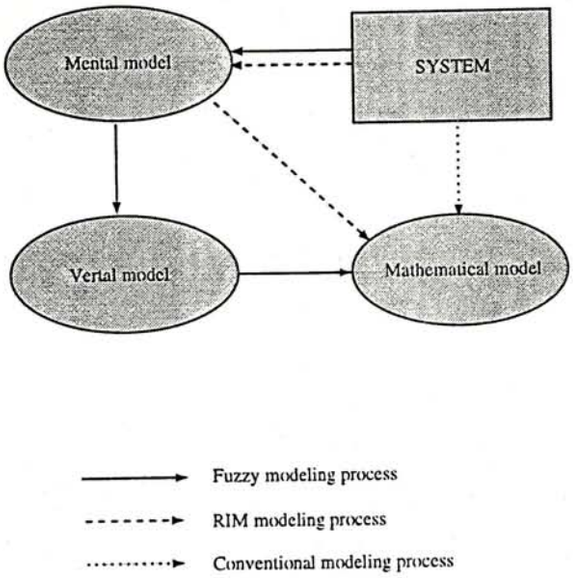

Figure 3.1: 3 different means of system modeling

the mental model formed in his brain for that task can be regarded as expert knowledge. That particular model or input-output relation is the primitive "optimal" function we are trying to recover in the expert knowledge extraction process. The process of fuzzy system modeling from human expert knowledge can be stated as follows:

1. provide a structure of the model. This process includes determining the number of input and output variables and partitioning the input output variables into fuzzy sets.

2. extract expert knowledge by means of linguistic fuzzy if-then rules and membership functions.

3. integrate the extracted knowledge and the model structure to form the final input-output relation.

We now pose a fundamental question: Is the use of linguistic fuzzy if-then rules and membership functions a good means of extracting the primitive human expert knowledge (mental models stored in his brain)? The answer to this question is negative. When the expert is asked to express his knowledge in terms of linguistic fuzzy if-then rules, he try to formulate a *verbal model* from the *mental model.* See **Fig. 3.1**. This process is very subjective as different experts (assume that they have the same mental model) will give out different sets of fuzzy if-then rules and membership functions. Besides, an expert in controlling a plant may not be an expert in translating his implicitly stored knowledge to linguistic if-then descriptions. As a result, the linguistic fuzzy if-then rules and membership functions given out are only very coarse approximation of the primitive knowledge. Furthermore, the ability of the human expert to give out precise linguistic if-then descriptions declines as the number of set levels and the number of variables increase. It may be quite easy for an expert to give out a linguistic if-then rule such as "**If A is LARGE then B is ...**". The expert may also give out the rule "**If A is QUITE LARGE and B is QUITE SMALL then C is ...**" without much difficulty. However, it may not be so easy for the expert to decide the consequent of the following rule:

"**If A is A LITTLE BIT MORE THAN QUITE LARGE and B is QUITE SMALL and C is A LITTLE BIT LESS THAN SMALL then D is ...**"

The situation becomes even worser when the number of set levels and antecedent variables increases. Because of this, the accuracy of fuzzy modeling may not improve with increasing number of linguistic fuzzy if-then rules given. To conclude, the use of linguistic fuzzy if-then rules and membership functions is not a good

means of extracting the primitive human expert knowledge. Then, does there exist a better method? The answer is positive. We argue that it is better to obtain sampled input-output real data during real time control by the expert. The reason is that the real data points we obtained is actually located on the "optimal" mental model (ignore the noise in the sampling process). This expert knowledge extraction method can be regarded as a process of "sampling", not a process of "approximation" as in conventional fuzzy case where human experts give out coarse linguistic fuzzy if-then descriptions. Unlike the fuzzy case, the accuracy of modeling in this case can certainly be improved by taking more samples and the modeling problem reduces to an ordinary non-fuzzy approximation inference. It can be based on the well developed approximation theory.

3.2 Role of conventional fuzzy inference doubtful if the expert knowledge is in the form of sampled input-output data

The second direction in the development of fuzzy models, inspired by classic systems theory and recent developments in neural networks, is based on the use of input-output data. In the language of systems theory, this approach can be regarded as process of a system identification.

The identification of a fuzzy system consists of two major phases. The first phase is the identification of the structure of the fuzzy model (structure identification) and the second is the estimation of the parameter values of the fuzzy

model (parameter identification). Broadly speaking, structure identification includes the determination of the input and output variables, the relationships between the variables (the structure of the rules), the number of rules in the rule base, and the partitioning of the input and output variables into fuzzy sets. In general, structure identification is a difficult and extremely ill-defined process, more an art than a science, and not readily amenable to automated techniques.

Wang and Mendel [9][36] propose a method to generate fuzzy if-then rule from numerical real data. Their method first divide the input and output spaces into fuzzy regions. This involves the design of membership functions for each fuzzy variable. Then the consequent of each fuzzy rule is determined by least-squares from the data points. We observe that in their method, the membership functions are defined by the system designer, not by the expert who gives out primitive information. In this case, the membership functions can at the most be considered as a means to provide an extra degree of freedom to adjust the system, not as a piece of raw information. Also, when dealing with this type of information by fuzzy approximation, as the structure of the resulting fuzzy model is limited by the limited alternatives of membership function shapes and inference mechanisms, the resulting system is definitely nonlinear and hard to be analyzed by conventional methods. As a result, it is crystal clear that in this situation, it is more direct to do the approximation in the real domain, instead of assigning membership functions in an ad hoc manner.

3.3 Computational requirements

We have to distinguish between two types of computation time when addressing this issue: (1) The computation time required when the system model is developed; (2) The computation time required when the controller is actually used in real time. It is clear that the first type of computation time is actually of little importance in developing a system since all the time is used in the system design stage. It will not affect the actual system performance. On the other hand, the computation time when the system is actually used in real time is very important in accessing system performance. In fuzzy inference systems design, the computation time used in the system modeling stage is negligible since the system model is composed of (1) expert fuzzy if-then rules; (2) membership functions; (3) an inference mechanism. They can be considered as separate information and essentially zero computation time is required to derive them since they are given by human experts. However, there is considerable time required when the system model is actually used in real time inference. Every input has to pass through the following three stages: (1) Fuzzify the real input data via their respective pre-determined membership functions; (2) Perform inference in the fuzzy domain following a set of if-then rules; (3) Return a real inference result by defuzzifying the fuzzy consequent. On the contrary, if the system model is developed from real data in real domain, the situation is completely different. First, there may be considerable time required in developing the system model and representing it by function(s) especially when the system dimension is high. However, this time is used in the system modeling process and is not related to the actual system performance. When the system is used in real time, the input

has to pass through only one stage: Substitute the input to the appropriate real function(s) and get the respective output. It is crystal clear that this type of inference processes use less time than the fuzzy one which consists of three stages (Actually, we have compared our RIM method with the fuzzy method and proved our assertions).

3.4 Low transparency

In fuzzy inference, because of the fact that the expert knowledge extraction process by means of linguistic descriptions is imprecise and the fact that design of fuzzy controllers is usually performed in an ad hoc manner, the preliminary controller constructed usually requires fine adjustment. This process involves the tuning of rule consequences or the membership functions. However, it is hard to justify the effect of parameters' change on the final control surface because of the very nonlinear fuzzy inference mechanism. As a result, the tuning process is a very random and trial-and-error process. The large degree of tuning freedom provided by fuzzy inference (e.g. different width and shape of membership functions, number of set levels, consequences of rules) makes this process very difficult if the dimension of the problem is high. Besides, the fuzzy controller constructed for the nominal plant may later perform inadequately if significant and unpreductable plant parameter variation occur. One may argue that we can apply adaptive fuzzy inference to solve the problem. However, as mentioned before, because of the very nonlinear fuzzy inference mechanism acting as a barrier between the system parameters and the final control surface, it is hard to define an optimal adaptive law.

3.5 Analytical difficulties

Given a control system, the first and foremost question about its various properties is whether it is stable, because an unstable control system is typically useless and potentially dangerous. Stability analysis of fuzzy system has been studied for a period of time. Braae and Rutherford [27][28] proposed a linguistic phase plane trajectory to analyze and improve the stability of fuzzy systems by exchanging the control rules. Kickert and Mamdani [26] use the describing function method to evaluate the stability of fuzzy control systems. B. Kiszka [29] introduced the energetistic stability of fuzzy dynamic systems and developed an entropy for fuzzy system. Besides, De Glas [24] and A. Kania [23]. use the concept of α stability for analyzing fuzzy systems, in which the distinction between stability and instability is removed and a real value between 0 and 1 is used to describe the "degree of stability" of a fuzzy system. However, all the above stated methods are in fact quite restrictive and have only limited applicability. The principal reason is that the design of fuzzy controllers has relied on ad hoc techniques. That is, the controllers were not synthesized using an underlying theory but were arrived at by trial and error. Also, by allowing the state of the system to be described by a fuzzy set, notions of unboundedness become ambiguous. The other reason is that fuzzy system is itself definitely nonlinear. The limited variety of membership functions and inference mechanisms imposes structural restrictions on the final model. We actually show that the most linear type of fuzzy controller (i.e. using triangular memberships functions and algebraic product as the logical AND operation) has an equivalent piecewise-polynomial representation in real domain. This limit the applicability of linear

system theory to their analysis and design. All the above hinder the use of conventional control analysis techniques (e.g. Liapunov's method) for designing and analyzing fuzzy systems.

Chapter 4

Real Intelligent Mapping

Due to the reasons stated in the previous chapter, we argue that it is better to obtain sampled input-output real data points during real time control by the human expert instead of asking the expert to give out linguistic fuzzy if-then rules and membership functions. The inference mechanism is now represented by one or more functions directly relates the real input and output; the usual fuzzification and defuzzification steps are eliminated. The new formulation is named Real Intelligent Mapping (RIM)[1] as it retains the spirit of translating the human expert knowledge into computer algorithms. Obviously, the problem is now an ordinary non-fuzzy approximation inference and it can be based on the well developed approximation theory. Compared with the fuzzy inference the new system is more transparent and is more easy to implement. We now states the idea of Real Intelligent Mapping precisely, we write:

$$\mathbf{x} = [\mathrm{x}_1 \mathrm{x}_2 \cdots \mathrm{x}_m]^T, \tag{4.1}$$

a real vector of $R^m \times R^n$ to represent the set of m sampled input, with each input data point in n-dimensional:

$$\mathbf{x}_i = [x_{i1} x_{i2} \cdots x_{in}]. \tag{4.2}$$

and

$$\mathbf{y} = [y_1 y_2 \cdots y_m]^T, \tag{4.3}$$

a real vector of R^m to represent the set of m sampled output, with each output corresponds to one of the m input.

Formulated in this way, the set of m input-output sampled data points for system modeling can be represent by the vector $[\mathbf{x}\ \mathbf{y}]$.

From the sampled input-output data points, we are going to identify one or more functions $f_1, f_2, \cdots, f_j$ that "best" represents the mapping between $\mathbf{x}$ and $\mathbf{y}$, where each f is a mapping from R^n to R. Afterward, when we are given a input $\mathbf{x}' = [x'_1 x'_2 \cdots x'_n]$, the output y' can be inferred as:

$$y' = g(\mathbf{x}') \tag{4.4}$$

where g is a mapping from R^j to R and it takes the role of "combining" the output of the j functions $f_1, f_2, \cdots, f_j$. When $j = 1$, we have $g = f$ and the system is represented by one real equation. When $j > 1$, we have jointed system. We see that a RIM system is completely characterized by the functions f_i and g, and the task of design is to determine these two functions by whatever means.

We now restate our problem: Given m sampled input-output data points, the task is to identify one or more functions $f_1, f_2, \cdots, f_j$ that "best" represents the

mapping between the input and output together with g to combine the output of fs giving the final output. There exists many alternatives to this problem. The following are two examples: (1) As we mentioned before, the mathematical representation of the "optimal" function representing the expert knowledge is very complicated and is "unknown" to us. So we try to take sampled data points and from the characteristics that the unknown "optimal" function should possess, find a "simpler" function that is "close" in some sense to it; or (2) We find a function which interpolate all the sampled data points. There exist many methods from approximation and interpolation theory to solve the problem. For the first example, regression is a well-known technique. For the second example, the data points can be interpolated by polynomials or spine functions. Whereas the choice is purely the designer's favor, a common question is how we can quantify the goodness of the inference provided by the designed function.

Many applications require "on-line" identification instead of "off-line." An identification method is said to be of the "off-line" type when one collects a large amount of input and output data for the system which may be stored in a computer or recorded in some manner. These date are then processed in a batch to estimate the parameters of the model and obtain the best fit according to a prescribed cost function. In off-line identification, there is a greater flexibility in selecting computational methods without any restriction on computing time. As a result, the accuracy of the estimates can be made fairly high. However, we can also model the system as long as the sampled data points are available. An identification is said to be "on-line" type if it satisfies the following conditions:

1. all the data need not be stored

2. a recursive algorithm is used for adjusting the estimates of the parameter after each sampling instant

3. the amount of computation required for "model adjustment" is a fraction of the sampling period.

A large variety of methods have been applied to system modeling, both on-line and off-line. The methods can be classified in many ways; one scheme for classification is given below.

1. Classical Methods: (mostly off-line)

   (a) Frequency Response Identification

   (b) Impulse response identification by deconvolution

   (c) Step response identification

   (d) Identification from correlation functions

2. Equation-error Approach: (batch-processing)

   (a) Least-squares

   (b) Generalized Least-squares

   (c) Maximum likelihood

   (d) Minimum variance

   (e) Gradient Methods

3. Model Adjustment Techniques:

   (a) Recursive Least-squares

(b) Recursive generalized least-squares

(c) Instrumental variables

(d) Bootstrap

(e) Recursive maximum likelihood

(f) Recursive correlation

(g) Stochastic approximation

Now we ask a question: What characteristics should a "good" approximation possess? The following are two common requirements:

1. The data points should be approximated in a way that the approximation between the data points should not be too oscillatory. There is a common problem when we try to interpolate a large number of data points by a polynomial function. In that case the high order requirement of the polynomial will makes the approximation between the data points very oscillatory.

2. The structure of the resulting system should not be too complicated or nonlinear which may hinder further analysis of the system. In other words, we try to find a simple method to approximate them.

From the above, we see that the choice of using which particular approximation method is usually problem dependent. In other words, a method that works well for one problem may not be so efficient for another problem. To choose a suitable method to solve a particular approximation problem, one first need to get some knowledge of the *quantity*, the *dimension*, and the *geometric*

distribution of the data to be approximated. This may be done by some statistical methods. However, this is not always possible especially when there are numerous data points to be approximated or when the dimension of the problem is high.

In the next chapter, we introduce the use of Dirichlet tessellation for the implementation of RIM systems, which is general enough to attack approximation problems regardless of data distribution. Roughly speaking, the method partitions the state-space data points into simplexes to form the input-output relation and then represents each simplex by a linear equation. The resulting approximation is piecewise-linear and it is in fact a kind of linear spine interpolation approximation inference.

Chapter 5

Design of Real Intelligent Mapping Systems Using Dirichlet Tessellation

RIM has already been used to solve the well-known truck backing-up problem by using linear regression [1]. However, linear regression is only one of the many approximation methods that can be used to design a RIM system. To make the RIM systems design more problem independent, we introduce the use of Dirichlet tessellation for the implementation of RIM systems, which is general enough to attack approximation problems regardless of data distribution (Section 5.1). Roughly speaking, the method partitions the state-space data points into simplexes to form the input-output relations and then represents each simplex by a linear equation. The resultant approximation is piecewise-linear which makes qualitative analysis of the RIM system possible. Also, real time computation time is saved when compared to the conventional fuzzy inference

since the input has to pass through only one stage to get the desired output, namely, substitute the input value to the appropriate *real* function. We propose a method for their identification which is optimal in the sense of *Least-Squares* (Section 5.2). The method can handle different types of expert knowledge and it has been used successfully to solve some well-known problems such as balancing an inverted pendulum, inverted pendulum with cart, truck backing-up, and chaotic time series prediction (Section 5.3). Also, an interactive CAD platform has been developed to enhance the RIM design based on our proposed method (Section 5.4).

5.1 Dirichlet tessellation for function approximation

Dirichlet tessellation [4][5] was proposed by Rogers in 1964 and has been used extensively in finite element analysis. Before showing how it is used for function approximation, we first define a function approximation problem as follows. Given: a set of data points represented by an ordered set of vectors in k-dimensional space. The task is to identify one or more functions that best approximate all the data points in a suitable sense.

The idea of using *Dirichlet tessellation* for function approximation is as follows. Suppose n data points in a k-dimensional space are to be approximated. We first project the n data points onto the input space to give n $(k-1)$-dimensional data points. Then we try to triangulate the n projected data points to form packed simplexes without any overlapping where each simplex is made up of k data points (See **Fig. 5.1** for $k = 3$ and $n = 20$ where the simplexes in

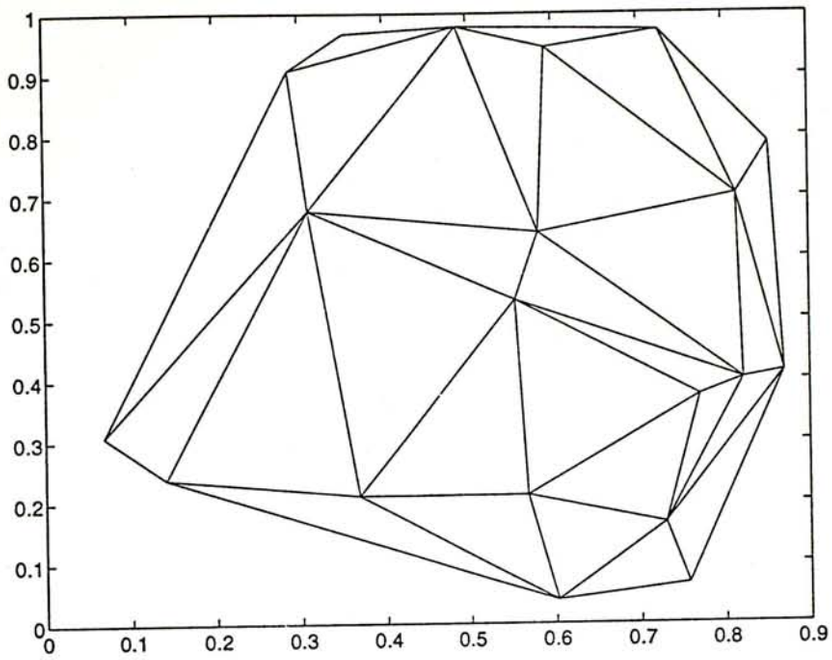

Figure 5.1: Triangulation of 20 random data points in the 2-D input space

this case are triangles and the points are generated at random).

For the case $k = 3$, the 3 data points in the 2-dimensional input space constituting each triangle in fact produce a plane in the original 3-D space before projection. We then represent each such plane by a linear function

$$y = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 \tag{5.1}$$

where y is the output, x's are the inputs and α's are the parameters of the function which can be easily determined from the three true data points making up the function. Generally, for the k-dimensional case, the equation becomes:

$$y = \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k. \tag{5.2}$$

As a result, the set of data points is approximated by a set of linear functions and the resulting system is piecewise-linear and continuous at the boundaries. **Fig. 5.2** shows the resulting system for the $k = 3$ and $n = 20$ case. When the

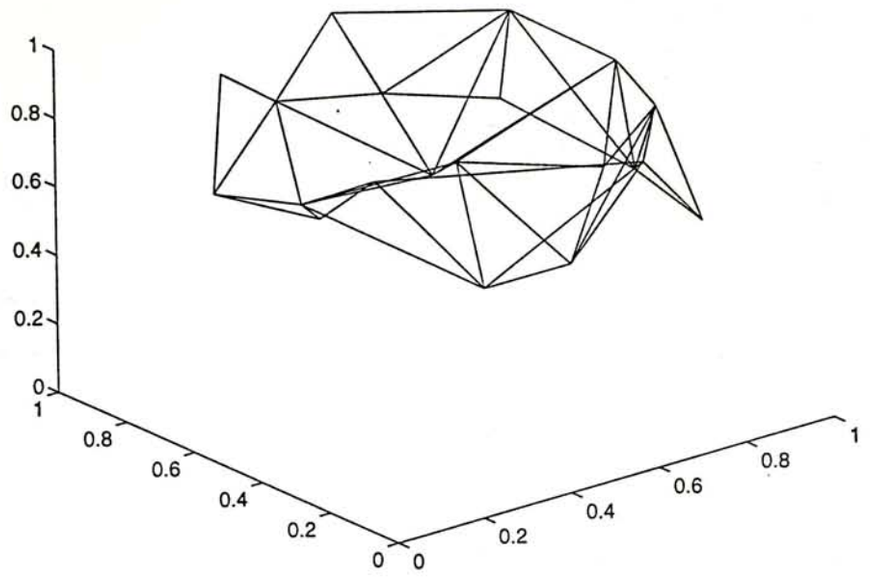

Figure 5.2: Control surface representing input-output relation for the 20 random data points

approximation method stated above is used for control system modeling, the finalized input-output relations will be made up of linear functions. The inference process becomes a simple substitution of input values into the appropriate function to obtain the desired response.

The rest of this section summarize the properties and the algorithm of tessellation

First, let us consider the 2-D case, the higher-dimensional case will be discussed later. Suppose the positions of n distinct points in the plane are given as data. We put into correspondence to each data point x a territory, i.e., the part of the plane, in which the points are closer to x than to any other data point. The resultant territories form a pattern of packed convex polygons and is called Dirichlet tessellation of the data points. **Fig. 5.3** shows the Dirichlet tessellation in bold lines. Each segment of the territory boundaries is in fact the

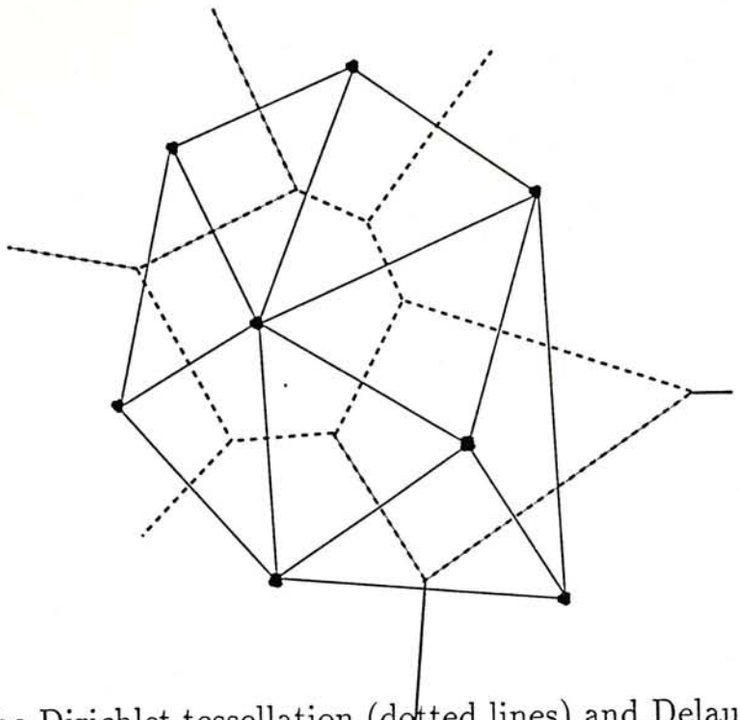

Figure 5.3: The Dirichlet tessellation (dotted lines) and Delaunay triangulation (solid lines) for a small-scale configuration

perpendicular bisector of the line joining two data points. If all pairs of data points sharing the same territory boundary segment are joined by straight lines, a pattern of packed triangles will be formed. The process is called Delaunay triangulation. **Fig. 5.3** shows the Delaunay triangles in faint lines.

In two dimensions three territorial boundaries meet at a vertex. Each vertex is located at the circumcenter of a Delaunay triangle formed by three data points. As a result, a vertex must be equidistant from all three of its forming points. In the tessellated pattern, each Delaunay triangle will have associated with it a unique vertex and vice versa. The vertexes are useful in constructing the data structure for storing the tessellated pattern.

In a k-dimensional Euclidean space the Delaunay triangles become simplexes and each simplex is made up with $k + 1$ data points. Each vertex in the tessellation is where the $k + 1$ territories meet and is the center of the hypersphere

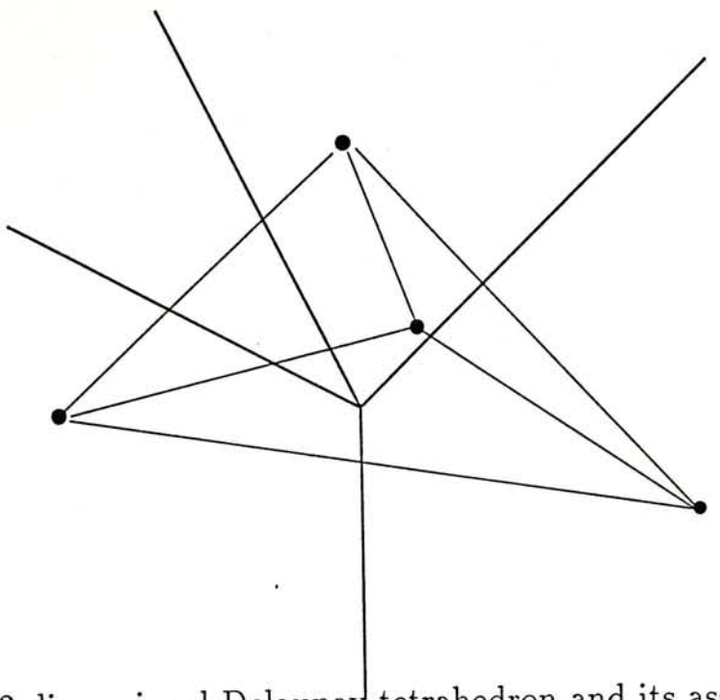

Figure 5.4: 3-dimensional Delaunay tetrahedron and its associated vertex

passing through all the data points constituting the associated simplex. Each contiguous pair of points is joined by a line that is an edge of some Delaunay simplexes. The territorial boundary shared by the contiguous point pair is a convex polygon lying in the $(k-1)$-dimensional hyperplane that bisects that edge. In three dimensions the territory of each data point is a convex polyhedron. **Fig. 5.4** shows a three-dimensional vertex and its associated Delaunay tetrahedron.

It is important to store the vertex structure of the tessellation in the form of a computer data structure before Dirichlet tessellation can be used to solve problems. Green and Sibson [5] store the Delaunay triangulation in the form of lists of contiguous points for each vertex. The following explains how they store the Delaunay triangulation in the two-dimensional case. The k-dimensional case is in fact a simple generalization of the two-dimensional case with extended lists. Suppose it were desired to store the vertex structure of the tessellation in **Fig.**

5.5. The eight data points in the plane give rise to seven vertices, $V_1, \ldots, V_7$. The territorial boundaries that extend to infinity can be considered as terminating in a vertex labeled ϕ. Each vertex is the circumcenter of three data points and a list of those points could be recorded for each vertex. In addition each vertex points to three other vertices, each one opposite one of the vertex's forming points. It is thus possible to record the structure by constructing two lists, each of length three. For each vertex in the structure, one list holds the forming points of the vertex (Delaunay triangles) and the coordinates of the points in the space are stored, the other holds the opposite neighboring vertices (**Table 5.1**) and they are stored as in terms of vertex numbers (their exact physical locations in the space need not be stored). In k dimensions each vertex will have $k+1$ forming points and $k+1$ associated neighboring vertices. With this data structure, we get all information concerning the tessellated pattern.

Vertex	*forming points*			*neighboring vertices*		
	1	2	3	1	2	3
V_1	P_6	P_4	P_5	V_4	ϕ	V_6
V_2	P_1	P_4	P_3	V_3	ϕ	V_7
V_3	P_2	P_3	P_4	V_2	V_4	V_5
V_4	P_2	P_5	P_4	V_1	V_3	ϕ
V_5	P_7	P_3	P_2	V_3	ϕ	ϕ
V_6	P_6	P_8	P_4	V_7	V_1	ϕ
V_7	P_1	P_8	P_4	V_6	V_2	ϕ

Table 5.1

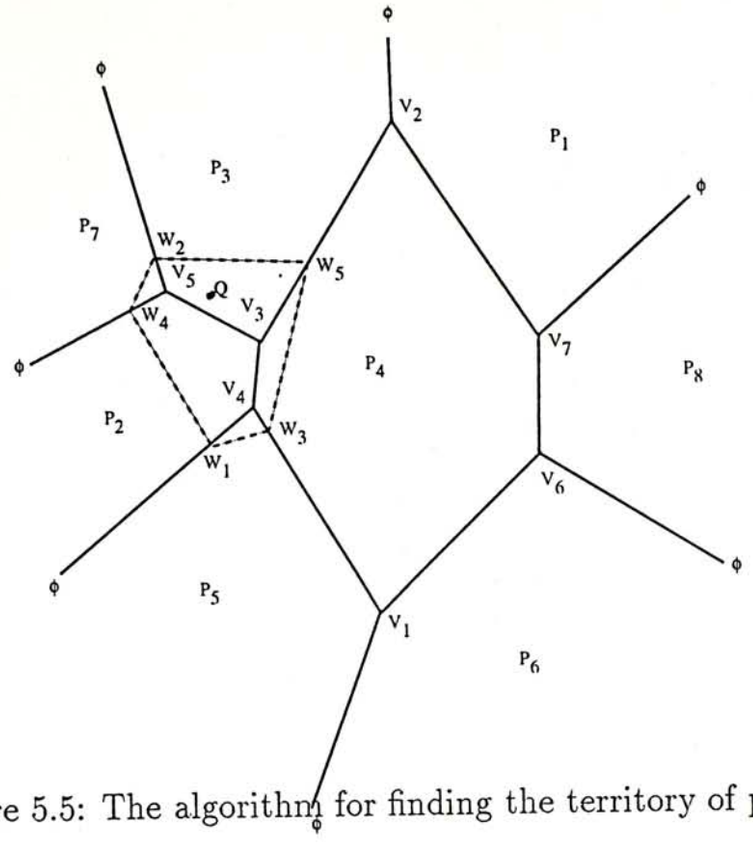

Figure 5.5: The algorithm for finding the territory of point Q

If it is possible to record the structure in the manner outlined above, then any number of points can be tessellated and triangulated by starting with a simple structure and building upon it. The starting pattern is the Delaunay simplex formed by the first $k+1$ points. This will give a tessellation containing one real vertex all of whose neighboring vertices will be ϕ.

Suppose we wish to insert a new point (Q in **Fig. 5.5**) within the current convex hull of the data points. The territory we wish to find is indicated by the dotted lines. The following algorithm is obtained from A. Bowyer [4] which shows clearly the tessellation procedure:

1. Identify a vertex currently in the structure that will be deleted by the new point (say V_4). Such a vertex is any that is nearer to the new point than to its forming points. There will always be at least one such vertex, as

the vertex corresponding to the Delaunay simplex in which the new point lies will always be deleted and the Delaunay simplexes completely fill the convex hull of the currently included points.

2. Perform a tree search through the vertex structure starting at the deleted vertex looking for others that will be deleted. This is an easy matter if the data are stored as indicated in **Table 5.1**. The result will be a list of all the vertices deleted by the new point Q. In this case the list will be: V_4, V_3, V_5.

3. The points contiguous to Q are all the points forming the deleted vertices: P_2, P_5, P_4, P_3, P_7.

4. An old contiguity between a pair of those points will be removed ($P_2 - P_4$ say) if all its vertices V_4, V_3 are in the list of deleted vertices.

5. In this case the new point has five new vertices associated with it: W_1, W_2, W_3, W_4, W_5. Compute their forming points and neighboring vertices. The forming points for each will be the point Q and k of the points contiguous to Q. Each line in the tessellation has k points around it (the line $V_3 - V_2$, for example, is formed by P_3 and P_4). The forming points of the new vertices and their neighboring vertices may be found by considering vertices pointed to by members of the deleted vertex list that are not themselves deleted, and finding the rings of points around them. Thus W_5 points outwards to V_2 from Q and is formed by P_3, P_4, Q.

6. The final step is to copy some of the new vertices, overwriting the entries of those deleted to save space.

5.2 Identification of a DT based RIM system by least-squares

When the information obtained for the design of a DT based RIM system consists of sampled input output (state-control) pairs which are recorded from successful control of the system, the quantity of sampled data points may be very large. Direct tessellation of the data points will result in numerous resulting linear functions. The following introduce a means to limit the number of generated functions. The method ensures that the resulting approximation is optimal in the sense of Least-Squares and the approach can be regarded as process of system identification. The identification of a DT based RIM systems model consists of two major phases. The first phase is the identification of the structure of the DT based RIM model (structure identification) and the second is the estimation of the parameter values of the model (parameter identification). Broadly speaking, structure identification includes determination of the input and output variables, the relationships between the variables , the number of linear functions, and the partitioning of the input and output variables into regions. Afterward, the parameters of the linear functions building up the system are found by least-squares.

As mentioned before, a Dirichlet tessellation based RIM controller is made up of piecewise-linear equations without any overlappings. Each of the linear equation can be represented as:

$$y = \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k \tag{5.3}$$

where k determines the dimension of the input vector. To illustrate the idea of

using least-squares approximation on DT based RIM controller, we first assume that $k = 2$. The resulting system is made up of piecewisely connectly straight lines and each of them can be represented by a linear equation:

$$y = ax + b \tag{5.4}$$

Assume that n data points are given as input, direct tessellation of the data points will result in a system consisting of $n - 1$ linear equations. This is not very practical when n is large. Our task is first to partition the input space into m regions (structure identification) and then to approximate the n data points by another $m + 1$ data points locating at the region boundaries by tessellating the $m + 1$ data points. In other words, The location of the $m + 1$ data points in the input domain should be given beforehand. Afterward, the task will be to find the output values of the $m+1$ data points such that the resulting tessellated system is globally optimal in the sense of least squares. The overall process is illustrated by the following example:

Example 5.1 Assume that n data points are given and we choose $m = 2$. The resulting system will be composed of 2 interconnecting linear equations (see **Fig. 5.6**):

$$\hat{y}_1 = ax + b \tag{5.5}$$

and

$$\hat{y}_2 = cx + d \tag{5.6}$$

The task is to find the parameters a, b, c and d such that the resulting systems is optimal in least-squares sense. Assume that the co-ordinates of the $m + 1 = 3$ data points are (x_1, y_1), (x_2, y_2) and (x_3, y_3). The two resulting linear

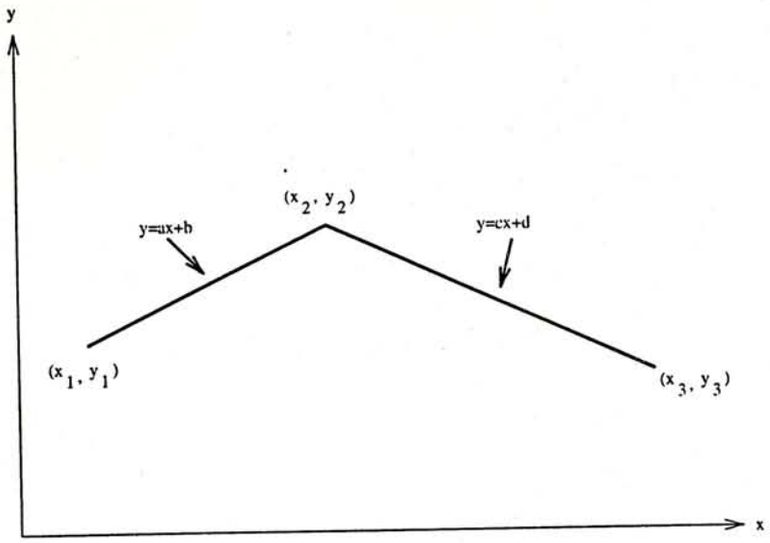

Figure 5.6: Two linear equations interconnected to form the system optimal in least-squares

equations can be written as:

$$\hat{y}_1 - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \tag{5.7}$$

and

$$\hat{y}_2 - y_2 = \frac{y_3 - y_2}{x_3 - x_2}(x - x_2) \tag{5.8}$$

rearranging, we have

$$\hat{y}_1 = (1 - \frac{x - x_1}{x_2 - x_1})y_1 + \frac{x - x_1}{x_2 - x_1}y_2 \tag{5.9}$$

$$\hat{y}_2 = (1 - \frac{x - x_2}{x_3 - x_2})y_2 + \frac{x - x_2}{x_3 - x_2}y_3 \tag{5.10}$$

For identifying the ys in globally least-squares sense, we have to combine the two subsystems into one equation.

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} 1 - \frac{x-x_1}{x_2-x_1} & \frac{x-x_1}{x_2-x_1} & 0 \\ 0 & 1 - \frac{x-x_2}{x_3-x_2} & \frac{x-x_2}{x_3-x_2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

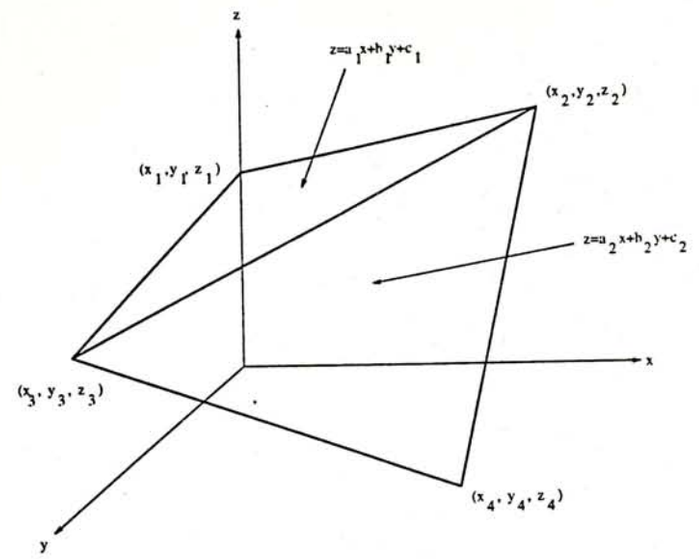

Figure 5.7: Two linear equations interconnected to form the system optimal in least-squares

In this representation, we can identify ys by least-squares. For the truck backing up example given later, the input is of order two and the final input-output relations is composed of piecewisely jointed linear equations. The following shows the least-square identification of the vertex values.

Assume that there is two piecewisely jointed triangles in the 3-D space as in **Fig. 5.7**. The task is to identify the vertex values z_1, z_2, z_3 and z_4 from the sampled input-output data points such that the overall system is "globally" optimal in the sense of least-squares.

From the figure. We have for subcontroller 1:

$$\hat{z}_1 = a_1 x + b_1 y + c_1 \tag{5.11}$$

and for subcontroller 2:

$$\hat{z}_2 = a_2 x + b_2 y + c_2 \tag{5.12}$$

Since the two equations should interpolate their vertex points, we have

$$z_1 = a_1 x_1 + b_1 y_1 + c_1 \tag{5.13}$$

$$z_2 = a_1 x_2 + b_1 y_2 + c_1 \tag{5.14}$$

$$z_3 = a_1 x_3 + b_1 y_3 + c_1 \tag{5.15}$$

$$z_2 = a_2 x_2 + b_2 y_2 + c_2 \tag{5.16}$$

$$z_3 = a_2 x_3 + b_2 y_3 + c_2 \tag{5.17}$$

$$z_4 = a_2 x_4 + b_2 y_4 + c_2 \tag{5.18}$$

substitute (5.16) into (5.15) and (5.17) into (5.15) we have

$$(z_2 - z_1) = a_1(x_2 - x_1) + b_1(y_2 - y_1) \tag{5.19}$$

$$(z_3 - z_1) = a_1(x_3 - x_1) + b_1(y_3 - y_1) \tag{5.20}$$

and

$$a_1 = \frac{(z_2 - z_1) - b_1(y_2 - y_1)}{x_2 - x_1} \tag{5.21}$$

$$b_1 = \frac{(z_3 - z_1) - a_1(x_3 - x_1)}{y_3 - y_1} \tag{5.22}$$

eliminate b_1 from (5.23) and a_1 from (5.24), we obtain

$$b_1 = \frac{(z_3 - z_1)(x_2 - x_1) - (z_2 - z_1)(x_3 - x_1)}{(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)} \tag{5.23}$$

$$a_1 = \frac{(z_2 - z_1)(y_3 - y_1) - (y_2 - y_1)(z_3 - z_1)}{(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)} \tag{5.24}$$

from (5.13), we have

$$c_1 = z_1 - a_1 x_1 - b_1 y_1 \tag{5.25}$$

if we rewrite (5.13) and (5.14) in the form

$$\hat{z}_1 = \alpha_1 z_1 + \beta_1 z_2 + \gamma_1 z_3 \tag{5.26}$$

$$\hat{z}_2 = \alpha_2 z_2 + \beta_2 z_3 + \gamma_2 z_4 \tag{5.27}$$

we have

$$\alpha_1 = \frac{x(y_2 - y_3) + y(x_3 - x_2)}{(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)} \tag{5.28}$$

$$\beta_1 = \frac{x(y_3 - y_1) + y(x_1 - x_3) - x_1 y_3 + y_1 x_3}{(x_2 - x_1)(y_2 - y_1) - (y_2 - y_1)(x_3 - x_1)} \tag{5.29}$$

$$\gamma_1 = \frac{x(y_1 - y_2) + y(x_2 - x_1) + x_1 y_2 - y_1 x_2}{(x_2 - x_1)(y_2 - y_1) - (y_2 - y_1)(x_3 - x_1)} \tag{5.30}$$

$$\tag{5.31}$$

similarly for subcontroller 2, we have

$$\alpha_2 = \frac{x(y_3 - y_4) + y(x_4 - x_3)}{(x_3 - x_2)(y_4 - y_2) - (y_3 - y_2)(x_4 - x_2)} \tag{5.32}$$

$$\beta_2 = \frac{x(y_4 - y_2) + y(x_2 - x_4) - x_2 y_4 + y_2 x_4}{(x_3 - x_2)(y_3 - y_2) - (y_3 - y_2)(x_4 - x_2)} \tag{5.33}$$

$$\gamma_2 = \frac{x(y_2 - y_3) + y(x_3 - x_2) + x_2 y_3 - y_2 x_3}{(x_3 - x_2)(y_3 - y_2) - (y_3 - y_2)(x_4 - x_2)} \tag{5.34}$$

$$\tag{5.35}$$

combine the two functions into one such that we can perform global identification of the parameters by least-squares

$$\begin{bmatrix} \hat{z}_1 \\ \hat{z}_2 \end{bmatrix} = \begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 & 0 \\ 0 & \alpha_2 & \beta_2 & \gamma_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}$$

For a system which is composed of more than two subsystems, linear programming is used to eliminates the redundant equations. For instance, in the design of the truck backing-up systems described later, a computer algorithm is developed to identify the input-output relation optimal in the sense of least-squares.

5.3 Examples

5.3.1 Defining the problem

Suppose the task is to design a complex control system which is so complicated that no mathematical model exists for it, or, that the mathematical model is strongly non-linear or complicated which makes its identification very difficult if not impossible. Usually, two types of information can be obtained in conventional fuzzy controller design problem: (1) the experience of human experts in controlling the system; and, (2) sampled input-output (state-control) pairs which are recorded from successful control of the system by human experts. The experience of human experts is usually expressed in terms of linguistic fuzzy if-then rules which state in what situation which action should be taken. The sampled input-output pairs are numerical data which give specific output values corresponding to each input situation.

Dirichlet tessellation can be used to design systems given either one kind of the aboved stated information. As we mentioned before, the use of linguistic if-then fuzzy rules is not a good means of extracting expert knowledge. However, we would like to show that Dirichlet tessellation can be used to solve the problem even if linguistic if-then rules are given. In that case, each linguistic rule is considered as a *real* data point in the space. In this section, four well known design problems are given as examples. The "Balancing an inverted pendulum" and "Inverted pendulum with cart" problems are solved by using information obtained form expert knowledge in the form of linguistic if-then rules while the "Truck backing-up" problem and the "Chaotic time series prediction" problem

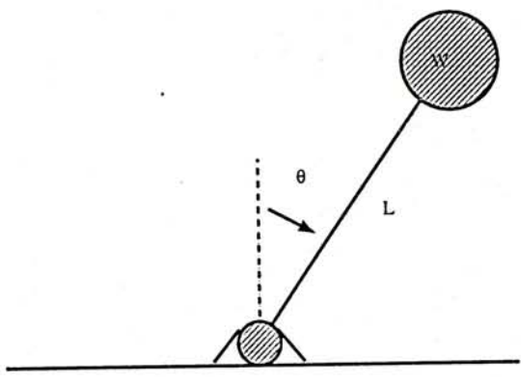

Figure 5.8: An inverted pendulum

are solved by using information obtained from sampled input-output (state-control) pairs.

5.3.2 Balancing an inverted pendulum

Balancing an inverted pendulum is a well-known control problem. The dynamics or the state-space model of an inverted pendulum can be stated as:

$$\theta(k+1) = \theta(k) + \Delta t \cdot \omega(k), \tag{5.36}$$

$$\omega(k+1) = \omega(k) + \frac{\Delta t}{J}(WL\sin(\theta(k)) + T(k)), \tag{5.37}$$

where θ specifies the angle between the pendulum and the normal, ω is the angular velocity of the pendulum, Δt is the time step, J, W, L are the inertia, the weight ,and the length of the pendulum respectively. $T(k)$ is the controller output torque.

We assume that only the experience of human in terms of linguistic fuzzy if-then rules is given for the system design. We use 5 set levels NB-"Negative Big", NS-"Negative Small", ZR-"ZeRo", PS-"Positive Small", PB-"Positive Big" for constructing the rules. Since there are two input variables to the system, the number of resulting if-then rules is 25 and we observe that this amount of rules allows the controller to balance the pendulum satisfactorily. The following is one of the fuzzy rules obtained from human experience:

if θ is PB and ω is PS then T is NB.

In conventional fuzzy inference, θ, ω and T in the above-stated rule are considered as linguistic fuzzy variables. And the linguistic descriptions "Positive Big", "Positive Small" and "Negative Big" used in the rule are in fact fuzzy sets. However, in RIM systems design, for each fuzzy variable, choose a value for which the membership function attains its maximum. So, the example rule stated above can be interpreted in RIM systems design as:

if θ' is 90 and ω' is 10 then T' is -200

where θ', ω' and T' are real variables with real constants 90, 10 and -200 assigned to them respectively. It can be easily realized that by this interpretation, each if-then rule in fact represents one real data point in the three-dimensional space. Assume that the co-ordinates of the three-dimensional space is given by (x, y, z). The example rule is in fact the point (90,10,-200) in the 3-D space. As a result of this, the rule base is composed of 25 real data points in the three-dimensional space. Our task is to approximate all the data points by one or more equations. The input or state part of the 25 data points in fact form a grid in a two-dimensional plane. Solving the problem by Dirichlet tessellation consists of the

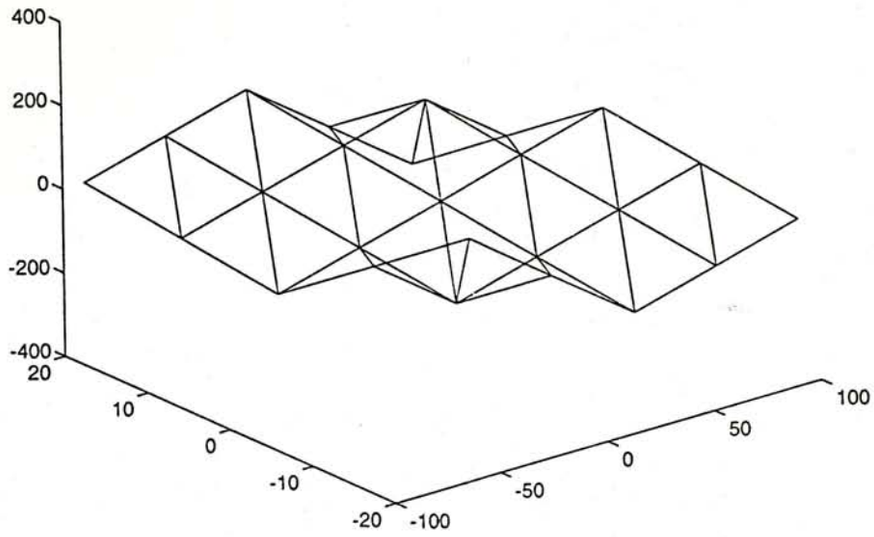

Figure 5.9: Control surface representing input output relations

following two steps; *Step 1* triangulate the data points by Dirichlet tessellation to form packed triangles (or simplexes in case of higher dimensional input vector) without overlapping. *Step 2* represents each triangle (or simplex) by a linear equation.

$$y = k_1 x_1 + k_2 x_2 + \ldots + k_n x_n + k_{n+1} \tag{5.38}$$

where n is the dimension of the input vector. In this problem, $n = 2$ since there are two input variables θ and ω. So equation (5) becomes:

$$T = k_1 \theta + k_2 \omega + k_3. \tag{5.39}$$

The parameters of each equation can be found as follows: Let $(\theta_1, \omega_1, T_1)$, $(\theta_2, \omega_2, T_2)$ and $(\theta_3, \omega_3, T_3)$ be the co-ordinates of the three data points forming the triangle. $\mathbf{k}$ be the vector $[k_1\ k_2\ k_3]$ which represents a vector of parameters to

be found. $\mathbf{T}=[T_1\ T_2\ T_3]$ which represents the output vector. $\mathbf{x}$ be the matrix

$$\begin{bmatrix} \theta_1 & \omega_1 & 1 \\ \theta_2 & \omega_2 & 1 \\ \theta_3 & \omega_3 & 1 \end{bmatrix}$$

Then,

$$\mathbf{T} = \mathbf{k}\mathbf{x}^T \tag{5.40}$$

and,

$$\mathbf{T}\mathbf{x} = \mathbf{k}\mathbf{x}^T\mathbf{x} \tag{5.41}$$

$$\mathbf{T}\mathbf{x}[\mathbf{x}^T\mathbf{x}]^{-1} = \mathbf{k}\mathbf{x}^T\mathbf{x}[\mathbf{x}^T\mathbf{x}]^{-1} \tag{5.42}$$

$$\mathbf{k} = \mathbf{T}\mathbf{x}[\mathbf{x}^T\mathbf{x}]^{-1} \tag{5.43}$$

As a result of this, a piecewise-linear control surface is constructed which is made up of 32 linear equations (see **Fig. 5.9**). When an input is given to the system, the inference process is in fact a simple substitution of the input data to the appropriate linear equation to get the real output.

5.3.3 Balancing an inverted pendulum with cart

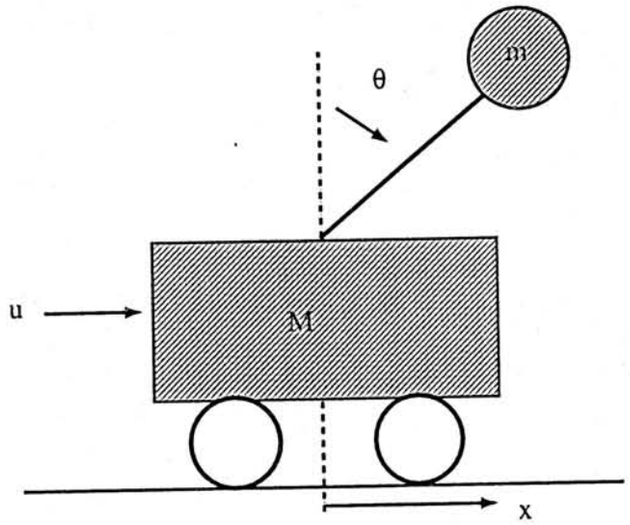

Figure 5.10: An inverted pendulum with cart

In the previous section, we demonstrated the use of Dirichlet Tessellation to balance an inverted pendulum which is fixed on a bearing. A torque is applied to it according to two input parameters, namely, angle with normal θ and angular velocity ω. Now, we try to balance an inverted pendulum which is fixed on a cart where the cart is free to move horizontally. There are two goals for the problem: (1) to balance the pendulum in vertical position; and (2) at the same time keep the cart in position $x = 0$. This problem is more complicated then the previous one as it involves three input parameters instead of two. The dynamics or the state-space model of an inverted pendulum can be stated as:

$$\dot{\theta} = \omega, \tag{5.44}$$

$$u = \dot{x}, \tag{5.45}$$

$$\dot{\omega} = \frac{g \sin\theta + \cos(\frac{-u - ml\ddot{\theta}^2 \sin\theta}{M+m})}{\frac{4}{3} + \frac{m\cos^2\theta}{M+m}}, \tag{5.46}$$

where θ specifies the angle between the pendulum and the normal, ω is the angular velocity of the pendulum, x is the horizontal position of the cart, Δt is the time step, g is the gravitational acceleration, M is the mass of the cart, m, l are the mass and the length of the pendulum respectively. u is the velocity of the cart to be controlled. We assume that only the experience of human in terms of linguistic if-then rules are given. Because of the fact that there are three input variables in this problem, in order to prevent the explosion in the number of generated rules, we only use 3 set levels NB-"negative Big", ZR-"ZeRo" and PB-"Positive Big" for constructing the rule base and observe that they are enough for performing the task satisfactorily. The number of resulting if-then rule is $3 \times 3 \times 3 = 27$. The following is one of the rules obtained from human experience.

if x is ZR and ω is PB and θ is PB then u is PB

Similar to the previous example, the above stated rule represents a data point (0, 50, 90, 10) in four-dimensional space. As a result of this, the rule base is composed of 27 real data points in the 4-D space. We try to partition the data points by Dirichlet tessellation to form packed tetrahedron (instead of triangles in 3-D case) without overlapping. Then, we represent each tetrahedron by a linear equation.

$$u = k_1\theta + k_2\omega + k_3 x + k_4 \tag{5.47}$$

The parameters ks in the equation can be found by matrix inversion method stated before. When an input is given to the system, the inference process is

a simple substitution of the input data to the appropriate linear equation to get the real output. **Fig. 5.11** shows the tessellation of the input space into tetrahedrons.

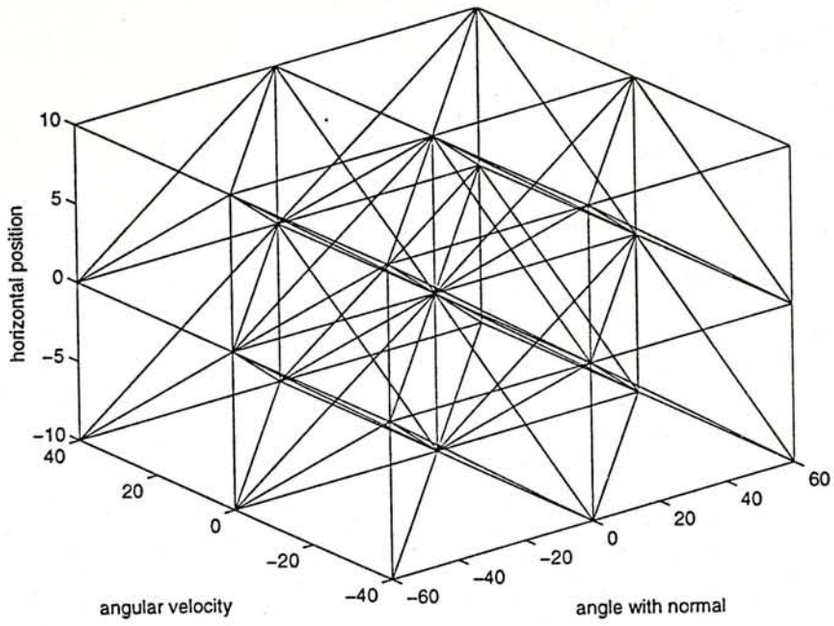

Figure 5.11: Tessellation of the input space of the inverted pendulum with cart problem

5.3.4 Truck backing-up

Backing a truck to a loading dock is a severely non-linear control problem for which no traditional control system design method exist. The dynamics of the truck is governed by the following set of equations:

$$x(t+1) = x(t) + \cos[\phi(t) + \theta(t)] + \sin[\theta(t)]\sin[\phi(t)], \tag{5.48}$$

$$y(t+1) = y(t) + \sin[\phi(t) + \theta(t)] - \sin[\theta(t)]\sin[\phi(t)], \tag{5.49}$$

$$\phi(t+1) = \phi(t) - \sin^{-1}[\frac{\sin(\theta(t))}{2}], \tag{5.50}$$

where x and y specify the position coordinates, ϕ is the angle of the truck with the horizontal, and θ is the steering angle of control. In this example, we assume that the expert knowledge is given by sampled input-output (state-control) pairs which are recorded from successful control by the human controller

and see how RIM deals with this kind of expert knowledge. We use the same set of data used by Wang and Mendel [9]. Their paper has already introduced a means of generating fuzzy rules from numerical data for a fuzzy inference system. Now, we do the similar for the RIM system in real domain by using *recursive least-squares identification method* [13]. This method assures that the resulting approximated control surface is optimal in the sense of *least-squares.* The recursive identification algorithms are estimators of the type:

$$\hat{\theta}_k = \hat{\theta}_{k-1} + P_k \phi_k \varepsilon_k, \tag{5.51}$$

$$\varepsilon_k = y_k - \phi_k^T \hat{\theta}_{k-1}, \tag{5.52}$$

$$P_k = P_{k-1} - \frac{P_{k-1}\phi_k \phi_k^T P_{k-1}}{1 + \phi_k^T P_{k-1} \phi_k}, \tag{5.53}$$

with the parameter estimate $\hat{\theta}_k$, the regressor ϕ_k, the prediction error ε_k, and the matrix P_k, which are all evaluated at time $k = 1, 2, 3, \ldots$

As the first step, we have to determine the structure of the resulting controller. This involves the partition of the input variables into regions. We have tried different partitions of input space and finally observe that the approximation is very satisfactory if we use 5 levels for the input variable x and 8 levels for the input variable ϕ, so the number of if-then rules is 40, the task is to determine the real output or consequent for each rule. This method of rule (RIM type) generation consists of the following steps:

1. Set all consequents of the rules to zero. As a result, the 40 rules are represented as real data points (x,ϕ,0).

2. Triangulate the rule points by Dirichlet tessellation to form triangles without overlapping.

3. Represents each triangle by a linear equation:

$$\theta = k_1 x + k_2 \phi + k_3. \tag{5.54}$$

The resulting control surface representing input-output relations after initializing all rules to zero is shown in **Fig. 5.12.**

4. Recursively update the equations by recursive least-squares identification method as each real input-output sample data point is given to the system. **Fig. 5.13** shows the input-output control surface after recursive least-squares identification.

Some example truck trajectories generated from the above-designed RIM controller are shown in **Fig. 5.14.** Another important property of using recursive least-squares identification is that it is very easy to modify the rule base as new data become available; i.e., to make the system adaptive.

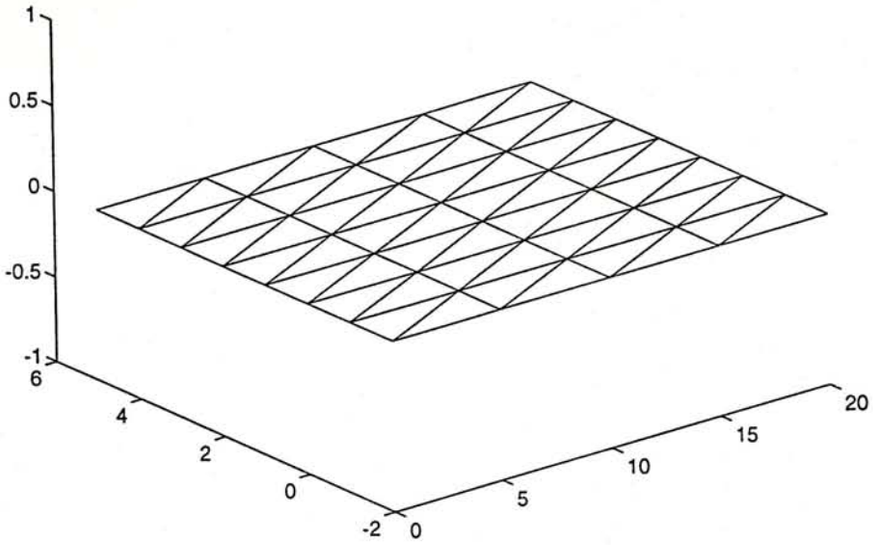

Figure 5.12: Control surface representing input output relations before recursive identification

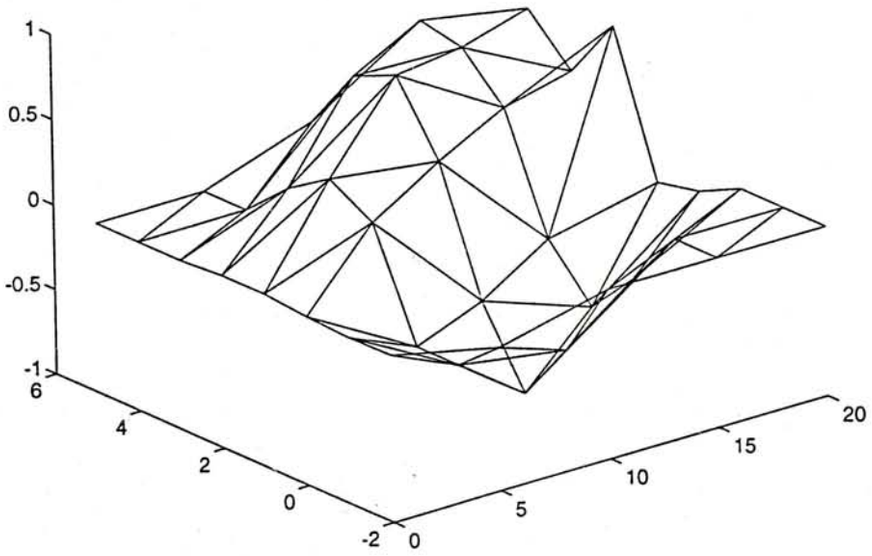

Figure 5.13: Control surface representing input output relations after recursive identification

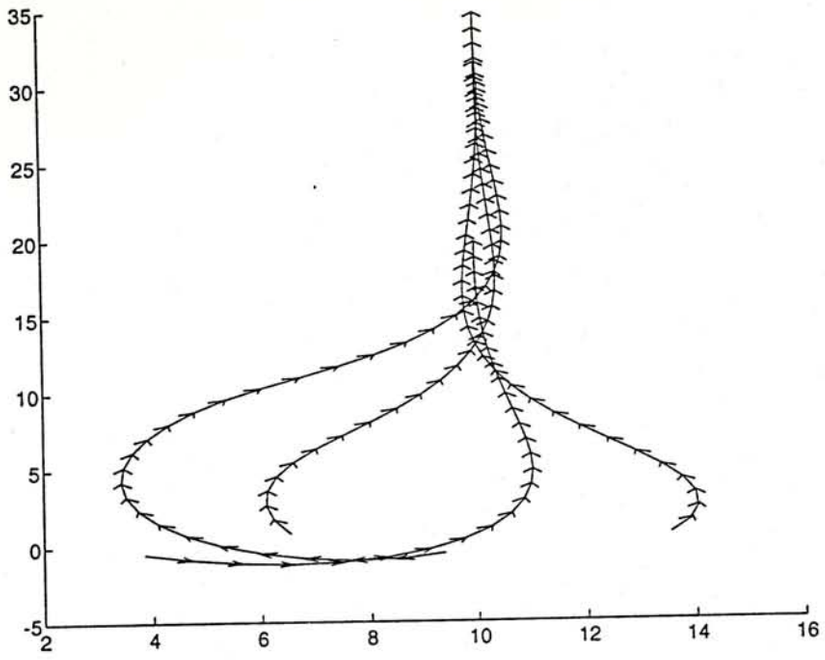

Figure 5.14: Example truck trajectories generated by RIM

5.3.5 Chaotic time series prediction

The problem of time-series prediction can be stated as: let $z(k)(k = 1, 2, 3...)$ be a time series, given $z(k-m+1), z(k-m+2), \ldots, z(k)$, determine $z(k+l)$, where m and l are fixed positive integers; i.e., determine a mapping from $z(k-m+1), z(k-m+2), \ldots, z(k) \in R^m$ to $z(k+l) \in R$. Some past samples of $z(k)$ are usually available which are used to determine the mapping.

Our approach to this problem is to use parametric models to represent the time $z(k)$. For example, the following autoregressive (AR) model may be used:

$$z(k+1) = \sum_{i=1}^{m} a_i z(k-i+1) + v(k) \tag{5.55}$$

where $v(k)$ is a white noise sequence. m is the dimension of the input vector, which equals the dimension of tessellation in our case. We assume that $m = 2$ is used in the example. The parameters a_i are estimated using the known values of

$z(k)$. The model fits well in to the RIM system we propose where each function is represented as

$$z(k+1) = k_1 z(k) + k_2 z(k-1) + k_3. \tag{5.56}$$

Chaotic time series is used as the source signal and the series can be produced by the following equation:

$$x(i) = x(i-1) + \delta[\frac{0.2x(i-\tau)}{1+x^{10}(i-\tau)} - 0.1x(i-1)] \tag{5.57}$$

where δ is the time step. When $\tau > 17$ equation (5.66) shows chaotic behavior. Higher values of τ give higher degree chaos. In this example, we choose $\tau = 40$.

In this example, we tried using the direct tessellation of the sampled input-output pairs to form the control surface without any rule generation process. However, this method may not be appropriate if the number of given data points for constructing the mapping is large because it may cause an explosion in the number of functions. Here we merely want to show the flexibility of using Dirichlet tessellation to solve a problem.

Specifically, assume that $z(1), z(2), \ldots, z(M)$ are given; then we form $M-m$ desired input-output pairs which are used to construct the control surface. **Fig. 5.15** shows the control surface formed for $m = 2$ and $M - m = 50$ by direct tessellation of data points. **Fig. 5.16** shows the prediction result generated by RIM.

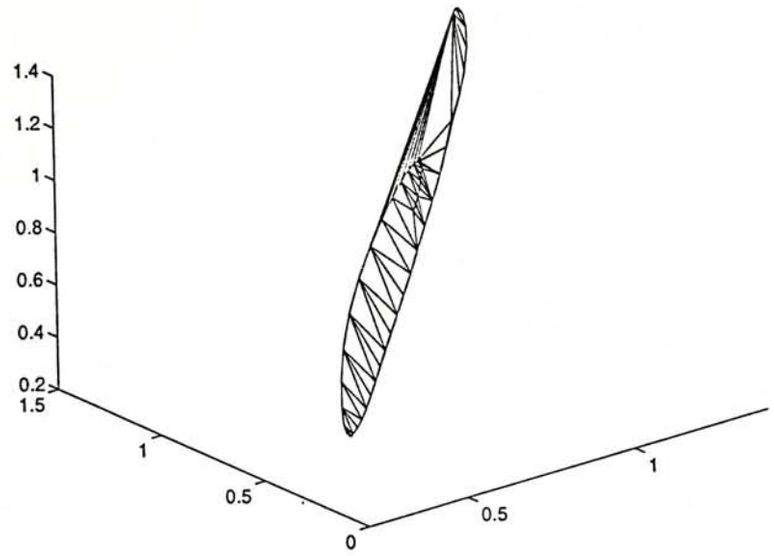

Figure 5.15: Input-output control surface by direct tessellations of data points

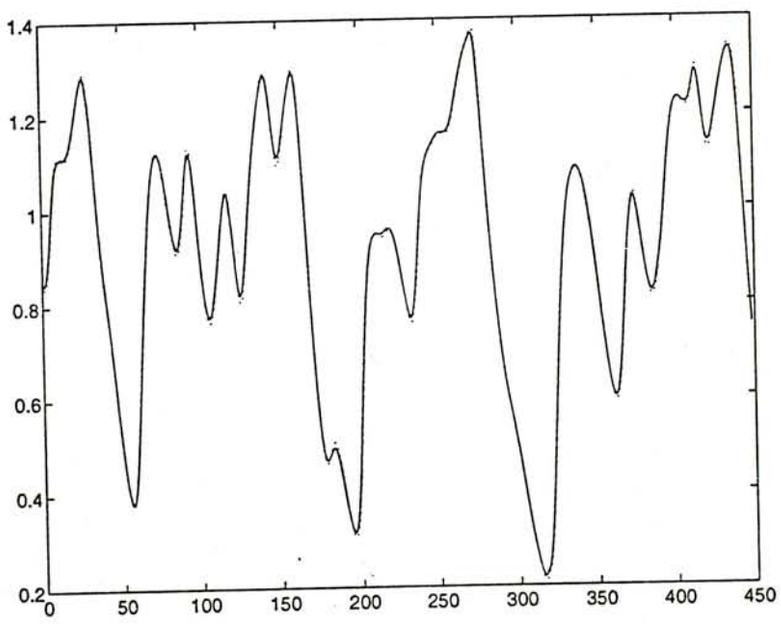

Figure 5.16: Prediction result by RIM, where solid line representing the series to be predicted and dotted line representing the predicted results

5.4 Interactive CAD platform for RIM systems design

In the design process of a fuzzy inference system, three types of information should be provided: (1) a set of fuzzy if-then rules; (2) membership functions for the fuzzy variables; (3) a fuzzy inference mechanism. Usually, the designer has to fine tune the system by means of changing either the fuzzy if-then rules or the membership functions, or both. However, the effect of changing these two types of information on the final input-output relation is usually not clearly understood by the designer because of the very nonlinear fuzzy inference mechanism acting as a barrier between them. This makes the fine tuning of the system a nearly trial and error process. On the other hand, in a RIM based system, as the input-output relation is consisting of *real* functions relating the input with the output, the effect of changing or adjusting each data points on the final input-output relation can be readily visualized. Also, as the input-output relations is made up of piecewise-linear functions, it is possible to carry out conventional control analysis such as stability analysis.

The high transparency of a RIM system makes it possible for us to place the system on a CAD control system design platform to carry out real time interactive design and analysis. We have designed one which allows the designer to interactively adjust the system parameters. The input-output relation together with the state trajectories are shown to allow the designer to adjust the system in real time.

As we mentioned before, two kinds of information are available: (1) the experience of the human controller; and, (2) sampled input-output data points

from successful control. In a design problem, each of the two kinds of information alone is usually incomplete. Although the system is successfully controlled by a human controller, some information will be lost when the human controller expresses his experience in terms of linguistic fuzzy if-then rules. Consequently, linguistic rules alone are usually not enough for designing a successful control system. On the other hand, the information from sampled input-output data pairs is usually not enough for a successful design, because the past operations usually cannot cover all the situations the control system will face. Through the use of the CAD platform, we can develop a general approach which combines both kinds of information into a common framework. **Fig. 5.17-5.18** show the CAD platforms for the design of truck backing-up and inverted pendulum examples respectively.

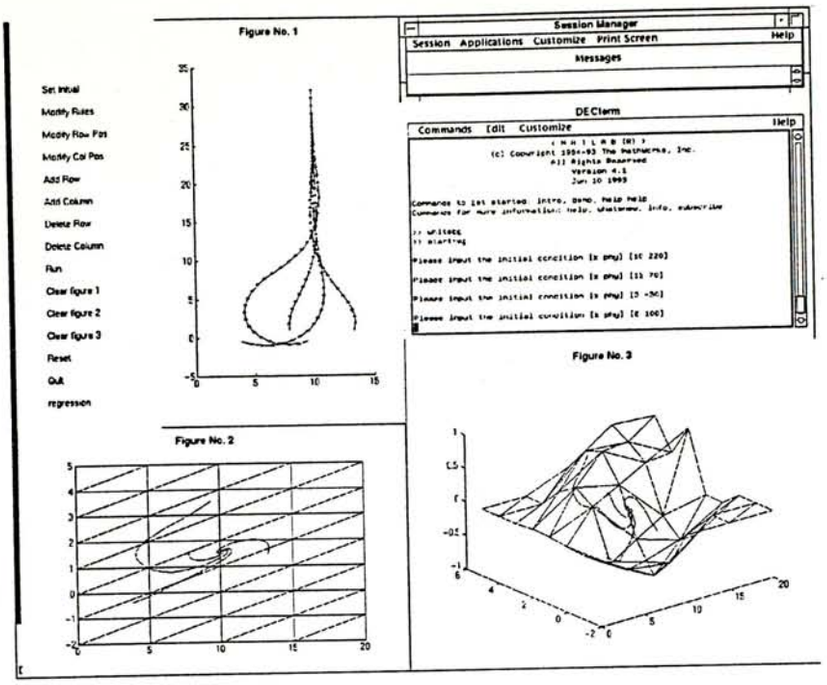

Figure 5.17: CAD platform for the development of Truck backing up RIM system

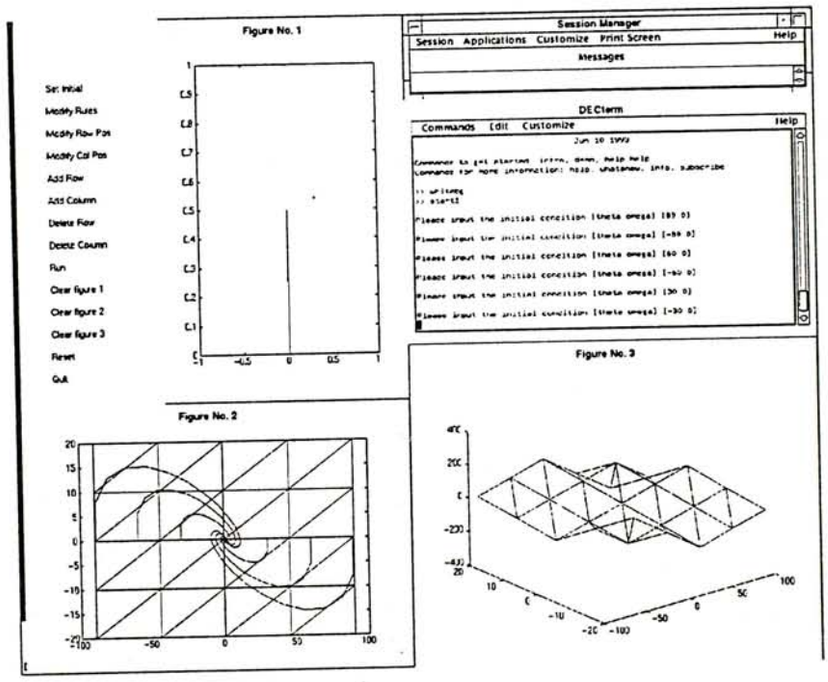

Figure 5.18: CAD platform for the development of Inverted pendulum RIM system

In this chapter, we proposed a new means of RIM system design which bases on partitioning the data points to form a set of real functions which approximate

the overall system. This method has been used successfully to solve several well-known problems given different types of expert knowledge. It shows that RIM has a merit of high transparency as the overall design and inference process is done in the real domain. This property encourages the further development of RIM to tackle more complicated problems that require the support of qualitative analysis which is difficult to be carried out for fuzzy inference systems.

Chapter 6

Analysis of Dirichlet tessellation based Real Intelligent Mapping Systems

Given a control system, the first and foremost question about its various properties is whether it is stable, because an unstable control system is typically useless and potentially dangerous. Stability analysis of fuzzy system has been studied for a period of time. Braae and Rutherford [27][28] proposed a linguistic phase plane trajectory to analyze and improve the stability of fuzzy systems by exchanging the control rules. Kickert and Mamdani [26] use the describing function method to evaluate the stability of fuzzy control systems. B. Kiszka [29] introduced the energetistic stability of fuzzy dynamic systems and developed an entropy for fuzzy system. Besides, De Glas [24] and A. Kania [23]. use the concept of α stability for analyzing fuzzy systems, in which the distinction between stability and instability is removed and a real value between 0 and 1 is used

to describe the "degree of stability" of a fuzzy system. However, all the above stated methods are in fact quite restrictive and have only limited applicability. The principal reason is that the design of fuzzy controllers has relied on ad hoc techniques. That is, the controllers were not synthesized using an underlying theory but were arrived at by trial and error. Also, by allowing the state of the system to be described by a fuzzy set, notions of unboundedness become ambiguous. The other reason is that fuzzy system is itself definitely nonlinear. All the above hinder the use of conventional control analysis techniques (e.g. Liapunov's method) for designing and analyzing fuzzy systems.

On the other hand, in the recently proposed Real Intelligent Mapping [1], it is argued that transformation of the real input data to the fuzzy domain and then back to the real domain again in the conventional fuzzy inference approach is essentially unnecessary in many applications. We introduced the use of Dirichlet tessellation (DT) for the implementation of RIM systems. In this chapter, a method for the stability analysis and design of the above stated systems is proposed. The analysis is based on the well-known Liapunov's direct and indirect method which is difficult to be applied to conventional fuzzy systems because of the reasons mentioned before. We first study the local stability property of a DT based RIM system by Liapunov's Linearization method with the inverted pendulum problem and the truck backing-up problem used as examples (Section 6.1.1 and 6.1.2). Then we analyze the globally stability property of DT based RIM system by a method based on Liapunov's direct method (Section 6.2). Also, a method for the design of DT based RIM systems is introduced which is based on conventional linear system theory (Section 6.3). In Section 6.4, a method specialized for analyzing Second order DT based RIM systems is introduced.

We then elaborate the stability analysis method to piecewise-nonlinear systems and show that the method can be applied to a class of conventional fuzzy system by establishing an equivalent piecewise-polynomial representation of it in real domain (Section 6.5).

6.1 Local Stability Analysis of DT Based RIM Systems

Local stability of a system concerns the behavior of the system near its equilibrium point. Loosely speaking, a system is described as stable if starting the system somewhere near its desired operating point implies that it will stay around the point ever after. In this section, Lyaponov's linearization method is used to analyze the local stability property of a DT based RIM system. It is a formalization of the intuition that a nonlinear system should behave similarly to its linearized approximation for small range motions. After local linearization of the system near the equilibrium point, linear control theory can be applied for local stability analysis. We now give a formal definition of local stability.

Definition 6.1[10] *The equilibrium state* $\mathbf{x} = \mathbf{0}$ *is said to be stable if, for any* $R > 0$, *there exists* $r > 0$, *such that if* $\|\mathbf{x}(0)\| < r$, *then* $\|\mathbf{x}(0)\| < R$ *for all* $t \geq 0$. *Otherwise, the equilibrium point is unstable.*

□

A autonomous (time-invariant) non-linear system can be represented by its state equation:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{6.1}$$

assume that $\mathbf{f}(\mathbf{x})$ is continuously differentiable. Then the system dynamics can be written as

$$\dot{\mathbf{x}} = (\frac{\partial \mathbf{f}}{\partial \mathbf{x}})_{\mathbf{x}=\mathbf{0}}\mathbf{x} + \mathbf{f}_{h.o.t}(\mathbf{x}) \tag{6.2}$$

where $\mathbf{f}_{h.o.t}(\mathbf{x})$ stands for higher-order terms in $\mathbf{x}$. The above Taylor expansion starts with the first-order term, due to the fact that $\mathbf{f}(\mathbf{0}) = \mathbf{0}$, since $\mathbf{0}$ is an equilibrium point. Let's use the constant matrix $\mathbf{A}$ to denote the Jacobian matrix of $\mathbf{f}$ with respect to $\mathbf{x}$ at $\mathbf{x} = \mathbf{0}$ (an $n \times n$ matrix of elements $\frac{\partial f_i}{\partial x_j}$)

$$\mathbf{A} = (\frac{\partial \mathbf{f}}{\partial \mathbf{x}})_{\mathbf{x}=\mathbf{0}} \tag{6.3}$$

Then, the system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \tag{6.4}$$

is called the linearization (or linear approximation) of the original nonlinear system at the equilibrium point $\mathbf{0}$.

The following theorem relates the stability of the linearized system (6.4) with that of the original nonlinear system (6.1).

Theorem 6.1 (Lyapunov's linearization method)[10]

- If the linearized system is strictly stable (i.e, if all eigenvalues of $\mathbf{A}$,or equivalently, all root of the characteristic equation $|s\mathbf{I}-\mathbf{A}| = 0$ are strictly in the left-half complex plane), then the equilibrium point is asymptotically stable (for the actual nonlinear system).

- If the linearized system is unstable (i.e, if at least one eigenvalue of $\mathbf{A}$, or equivalently, at least one root of the characteristic equation $|s\mathbf{I} - \mathbf{A}| = 0$ is strictly in the right-half complex plane), then the equilibrium point is unstable (for the nonlinear system).

- If the linearized system is marginally stable (i.e, all eigenvalues of $\mathbf{A}$, or equivalently, all roots of $|s\mathbf{I} - \mathbf{A}| = 0$ are in the left-half complex plane, but at least one of them is on the $j\omega$ axis), then one cannot conclude anything from the linear approximation (the equilibrium point may be stable, asymptotically stable, or unstable for the nonlinear system).

□

Now, we apply this method to investigate the local stability property of the proposed DT based RIM system. The inverted pendulum and truck backing up problems are given as examples.

6.1.1 Balancing an inverted pendulum

The dynamics or the state-space model of an inverted pendulum can be stated as:

$$\dot{\theta} = \omega, \tag{6.5}$$

$$\dot{\omega} = \frac{1}{J}(WL\sin\theta + T). \tag{6.6}$$

where θ specifies the angle between the pendulum and the normal, ω is the angular velocity of the pendulum, J, W, L are the inertia, the weight ,and the length of the pendulum respectively. T is the controller output torque.

As mentioned before, a DT based RIM controller is designed to balance the inverted pendulum to its vertical position. The controller is made up of piecewise-linear equations without any overlappings. Accordingly, only one equation is responsible for a particular input domain. For the local stability analysis of the system, we assume that it is possible for us to partition the input

space in such a way that only one equation is responsible for the equilibrium point. That is, we should partition the input space such that the equilibrium point is not located on the regions' boundaries. The equation can be represented by:

$$T = k_1\theta + k_2\omega + k_3. \tag{6.7}$$

As we know that for inverted pendulum problem, there will be no output torque applied when the system is at its equilibrium point (i.e. $\theta = 0$, $\omega = 0$). As a result, we have $k_3 = 0$. The above equation becomes:

$$T = k_1\theta + k_2\omega. \tag{6.8}$$

Substitute (6.8) into (6.6), we have:

$$\dot{\omega} = \frac{1}{J}(WL\sin\theta + k_1\theta + k_2\omega) \tag{6.9}$$

The linearized system matrix about the equilibrium point (i.e. $\theta = 0$, $\omega = 0$) is given by

$$\mathbf{A} = \left[\begin{array}{cc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{array}\right]_{\mathbf{x}=\mathbf{0}}$$

where

$$\frac{\partial f_1}{\partial x_1} = 0 \tag{6.10}$$

$$\frac{\partial f_1}{\partial x_2} = \omega \tag{6.11}$$

$$\frac{\partial f_2}{\partial x_1} = \frac{WL + k_1}{J}\theta \tag{6.12}$$

$$\frac{\partial f_2}{\partial x_2} = \frac{k_2}{J}\omega \tag{6.13}$$

Finally, the linearized system can be represented as:

$$\left[\begin{array}{c} \dot{\theta} \\ \dot{\omega} \end{array}\right] = \left[\begin{array}{cc} 0 & 1 \\ \frac{WL+k_1}{J} & \frac{k_2}{J} \end{array}\right] \left[\begin{array}{c} \theta \\ \omega \end{array}\right]$$

As a result, we can investigate the local stability of the system from the locations of the eigenvalues of the matrix $\mathbf{A}$,or equivalently, from the locations of the root of the characteristic equation $|s\mathbf{I} - \mathbf{A}| = 0$.

6.1.2 Truck backing-up

The dynamics of the truck is governed by the following set of equations:

$$\dot{x} = \cos\theta \sin\phi, \tag{6.14}$$

$$\dot{y} = \cos(\phi - \theta) - \sin\theta\cos\phi, \tag{6.15}$$

$$\dot{\phi} = -\sin^{-1}[\frac{\sin\theta}{2}], \tag{6.16}$$

where x and y specify the position coordinates, ϕ is the angle of the truck with the normal, and θ is the steering angle of control.

Similar to the previous example, the controller equation responsible for the equilibrium point is:

$$\theta = k_1 x + k_2 \phi. \tag{6.17}$$

Substitute (6.17) into (6.14) and (6.16), we have:

$$\dot{x} = \cos(k_1 x + k_2\phi)\sin\phi \tag{6.18}$$

and

$$\dot{\phi} = -\sin^{-1}[\frac{\sin(k_1 x + k_2\phi)}{2}] \tag{6.19}$$

The linearized system matrix about the equilibrium point (i.e. $\theta = 0$, $\omega = 0$) is given by

$$\mathbf{A} = \left[\begin{array}{cc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{array}\right]_{\mathbf{x=0}}$$

where

$$\frac{\partial f_1}{\partial x_1} = 0 \tag{6.20}$$

$$\frac{\partial f_1}{\partial x_2} = \phi \tag{6.21}$$

$$\frac{\partial f_2}{\partial x_1} = -\frac{k_1}{2}x \tag{6.22}$$

$$\frac{\partial f_2}{\partial x_2} = -\frac{k_2}{2}\phi \tag{6.23}$$

Finally, the linearized system can be represented as:

$$\begin{bmatrix} \dot{x} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-k_1}{2} & \frac{-k_2}{2} \end{bmatrix} \begin{bmatrix} x \\ \phi \end{bmatrix}$$

As a result, we can investigate the local stability of the system from the locations of the eigenvalues of the matrix $\mathbf{A}$,or equivalently, from the locations of the root of the characteristic equation $|s\mathbf{I} - \mathbf{A}| = 0$.

6.2 Global stability analysis of DT based RIM systems

K. Tanaka and M. Sugeno [22] proposed a method for the design and analysis of their "Takagi and Sugeno's fuzzy model" based fuzzy system. We observe that DT based RIM systems is a special case of their model when the membership functions are described by crisp set instead of fuzzy set. In this section, we apply their method to the stability analysis of DT based RIM system. A sufficient condition which guarantees the stability of piecewise jointed systems based on the Lyapunov's direct method is given. We first derive the method for piecewise-linear systems and apply it in the design of a Dirichlet tessellation based RIM

system. Then, we elaborate the method to piecewise-nonlinear system.

As mentioned in Section 5.2, a Dirichlet tessellation based RIM controller is made up of piecewise-linear equations without any overlappings. In other words, only one equation is responsible for a particular input domain and the tessellated pattern covers the whole input space. As a result of this, let $x(k)$ be the state of the system at time k, every linear subsystems mentioned above can be represented in the matrix form as $\mathbf{A}_i\mathrm{x}(k)$, where i distinguishs the individual subsystems, $\mathbf{x}(k) \in R^n$, $\mathbf{A}_i \in R^n \times R^n$, $\mathrm{x}(k) = [x(k)\, x(k-1) \cdots x(k-n+1)]^T$, and

$$\mathbf{A}_i = \begin{bmatrix} a_1^i & a_2^i & \cdots & a_{n-1}^i & a_n^i \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

The output of the RIM system is inferred as follows:

$$\begin{bmatrix} \mathrm{x}_1(k+1) \\ \mathrm{x}_2(k+1) \\ \vdots \\ \mathrm{x}_n(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1\mathrm{x}(k) \\ \mathbf{A}_2\mathrm{x}(k) \\ \vdots \\ \mathbf{A}_n\mathrm{x}(k) \end{bmatrix}$$

The analysis of DT based RIM systems is based on the well-known Lyapunov's stability theorem stated as follows:

Theorem 6.2 [30]. Consider a discrete system described by

$$\mathbf{x}(k+1) = f(\mathbf{x}(k)), \tag{6.24}$$

where $\mathbf{x}(k) \in R^n$, $f(\mathbf{x}(k))$ is an $n \times 1$ function vector with the property that

$$f(\mathbf{0}) = \mathbf{0} \quad \text{for all } k. \tag{6.25}$$

Suppose that there exists a scalar function $V(\mathbf{x}(k))$ continuous in $\mathbf{x}(k)$ such that

1. $V(\mathbf{0}) = 0$,
2. $V(\mathbf{x}(k)) > 0$ for $\mathbf{x}(k) \neq \mathbf{0}$,
3. $V(\mathbf{x}(k))$ approaches infinity as $\|\mathbf{x}(k)\| \to \infty$,
4. $\Delta V(\mathbf{x}(k)) < 0$ for $\mathbf{x}(k) \neq \mathbf{0}$.

Then the equilibrium state $\mathbf{x}(k) = \mathbf{0}$ for all k is asymptotically stable in the large and $V(\mathbf{x}(k))$ is a Lyapunov function.

□

Theorem 6.3. The equilibrium of a DT based RIM system, (6.19), is globally asymptotically stable if there exists a common positive definite matrix $\mathbf{P}$ for all the subsystems such that

$$\mathbf{A}_i^T \mathbf{P} \mathbf{A}_i - \mathbf{P} < 0 \quad \text{for } i \in 1, 2, \ldots, l. \tag{6.26}$$

□

Proof. Consider the scalar function $V(\mathbf{x}(k))$ such that

$$V(\mathbf{x}(k)) = \mathbf{x}^T(k) \mathbf{P} \mathbf{x}(k), \tag{6.27}$$

where $\mathbf{P}$ is a positive definite matrix. This function satisfies the following properties:

1. $V(\mathbf{0}) = 0$,

2. $V(\mathbf{x}(k)) > 0$ for $\mathbf{x}(k) \neq \mathbf{0}$,

3. $V(\mathbf{x}(k))$ approaches infinity as $\|\mathbf{x}(k)\| \rightarrow \infty$,

we have:

$$\begin{aligned}
\Delta V(\mathbf{x}(k)) &= V(\mathbf{x}(k+1)) - V(\mathbf{x}(k)) && (6.28)\\
&= \mathbf{x}^T(k+1)\mathbf{P}\mathbf{x}(k+1) - \mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k) && (6.29)\\
&= (\sum_{i=1}^{n} w_i\mathbf{A}_i\mathbf{x}(k))^T\mathbf{P}\sum_{i=1}^{n} w_i\mathbf{A}_i\mathbf{x}(k) - \mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k) && (6.30)\\
&= \mathbf{x}^T(k)\{\sum_{i=1}^{n} w_i\mathbf{A}_i^T\mathbf{P}\sum_{i=1}^{n} w_i\mathbf{A}_i - \mathbf{P}\}\mathbf{x}(k) && (6.31)\\
&= \mathbf{x}^T(k)\{\sum_{i,j=1}^{n} w_iw_j\mathbf{A}_i^T\mathbf{P}\mathbf{A}_j - \mathbf{P}\}\mathbf{x}(k) && (6.32)\\
&= \mathbf{x}^T(k)\{\sum_{i=1}^{n}(w_i)^2\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i - \mathbf{P}\}\mathbf{x}(k) && (6.33)\\
& && (6.34)
\end{aligned}$$

for DT based RIM system, since at one time only one function is fired. Let it be function l, we have:

$$\Delta V(\mathbf{x}(k)) = \mathbf{x}^T(k)\{\mathbf{A}_l^T\mathbf{P}\mathbf{A}_l - \mathbf{P}\}\mathbf{x}(k) \qquad (6.35)$$

we obtain

$$\Delta V(\mathbf{x}(k)) < 0. \qquad (6.36)$$

By Theorem 6.2, $V(\mathbf{x}(k))$ is a Lyapunov's function and the RIM system is globally asymptotically stable.

□

This theorem is reduced to the Lyapunov's stability theorem for linear discrete systems when $n = 1$.

This theorem can be applied to the stability analysis of a nonlinear system which is approximated by a piecewise linear function. Like the RIM systems by Dirichlet Tessellation.

Next, a necessary condition for ensuring the existence of a common $\mathbf{P}$ is given.

Theorem 6.4 [22]. Assume that $\mathbf{A}_i$ is a stable and nonsingular matrix for $i = 1, 2, \ldots, n$. $\mathbf{A}_i\mathbf{A}_j$ is a stable matrix for $i, j = 1, 2, \ldots, n$ if there exists a common positive definite matrix $\mathbf{P}$ such that

$$\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i - \mathbf{P} < 0. \tag{6.37}$$

□

Proof. From (6.37), we obtain

$$\mathbf{P} - (\mathbf{A}_i^{-1})^T\mathbf{P}\mathbf{A}_i^{-1} < 0. \tag{6.38}$$

since $(\mathbf{A}_i^{-1})^T = (\mathbf{A}_i^T)^{-1}$. Therefore, $\mathbf{P} < (\mathbf{A}_i^{-1})^T\mathbf{P}(\mathbf{A}_i^T)^{-1}$ for $i = 1, 2, \ldots, n$. Since $\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i - \mathbf{P} < 0$ from (6.37), the following inequality holds for $i, j = 1, 2, \ldots, n$:

$$\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i < (\mathbf{A}_i^{-1})^T\mathbf{P}(\mathbf{A}_i^T)^{-1}. \tag{6.39}$$

From the inequality, we obtain $\mathbf{A}_j^T\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i\mathbf{A}_j - \mathbf{P} < 0$. Therefore, $\mathbf{A}_i\mathbf{A}_j$ must be a stable matrix for $i, j = 1, 2, \ldots, n$.

□

Theorem 6.4 shows that if one of the $\mathbf{A}_i\mathbf{A}_j$'s is not a stable matrix, then there does not exist a common $\mathbf{P}$. In order to check the stability of the system, we must find a common positive definite $\mathbf{P}$. It is difficult to find a common positive definite matrix $\mathbf{P}$ as effectively as possible. So, the following simple procedure is used. The procedure consists of two steps.

1. We find a positive definite matrix $\mathbf{P}_i$ such that

$$\mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_i - \mathbf{P}_i < 0. \tag{6.40}$$

for $i = 1, 2, \ldots, n$. It is possible to find a positive definite matrix $\mathbf{P}_i$ if $\mathbf{A}_i$ is a stable matrix.

2. Next, if there exists $\mathbf{P}_j$ in $\{\mathbf{P}_i | i = 1, 2, \ldots, n\}$ such that

$$\mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i - \mathbf{P}_j < 0. \tag{6.41}$$

for $i = 1, 2, \ldots, n$, then we select $\mathbf{P}_j$ as a common $\mathbf{P}$. If the second step has not succeed, go back to the first step.

6.3 Design of a stable DT based RIM system

We have considered the conditions for the stability of a DT based RIM system by using Lyapunov's direct method in the previous section. In this section, we propose a design method of a model based RIM controller. The controller can be designed so as to guarantee the stability of the RIM system.

A DT based RIM system consisting n subsystems can be represented as:

$$\begin{bmatrix} \mathbf{x}_1(k+1) \\ \mathbf{x}_2(k+1) \\ \vdots \\ \mathbf{x}_n(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1\mathbf{x}(k) \\ \mathbf{A}_2\mathbf{x}(k) \\ \vdots \\ \mathbf{A}_n\mathbf{x}(k) \end{bmatrix}$$

By taking the Laplace's transform, We can apply the well-known root locus method to design the system by the following steps:

1. Choose an appropriate proportional gain K for the controller such that all the roots of the characteristic equations of the subsystems $\mathbf{A}_i$s lie inside the unit circle $\|z\| = 1$ in the z-plane. This ensures that all the subsystems are stand alone stable systems.

2. Check the stability of the overall system by the procedure to find a common $\mathbf{P}$ discussed in the previous section. If the system is not stable, go back to step 1.

Example 6.2 [22]. Let us consider the following two linear systems:

L_1: $x(k+1) = 2.178x(k) - 0.361x(k-1) + 0.603u(k)$

L_2: $x(k+1) = 2.256x(k) - 0.361x(k-1) + 1.120u(k)$

We try to stabilize the overall system using a linear controller with a proportional gain K. The controller can be described as:

$$u(k) = Kx(k) \tag{6.42}$$

The two systems becomes:

S_1: $x(k+1) = (2.178x(k) - 0.603K) - 0.361x(k-1)$

S_2: $x(k+1) = (2.256x(k) - 1.120K) - 0.361x(k-1)$

Here we assume that reference input $r(k) = 0$. Next, we utilize the root locus method to determine the parameter K. It is not always necessary to utilize the root locus method. For example, we may use the technique of a Bode diagram or pole assignment. From **Fig. 6.1** and **Fig. 6.2**, it is well known that the stability boundary in the z-plane is the unit circle $|z| = 1$. We can stabilize the linear subsystems of S_1 and S_2 when we choose a gain K such that $0.980 < K < 6.25$ and $0.80 < K < 3.23$, respectively. Therefore, in order to stabilize the overall system, we must choose a gain K such that $0.98 < K < 3.23$.

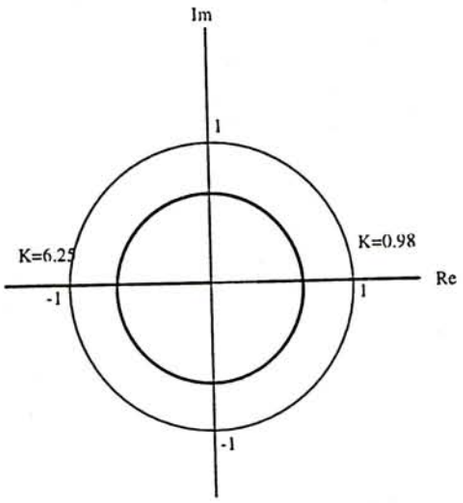

Figure 6.1: Root locus for system 1

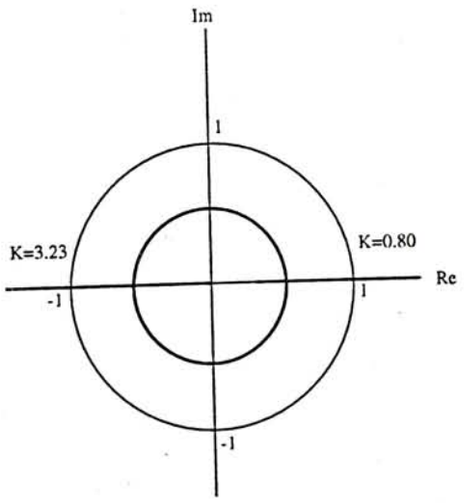

Figure 6.2: Root locus for system 2

Lastly, we check the stability of the overall system using the procedure to find a common $\mathbf{P}$ for the subsystems. For the linear subsystems S_1 and S_2, we have:

$$\mathbf{A}_1 = \begin{bmatrix} 2.178 - 0.603K & -0.588 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{A}_2 = \begin{bmatrix} 2.256 - 1.120K & -0.361 \\ 1 & 0 \end{bmatrix}$$

Here, we choose $K = 1.12$ and choose $\mathbf{P}$ such that

$$\mathbf{P} = \begin{bmatrix} 2.0 & -1.3 \\ -1.3 & 1.0 \end{bmatrix}$$

Then the condition $\mathbf{A}_i^T \mathbf{P} \mathbf{A}_i - \mathbf{P} < 0$ is satisfied and the overall system is globally asymptotically stable.

6.4 A method for analyzing Second order DT based RIM systems

The above stated method for the analysis of global stability of DT based RIM systems has one drawback. Namely, it may be difficult to find the common positive definite matrix $\mathbf{P}$. S. Kawamoto, K. Tada, A. Ishigame and T. Taniguchi [37] proposed a method of finding the common $\mathbf{P}$ graphically for the Tagaki and Sugeno's fuzzy model. Since the DT based RIM system is a special case of that model with crisp sets instead of fuzzy set. Accordingly, we can use their method for finding the common $\mathbf{P}$. The following is an approach for finding the whole region in where a 2×2 real matrix $\mathbf{P}$ exists. It provides a guidance for finding the $\mathbf{P}$ in a more systematic way. For details, please refer to [37].

A DT based RIM system can be represented as:

$$\mathbf{x}(k+1) = \sum_{i=1}^{n} w_i A_i \mathbf{x}(k), \tag{6.43}$$

where n is the number of linear subsystems that make up the overall system and

$$w_i = \begin{cases} 1 & \text{if the function } i \text{ is fired} \\ 0 & \text{if the function } i \text{ is not fired} \end{cases}$$

We know that the equilibrium of the above system is asymptotically stable in the large if there exists a common positive definite matrix $\mathbf{P}$ for all the subsystems such that $\mathbf{A}_i$ is stable and nonsingular, and

$$\mathbf{A}_i^T \mathbf{P} \mathbf{A}_i - \mathbf{P} < 0 \tag{6.44}$$

From the above equation, we have

$$\mathbf{A}_1^T \mathbf{P} \mathbf{A}_1 - \mathbf{P} = -\mathbf{Q}_1 < 0 \tag{6.45}$$

$$\mathbf{A}_2^T \mathbf{P} \mathbf{A}_2 - \mathbf{P} = -\mathbf{Q}_2 < 0 \tag{6.46}$$

$$\vdots \tag{6.47}$$

$$\mathbf{A}_m^T \mathbf{P} \mathbf{A}_m - \mathbf{P} = -\mathbf{Q}_m < 0 \tag{6.48}$$

where matrices $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_m > 0$. We assume in this approach that the common positive definite matrix $\mathbf{P}$ is a 2×2 real one, that is,

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{21} \end{bmatrix}$$

and rewrite it without a loss of generality, according to $p_{12} \geq 0$, as

$$\mathbf{P} = \begin{bmatrix} p_1 & \pm 1 \\ \pm 1 & p_2 \end{bmatrix}, p_1 = \frac{p_{11}}{|p_{12}|}, p_2 = \frac{p_{22}}{|p_{12}|}$$

Setting

$$\mathbf{A}_1 = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$$

and

$$\mathbf{Q}_1 = \begin{bmatrix} q_1 & q_2 \\ q_3 & q_4 \end{bmatrix}$$

and substituting these forms into (6.47) yield

$$q_1 = -\{(a_1^2 - 1)p_1 + a_3^2 p_2 \pm 2a_1 a_3\}, \tag{6.49}$$

$$q_2 = -\{a_1 a_2 p_1 + a_3 a_4 p_2 \pm (a_1 a_4 + a_2 a_3 - 1)\}, \tag{6.50}$$

$$q_3 = -\{a_2^2 p_1 + (a_4^2 - 1)p_2 \pm 2a_2 a_4\}. \tag{6.51}$$

Since $\mathbf{P} > 0$, we have

$$p_1 > 0, \tag{6.52}$$

$$p_1 p_2 > 1, \tag{6.53}$$

and the condition $\mathbf{Q}_1 > 0$ gives $q_1 > 0$, by rewriting it,

$$(a_1^2 - 1)p_1 + a_3^2 p_2 \pm 2a_1 a_3 < 0 \tag{6.54}$$

Also, from $q_1 q_3 - q_2^2 > 0$, we get

$$a_2^2 p_1^2 + a_3^2 p_2^2 - \{(a_1 a_4 - a_2 a_3)^2 - (a_1^2 + a_4^2) + 1\}p_1 p_2 \tag{6.55}$$

$$\pm 2a_2(a_4 - a_1)p_1 \pm 2a_3(a_1 - a_4)p_2 \tag{6.56}$$

$$+\{(a_1 a_4 - a_2 a_3)^2 - 2(a_1 a_4 + a_2 a_3) + 1\} < 0 \tag{6.57}$$

Then, we can construct the $\mathbf{P}^{(1)}$-region, which is the $\mathbf{P}$-region for (6.47), and satisfies conditions (6.54)-(6.59) in the $p_1 - p_2$ plane. Repeating for (6.48)-(6.50), we have $\{\mathbf{P}^{(1)}, \mathbf{P}^{(2)}, \cdots, \mathbf{P}^{(m)}\}$ and the $\mathbf{P}$-region by $\mathbf{P} = \mathbf{P}^{(1)} \cap \mathbf{P}^{(2)} \cap$

$\cdots \cap \mathbf{P}^{(m)}$. Thus each point (p_1, p_2) included in the **P**-region has the common positive definite matrix **P** as

$$\mathbf{P} = |p_{12}| \begin{bmatrix} p_1 & \pm 1 \\ \pm 1 & p_2 \end{bmatrix}, (p_{12} \geq 0)$$

In addition, it should be noticed that $p_{12} = 0$ corresponds to a trivial case.

6.5 Piecewise-polynomial real domain representation of a class of fuzzy controller and its stability

Now we elaborate the method stated in Section 6.3 to study the stability of piecewise-nonlinear systems. Suppose there is a system formed by piecewise-jointed nonlinear subsystems. Each non-linear subsystem can be considered as a stand-alone system. Since Liapunov's direct method is also applicable to non-linear systems, we can apply the method stated in the previous section to analyze this kind of systems. Suppose we can identify a common Liapunov's function V for all the nonlinear subsystems, then we can describe the overall nonlinear system as globally asymptotically stable.

In this section, we are going to establish an equivalent piecewise-polynomial representation of a class of conventional fuzzy system in real domain. By doing this, we can apply the above stated stability analysis to this kind of fuzzy system.

The class of fuzzy controllers use algebraic product as the logical AND operator and correlation-product inference method. Also, all the fuzzy variables have triangular membership functions. This kind of fuzzy controller is widely used for

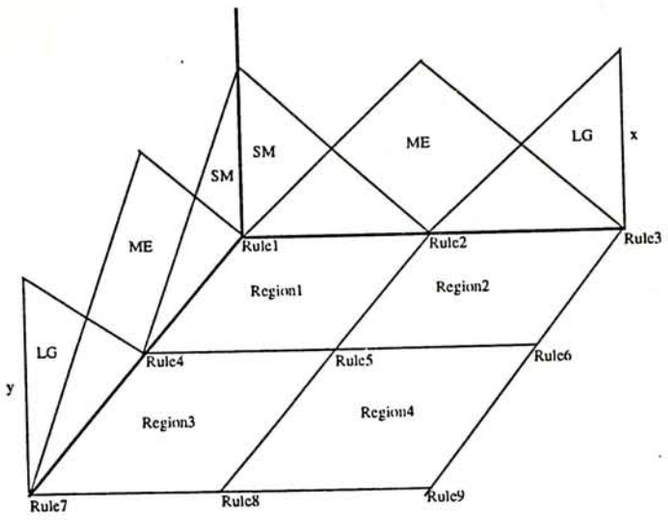

Figure 6.3: Membership functions of the fuzzy system

control applications [31] because of its simplicity. It provides smooth response [32] and its architecture is supported by existing analog/digital fuzzy processors [33] [34]. To simplify the discussion, we start with a two-input variables fuzzy controller. We assume that each of the fuzzy input variable has three set values: (1) Small (SM) with $\mu_{SM}(0) = 1$; (2) Medium (ME) with $\mu_{ME}(1) = 1$; and (3) Large (LG) with $\mu_{LG}(2) = 1$. The membership functions of the variables are shown in **Fig. 6.3.**

There are altogether $3 \times 3 = 9$ rules for the controller. If the fuzzy controller is defined as before, we observe that only four of the nine rules are fired at one time depending on the input. The input space can in fact be divided into four separate regions as in **Fig. 6.3.** Every time the input fall into one specific region, only the four rules located at the corners of that region are fired. For instance, if an input (x, y) is given which is within region 1. Then only rule number 1, 2, 4 and 5 are fired and the output can be inferred as:

$$z = \frac{\mu_1 r_1 + \mu_2 r_2 + \mu_4 r_4 + \mu_5 r_5}{\mu_1 + \mu_2 + \mu_4 + \mu_5} \tag{6.58}$$

where r_1, r_2, r_4 and r_5 are the consequent values of rule 1, 2, 4 and 5 respectively and μs are their corresponding weights. If algebraic product is used as the logical AND operator and correlation-product inference method are used. It can be easily shown that:

$$\mu_1 = (1-x)(1-y) \tag{6.59}$$

$$\mu_2 = x(1-y) \tag{6.60}$$

$$\mu_4 = (1-x)y \tag{6.61}$$

$$\mu_5 = xy \tag{6.62}$$

As a result, the output can be represented as:

$$z = (1-x)(1-y)r_1 + x(1-y)r_2 + (1-x)yr_4 + xyr_5 \tag{6.63}$$

$$= (r_1 - r_2 - r_4 + r_5)xy + (r_2 - r_1)x + (r_4 - r_1)y + r_1 \tag{6.64}$$

which is a polynomial in x and y. Similar for region 2, we can show that:

$$z = \frac{\mu_2 r_2 + \mu_3 r_3 + \mu_5 r_5 + \mu_6 r_6}{\mu_2 + \mu_3 + \mu_5 + \mu_6} \tag{6.65}$$

and,

$$\mu_2 = (2-x)(1-y) \tag{6.66}$$

$$\mu_3 = (x-1)(1-y) \tag{6.67}$$

$$\mu_5 = (2-x)y \tag{6.68}$$

$$\mu_6 = (x-1)y \tag{6.69}$$

As a result, the output can be represented as:

$$z = (2-x)(1-y)r_2 + (x-1)(1-y)r_3 + (2-x)yr_5 + (x-1)yr_6 \tag{6.70}$$

$$= (r_2 - r_3 - r_5 + r_6)xy + (r_3 - r_2)x + (r_3 - 2r_2 + 2r_5 - r_6)y \tag{6.71}$$

$$+ (2r_2 - r_3) \tag{6.72}$$

which is also a polynomial in x and y. We can do the similar for region 3 and region 4 to find their corresponding polynomial. As a result, the overall fuzzy controller can be represented by piecewise-jointed polynomials in real domain with each sub-controller represented as:

$$z = c_1xy + c_2x + c_3y + c_4 \tag{6.73}$$

For n input system, each subcontroller can be represented by the following polynomial:

$$x_{n+1} = c_1x_1x_2\cdots x_n + c_2x_2x_3\cdots x_n + c_3x_1x_3\cdots x_n + \cdots \tag{6.74}$$

$$+ \; c_{n+1}x_1x_2\cdots x_{n-1} + c_{n+2}x_3x_4\cdots x_n + \cdots \tag{6.75}$$

we can apply the method stated in section 6.3 to analyze this kind of systems. Suppose we can identify a common Liapunov's function V for all the nonlinear subsystems, then we can describe the overall nonlinear system as globally asymptotically stable. However, in the design and analysis of this kind of system, we cannot rely on linear system theory since the subsystems are not linear.

In this chapter, a method for the stability analysis of Dirichlet tessellation based RIM systems is proposed. Also, a method for the design of DT based RIM systems is proposed which is based on conventional linear system theory We then apply the analysis method to piecewise-nonlinear systems and show that the method can also be applied to a class of conventional fuzzy system by establishing an equivalent piecewise-polynomial representation of it in real domain. However, in the design and analysis of that kind of fuzzy system, we cannot rely on linear system theory since the subsystems are not linear. We conclude that our DT based RIM system design can be based on linear system

theory (instead of trial-and-error) which is difficult to be done in conventional fuzzy system design.

Chapter 7

Conclusion

In this thesis, we argue that transformation of the real input data to the fuzzy domain and then back to the real domain again in the fuzzy inference approach is essentially unnecessary in many applications. Instead, the fuzzy inference procedure is replaced by a mapping operating directly on real data. The new approach is named Real Intelligent Mapping (RIM). We proposed a new means of RIM system design which is based on partitioning the data points to form a set of real functions approximating the overall system. This method has been used successfully to solve several well-known problems given different types of expert knowledge. It shows that RIM has a merit of high transparency as the overall design and inference process is done in the real domain. A method for the stability analysis of Dirichlet tessellation based RIM systems is proposed. Besides, a method for the design of stable DT based RIM systems is introduced which is based on conventional linear system theory. This property encourages the further development of RIM to tackle more complicated problems that require the support of qualitative analysis which is difficult to be carried out for fuzzy

inference systems. We also apply the analysis method to piecewise-nonlinear systems and show that the method can also be applied to a class of conventional fuzzy system by establishing an equivalent piecewise-polynomial representation of it in real domain. However, in the design of that kind of fuzzy system, we cannot rely on linear system theory since the subsystems are not linear.

We are not trying to ignore fuzzy inference as a way to solve a fuzzy expert knowledge oriented problem. However, the difficulty in analyzing a fuzzy inference based system hinders its further development to handle more sophisticated control problems. It is clear that the method we propose is not the only solution. We merely point out that even fuzzy inference with membership functions is one of the many methods for solving problems and different methods may have their own merits.

Bibliography

[1] C. P. Kwong, "Fuzzy inference without membership function," presented at the *Workshop on the Future Directions of Fuzzy Theory and Systems*, Hong Kong, Oct. 1993.

[2] W. L. Yeung and C. P. Kwong, "Design of Real Intelligent Mapping Systems Using Dirichlet Tessellation," submitted to *IEEE trans. Syst. Man Cybern.*.

[3] W. L. Yeung and C. P. Kwong, "Analysis of Dirichlet tessellation based Real Intelligent Mapping Systems," in preparation.

[4] A. Bowyer, "Computing Dirichlet tessellations," *The Computer Journal*, vol. 24, No. 2, pp. 162-166, 1981.

[5] P. J. Green and R. Sibson, "Computing Dirichlet tessellations in the plane," *The Computer Journal*, vol. 21, No. 2, pp. 168-173, 1978.

[6] L. A. Zadeh, "Fuzzy sets," *Informat. Control*, vol. 8, pp. 338–358, 1965.

[7] B. Kosko, "Neural Networks and Fuzzy Systems," *Prentice-Hall*, 1992.

[8] C. C. Lee, "Fuzzy logic in control system : fuzzy controller – part I and II," *IEEE Trans. Syst. Man Cybern.*, vol. SMC–20, pp. 404–43,5, Nov. 1990.

[9] L. X. Wang and Jerry M. Mendel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Trans. Syst. Man Cybern.*, vol. 22., No. 6, pp. 1414-1427, Nov. 1992.

[10] Jean-Jacques E. Slotine and Wei-ping Li, "Applied Nonlinear Control," *Prentice-Hall*, 1991.

[11] Erwin Kreyszig, "Introductory Functional Analysis with Applications," *John Wiley and Sons*, 1978.

[12] T. L. Vincent, A. I. Mees, and L. S. Jennings, "Dynamics of Complex Interconnected Biological Systems," *Boston: Birkhäuser*, 1990.

[13] R. Johansson, "System Modeling and Identification," *Prentice-Hall*, 1993.

[14] M. Vidyasagar, "Nonlinear Systems Analysis," 2nd edition, *Prentice-Hall*, 1993.

[15] Rogers, C. A., "Packing and Covering," *Cambridge Mathematical Tracts No. 54, Cambridge University Press*, 1964.

[16] George J. Klir and Tina A. Folger, "Fuzzy sets, uncertainty, and information," *Prentice-Hall*, 1992.

[17] D. H. Owens, "Multivariable and optimal systems," *Academic press*, 1981.

[18] N. K. Sinha and B. Kuszta, "Modeling and identification of dynamic systems," *Van Nostrand Reinhold Electrical/Computer Science and Engineering Series*, 1983.

[19] John C. Doyle, Bruce A. Francis and Allen R. Tannenbaum, "Feedback Control Theory," *Maxwell macmillan international editions*, 1992.

[20] B. Kosko, "Fuzzy systems as universal approximators," *Proc. FUZZ-IEEE'92*, pp.1153-1162, 1992.

[21] L. X. Wang, "Fuzzy systems are universal approximators," *Proc. FUZZ-IEEE'92*, pp.1163-1169, 1992.

[22] Kazuo Tanaka and Michio Sugeno, "Stability analysis and design of fuzzy control systems," *Fuzzy Sets and Systems*, vol. 45, pp. 135-156, 1992.

[23] Aleksander A. Kania, "On Stability of Formal Fuzziness Systems," *Information Science*, vol. 22, pp. 51-68, 1980.

[24] Michel De Glas, "Invariance and Stability of Fuzzy Systems," *Journal of Mathematical Analysis and Applications*, vol. 99, pp. 299-319, 1984.

[25] R. M. Tong, "Analysis and control of fuzzy systems using finite discrete relations," *Int. J. Control*, vol. 27, No. 3, pp. 431-440, 1978.

[26] W. J. M. Kickert and E. H. Mamdani, "Analysis of a fuzzy logic controller," *Fuzzy Sets and Systems*, vol. 1, pp. 29-44, 1978.

[27] M. Braae and D. A. Rutherford, "Selection of parameters for a fuzzy logic controller," *Fuzzy Sets and Systems*, vol. 2, pp. 185-199, vol. 2.

[28] M. Braae and D. A. Rutherford, "Theoretical and linguistic aspects of the fuzzy logic controller," *Automatic*, bol. 15, pp, 553-577, 1979.

[29] Jerzy B. Kiszka, Madan M. Gupta and Peter N. Nikiforuk, "Energetistic Stability of Fuzzy Dynamic Systems," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-15, No. 6, Nov/Dec 1985.

[30] B. C. Kuo, *Digital Control Systems* (Holt-Saunders, 1980).

[31] M. Sugeno. "An introductory survey of fuzzy control," *Informat. Sci.*, 36:59-83, 1985.

[32] C. G. Moore, C. J. Harris and M. Brown. *Intelligent Control: Aspects of Fuzzy Logic and Neural Networks.* World Scientific, 1993.

[33] C. C. Lee, "Fuzzy Logic in control system : fuzzy controller - part I and II," *IEEE Trans. Syst. Man Cybern.*, SMC-20, pp. 404-435, Nov. 1990.

[34] M. Togai and H. Watanabe, "A VLSI Implementation of a Fuzzy-Inference Engine: Toward an Expert System on a Chip," *Informat. Sci.*, 38:147-163, 1986.

[35] Eyckhoff P., "System Identification: Parameter and State Estimation," *Wiley: London*, 1974.

[36] L. X. Wang and Jerry M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Trans. Neural Networks*, vol. 3, No. 5, pp. 807-814, Sep. 1992.

[37] S. Kawamoto, K. Tada, A. Ishigame and T. Taniguchi, "An approach to stability analysis of second order fuzzy systems," *IEEE Inter. Conf. on Fuzzy Systems*, pp. 1427-1434, 1992.

[38] K. Tanaka and M. Sano, "A robust stabilization problem of fuzzy control systems and its application to backing up control of a truck-trailer," *IEEE Trans. Fuzzy Systems*, vol. 2, No. 2, pp. 119-134, 1994.

[39] Hao Ying, "Analytical structure of a two-input two-output fuzzy controller and its relation to PI and multilevel relay controllers," *Fuzzy Sets and Systems*, 63, pp. 21-33, 1994.

[40] Dimitar P. Filev and Ronald R. Yager, "On the analysis of fuzzy logic controllers," *Fuzzy Sets and Systems*, 68, pp. 39-66, 1994.

[41] Heidar A. Malki, Huaidong Li and Guanrong Chen, "New design and stability analysis of fuzzy proportional-derivative control systems," *IEEE Trans. Fuzzy Systems*, vol. 2, No. 4, pp. 245-256, 1994.

[42] Tor A. Johansen, "Fuzzy model based control: stability, robustness, and performance issues," *IEEE Trans. Fuzzy Systems*, vol. 2, No. 3, pp. 221-234, 1994.

[43] Byung Soo Moon, "Equivalance between fuzzy logic controllers and PI controllers for single input systems," *Fuzzy Sets and Systems*, 69, pp. 105-113, 1995.

[44] Francois Guely and Patrick Siarry, "A centred formulation of Takagi-Sugeno rules for improved learning efficiency," *Fuzzy Sets and Systems*, 62, pp. 277-285, 1994.

[45] Yager, Ronald R. and Filev, Dimitar P., "Essentials of fuzzy modeling and control," *J. Wiley : New York*, 1994.