Spring 2003

Fuzzy System Identification Based Upon a Novel Approach to Nonlinear Optimization

Raymond Scott Starsman
Old Dominion University

FUZZY SYSTEM IDENTIFICATION BASED UPON A NOVEL APPROACH TO NONLINEAR OPTIMIZATION

by

Raymond Scott Starsman
B.S. May 1986, United States Naval Academy
B.S. December 1991, Naval Postgraduate School
M.S. December 1991, Naval Postgraduate School

A Dissertation Submitted to the Faculty of Old Dominion University in Partial Fulfillment of the Requirement for the Degree of

DOCTOR OF PHILOSOPHY

ELECTRICAL ENGINEERING

OLD DOMINION UNIVERSITY
May 2003

Approved by:

[illegible]
Oscar R. González (Director)

[illegible]
Stephen A. Zahorian (Member)

[illegible]
W. Steven Gray (Member)

[illegible]
John J. Swetits (Member)

ABSTRACT

FUZZY SYSTEM IDENTIFICATION BASED UPON A NOVEL APPROACH TO NONLINEAR OPTIMIZATION

Raymond Scott Starsman
Old Dominion University, 2003
Director: Dr. Oscar R. González

Fuzzy systems are often used to model the behavior of nonlinear dynamical systems in process control industries because the model is linguistic in nature, uses a natural-language rule set, and because they can be included in control laws that meet the design goals. However, because the rigorous study of fuzzy logic is relatively recent, there is a shortage of well-defined and understood mechanisms for the design of a fuzzy system. One of the greatest challenges in fuzzy modeling is to determine a suitable structure, parameters, and rules that minimize an appropriately chosen error between the fuzzy system, a mathematical model, and the target system. Numerous methods for establishing a suitable fuzzy system have been proposed, however, none are able to demonstrate the existence of a structure, parameters, or rule base that will minimize the error between the fuzzy and the target system.

The piecewise linear approximator (PLA) is a mathematical construct that can be used to approximate an input-output data set with a series of connected line segments. The number of segments in the PLA is generally selected by the designer to meet a given error criteria. Increasing the number of segments will generally improve the approximation. If the location of the breakpoints between segments is known, it is a straightforward process to select the PLA parameters to minimize the error. However, if the location of the breakpoints is not known, a mechanism is required to determine their locations. While algorithms exist that will determine the location of the breakpoints, they do not minimize the error between data and the model. This work will develop theory that shows that an optimal solution to this nonlinear optimization problem exists and demonstrates how it can be applied to fuzzy modeling.

This work also demonstrates that a fuzzy system restricted to a particular class of input membership functions, output membership functions, conjunction operator, and defuzzification technique is equivalent to a piecewise linear approximator (PLA). Furthermore, this work develops a new nonlinear optimization technique that minimizes the error between a PLA and an arbitrary one-dimensional set of input-output data and solves the optimal breakpoint problem. This nonlinear optimization technique minimizes the approximation error of several classes of nonlinear functions leading up to the generalized PLA. While direct application of this technique is computationally intensive, several paths are available for investigation that may ease this limitation. An algorithm is developed based on this optimization theory that is significantly more computationally tractable. Several potential applications of this work are discussed including the ability to model the nonlinear portions of Hammerstein and Wiener systems.

ACKNOWLEDGEMENTS

Achieving an accomplishment that has been a personal goal since childhood is an intensely fulfilling and rewarding experience. Of course, attaining this required assistance from more family, friends, coworkers, and supervisors than is possible to acknowledge here.

The United States Navy, through which I have earned my Bachelor's and Master's degrees, has been exceptionally supportive in providing the time and financial assistance that made this possible. The Navy has long provided superb assistance to sailors looking to advance their education. Without this help, I could never have completed this program.

Through the course of this work, I have had numerous supervisors that have provided invaluable encouragement and assistance. Firstly, I must thank RADM John Gauss, USN (Ret) who truly opened the door that allowed me to begin this work. His intelligence and persistence in the delivery of Command and Control capability to the Fleet continues to inspire me today. Col Thomas Andrew, USAF (Ret) and CAPT Pamela Mulvehill, USN (Ret) were instrumental in providing the time necessary to complete the coursework requirements of this program. Without their support and understanding, you would not be reading this today. CAPT Ros Poplar, USN and CAPT Marty Jenkins, USN provided me the time and encouragement to continue this work in the incredibly difficult environment of a ship at sea. Lastly, I must extend my most sincere appreciation to Ms. Monica Shephard who serves as the Director for Command, Control, Communications, Computers, Combat systems, and Intelligence at the Combined Fleet Forces Command. Her pride in the achievements of her subordinates and her unparalleled support for her people are a wonderful example to all those around her. Thank you all.

The effect that friends and colleagues have had on this work are equally important though in more subtle ways. Their encouragement and support made it possible to continue to push on even when the pressures of work threatened to derail me. While the people who have helped me is to numerous to mention here, I would specifically like to thank Hugh Walsh, Laura Reid, Rogers Peters, Sally Schornak, Henry Huddleston, Richard Payne, Joe Farbo, Brian

Watson, Ted Hill, Sara Spath, Danelle Barrett, John Stafford, Jon Kaltwasser, Rajeev Parekh, Leigh Armistead, and Al Allison.

The wonderful faculty at Old Dominion University also deserve my most profound gratitude. Dr. Zahorian taught me several important lessons, the most memorable being that the more you learn, the more you realize you know almost nothing. Dr. Swetits presented approximation theory in such a way that even a relative novice (such as myself) could quickly grasp the core concepts. Dr. Gray provided me probably the single most important piece of advice as I approached this problem. His advice to distill a problem to it's most fundamental components proved the key insight I needed to solve this problem. Finally, my greatest debt of gratitude goes to Dr. Gonzalez. He has faithfully shepherded me through this difficult process and provided more assistance with research and funding than I had any right to expect.

Those who provided the most continuous and intense support are my family. My parents, Ray and Marsha Starsman, gave me an undying love of education that inspired me to attempt this effort. My mother- and father-in-law, Ed and Jean Keller, have been absolutely wonderful, filling in for me when my studies made me otherwise unavailable. I could not have done this without them. Thanks also to my sons, Ray, Mike, and Brian, who had to endure an occasionally preoccupied father. I hope you inherit my love of education. Finally, my deepest and most profound gratitude goes to my wife, Stacey. You have put up with seeing a lot of my back at the computer and yet have always supported me as I pursued this goal. Thank you from the bottom of my heart.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

In the past thirty-five years, fuzzy logic has grown in stature from a curious extension of Boolean logic into a powerful tool capable of solving problems of great complexity. The observation that fuzzy logic provides a mechanism through which some of the subtleties of human thought can be encoded, stored, utilized, and expressed powers the growth of the use of fuzzy logic in applications. Processes requiring human intervention and not yielding to solution through conventional techniques or even artificial intelligence methods are prime candidates for the use of fuzzy systems.

This work provides an analysis of a class of single-input fuzzy systems and develops several techniques applying this analysis to several nonlinear modeling problems. The extension of this analysis to multiple input dimensions is investigated, but several challenges remain.

1.1 Fuzzy Logic

Fuzzy logic was first formalized in [58] as a superset of Boolean logic capable of resolving the logical paradoxes stymieing traditional logic approaches. Control systems were among the first fields to accept fuzzy logic as a tool to be used in solving engineering problems. One of the first applications of fuzzy logic to control problems was discussed in [32] and provided the fertile ground for most future fuzzy control work.

The initial approach for fuzzy control system design focused on the ease of mapping linguistic commands to a fuzzy system [32]. Linguistic designs are the result of encoding a human expert's knowledge into a fuzzy system and were used to solve several difficult industrial control problems. This technique relies upon the ability of the expert to express his or her knowledge of a system's dynamics in terms of fuzzy sets. While powerful, this technique has two key weaknesses: too many design variables and limited analysis tools. A hallmark of fuzzy systems is the flexibility in their representation. However, this flexibility yields a tremendous number of system variables that must be adjusted to tune performance. Manual tuning of even simple fuzzy systems can be a difficult and time-consuming process. Furthermore, there exists no comprehensive analysis regarding the stability and performance of fuzzy control systems. The only available results consider specific cases.

Numerous techniques have been developed in response to the problems with manual tuning. The Fuzzy Model Reference Learning Controller (FMRLC) [39] is one of several complete design procedures. Many other techniques make use of gradient descent [49], neural network approaches [29, 52], genetic algorithms [17], and other procedures [cf. 38] to automatically tune a fuzzy controller. The development of techniques for the analysis of fuzzy control systems is also vital to continued progress. The stability of several classes of fuzzy adaptive control systems is discussed in [36, 39, 56].

Fuzzy logic has been used not only in the direct design of controllers but also in indirect designs. In the latter case, fuzzy logic is first used for approximation and identification of the nonlinear systems. This is possible since fuzzy systems have been proven to be universal function approximators [26, 44, 55] which together with a dynamical component (such as time delays and recursion) can approximate dynamical systems. Radial basis functions [8], clustering [27], and others [12] are some of the techniques that have been applied to the dynamic system identification problem.

1.2 Fuzzy System Design Issues

While the use of fuzzy logic in system identification is well established, there remain a large number of unresolved issues. The goal of this work is to investigate a specific class of fuzzy systems and to develop a technique to determine the fuzzy system parameters based on a given error criteria. This class of fuzzy systems is useful for the solution of many problems and it contains only triangular membership functions that overlap perfectly (meaning that the apex of triangle *n* occurs at the same point as the base of triangles *n-1* and *n+1*). This class also uses a fuzzy singleton output set and centroidal defuzzification.

The fuzzy logic system designer is faced with a series of design decisions for which very little guidance is available. Decisions regarding the very nature of a fuzzy system are left to the whim of the designer. While the ambiguity of the structure of a fuzzy system was part of its early charm, engineers serious about the application of fuzzy logic to critical control problems require deeper understanding of the structural issue of fuzzy systems. A byproduct of this work will be a more complete exploration of a class of fuzzy systems that may then be used in an engineering environment backed by analysis.

By no means does this work answer all the questions associated with this single class of fuzzy systems. Rather, several basic questions are answered definitively and several paths are laid out for future work that will resolve some of the unanswered issues.

1.3 Piecewise-Linear Approximation

It is shown in this work that the specific fuzzy system described above, is equivalent to a Piecewise Linear Approximator (PLA). This equivalence makes it possible to use available results to identify a system with a PLA and then to map the PLA into a fuzzy system. The central issue in solving the identification problem is closely related to optimal knot placement for the PLA. The solution to the hitherto unsolved PLA optimal knot placement problem is the key contribution of this work. Several algorithms are developed that take advantage of this result and several basic fuzzy logic and control problems are investigated with this method.

Besides fuzzy systems, piecewise linear constructs are used in a number of fields and may benefit from this work.

1.4 Application to Engineering

Piecewise-linear analysis and approximators have found wide application within engineering. Piecewise linear constructions have the advantage of being able to model any continuous system as well as a formulation that is relatively easy to understand. Of particular note is the use of these techniques within the fields of circuit theory, nonlinear control, and nonlinear systems identification.

Piecewise-linear analysis has been widely applied to the modeling of nonlinear characteristics of electronic devices and to study a large class of nonlinear resistive networks [9-11, 23-25, 28]. PLAs are used to express the circuits as a piecewise-linear system with linear boundaries and regions expressed as equations. A great deal of research has been conducted in this line of work resulting in a formal structure with an error-function similar to the one used in the nonlinear optimization problem in this work [9, 10, 11, 23, 24].

Piecewise linear models are simple to use in the modeling of nonlinear circuits because the linear regions and boundaries can be directly determined by experiment. Otherwise, if the linear regions and boundaries are not known explicitly, it would not be as simple to use them. In this case, input-output data is the only information available and nonlinear system identification can be used. PLAs have also been used in nonlinear system identification of a large class of systems [6, 33, 37, 48, 53]. The solution of the knots placement problem to be presented in this work, makes it also possible to use PLAs in nonlinear system identification.

The use of PLAs in nonlinear system identification is especially attractive when the nonlinear system consists of linear subsystems interconnected with static nonlinearities. Van Pelt and Bernstein [53] give a good summary with references to this general class of systems. Two special cases are the Wiener class of systems where the nonlinearity follows the linear subsystem and the Hammerstein class where the nonlinearity is ahead of the linear system. Both of these cases will be considered in this work.

An advantage of modeling nonlinear systems with PLAs is that there exist analysis results. Sontag in [50] analyzes the regulation problem of nonlinear systems modeled with PLAs. In [40] Petit compiles other known analytical results for closed-loop systems where the nonlinearities are modeled with PLAs. Since the equivalence between a class of fuzzy systems and PLAs is shown in this work, it will now be possible to analyze closed-loop systems that contain this type of fuzzy system.

1.5 Problem Formulation

A key product of this work is the development of an identification technique for determining an optimal fuzzy model for static nonlinearities in Wiener and Hammerstein discrete-time systems. Constraints on the structure of the fuzzy system will be described and implementation will be discussed. The application of this technique to model higher-order systems will also be addressed.

1.6 Example Systems

Throughout this work, example systems will be used to enhance the discussion of various topics. Three examples are used to help cover the wide array of topics discussed within. The first example is a purely heuristic one primarily helpful in the development of fuzzy logic techniques. The other two examples are nonlinear systems useful in the discussion of the application of the techniques developed in this work in nonlinear systems identification.

1.6.1 Example 1

Consider the steering mechanism on a ship shown in Figure 1 where angles and angular rates increase in the clockwise direction. When the ship's wheel (similar to the automotive steering wheel) is turned to the left, the ship's rudder is turned a

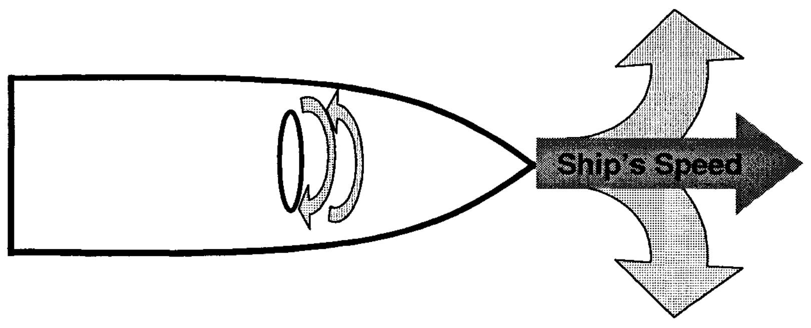

Figure 1 – Example fuzzy system.

corresponding amount to the left and, the turn rate of the ship decreases until a steady state turn rate is achieved based upon the angle of the rudder and the speed of the ship. When the wheel is turned to the right, the rudder is turned to the right and the ship's turn rate increases until a new turn rate is achieved based upon the angle of the rudder and the speed of the ship. In this case, the inputs to this system are the amount the ship's wheel is turned and the speed of the ship. The goal is to model the behavior of the ship in response to the input.

1.6.2 Example 2

Throughout this work, the simple discrete-time nonlinear system shown in Figure 2 is used as a sample system to illustrate the setup of a fuzzy system identifier and determine the parameters by solving a nonlinear optimization problem.

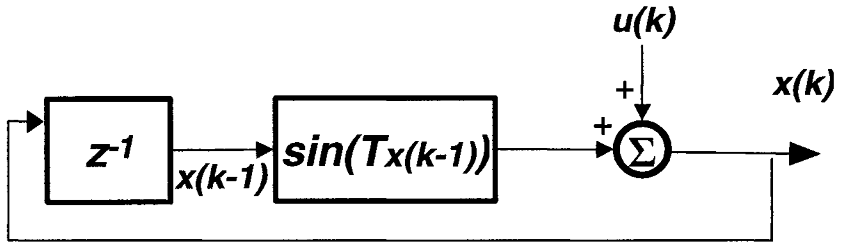

Figure 2 – Sample Wiener nonlinear system.

This system is described by the following equation:

$$x(k)=\sin(Tx(k-1)) + u(k) \tag{1}$$

where T is the time interval between samples, $x(k)$ is the value of x at the k^{th} sample, and $u(k)$ is the value of the input u at the k^{th} sample. It is a nonlinear example with a single step delay for which the one-dimensional solution is completely presented in this work. This example corresponds to the Wiener class of systems.

While this example uses a continuous nonlinearity, the next example system will introduce a piecewise nonlinearity.

1.6.3 Example 3

A third sample system is introduced for the purposes of comparing it to some of the results discussed in [53]. A representation of general nonlinear systems with a generalized Hammerstein model is shown in Figure 3. The generalization is introduced when the feedback static nonlinearity h_0 is not zero.

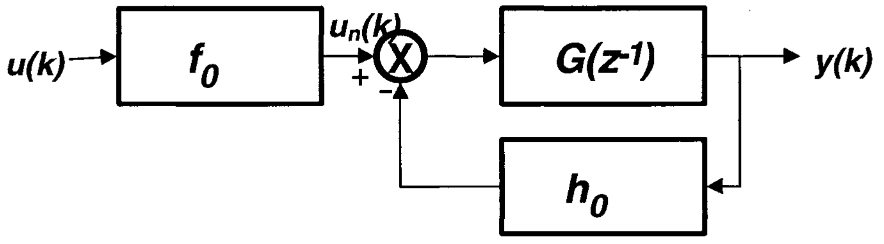

Figure 3 – Sample Hammerstein nonlinear feedback model.

The example used in this work corresponds to the first example discussed in [53] and is a Hammerstein system where

$$G(z^{-1}) = \frac{0.5992 + 0.5679z^{-1}}{1 - 1.706z^{-1} + 0.8521z^{-2}}, \tag{2}$$

$$f_0(u) = \begin{cases} u + 0.25 & u \leq -0.25 \\ 0 & -0.25 < u < 0.25 \\ u - 0.25 & u \geq 0.25 \end{cases}, \text{ and} \tag{3}$$

$$h_0 = 0. \tag{4}$$

1.7 Overview

This work proceeds by first laying the foundation for Fuzzy Logic in Chapter 2 and Approximation in Chapter 3. Chapter 4 proves that an optimal piecewise linear approximator to a given set of input/output data exists and that a fuzzy system can be developed such that it behaves precisely as the piecewise linear approximator. Chapter 5 uses the results in Chapter 4 to demonstrate the ability to optimally map a fuzzy system to a given set of data points. A sub-optimal algorithm is developed to overcome some of the computational intensity of the optimal mapping algorithm. This sub-optimal algorithm is applied to several different problems, most notably the approximation of a Wiener and Hammerstein nonlinear system. Future issues and additional research are outlined in Chapter 6.

2.0 INTRODUCTION TO FUZZY LOGIC

Fuzzy (multi-valued) logic is an extension of the Boolean (two-valued) logic that can directly take into account the uncertainty present in many real world logical decisions [58], handling the subtleties and paradoxes that stymie conventional logic.

2.1 Basic Terms and Concepts

Only those fuzzy logic terms and concepts used in support of this work are described in this chapter. A more complete definition of fuzzy logic is available, for example, in [58] and [39].

The term 'universe of discourse' refers to the range of values assumed by any single property or state of a system.

2.1.1 Membership Function

A membership function (MF) is a mapping from universe of discourse in $\Re$ to a number in the interval [0,1] representing the membership of the input value in the fuzzy output space. Typical fuzzy membership functions include the Gaussian, a triangle, and a trapezoid. The only limitation placed on the MFs is that their output be restricted to [0, 1]. Figure 4 is an example of five typical MFs.

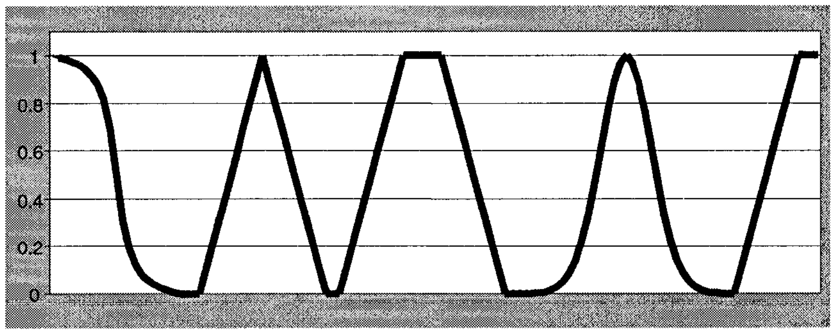

Figure 4 – Sample membership functions.

The degree to which a membership function responds to an input is referred to as it's activation level. For example, the activation function of a triangular membership function with it's left corner at d_1, it's center at d_2 and it's right corner at d_3 could be described as:

$$a(x) = \begin{cases} \dfrac{x-d_1}{d_2-d_1} & : \quad d_1 < x \leq d_2 \\ \dfrac{-x+d_3}{d_3-d_2} & : \quad d_2 < x < d_3 \\ 0 & : \quad elsewhere \end{cases}$$

A special type of MF used in some applications is the fuzzy singleton [39]. A fuzzy singleton at d_1 is defined as:

$$a(x) = \begin{cases} 1 : x = d_1 \\ 0 : \text{otherwise} \end{cases}$$

The fuzzy singleton resembles the discrete-time delta function.

2.1.2 Fuzzy Set

A fuzzy set is a collection of membership functions related to a single property defined over the same universe of discourse. In Example 1, three variables are important in steering operations: wheel position, ship's speed, and turn rate. Consequently, MFs would be defined for these three variables. To describe the position of the ship's wheel, seven MFs could be selected to be {*Hard Left, Left Full, Left Standard, Amidships, Right Standard, Right Full, Hard Right*}. The ship's speed in the forward direction can be described for steering purposes with only four MFs: {Stopped, Slow, Average, Fast}. The MFs defined for each variable form a fuzzy set. For convenience of notation, the activation levels of a fuzzy set are often written as a vector. For example, the ship's wheel fuzzy set might be written as {0, 0, 0, 0.4, 0.6, 0, 0}. Note that more than one member of a fuzzy set can be activated simultaneously, a hallmark of fuzzy logic.

A fuzzy set representing the ship's wheel input is shown in Figure 5.

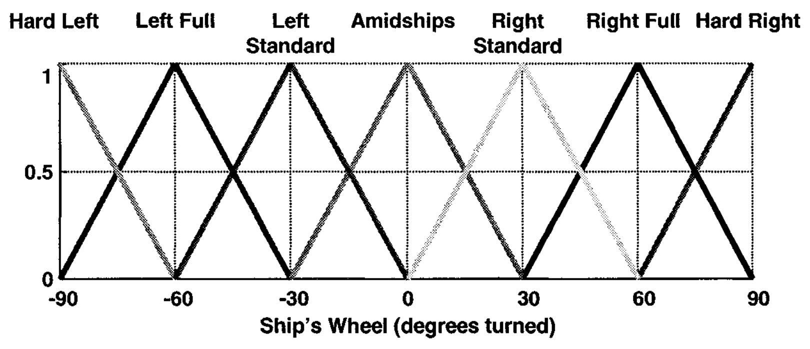

Figure 5 – Membership functions for the position of ship's wheel.

This set consists of 7 membership functions. The value of this set where the ship's wheel is turned to -15 degrees is expressed in vector form is {0, 0, 0.5, 0.5, 0, 0, 0} as only the 3rd and 4th MF are activated, each to a level of 0.5. The real advantage of fuzzy logic is that a linguistic expression of a concept can be readily mapped to a quantitative representation. In the U. S. Navy, the command from the Conning Officer to the Helmsman to turn the ship to the right would be "Right Standard Rudder". As shown in the figure above, this command has a fuzzy logic representation and can be understood as the fuzzy membership function.

2.1.3 Fuzzy Operations

Fuzzy logic operations are supersets of their Boolean counterparts. The most fundamental and commonly used are the AND, the OR, and the NOT functions. The fuzzy AND and fuzzy OR functions are often referred to as fuzzy conjunctions as they combine the inputs of multiple fuzzy variables. Whereas these operations are explicitly and uniquely defined in a Boolean logic environment, there are no such limitations in fuzzy logic.
Two definitions for the fuzzy AND operator are

- $F_1(x_1)$ AND $F_2(x_2)$ AND ...AND $F_n(x_n) \equiv \min(F_1(x_1), F_2(x_2), \ldots, F_n(x_n))$
- $F_1(x_1)$ AND $F_2(x_2)$ AND ...AND $F_n(x_n) \equiv F_1(x_1) \bullet F_2(x_2) \bullet \ldots \bullet F_n(x_n)$.

Two definitions for the fuzzy OR operator are

- $F_1(x_1)$ OR $F_2(x_2)$ OR ...OR $F_n(x_n) \equiv \max(F_1(x_1), F_2(x_2), \ldots, F_n(x_n))$
- $F_1(x_1)$ OR $F_2(x_2)$ OR ...OR $F_n(x_n) \equiv 1-(1- F_1(x_1))\bullet(1- F_2(x_2)) \bullet\ldots\bullet(1- F_n(x_n))$.

The fuzzy NOT is a unary operator usually defined as

- NOT$(F_1(x_1)) \equiv 1- F_1(x_1)$.

Note that if the fuzzy variables $F_n(x_n)$ were limited to the two logical values of 0 and 1 (and hence converted into a Boolean representation), the above operations would all collapse to their Boolean counterparts.

2.1.4 Fuzzy Decisions

A fuzzy decision is a fuzzy IF-THEN conclusion drawn on one or more fuzzy sets. A typical fuzzy decision for the ship's steering system is:

IF *Ship's Wheel* **IS** *Left Standard* **AND** *Speed* **IS** *Fast* **THEN** *Turn Rate* **IS** *Large Negative.*

The arguments of the fuzzy conjunction are evaluated and the output membership function is activated to this level. In the ship steering example, consider the case where the fuzzy membership function for *Left Standard* in the *Ship's Wheel* fuzzy set is activated to a degree of 0.6 and the *Fast* membership function in the *Speed* fuzzy set is activated to a degree of 0.8. The fuzzy IF-THEN rule above would yield the *Large Negative* membership function of the *Turn Rate* fuzzy set being activated to a degree equal to 0.8 **AND** 0.6. In the case where a multiplicative fuzzy AND is being used the *Negative* membership function would be activated to a degree of 0.48.

2.1.5 Rule Base

The rule base is the set of fuzzy decisions that specify the fuzzy output(s) for all relevant combinations of fuzzy sets from the input variables. Each rule is in the form of a fuzzy decision as described in Section 2.1.5. While the IF-THEN rules could be simply listed, a more convenient and readable notation is to build a table, an example of which is shown in Table 1 below.

Table 1 - Fuzzy rule base for Example 1

		Ship's Wheel						
		Hard Left	Left Full	Left Standard	Amidships	Right Standard	Right Full	Hard Right
Speed	Stopped	Zero	Zero	Zero	Zero	Zero	Zero	Zero
	Slow	Negative	Negative	Slight Negative	Zero	Slight Positive	Positive	Positive
	Average	Large Negative	Negative	Negative	Zero	Positive	Positive	Large Positive
	Fast	Large Negative	Large Negative	Negative	Zero	Positive	Large Positive	Large Positive

This table shows the ship's steering example system with two input states (*Ship's Wheel* and *Speed*). Each possible conjunction of the two input fuzzy sets on the table represents a separate IF-THEN rule, the conclusion of which is the activation of the fuzzy output variable specified at the intersection.

2.1.6 Defuzzification

Membership functions are used to convert real-valued inputs into fuzzy signals; Defuzzification reverses the process. Like the rest of fuzzy logic, there are many implementations of the defuzzification process. This work focuses on one of the more common implementations (the centroid) having specific features that will be taken advantage of in Chapters 4 and 5.

Because fuzzy MFs can and do overlap, usually there will be multiple conclusions derived from a single presentation of input data to a fuzzy rule base. For example, at a given instant the fuzzy rule base could produce the following output activations:

Negative activated to 0.2

Slight Negative activated to 0.7

Zero activated to 0.5

Zero activated to 0.1

In order to convert these results from a fuzzy value into a value useful to a conventional control system we must apply the defuzzification process.

The defuzzification process used in this work uses the centroid calculation in order to find a weighted average of the activated output MFs. The centroid of the aggregated weighted output MFs is used as the defuzzified output. The equation for centroid calculation is:

$$y = \frac{\sum_{k=1}^{M} c_k A_k}{\sum_{k=1}^{M} A_k}$$

where M is the number of activated output MFs, c_k is the centroid of the k^{th} output MF, and A_k is the area of the k^{th} output MF.

As noted in Section 2.1.1, the fuzzy singleton is a special type of MF with its entire weight existing at a single point (c_k). By definition the area under a singleton MF is equal to the level it is activated. This means that when fuzzy singleton k is activated to strength a_k it's area is equal to a_k. Therefore, when fuzzy singletons are used as the output MFs, the centroid calculation equation becomes:

$$y = \frac{\sum_{k=1}^{M} c_k a_k}{\sum_{k=1}^{M} a_k} \tag{5}$$

where a_k is the activation level of the k^{th} output MF.

2.1.7 Wiener System Example

For illustration consider Example 2, the Wiener nonlinear system. If the linear subsystem is known then there are only two variables of interest: the input and output of the nonlinearity. A universe of discourse is selected for each input variable as well as the size and shape of the MFs. In this example, this design is done by hand. The MFs selected for the input, *x(k-1)*, are shown in Figure 6. In this case, the assignment is made without consideration for the behavior of the system. In any real application, the shape, number, and size of these MFs occupy a great deal of the designer's time and is the focus of much of this research.

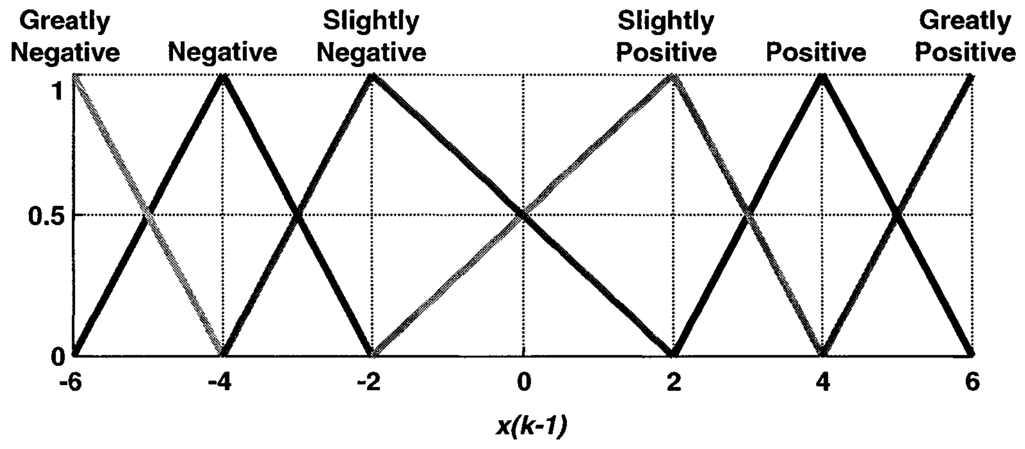

Figure 6 – Membership functions for sample system.

If a conventional output fuzzy set were being used to represent the output state, a similar set of output MFs would also be developed. In this work fuzzy singletons will be used. The fuzzy singleton set can be simply described by the centers of the singletons: {-0.9, -0.75, -0.3, 0.3, 0.75, 0.9}.

The next step is to develop a rule base that will appropriately model the system. Using the *F(•)* operator to denote the fuzzification of the state. A typical rule might be:

IF *F(x(k-1))* IS *Slightly Negative* THEN *F(x(k))* IS *–0.1*

The –0.1 term in the above rule is the fuzzy singleton representation of the model output state. Because this is a single input state system, the rule base will consist of a rule for each of the MFs in the input fuzzy set, 6 rules in this example. Table 2 represents the six rules in tabular form as discussed above.

Table 2 - Fuzzy rule base for sample problem

x(k-1)					
Greatly Negative	Negative	Slightly Negative	Slightly Positive	Positive	Greatly Positive
0.3	-0.75	-0.9	0.9	0.75	-0.3

The final step in this example is to take the fuzzy singletons activated by the rule base and defuzzify them into a numeric output. The centroidal defuzzification given by (5) is used for this example.

Having laid out the fuzzy model for the example, the system can be simulated and the results compared with actual system. Consider the case where *x(k-1)*=0. At this point rules 3 and 4 are activated each at strength 0.5. The system is defuzzified by applying equation 5:

$$y = \frac{-2 \cdot 0.5 + 2 \cdot 0.5}{0.5 + 0.5} = 0$$

Consider another case where *x(k-1)*=2. At this point only rule 4 is activated and at a strength of 1 yielding an output result of 1.0. The results of this approximation for –6, -4, ... 6 are shown in Table 3 where *x(k-1)*is the input signal, *x(k)*is the output of the actual system, *xf(k)*is the output of the fuzzy system, and *e(k)* is the 1-norm error between the actual system and the approximated system.

Table 3 - Results of fuzzy approximation of sample system

x(k-1)	x(k)	xf(k)	e(k)
-6	0.2794	0.3	0.0206
-4	-0.7568	-0.75	0.0068
-2	-0.9093	-0.9	0.058529
0	0.0	0.0	0.0
2	.9093	0.9	0.0993
4	0.7568	0.75	0.0068
6	-0.2794	-0.3	0.0206

These results demonstrate the ability of a simple fuzzy system to approximate a nonlinear system to a reasonable accuracy using simple heuristic parameter assignments.

2.2 Fuzzy System

The term fuzzy system refers to any system that processes traditional input(s) through input fuzzy set(s), passes that result through a fuzzy rules set, and produces a traditional output with a defuzzifying mechanism. An n^{th} dimension fuzzy system refers to a fuzzy system with n input dimensions. A block diagram of a typical fuzzy system is shown in Figure 7.

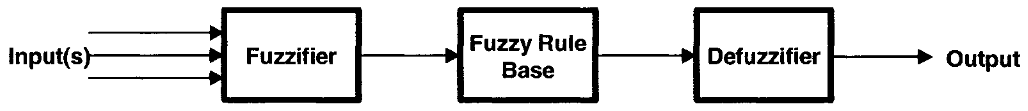

Figure 7 – Typical fuzzy system.

2.3 Application to Control

Fuzzy logic has been extensively applied in control systems [56, 32, 39, 21, and 36]. The attractiveness stems from the ability to describe a system's operation and control in heuristic terms that bridge the gap between system experts and control system designer. The use of fuzzy logic greatly expands the number of tools in the control designer's toolbox and provides new solutions to many of the difficult challenges posed to the modern control engineer.

However, difficulties arise in applying fuzzy logic to a complex control problem. Numerous parameters must be set to complete a mapping between a fuzzy system and a real system. These parameters include the input membership function type and shape parameters, the output membership function type and shape parameters, and the relationship between fuzzy inputs and fuzzy outputs.

The determination of those parameters has traditionally been accomplished manually by tuning a fuzzy system's parameters by hand to meet design specifications. However, this is a time-consuming and frustrating task for systems of any significant complexity.

2.4 Open Issues

There are two pressing problems facing anyone designing a fuzzy control system. Firstly, the fuzzy sets that represent the system inputs must be designed. For low-order linear systems this is not too difficult and can be accomplished manually. However, for higher order nonlinear systems this becomes a tremendous challenge. Not only must the designer select an appropriate number of MFs for each input, but each MF must be shaped to properly represent the system and interact with the other fuzzy sets.

Next, the rule base must be developed which interconnects all of the members of the fuzzy sets and produces the ultimate system output. The number of rules activated is generally equal to:

$$\prod_{n=1}^{N} M_n$$

where N is the number of system inputs and M_n is the number of membership functions associated with input n. Clearly, this leads to an enormous number of rules for even a moderately complex system requiring some sort of automated and/or adaptive scheme.

2.5 Application as an Approximator

Fuzzy systems are capable of modeling the behavior of a nonlinear system as demonstrated in the example in Section 2.1.7. This similarity between fuzzy systems and mathematical approximation techniques is worthy of a more complete examination as application of approximation techniques may help overcome some of the difficulties of designing fuzzy systems. Before pursuing this similarity further, it is important to lay out the fundamental theories of approximation.

3.0 INTRODUCTION TO FUNCTION APPROXIMATION

For the purposes of this work, function approximation is defined as the modeling of a function known only through a set of data points. The model is selected from a given class of function approximations based on the minimization of some error criteria or, at least, on the reduction of the error to a given tolerance.

Function approximation techniques are used in many different applications including:

- Extrapolation,
- Data reduction,
- Compression, and
- Function estimation [7, 46, and 14].

Approximation theory is the foundation upon which a good portion of this work is built. This section provides a brief introduction to this material.

3.1.1 Function Approximation

A function or system approximation problem contains three primary elements: (1) the function, system, or data to be approximated (*f*), (2) the set of approximation functions or systems (*A*), and (3) a measure of performance to select an approximation from *A* [42].

In this work, only functions that are continuous on an interval on the real line, $\Re$, I=[*a, b*] are considered. This set of functions is denoted by *C[a, b]* with norm

$$\|f\|_\infty = \sup_{x \in I} |f(x)|.$$

3.1.2 Polynomial Approximation

The set of polynomials is often used to approximate functions. A generalized polynomial of the form:

$$P(x) = \sum_{n=0}^{N} a_n g_n(x) \qquad (6)$$

where $g_n(x) \in C[a, b]$, $a_n \in \Re$ and N is the number of functions forming the generalized polynomial [46]. A simple polynomial is a subset of the generalized polynomial given in (6) where $g_n(x)=x^n$.

While generalized polynomials are useful in many applications, simple polynomials are often used "because they can be evaluated, differentiated, and integrated easily and in finitely many steps using just the basic arithmetic operations of addition, subtraction and multiplication" [13] and can approximate continuous functions within a finite error bound.

Theorem 1 (Weierstrass Approximation Theorem) [42]

> *Let f be $\in C[a, b]$. To each $\varepsilon > 0$ there corresponds a polynomial P such that $\| f - P \|_\infty < \varepsilon$. Thus $| f(x) - P(x) | < \varepsilon$ for all $x \in [a, b]$.*

The Weierstrass Approximation Theorem shows that there exists a polynomial that can approximate any continuous function to an arbitrary accuracy. In fact, a simple polynomial of degree N or greater can approximate a function with N noise-free samples exactly at each sample.

Theorem 2 (Uniqueness Theorem) [42]

> *Let A be a subspace of a normed linear space $\Re$ that is also a convex set. Then, for all $f \in \Re$, there is at most one best approximation from A to f.*

The Uniqueness Theorem shows that, under certain circumstances, the best approximation is unique. As will be shown in the next chapter, the conditions of the Uniqueness Theorem are not met by the approximations used in this work.

However, there are several significant problems with polynomial approximation to be discussed in the following subsections.

- A local perturbation in the function to be approximated can affect the global quality of the approximation [13]. In other words a change in behavior of the function isolated between two knots changes the resulting approximation over the entire range.
- Increasing the degree of the approximation function can decrease the accuracy of the approximation in the regions between the provided fit points [7]. This phenomenon is known as overfitting.

3.1.2.1 Local Perturbation Effects

Because all of the parameters of an approximating polynomial are determined by the entire set of data points, the addition of a data point for consideration in the approximation can have dramatic consequences on the entire polynomial approximation. The example in Figure 8 shows the function to be approximated in the solid line with the

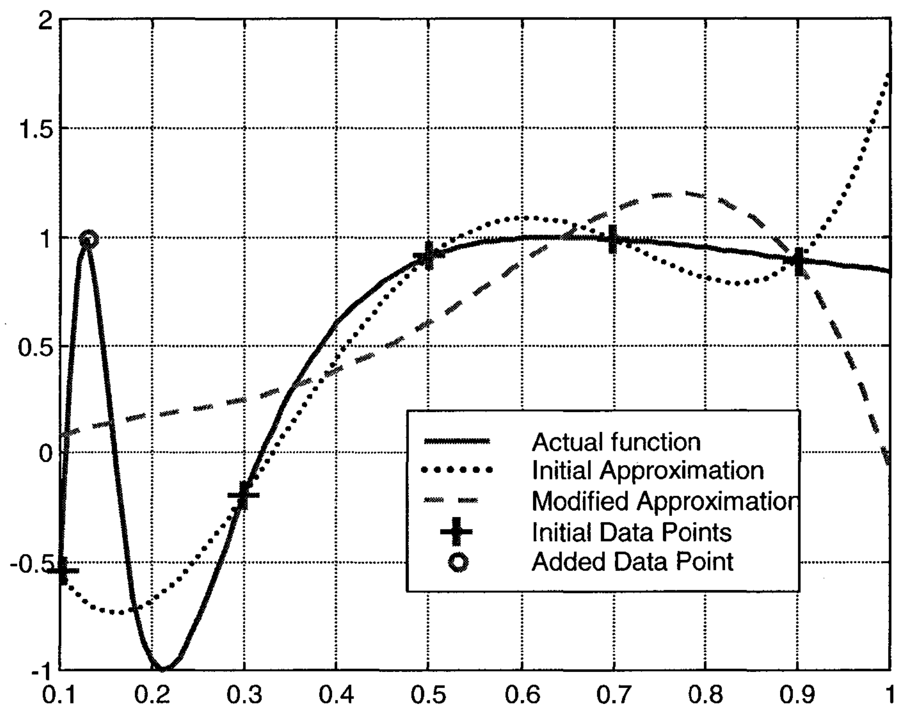

Figure 8 – Global effects of local perturbations on polynomial approximations

selected data points as red +s. The 4th degree polynomial approximation of the function calculated from the initial data points is shown as a dotted blue line. A data point is added to the initial set and is shown as a magenta circle. The approximation is recalculated to take this new point into account and the result is shown in the cyan dashed line. As this figure clearly illustrates, the entire approximation has been drastically changed by the addition of a single point.

3.1.2.2 Overfitting

An example of the overfitting problem is shown in Figure 9. This plot shows the root mean square error of the approximation of 0th through 9th degree polynomial to a sine curve from 0 to 3 perturbed by zero-mean Gaussian noise with a standard deviation of 0.05. The polynomials were fit to ten evenly spaced points and the error was calculated for each polynomial degree against the actual function.

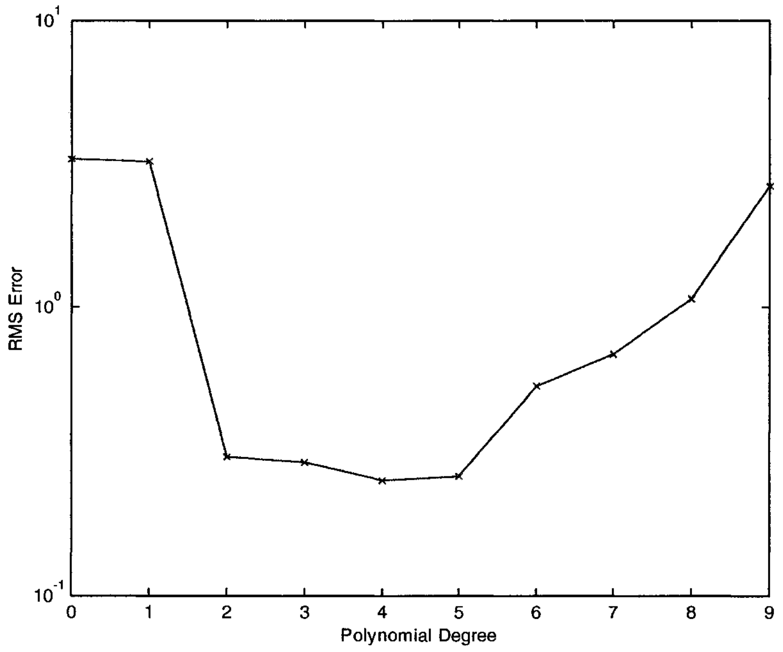

Figure 9 – Simple polynomial approximation error.

From Figure 9, it is clear that the approximation error initially decreases as the polynomial degree is increased. However, at some point the approximation error increases as the polynomial begins to over fit the data and lose the general sense of the curve.

3.1.3 Piecewise Polynomial Approximation

The piecewise polynomial representation of an approximator was developed the susceptibility of the polynomial approximator to overfitting and local perturbations [7]. Rather than fitting a single polynomial to the entire set of data, the approximation interval is broken into $N-1$ subintervals and a low-degree polynomial is fit to each sub-interval. The N breaks between polynomials are known in approximation parlance as *knots*.

Because the approximation is broken into intervals, the effect of perturbations in the input data is limited to intervals that are near the perturbations. Intervals that are distant from the perturbation remain unaffected. The overfitting problem is eased because the order of the polynomial does not have to be increased to account for higher order curves. Each interval is approximated with a relatively low-degree polynomial and global complexity is accounted for by increasing the number of regions.

Define D as a sequence of increasing points in I:

$$D=\{d_0 \ldots d_N : a=d_0<d_1 \ldots <d_N =b\}.$$

D partitions I into N intervals where:

$$I_n=[d_{n-1}, d_n),\ n=1, \ldots, N.$$

Let $p_{r,n}(x)$ denote a polynomial function of degree r on D where $d_{n-1}\leq x<d_n$ and is zero everywhere else and where $p_{r,n-1}(d_{n-1})= p_{r,n}(d_{n-1})$. This last condition ensures that the piecewise polynomial to be formed with these functions is continuous.

An r^{th} dimensional piecewise polynomial approximator (PPA) is then defined as

$$P_r(x)=\sum_{n=1}^{N} p_{r,n}(x) \tag{7}$$

Theorem 1 indicates that a polynomial can be used to approximate a set of data points to arbitrary accuracy, with the degree of the polynomial being increased to reduce the error between the approximation and the input data set. In contrast, the bound on the accuracy of a PPA is dependent upon the spacing between the knots. Before proceeding to a characterization of this error, it is necessary to introduce two definitions.

Let h be the maximum spacing between knots

$$h = \max_{n=1,\ldots N} d_n - d_{n-1} \tag{8}$$

Additionally, the modulus of continuity is defined for a function *f(x)* with domain M and range P where M and P are metric spaces with distances ρ_0 and ρ respectively. The modulus of continuity is defined as:

$$\omega_f(\delta) = \sup_{\rho_0(x_1,x_2)\le\delta} \rho(f(x_1,x_2)) \tag{9}$$

where x_1 and $x_2 \in M$. [47]

Theorem 3 [42]

The least maximum error between a function f $\in$ *C[a,b] and its piecewise polynomial approximation* $P_r(x)$ *as defined in (7) satisfies the inequality*

$$\min\|f - P_r\|_\infty \le \omega_f\left(\tfrac{1}{2}(r+1)h\right)$$

where $\omega_f(\bullet)$ *is the modulus of continuity of f defined in (9), h is the largest interval as defined in (8) and r is the degree of each piecewise polynomial.*

Thus the error of a PPA approximation with a given degree (r) to a function f with a known or estimated modulus of continuity (ω_f) is bounded by the maximum spacing between knots (h).

3.1.4 Piecewise Linear Approximation

The piecewise linear approximator (PLA) is a piecewise polynomial approximator of degree *1*. This is the simplest PPA as it is a collection of connected line segments. Although the PLA is not smooth and its derivative is not continuous, this type of approximator may be used in cases where a smooth approximation is not required and can be used as a parameter finder for differentiable fuzzy systems.

As a parameter finder, a satisfactory piecewise linear solution will be found. However, rather than placing and sizing triangular fuzzy MFs, differentiable MFs (typically based on exponential functions) could be set based on these parameters. This would yield a smooth fuzzy system based upon a similar solution in the piecewise linear case. Further research is required to address use of parameter finding algorithms derived from PLAs.

The PLA used in this work is

$$p(x) = \begin{cases} m_1 x + b_1 : d_0 \leq x \leq d_1 \\ \vdots \\ m_{N+1} x + b_{N+1} : d_N \leq x \leq d_{N+1} \end{cases} \tag{10}$$

where $d_0 < d_1 < \ldots < d_{N+1}$ are the knots of the piecewise polynomial, m_n is the slope of the n^{th} segment, and b_n is the y-intercept of the n^{th} segment. Theorem 3 applies to this subset and arms us with the knowledge that the PLA is capable of approximating any given data to an arbitrary degree.

While there exist mechanisms to determine the m_n and b_n parameters such that the resultant PLA optimally approximates a given set of input data, there is no known mechanism for determining the location of the knots of the PLA by minimizing a defined error condition [14]. Dierckx presents an algorithm for univariate and bivariate knot placement in [14]. While these algorithms provide a good fit of the curve or surface to data, the solutions are suboptimal and are limited to a maximum of two input variables.

3.2 Application to System Identification

Traditional system identification methods fall into two major categories: parametric identification where certain parameters of an unknown plant are modeled and non-parametric where physical response characteristics of the plant are modeled [20, 5]. For the purposes of this work, non-parametric system identification will be used with a fuzzy system modeling an unknown plant's response characteristics. It is required that a system identifier emulate a given input/output mapping, which is precisely what an approximator does. This type of modeling is also called black-box modeling and is described in detail in [22].

3.3 Open Issues

While approximation is a well-studied topic, there remain several important open areas of research [14]. One of the key issues addressed in this work is the development of theory that addresses optimal breakpoint placement.

3.3.1 Multivariate Adaptive Knot Placement

As many systems in a discrete-time control systems context are multi-input, extensive research and development in the field of multivariate approximation needs to be conducted to support this work. Ultimately, a multivariate adaptive knot placement algorithm must be developed. While Dierckx presents a univariate and bivariate adaptive knot placement algorithm in [14], there are no results for the general multivariate case. Some initial investigation into multivariate knot placement is discussed in Appendix A.

3.3.2 Approximation of Dynamic Systems

A static system is defined as a system where the system output is dependent only upon the current input to the system. A system such as $y=sin(x)$ is a static system.

A dynamic system is dependent upon the system inputs as well as the previous state(s) of the system. The Wiener sample system given by (1) and the Hammerstein sample systems given by (2), (3), and (4) presented in Chapter 1 are examples of dynamic systems. The techniques in this work are especially useful for modeling Wiener and Hammerstein systems where the nonlinearity can be approximated as a static mapping.

4.0 OPTIMAL PARAMETER DETERMINATION OF ONE-DIMENSIONAL FUZZY SYSTEMS

4.1 Introduction

The optimal determination of parameters of a fuzzy system is a nonlinear problem. While a variety of parameter determination mechanisms exist, most do not provide optimal results. Instead they rely on heuristics, gradient descent, genetic algorithms or other techniques that are non-optimal in a nonlinear setting. The goal of this work is to optimally determine the parameters of a one-dimensional fuzzy system. The approach followed to meet this goal is to first establish a relationship between a fuzzy system and a PLA and then prove that the parameters of a PLA can be determined such that the error between the PLA and a set of input/output data points is minimized.

While there are algorithms for determining PLA parameters based upon a data set [14], the approximation error and optimality of the solution remain unexplored. This chapter develops the necessary theory to demonstrate the existence of an optimal solution to the parameter determination problem for PLAs and their extension to one-dimensional fuzzy systems.

As one-dimensional fuzzy systems are the simplest incarnation of fuzzy systems they are used for the initial study of the problem of fuzzy system structure and parameter determination. Besides having a single input variable these systems are simpler than all higher-order fuzzy systems as there is no need for a conjunction operator (a fuzzy AND or OR). These simple systems are investigated in order to develop the concepts necessary to study higher-order fuzzy systems. While the systems are simple, they are not trivial; the development of non-linear optimization techniques in order to solve the approximations is a complex problem that does not have an existing solution.

Of course, the ultimate goal of this research is to provide a general solution to the problem of *N*-dimensional fuzzy system parameter determination. This extension introduces significant complexity to the solution of this problem and is discussed in Appendix A.

4.2 One-Dimensional Fuzzy Systems

For the purposes of this work, a special class of one-dimensional fuzzy systems will be used. They are restricted in the type of membership function, the nature of the MF overlap, and the type of output MF. While the restrictions limit some of the flexibility fuzzy system designers may be accustomed to, a framework is provided that permits a formal analysis, a benefit that more than compensates the loss of flexibility.

The one-dimensional fuzzy system shall be restricted in the following manner:

1. Triangular MFs will be used where the n^{th} MF is given by:

$$a_n(x) = \begin{cases} \dfrac{x-d_{n-1}}{d_n-d_{n-1}} & d_{n-1} < x \le d_n \\ \dfrac{-x+d_{n+1}}{d_{n+1}-d_n} & d_n < x < d_{n+1} \\ 0 & elsewhere \end{cases} \tag{11}$$

   where d_{n-1}, d_n, d_{n+1} are the n-1, n, and n+1 breakpoints; d_0=-∞ and d_{N+1}=∞; and $1 \le n \le N$.

   A graphical representation of a set of arbitrarily selected triangular MFs is shown in Figure 10.

2. The output MF associated with input MF n is a fuzzy singleton.

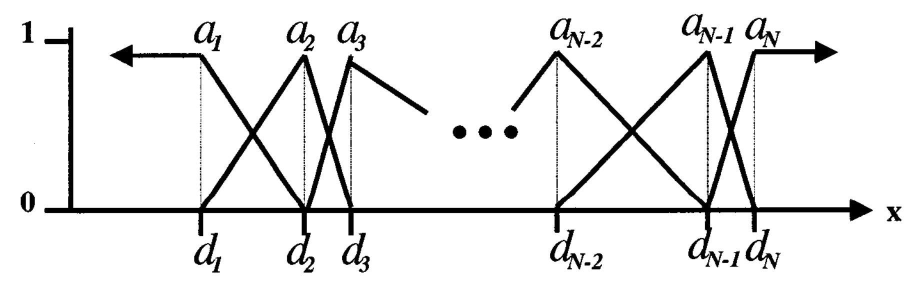

Figure 10 – Triangular membership functions with knots at d_k.

The preceeding definition yields a set of overlapping triangles such that a maximum of two MFs are non-zero at any input value x and that the summed activation at any x is identically equal to one as illustrated in Figure 11 and proven in Lemma 1.

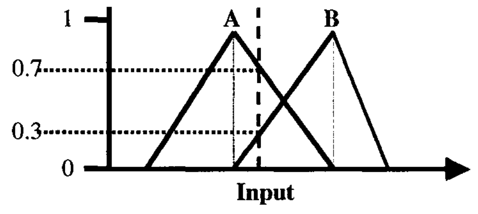

Figure 11 – Triangular membership functions.

Lemma 1.

*In a one-dimensional fuzzy system with N MFs given by (*11*) the sum of the activation strengths of all active MFs for any $d_0 \le x \le d_{N+1}$ is one.*

Proof:

Consider $d_n<x<d_{n+1}$ where $0 < n \le N$. In this interval, only $a_n(x)$ and $a_{n+1}(x)$ are non-zero and equal to:

$$a_n(x) = \frac{(-x + d_{n+1})}{d_{n+1} - d_n}$$

$$a_{n+1}(x) = \frac{(x - d_n)}{d_{n+1} - d_n}$$

Summing the two activation functions active for any x in $d_n < x < d_{n+1}$ gives :

$$a_{n+1}(x) + a_n(x) = \frac{(x - d_n)}{d_{n+1} - d_n} + \frac{(-x + d_{n+1})}{d_{n+1} - d_n}$$

$$a_{n+1}(x) + a_n(x) = 1$$

Now consider where $x=d_n$. At these points, only $a_n(x)$ is non-zero and is equal to:

$$a_n(x) = \frac{-d_n + d_{n+1}}{d_{n+1} - d_n}$$
$$a_n(x) = 1$$

■

From Lemma 1, the denominator in the centroid calculation equation (5) is always equal to one yielding the simplified centroid equation for defuzzification:

$$y(x) = \sum_{n=1}^{N} c_n a_n. \tag{12}$$

4.3 Mapping Between One-Dimensional Fuzzy Systems and the PLA

This work rests upon the foundation of approximation theory and requires that a mapping from a fuzzy system of the form given above to a PLA of the form given in (10) exists.

Theorem 4.

A fuzzy system with MFs given by (11) *and defuzzified by the centroid equation given in* (12) *is representable by the piecewise linear approximator given in* (10).

Proof:

This theorem is proven by demonstrating that for all x: (a) the input/output representation of the fuzzy system consists of line segments between every breakpoint (d_n) and (b) that the system is continuous at each d_n. Equation (12) is simply a linear combination of weighted fuzzy MF activation levels. Therefore, in the n^{th}, $0 \leq n \leq N$, region in the one-dimensional input space $x \in [d_n, d_{n+1}]$, equation (12) can be solved explicitly in the following manner:

$$\begin{aligned} &a_i(x) = 0 \text{ for } i < n \quad or \quad i > n+1 \\ &a_n(x) = \frac{-x + d_{n+1}}{d_{n+1} - d_n} \\ &a_{n+1}(x) = \frac{x - d_n}{d_{n+1} - d_n} \end{aligned} \tag{13}$$

Expanding (12) by substituting the activation functions solved in (13) yields:

$$y(x) = \frac{c_n(-x + d_{n+1})}{d_{n+1} - d_n} + \frac{c_{n+1}(x - d_n)}{d_{n+1} - d_n}.$$

Gathering like terms yields:

$$y(x) = \frac{c_{n+1} - c_n}{d_{n+1} - d_n} x + \frac{c_n d_{n+1} - c_{n+1} d_n}{d_{n+1} - d_n}. \qquad (14)$$

Equation (14) defines an affine mapping from the inputs x is to the outputs y completing the first part of the proof. Continuity at each d_n is demonstrated by testing the *n-1* and the *n* segments around the point $x=d_n$. The equation for the *n-1* segment is derived from (14) and is:

$$y(x) = \frac{c_n - c_{n-1}}{d_n - d_{n-1}} d_k + \frac{c_{n-1} d_n - c_n d_{n-1}}{d_n - d_{n-1}}$$

$$y(x) = \frac{(c_n - c_{n-1}) d_n + c_{n-1} d_n - c_n d_{n-1}}{d_n - d_{n-1}}$$

$$y(x) = \frac{c_n d_n - c_n d_{n-1}}{d_n - d_{n-1}}$$

$$y(x) = c_n$$

Where $x=d_n$ the equation for the n^{th} segment is:

$$y(x) = \frac{c_{n+1} - c_n}{d_{n+1} - d_n} d_n + \frac{c_n d_{n+1} - c_{n+1} d_n}{d_{n+1} - d_n}$$

$$y(x) = \frac{(c_{n+1} - c_n) d_n}{d_{n+1} - d_n} + \frac{c_n d_{n+1} - c_{n+1} d_n}{d_{n+1} - d_n}$$

$$y(x) = \frac{(-c_n) d_n}{d_{n+1} - d_n} + \frac{c_n d_{n+1}}{d_{n+1} - d_n}$$

$$y(x) = c_n$$

Both the linearity of the system between breakpoints (d_ns) and the continuity of the system at the breaks is proven.

In order to prove the uniqueness of this solution, the representation of the PLA to the fuzzy system between any two adjacent breakpoints is examined. The representation of the fuzzy system in (14) from $d_{n-1} \leq x < d_n$ can be made equivalent to the PLA line segment on $d_{n-1} \leq x < d_n$, $y=m_nx+b_n$, by equating the x-coefficient and the constant coefficient parts and solving for c_n and c_{n+1}.

$$c_{n+1} - c_n = (d_{n+1} - d_n)m_n$$

$$c_n d_{n+1} - c_{n+1} d_n = (d_{n+1} - d_n)b_n$$

Since $d_{n-1}<d_n$, there exists a unique solution for c_n and c_{n-1}. ■

Theorem 4 states that a class of one-dimensional fuzzy systems is essentially a continuous piecewise-linear function. This similarity indicates that results discovered for PLAs may be applied to fuzzy systems restricted as discussed earlier.

The goal of the next section is to demonstrate that, given an input/output set of data, a set of PLA parameters can be optimally determined. The solution of the PLA definition provides the position of the breakpoints $d_0 < d_1 < \ldots < d_{N+1}$ as well as the m_ns and b_ns from the equations for the $N+1$ segments given in (10).

The final step in this work will be to show that an arbitrary PLA can be mapped to a fuzzy system. This work is discussed in Chapter 5.

4.4 Optimal Determination of a Piecewise-Linear Approximator from Data

In this section, the PLA shall be shown to be capable of being expressed in a single equation as a sum of shifted and scaled absolute value functions. Decomposing the PLA into simpler forms of one or two parameters will precede the concept of optimally determining the parameters of an N-break PLA. Optimally determining the parameters of these simple components will illuminate the key concepts necessary to solve the general PLA problem.

A PLA is a series of connected line segments that represent some underlying function or set of data. A typical PLA is shown in Figure 12.

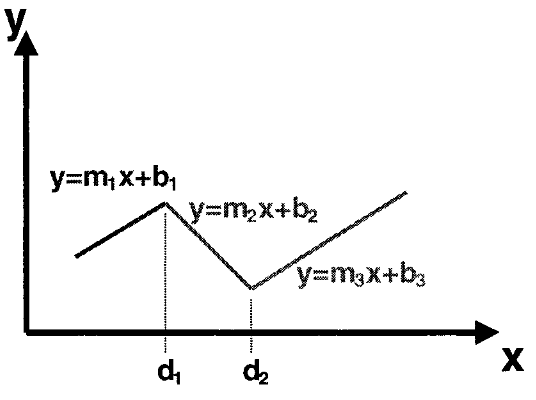

Figure 12 – Typical PLA.

A simple piecewise expression of an N-break PLA is:

$$y(x)=\begin{cases} m_1x+b_1 : x<d_1 \\ m_2x+b_2 : d_1 \le x < d_2 \\ \vdots \\ m_Nx+b_N : d_{N-1} \le x < d_N \\ m_{N+1}x+b_{N+1} : d_N \le x \end{cases} \tag{15}$$

where $d_n < d_{n+1}$ where $1 \le n < N$. In order to satisfy the continuity requirement $y(d_n)=y(d_{n+1})$ where $1 \le n < N$. Evaluating (15) at each breakpoint yields:

$$m_nd_n+b_n=m_{n+1}d_n+b_{n+1}$$

where $1 \le n < N$.

Solving for b_{n+1} yields:

$$b_{n+1}= (m_n - m_{n+1})d_n+b_n$$

Therefore, with known m_i, d_n, and b_1 where $i=1...N+1$, $n=1...N$, the PLA is completely determined.

However, in order to optimally determine the parameters, it is useful to express the PLA in a single equation.

Theorem 5.

A PLA with N breakpoints given by (15) with known m_i, d_n, and b_1, where $i=1...N+1$, $n=1...N$, can always be stated in an equivalent form as:

$$y(x) = a_0 x + c + \sum_{n=1}^{N} a_n |x - d_n| \tag{16}$$

with a suitable choice of a_0, c, $a_1, ..., a_N$, $d_1, ..., d_N$.

Proof.

Select an arbitrary n where $1 \leq n \leq N$. The summation in (16) can be broken into two pieces around n and written as:

$$y(x) = a_0 x + c + \sum_{m=1}^{n} a_m |x - d_m| + \sum_{m=n+1}^{N} a_m |x - d_m| \tag{17}$$

In the interval $d_n \leq x \leq d_{n+1}$, the sign of each $x\text{-}d_m$ term does not change. Wherever $m \leq n$, the term $x\text{-}d_m$ is greater than 0 and wherever $m > n$, the term $x\text{-}d_m$ is less than 0. Therefore, $|x\text{-}d_m| = x\text{-}d_m$ for $m \leq n$ and $|x\text{-}d_m| = -(x\text{-}d_m)$ for $m > n$. Substituting into (17) yields:

$$y(x) = a_0 x + c + \sum_{m=1}^{n} a_m (x - d_m) - \sum_{m=n+1}^{N} a_m (x - d_m) \quad for\ d_n \leq x \leq d_{n+1} \tag{18}$$

In the same interval, (15) reduces to:

$$y(x) = m_{n+1} x + b_{n+1}. \tag{19}$$

Equating the coefficients in this interval of (18) and (19) yields:

$$slope: \quad a_0 + \sum_{k=1}^{n} a_k - \sum_{k=n+1}^{N} a_k = m_{n+1}$$

$$\text{int}ercept: \quad c - \sum_{k=1}^{n} a_k d_k - \sum_{k=n+1}^{N} a_k d_k = b_{n+1}$$

By solving the $N+1$ independent equations resulting from the $N+1$ x-coefficients a_0, a_1, ..., a_N are solved. The nature of the coefficients of the a_k terms guarantees that these equations are independent and that a solution exists. Once the a_k terms have been determined, they can be substituted into the constant-coefficient equations and these $N+1$ independent equations provide a solution for c, d_1, ..., d_N. Again the shifting sign of the d_k terms guarantees that these equations are independent and that a solution exists.

■

Given the equation for an *N*-break PLA shown in (16), it is necessary to determine values for a_n and d_n such that the equation yields a suitable approximation to a set of *M* points $\{(x_1, y_1), \ldots, (x_M, y_M)\}$. By Theorem 1, it has been shown that there exists a solution for a_n and d_n such that the error between the approximant and the data points is minimized for a given set of data. This work does not meet the criteria of Theorem 2 that the metric space be strictly convex as the 1-norm for error functions is used. Therefore, the uniqueness of a solution is not guaranteed by Theorem 2.

The presence of the absolute value term(s) in (16) makes this a nonlinear optimization problem, which is generally difficult to solve. Whereas linear error hyper-surfaces are convex and yield a single minimum over the entire error space, nonlinear error hyper-surfaces may not be convex and may have many multiple local minima as well as one or more absolute minima. There are many techniques for nonlinear optimization such as branch and bound [35, 15], nonlinear least-squares, simulated annealing [30], clustering [12, 27], neural networks [59] and genetic algorithms [16], but they suffer from one or more problems:

- Getting trapped at local minimum and returning sub-optimal results
- Very slow convergence
- Many iterations required imposing a huge computational load
- Initial solution must be close to actual solution

The goal of this work is to determine whether a global optimization is possible for several specific classes of nonlinear functions, to develop analytical optimization techniques that guarantee a solution in a finite number of steps, and to examine several practical aspects of implementing the developed nonlinear optimization solution.

The remainder of this chapter investigates the first of these goals: the examination of the existence of PLA parameters to minimize a given error criteria for a set of input/output data. The classes of nonlinear functions examined are those that make up the PLA described in (16). The central nonlinear function is the scaled and shifted absolute value function $y=a|x-d|$ where *a* is the scaling coefficient and *d* is the shift coefficient. This function is initially broken into the two fundamental nonlinear components and each is investigated separately. These results are then applied to the solution of the scaled and shifted problem. Finally, the N-dimensional PLA problem is addressed.

4.4.1 Optimal Determination of a Shifted Absolute Value Function

A shifted absolute value function is given by:

$$f(x,d) = |x-d|$$

where $d \in \Re$ shifts the break in the function from the origin along the x-axis. Given a set of data $\{(x_1, y_1), \ldots, (x_M, y_M)\}$, an error function is defined:

$$E(d) = \sum_{m=1}^{M} |f(x_m, d) - y_m| = \sum_{m=1}^{M} \left| |x_m - d| - y_m \right| \qquad (20)$$

The goal of the optimization is to find d such that $E(d)$ is minimized for a given set of data $\{(x_1, y_1), \ldots, (x_M, y_M)\}$.

Consider a simple system with 6 evenly spaced points with random amplitudes as shown in Figure 13.

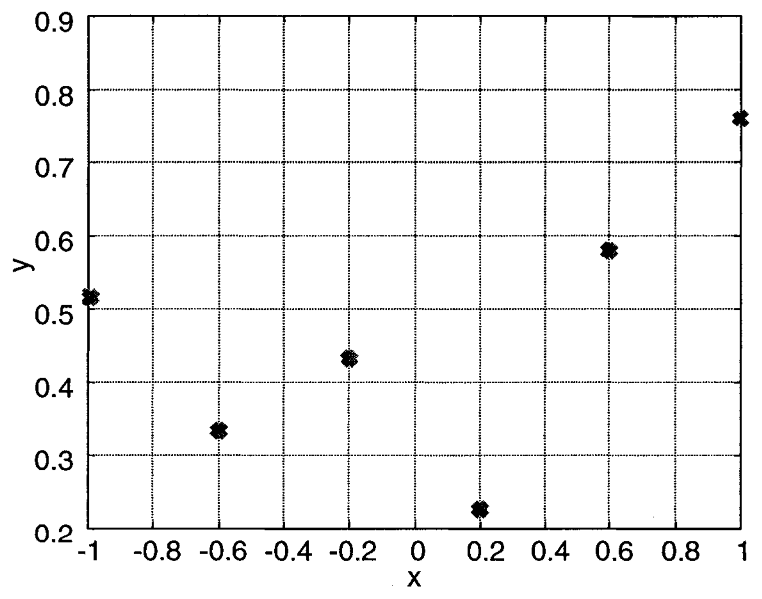

Figure 13 – Sample data system for shifted absolute value function.

The error function given in (20) is calculated with the data points given above and the results are shown in Figure 14.

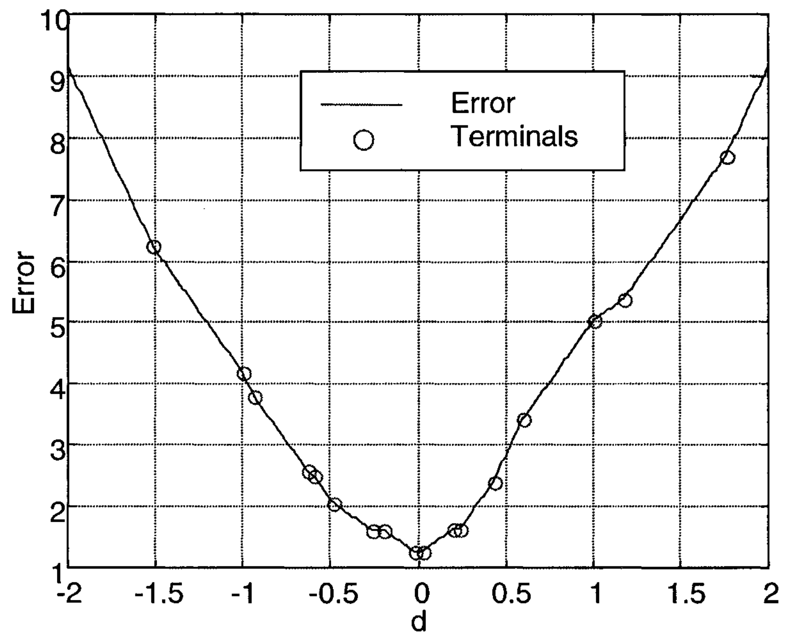

Figure 14 – Error plot of shifted ABS function.

An important observation regarding the error function in (20) is that it is piecewise linear. The terminals between linear sections occur wherever d is a solution to the equations $x_m=d$ or $|x_m-d|=y_m$. Between any two adjacent terminals, $E(d)$ is a line segment and, therefore, one terminal is the minimum for the line segment (if both terminals are equal, then the entire line segment is a minimum).

Theorem 6.

Given a set of finite data points $\{(x_1, y_1), \ldots, (x_M, y_M)\}$, an approximating function $f(x, d)=|x-d|$, and the error function defined in (20), the error is minimized at a point where $d=x_m$ or where d satisfies $|x_m-d|=y_m$ where $m=1\ldots M$.

Proof.

Create a set $A=\{x_1,\ldots, x_M\} \cup \{x_1 - y_1, \ldots, x_M - y_M\} \cup \{ x_1 + y_1, \ldots, x_M + y_M\}$ where the elements of A are the solution candidates for d. Sort A such that $\alpha_n \leq \alpha_{n+1}$ for all $n=1\ldots3M-1$ where α_n is the nth element of A. In the region $\alpha_n<d<\alpha_{n+1}$ neither x_m-d nor $|x_m-d|-y_m$ change sign for any m as sign changes can occur only where $d=\alpha_n$ for all $1\leq n\leq 3M-1$. Hence $||x_m-d|-y_m|$ consists of a single line segment on $\alpha_n<d<\alpha_{n+1}$ for all m. Therefore, $E(d)$ as calculated in (20) is also a line segment on $\alpha_n<d<\alpha_{n+1}$ and one (or both) of the endpoints (α_n, α_{n+1}) is the minimum of this line segment. Thus, the minimum of $E(d)$, where $\alpha_1<d<\alpha_{3M}$, occurs at one of the $3M$ points in A. The interior line segments are all accounted for by the points in A, however the segment endpoints at $d=-\infty$ and $d=\infty$ are not included in A and must be considered as potential minima of the error function. However, when $d=-\infty$ or $d=\infty$, the term $|x_m-d|$ is ∞ for all x_m and hence the term $||x_m-d|-y_m|$ is ∞ for all m. Therefore, $E(\pm\infty)=\infty$. Considering $E(d)$ as a collection of line segments where $\alpha_1<d<\alpha_{3M}$, the minimum of this function is one of the endpoints of the interior segments.

■

This theorem shows that there are a finite number of solutions to the shifted absolute value optimization problem. In fact, the number of potential solutions is upper bounded by $3M$ where M is the number of data points.

The example below demonstrates the power of this theorem and the algorithms that can be developed from it. Consider the function *y=sin(x)+cos(2x)* where *x* is sampled every half-unit from –2 to 2 as shown in Figure 15. This function was selected because it yields

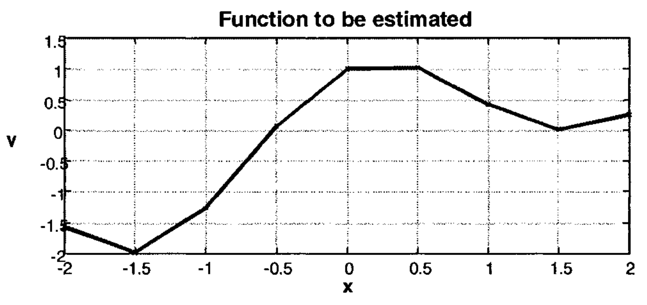

Figure 15 – Sample function.

an error function (20) with a local minimum in addition to the global minimum when approximated with a shifted absolute value function. A function $f(x)$ has a local minimum at x_l if $f(x_l+\alpha)>f(x_l)$ *for* $\alpha \in (-\varepsilon, \varepsilon)$ where ε is a small positive real number. It is only locally minimum if there exists x such that $f(x_l)>f(x)$. Local minima trap gradient descent type algorithms and can fool them into believing they have achieved the global minimum when, in fact, they have been trapped in a local minimum. Most of these algorithms include local minimum discovery mechanisms, but there is never a guarantee that an algorithm has not been trapped.

Evaluating the error function for $-2 \le d \le 2$ at intervals of 0.01 yields the results shown in Figure 16. This is the 'brute-force' method of solving a problem of this type. The

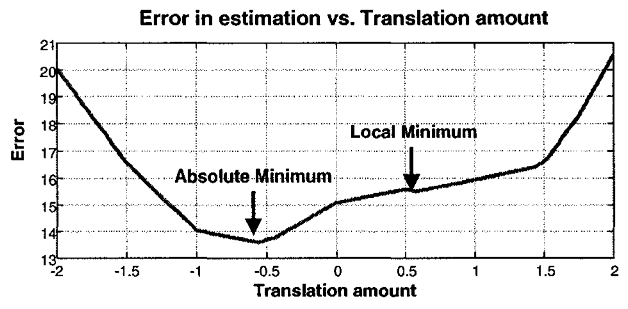

Figure 16 – PLA estimation of sample function.

function has a global minima at $d=-0.56$. However, there is also a local minima at $d=0.58$. While it can be applied reasonably effectively to a simple problem like this one, it quickly fails as the complexity and dimensionality of the problem grows. The results above demonstrate the nonlinear behavior of the function.

The proof of Theorem 6 suggests an algorithm for a single-pass numeric solution to this problem. The method is outlined as follows:

1. Form the set A of solution candidates

   For each input data pair (x_m, y_m):

   - Add an element equal to x_m
   - If $y_m > 0$:
     - Add an element equal to $x_m - y_m$
     - Add an element equal to $x_m + y_m$
2. Use (20) to calculate the error, substituting each element in A for d.
3. The point with the lowest error is the global minima for the shifted absolute value function.

When this algorithm was performed on the data shown in Figure 15, d was found to be –0.5609.

This first optimization algortihm illustrates the method with which the remaining PLA component functions are solved. First, a piecewise-linear representation of the error hypersurface is determined. The corners of each linear region are points, one of which must be the minimum for the piecewise-linear hypersurface. The smallest minimum error for the aggregation of the linear regions identifies the point minimizing the error function.

It has thus been shown that a global minimum exists for the error function given by (20) where an input/output data set is approximated by a shifted absolute value function. The next nonlinear approximator examined is a scaled absolute value.

4.4.2 Optimal Determination of a Scaled Absolute Value Function

A scaled absolute value function is given by:

$$f(x,a) = a|x|$$

where $a \in \Re$ scales the absolute value function. Given a set of data $\{(x_1, y_1), \ldots, (x_M, y_M)\}$, an error function is defined:

$$E(a) = \sum_{m=1}^{M} |f(x_m, a) - y_m| = \sum_{m=1}^{M} \left| a|x_m| - y_m \right| \qquad (21)$$

The goal of the optimization is to find an a such that $E(a)$ is minimized for a given set of data $\{(x_1, y_1), \ldots, (x_M, y_M)\}$.

As in (20) above, the error function in (21) is piecewise linear. The breaks between linear sections occur wherever x_m is 0 or where x_m satisfies the relationship $a|x_m|=y_m$. Between these breaks, $E(a)$ is a line segment and, therefore, one terminal is the minimum for the line segment.

Theorem 7.

Given a set of finite data points $\{(x_1, y_1), \ldots, (x_M, y_M)\}$, an approximating function $f(x, a)=a|x|$, and the error function defined in (21), the error is minimized at a point where $x_m=0$ or where $a|x_m|=y_m$.

Proof.

Create a set $A=\{0\} \cup \{ y_1/x_1, \ldots, y_M/x_M \} \cup \{ -y_1/x_1, \ldots, -y_M/x_M \}$ where the elements of A are solution candidates for a. Sort A such that $\alpha_n \leq \alpha_{n+1}$ for all $n=1\ldots2M+1$ where α_n is the nth element of A. In the region $\alpha_n < a < \alpha_{n+1}$ neither $a|x_m|$ nor $a|x_m|-y_m$ change sign for any m as sign changes can occur only $d=\alpha_n$ for all $1 \leq n \leq 3M-1$. Hence $|\, a|x_m|-y_m \,|$ consists of a single line segment on $\alpha_n<a<\alpha_{n+1}$ for all m. Therefore, $E(a)$ as calculated in (20) is also a line segment on $\alpha_n<a<\alpha_{n+1}$ and one (or both) of the endpoints (α_n, α_{n+1}) is the minimum of this line segment. Thus, the minimum of $E(a)$, where $\alpha_1<a<\alpha_{3M}$, occurs at one of the $2M$ +1 points in $\boldsymbol{A}$. The interior line segments are all accounted for by the points in $\boldsymbol{A}$, however the segment endpoints at $a=-\infty$ and $a=\infty$ are not included in $\boldsymbol{A}$ and must be considered as potential minima of the error function. However, when $a=-\infty$ or $a=\infty$, the term $|a|x_m|-y_m|$ is ∞ for all m. Therefore, $E(\pm\infty)=\infty$. Considering $E(a)$ as a collection of line segments where $\alpha_1<a<\alpha_{3M}$, the minimum of this function is one of the endpoints of the interior segments.

■

The next step in increasing the complexity of the nonlinear approximator is to combine the results of the previous two sections (Theorems 6 and 7) into a single scaled and shifted function.

4.4.3 Optimal Determination of a Scaled and Shifted Absolute Value Function

The previous two examples demonstrated the optimization of a non-linear function with a single variable being optimized. The next step is to examine an aggregate of the shifted and scaled functions and develop a non-linear optimization approach for this function of two variables.

A scaled and shifted absolute value function is given by:

$$f(x,a,d)=a|x-d|$$

where $a\in\Re$ scales the absolute value function and $d\in\Re$ shifts it along the x-axis. Given a set of data $\{(x_1, y_1), \ldots, (x_M, y_M)\}$, an error function is defined:

$$E(a,d)=\sum_{m=1}^{M}|f(x_m,a,d)-y_m|=\sum_{m=1}^{M}\big|a|x_m-d|-y_m\big| \qquad (22)$$

The goal of the optimization is to find an a and d such that $E(a, d)$ is minimized for $\{(x_1, y_1), \ldots, (x_M, y_M)\}$. Unlike the previous examples, this yields a three-dimensional error surface rather than the two-dimensional error curve found in the previous two problems.

As in the previous two examples, the absolute value nonlinearity boundaries are formed at $x_m=d$ and at $a|x_m-d|=y_m$. The second boundary can be written as two equations where $y_m=ax_m-ad$ and $y_m=ad-ax_m$ depending on the sign of the term x_m-d. However, these two equations are not linear in the two parameters that are the subject of the optimization, a and d. A new variable, b, is defined such that $b=ad$. Substituting, the three equations for the boundaries become:

$$\begin{aligned} ax_m &= b \\ y_m &= ax_m - b \\ y_m &= b - ax_m \end{aligned} \tag{23}$$

These equations are linear in a and b and specify $3M$ lines that partition the a-b plane into some number of hypersegments. Between partitions given by (23) none of the arguments of the absolute value nonlinearities in (22) can change sign and, therefore, (22) can be rewritten as:

$$E(a,b) = \sum_{m=1}^{M} \pm \left[(ax_m - b) \pm y_m \right] \tag{24}$$

where the $\pm$ represents a positive or negative sign that is fixed everywhere within the bounds given by (23). The error surface described in the bounded region described by (24) is linear in a and b everywhere in that region. The set of $3M$ intersecting lines given in (23) provide a lattice work across the entire error surface and divide it into piecewise linear regions.

An example of this is shown in Figure 17. Here, a region with six vertices is completely bounded by six lines.

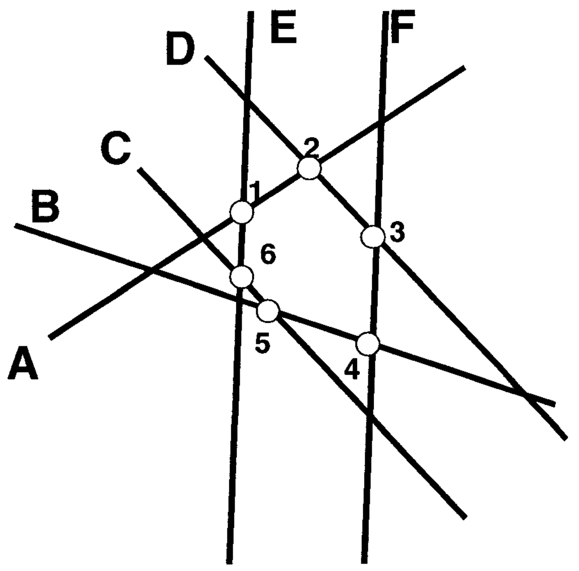

Figure 17 – Planar partitioning.

A useful property of a planar region completely bounded by lines is that the minimum of that region must occur at the intersection of two of the bounds.

Lemma 2.

Given a planar region P formed by the equation $E(a,b)=c_1 a + c_2 b + c_3$ completely bounded by K planes $\{d_{11} a + d_{21} b = d_{31}, \ldots, d_{1K} a + d_{2K} b = d_{3K}\}$. None of the bounding planes can be parallel with P. E(a, b) is minimized on P at the intersections of two of the K bounding planes with the plane $c_1 a + c_2 b + c_3$.

Proof.

Assume the minimum occurs inside *P* but not on any of the bounding planes and is equal to $c_1 a' + c_2 b' + c_3$. Examine the point (a'', b'') where $a''=a'-\varepsilon_a/c_1$ and $b''=b'-\varepsilon_b/c_2$ and $\varepsilon_a>0$ and $\varepsilon_b>0$. Because (a', b') are in the interior of *P* an ε_a and an ε_b can be found such that (a'', b'') is also in *P*. $E(a'', b'') = c_1 a' - \varepsilon_a + c_2 b' - \varepsilon_b + c_3$ and is therefore less than the value at (a', b'). In the case where c_1 is zero, a'' is set to a' and $E(a'', b'') = c_1 a' + c_2 b' - \varepsilon_b + c_3$ and is still less than $E(a', b')$. The same applies to the case where c_2 is zero.

In the case where both c_1 and c_2 are zero, $E(a, b) = c_3$ everywhere and the minimum occurs at every point in P. Therefore, the minimum must occur on one of the bounding planes.

Assume that the minimum occurs on the p^{th} bounding plane, but not at an intersection with another bounding plane and is $E(a', b') = c_1 a' + c_2 b' + c_3$ where a' and b' satisfy $d_{1p} a' + d_{2p} b' = d_{3p}$. Substituting the bound plane condition into the minimum equation yields $E(a', b') = c_1 a' + (c_2 d_{3p} - d_{1p} a')/d_{2p} + c_3$ which can be rearranged to $E(a', b') = (c_1 - d_{1p}/d_{2p}) a' + c_2 d_{3p}/d_{2p} + c_3$. Consider the point (a'', b'') where $a''=a' - d_{2p}\, \varepsilon/(c_1 - d_{1p})$ where $\varepsilon>0$ and b'' resides on a corresponding location on the bounding plane. The minimum equation is written as $E(a'', b'') = (c_1 - d_{1p}/d_{2p}) a' - \varepsilon + c_2 d_{3p}/d_{2p} + c_3$. Because $E(a'', b'') < E(a', b')$, (a', b') cannot be the point that minimizes $E(a,b)$. In the case where $c_1=d_{1p}$, a'' is undefined. In this case, the substitution for b' would be used and provide the same result as above. In the case where $c_2=d_{2p}$ and $c_2=d_{2p}$, the bounding plane and P are parallel in violation of the given constraint. Therefore, the minimum cannot be restricted to occur in the interior of P or on one of the edges of P. The minimum of P must occur on at least one of the vertices of P.

■

Having shown that a planar region completely bounded by planes has a minimum value occurring at one of the vertices, it is useful to examine the case where a region is not completely bounded but extends out to infinity.

Lemma 3.

Given a planar region P formed by the equation $E(a,b)=c_1 a + c_2 b + c_3$ partially bounded by K planes $\{d_{11} a + d_{21} b = d_{31}, \ldots, d_{1K} a + d_{2K} b = d_{3K}\}$. Assume that planes 1 and 2 have only one intersection with any of the other K planes and therefore border the region P and extend unbounded. None of the K planes can be parallel with P. E(a, b) is minimized on P at the intersections of two of the K bounding planes with the plane $c_1 a + c_2 b + c_3$ or at $-\infty$.

Proof.

Assume the minimum occurs inside P but not on any of the bounding planes and is equal to $c_1 a' + c_2 b' + c_3$. Examine the point (a'', b'') where $a''=a'-\varepsilon_a/c_1$ and $b''=b'-\varepsilon_b/c_2$ and $\varepsilon_a>0$ and $\varepsilon_b>0$. As in Lemma 2, this point is less than the assumed minima and therefore the minimum must be on one of the bounding planes. Assume the minimum is on bounding plane 1 or 2 (those planes extending to infinity). If $E(a, b) \rightarrow -\infty$ as either of these planes extends towards infinity then the minimum of $E(a, b)$ is $-\infty$. If $E(a, b) \rightarrow \infty$ as both bounding planes extend toward infinity, than the minimum occurs at the intersection of two of the K bounding planes as in Lemma 2.

■

A theorem can be developed along the same lines as Theorems 6 and 7 proving that the error function of the scaled and shifted absolute value function is minimized at an a and b satisfying (23) at one of the M points in the input/output data set.

Theorem 8.

Given a set of finite data points $\{(x_1, y_1), \ldots, (x_M, y_M)\}$ and an approximating function $f(x, a, d)=a|x-d|$, the error function defined in (22) is minimized at a point where the bounding planes $x_m=d$ and $a|x_m-d|=y_m$ intersect on the error surface.

Proof.

Making the substitution $b=ad$ the error function becomes that shown in (24) between each of the partitions formed by (23). This error function describes a planar surface piecewise linear between the $3M$ planes $y_m=ax_m-b$, $y_m=-ax_m+b$, or $x_m=d$. If the minimum occurs within one of the regions completely bounded by the planes derived from (23), Lemma 2 dictates that the minimum occurs at one or more of the vertices of one of the regions bounded by the lattice. However, the error hypersurface may have regions that are only partially bounded and extend to infinity. By Lemma 3, the minimum at a partially bounded region either occurs at a vertex or is $-\infty$. However, the absolute value function in (22) dictates that the minimum is positive and therefore any unbounded region must have a finite minimum to be considered as the minimum. Therefore the minimum occurs at one of the vertices formed by the intersection of two of the *3M* bounding planes where they intersect the error surface.

■

This theorem shows that the minimum of the error function given in (22) for a scaled and shifted absolute value function occurs at one of a finite number of points in the (a, d) plane. The possible number of solutions is upper bounded by the total number of potential intersections of the *3M* bounding planes or $3M(3M-1)$ where M is the number of points in the input/output data set.

The scaled and shifted absolute value function is equivalent to a one-break PLA. The final step in developing this theory is extending this result to an N-break PLA. Before proceeding there, it is necessary to examine an N-dimensional piecewise-linear hypersurface and demonstrate that the global minimum always occurs at one of the vertices of the hypersurface.

4.4.4 Global Minimum of an *N*-Dimensional Piecewise-Linear Hyperspace

Given an N-dimensional Euclidean space and K N-1 dimensional hyperplanes where the kth hyperplane is given by

$$y = \mathbf{a}'_k\,\mathbf{x} \quad where \quad \mathbf{a}_k = \begin{bmatrix} a_{k,1} \\ \vdots \\ a_{k,N-1} \end{bmatrix} \quad and \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

where $y \in \Re$, $a_{k.n} \in \Re$, $\mathbf{a}_k \neq \mathbf{0}$, and $x_n \in \Re$.

Define a hypersurface, P, bounded by the hyperplanes $\mathbf{a}'_k\,\mathbf{x}\ \forall\ k$ where

$$E(\mathbf{x}) = \mathbf{c}'\mathbf{x} \quad where \quad \mathbf{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_{N-1} \end{bmatrix}$$

The vertices of P are the points $x \in P$ where N-1 of the bounding hyperplanes intersect.

Lemma 4.

Given a hypersurface P formed by the equation $E(x)=c'x$ where $c \in \Re^{N-1}$ and *$x \in \Re^{N-1}$ completely bounded by K hyperplanes $\{ a_1'x, \ldots, a_K'x \}$. None of the bounding hyperplanes can be parallel with P. E(x) is minimized on P at the intersections of N-1 of the K bounding hyperplanes with the hyperplane **c'x**.*

Proof.

Assume the minimum occurs inside *P* but not on any of the bounding planes and is equal to $\mathbf{x}=\acute{\mathbf{z}}$. Examine the point $\check{\mathbf{z}}$ where $\check{\mathbf{z}} = \acute{\mathbf{z}} - \Delta$ where

$$\Delta = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_{N-1} \end{bmatrix} \quad where \quad \delta_n = \begin{cases} \dfrac{\varepsilon_n}{c_n} & c_n \neq 0 \\ 0 & c_n = 0 \end{cases} \quad : n = 1 \ldots N-1, \varepsilon_n > 0$$

Because $\acute{\mathbf{z}}$ is in the interior of *P* a Δ can be found such that $\check{\mathbf{z}}$ is also in *P*. $E(\check{\mathbf{z}}) = \mathbf{c}'(\acute{\mathbf{z}}-\Delta)$ and is therefore less than $E(\acute{\mathbf{z}})$ except where $\mathbf{c}=\mathbf{0}$ in which case *y* is a constant (and therefore minimum) everywhere. Therefore, the minimum must occur on one of the bounding hyperplanes.

Next, assume that the minimum occurs at $\mathbf{x}=\acute{\mathbf{z}}$ which lies in *P* on the *k*th bounding hyperplane, but not at an intersection with another bounding hyperplane. Because $\acute{\mathbf{z}}$ lies on the *k*th bounding hyperplane, it must satisfy $\mathbf{c}'\acute{\mathbf{z}} = \mathbf{a}_k'\acute{\mathbf{z}}$. Consider a point $\check{\mathbf{z}}$ where $\check{\mathbf{z}} = \acute{\mathbf{z}} - \Delta$ where

$$\Delta = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_{N-1} \end{bmatrix} \quad where \quad \delta_n = \begin{cases} \dfrac{\varepsilon_n a_{k,n}^{\;2}}{c_n} & c_n \neq 0 \\ 0 & c_n = 0 \end{cases} \quad : n = 1 \ldots N-1, \varepsilon_n > 0$$

Because $\check{\mathbf{z}}$ remains on the bounding hyperplane and $E(\check{\mathbf{z}}) = \mathbf{c}'\,\acute{\mathbf{z}} - \mathbf{c}'\Delta$ is always less than $\mathbf{c}'\acute{\mathbf{z}}$, the minimum of *E*(**x**) must lie at the intersection of at least two of the bounding hyper planes. This logic can be continued until the minimum is shown to exist at one of the intersections of *N-1* of the bounding hyperplanes with *P* yielding a point in the *N*th dimensional space.

■

While this proves that the minimum occurs at a vertex, it does not prove that this minimum is unique. It is possible for the minimum to occur at multiple vertices and the hypersurfaces that connect them.

Lemma 5.

Given a hypersurface P formed by the equation E(x)=c'x where $\mathbf{c}\in\Re^{N-1}$ and $\mathbf{x}\in\Re^{N-1}$ *partially bounded by K hyperplanes {* $\mathbf{a}_1'\mathbf{x}, \ldots, \mathbf{a}_K'\mathbf{x}$ *}. None of the bounding hyperplanes can be parallel with P. E(x) is minimized on P either at the intersections of N-1 of the K bounding hyperplanes with the hyperplane* **c'x** *or approaches* $-\infty$.

Proof.

Assume the minimum occurs inside *P* but not on any of the bounding planes and is equal to **x=ź**. Examine the point **ž** where **ž** = **ź** - **Δ** where

$$\Delta = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_{N-1} \end{bmatrix} \quad where \quad \delta_n = \begin{cases} \dfrac{\varepsilon_n}{c_n} & c_n \neq 0 \\ 0 & c_n = 0 \end{cases} \quad : n = 1 \ldots N-1, \varepsilon_n > 0$$

Because **ź** is in the interior of *P* a **Δ** can be found such that **ž** is also in *P*. *E(***ž***)* = **c'(ź-Δ)** and is therefore less than *E(***ź***)* except where **c=0** in which case *y* is a constant (and therefore minimum) everywhere. Therefore, the minimum must occur on one of the bounding hyperplanes.

Next, assume that the minimum occurs at **x= ź** which lies in *P* on the *k*th bounding hyperplane, but not at an intersection with another bounding hyperplane. Because **ź** lies on the *k*th bounding hyperplane, it must satisfy **c'ź** = $\mathbf{a}_k$**'ź**. Consider a point **ž** where **ž** = **ź** - **Δ** where

$$\Delta = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_{N-1} \end{bmatrix} \quad where \quad \delta_n = \begin{cases} \dfrac{\varepsilon_n a_{k,n}^{\;2}}{c_n} & c_n \neq 0 \\ 0 & c_n = 0 \end{cases} \quad : n = 1 \ldots N-1, \varepsilon_n > 0$$

Because **ž** remains on the bounding hyperplane and *E(***ž***)* = **c' ź** – **c'Δ** is always less than **c'ź**, the minimum of *E*(**x**) must lie at the intersection of at least two of the bounding hyper planes. This logic can be continued until the minimum is shown to exist at one of the intersections of *N-2* of the bounding hyperplanes with *P* yielding a ray in the N^{th} dimensional space. If $E(\mathbf{x}) \rightarrow \infty$ as the ray $\rightarrow \infty$ then the minimum occurs where the ray intersects the next bounding plane. If the $E(\mathbf{x}) \rightarrow -\infty$ as the ray $\rightarrow \infty$ then the minimum is $-\infty$.

■

Lemmas 4 and 5 provide the tools necessary to analyze a piecewise linear function in $\Re^N$ and provide other criteria when searching this space for the minimum of this function.

4.4.5 Optimal Determination of a Piecewise-Linear Approximator

In Theorem 5 it was shown that a PLA could be expressed as:

$$f(x,\mathbf{a},\mathbf{d},c) = a_0 x + c + \sum_{n=1}^{N} a_n |x - d_n|, \quad \mathbf{a} = \begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} d_1 \\ \vdots \\ d_N \end{bmatrix} \tag{25}$$

where a_0, $c \in \Re$ are coefficients of an affine translation, $a_n \in \Re$ $(n=1, \ldots, N)$ scales the absolute value function and $d_n \in \Re$ $(n=1, \ldots, N)$ shifts it along the *x* axis. Given a set of data $\{(x_1, y_1), \ldots, (x_M, y_M)\}$, an error function is defined:

$$\begin{aligned} E(\mathbf{a},\mathbf{d},c) &= \sum_{m=1}^{M} |f(x_m,\mathbf{a},\mathbf{d},c) - y_m| \\ &= \sum_{m=1}^{M} \left| \sum_{n=1}^{N} \left[a_n |x_m - d_n| \right] + a_0 x_m + c - y_m \right| \end{aligned} \tag{26}$$

From this equation, $E(\mathbf{a}, \mathbf{d}, c)$ yields a $2N+2$-dimensional piecewise linear hypersurface. The goal of the optimization is to find an **a, d**, and *c* such that $E(\mathbf{a}, \mathbf{d}, c)$ is minimized for the given set of data.

The hypersurfaces that partition the error hyperspace given by (26) are formed wherever $x_m=d_n$ or $a_0x_m + c + a_1|x_m-d_1|+\ldots+ a_N|x_m-d_N|=y_m$ are satisfied. The second boundary condition can be written as $a_0x_m + c \pm[a_1x_m-a_1d_1]\pm[\ldots]\pm[a_Nx_m-a_Nd_N]=y_m$ which in turn yields 2^N equations, each with a different perturbation of the $\pm$ signs. However, these equations are not linear in a_n and d_n. A new variable, b_n, is defined where $b_n=a_n d_n$. Substituting, the boundary equations become:

$$\begin{aligned} &a_n x_m - b_n = 0 && N \text{ equations} \\ &a_0 x_m + c + \sum_{n=1}^{N} \pm \left[a_n x_m - b_n\right] = y_m && 2^N \text{ equations} \end{aligned} \tag{27}$$

These equations are linear in a_n, b_n, and c. Anywhere between these boundaries, equation (26) can be written as:

$$E(\mathbf{a},\mathbf{b},c) = \sum_{m=1}^{M} (\pm) \left[\left[\sum_{n=1}^{N} \pm (a_n x_m - b_n) \right] + a_0 x_m + c - y_m \right], \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix} \tag{28}$$

where the ± represents a positive or negative sign that is fixed everywhere within the given boundary. The error surface described in the bounded region described by (28) is linear in **a**, **b**, and c everywhere in that region. The intersecting lines given in (27) bound each linear region.

Theorem 9.

Given a set of finite data points $\{(x_1, y_1), \ldots, (x_M, y_M)\}$, a piecewise linear approximating function in the form of (25) where N is the number of breaks and the error function defined in (26), the error is minimized at a point where 2N+1 of the 2^N+N boundary equations intersect with the error function.

Proof.

The $2N+2$ dimensional piecewise-linear hypersurface defined by the error function (26) is made up of some number of completely bounded hyperplanes with an outer border of partially bounded hyperplanes. Assume the minimum occurs in one of the partially bounded regions. Lemma 5 shows that the minimum of a partially bounded linear region occurs either at a vertex or at $-\infty$. Because the error function in (26) is constructed with absolute value functions, the minimum this function can achieve is *0*. Therefore, the minimum cannot occur at $-\infty$ and must therefore occur at a vertex. Should the minimum fall within the one of the bounded regions in this structure, the minimum point will occur at one of the vertices where the boundary functions intersect with the error surface by Lemma 4.

■

This theorem states that the minimum of an *N*-break piecewise linear function is always found at one of a finite number of points in a *2N+2* –dimensional space. This bounds the number of possible solutions to the optimization problem as well as suggesting possible algorithms to solve the optimization.

4.5 Conclusion

This chapter began with a formalization of a specific class of fuzzy system and a proof that this class was equivalent to an *N*-dimensional PLA. An optimization algorithm was developed for a shifted absolute value function, a scaled absolute value function, and a shifted and scaled absolute value function in preparation for development of the general PLA optimization theory. Finally, a proof of the optimization of an *N*-dimensional PLA was developed. A proof of this nature has not been hitherto developed and is a novel contribution of this work.

The next chapter examines these results from a practical perspective and develops several algorithms that can be used in various applications including control systems.

5.0 APPLICATIONS AND ALGORITHMS

Having developed a mechanism for nonlinear optimization of a class of functions, it is useful to examine how these results can be applied to existing problems. This chapter first examines several simple nonlinear approximation problems and applies the above results to optimally fit a PLA to a set of data. An alternative (and non-optimal) algorithm is discussed that provides significantly improved performance and it is applied to problems of greater complexity. A simple system identification and adaptive control problem is discussed. Finally, this technique is applied to a Hammerstein system identification problem.

5.1 Optimal Determination of a PLA from Data

A method to minimize the error function given by (28) of the approximation of a set of input/output data by an *N*-break PLA is readily induced from Theorem 9. The method consists of the following steps:

1. Select the number of breakpoints, *N*, in the PLA. Selecting an *N* that is too large will result in an approximation that may overfit the input data. Selecting an *N* that is too small results in an approximation that cannot satisfactorily fit the data.
2. Provide *M* data points that are to be approximated.
3. Form the $M(N+2^N)$ equations forming the linear boundaries as in (27).
4. Initialize the lowest error to ∞.
5. Select every possible combination of $2N+2$ of the equations formed above. Solve each combination for **a**, **b**, *a0,* *c*, and the error.
6. Calculate the error of the function using equation (28).
7. If the error is lower than the previous best, store **a**, **b**, *a0,* and *c*.
8. Once all possible combinations are tried, the stored **a, b**, *a0,* and *c* minimize the error function (28).

The above algorithm was coded in MATLAB and is included in Appendix B.

This method was tested with a set of data taken from a true PLA function with **a**=[0.5, 0.75], **d**=[-1 3], *a0*=0.25, and *c*=-1.75. Five data points from this function [(-3, 3), (-1, 1), (1, 1), (3, 1), (5, 4)] were presented to the method above with the goal of finding a PLA with two breakpoints (N=2). The method found $M(N+2^N)$ or 30 boundary equations. The boundary equations were examined in all possible combinations of $2N+2$ equations yielding 593,775 possible vertices. Of these nearly six hundred thousand systems of equations, more than 92% were found to be singular, to have an element of **a** equal to zero, or to be poorly ordered ($a_{n+1} \leq a_n$).

The algorithm determined the correct values of the coefficients and produced the plot shown in Figure 18 where the line is the PLA produced form the data and the circles are the initial data points. Clearly the determination of the coefficients is optimal and the error is zero.

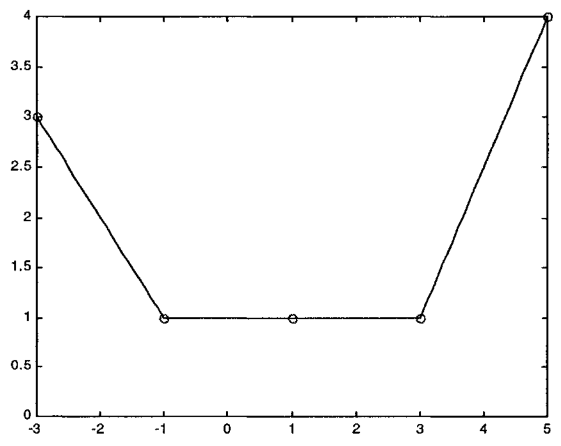

Figure 18 – PLA estimation of 5 points.

The same five points were fed to the algorithm, this time with the y-axis values perturbed by Gaussian random noise of magnitude of 0.1 and standard deviation of 1. The purpose of this is to examine the fitting of a 2-break PLA to points that cannot be perfectly approximated. The algorithm determined coefficients such that the error was 0.0814. The coefficients found were **a**=[0.5552, 0.6897], **d**=[-1.0000, 3.0000], $a0$=0.1833, and c=-1.7422. The result of this is shown in Figure 19.

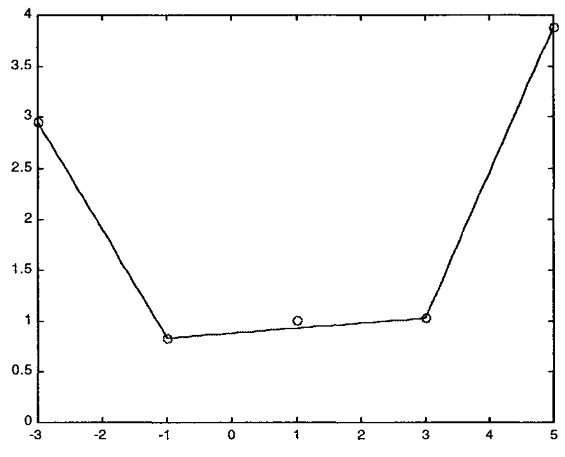

Figure 19 – PLA estimation of 5 points perturbed by noise.

While these examples are trivial, they demonstrate the capability of the algorithm to optimally fit a PLA to given data given only the number of breaks.

Additionally, this result can be compared to the error bound guaranteed by Theorem 3. Recall that the theorem bounded the error by $\omega_f(\frac{1}{2}(r+1)\,h)$ where r, the degree of the polynomial used in the PLA, is 1; h, the maximum distance between breaks, is in this case 2, and the modulus of continuity, $\omega_f(2)$ for the 2-break PLA is approximately 3. Therefore, the actual error of the function is several orders of magnitude less than the bound.

5.2 Solution Complexity and Complexity Reduction

As the number of data points (M) and the number of segments in the approximation (N) increases so do the number of vertices. The number of vertices are upper-bounded by selecting $2N+2$ out of the $M[N+2^N]$ possible equations or:

$$v = \binom{2N+2}{M\left[N+2^N\right]}.$$

The number of simultaneous systems of equations grows rapidly with the number of data points (*M*) included and explodes exponentially as additional breaks (*N*) are added. The table below shows the number of vertices associated with a system with a given number of segments and data points.

Table 4 - Number of error surface vertices for a given problem

		Number of Vertices									
		Number of Breaks									
		1	2	3	4	5	6	7	8	9	10
Data Points	8	10626	12271512	6.428E+10	2.274E+15	7.534E+20	2.905E+27	1.464E+35	1.015E+44	9.831E+53	1.332E+65
	16	194580	927048304	1.943E+13	2.692E+18	3.458E+24	5.167E+31	1.015E+40	2.76E+49	1.055E+60	5.666E+71
	32	3321960	6.43E+10	5.395E+15	2.96E+21	1.498E+28	8.818E+35	6.84E+44	7.368E+54	1.119E+66	2.393E+78
	64	54870480	4.282E+12	1.438E+18	3.141E+24	6.311E+31	1.475E+40	4.545E+49	1.949E+60	1.18E+72	1.007E+85
	128	891881376	2.795E+14	3.755E+20	3.273E+27	2.621E+35	2.441E+44	3E+54	5.133E+65	1.24E+78	4.232E+91
	256	1.438E+10	1.806E+16	9.71E+22	3.382E+30	1.081E+39	4.019E+48	1.973E+59	1.349E+71	1.303E+84	1.777E+98
	512	2.31E+11	1.162E+18	2.498E+25	3.478E+33	4.444E+42	6.601E+52	1.295E+64	3.539E+76	1.367E+90	7.45E+104
	1024	3.704E+12	7.453E+19	6.411E+27	3.569E+36	1.824E+46	1.083E+57	8.495E+68	9.283E+81	1.434E+96	3.13E+111

From this table, it is clear that even small problems are far too computationally intense to solve in this direct manner (a 6-segment line and 128 data points has over 2 X 10^{35} vertices). However, there may be means to dramatically simplify this problem and reduce the computations required into a reasonable realm. For example, if the above problem was partitioned into two hyperspaces of 64 data points on 3-segment lines, the two solutions have under 5 X 10^{12} vertices, still a tremendous number but much more manageable.

5.3 Descent Technique of Parameter Determination

While the theory presented to this point provides an optimal determination of parameters from a nonlinear error function, the cost, in terms of complexity, is tremendous for all but the simplest problems. There are several means for dealing with cost of solving this nonlinear optimization problem.

1. Develop novel methods for solving this type of problem that greatly reduce computational complexity.

2. Develop a method that trades guaranteed optimality for computational reduction.

The development of methods for more efficiently solving the nonlinear optimization problem is a significant effort in itself and beyond the scope of this work. However the development of a more practical method of arriving at good, if not optimal, solutions to these problems is a critical step in the development of examples demonstrating the utility of this work. Therefore, a technique fulfilling this requirement will be examined.

5.3.1 Descent Algorithm

Consider a space, S, in $\Re^N$ where there exist K hyperplanes. The intersection of N of these hyperplanes determines a point, P, in S. Leaving this point are N hyperlines resulting from the intersection of every combination of N-1 of the N hyperplanes that intersect to form P. Each of the N hyperlines is intersected at a point by, at most, K-N hyperplanes (not including intersections at P).

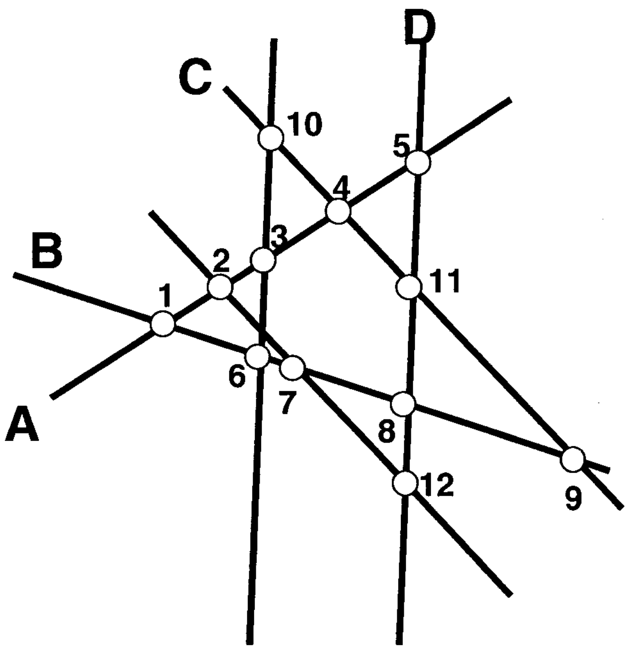

Figure 20 – Descent example.

Consider the case shown in Figure 20 where $N=2$ and $K=6$. The algorithm begins by randomly selecting a node (node 1 in this case). Note that this node is created from the intersection of lines A and B. First the errors of the nodes formed by the intersection of line A with other lines are checked (nodes 1, 2, 3, 4, and 5) and the error calculated with (28). Next the nodes on line B are checked (nodes 6, 7, 8, and 9) and the error calculated. The lowest error is saved and that node is set as the central node and the above algorithm repeated. For this example, assume that node 4 was found to be the node with lowest error. Since the nodes on line A have already been checked and found to have a higher error than node 4, only the nodes on line C need to be checked (nodes 10 and 11). For this example, assume that node 4 had a lower error than nodes 10 or 11. The algorithm would terminate and return node 4 as the local minimum. Because it is likely for nodes to go unchecked (node 12 in this example), this algorithm does not guarantee the return of an absolute minimum.

While this method does not guarantee the discovery of the absolute minimum, it does guarantee the discovery of a local minimum. This simple example doesn't truly demonstrate the significant decrease in computational complexity as there were only two dimensions in the problem's input space. However, savings in computation rise exponentially with an increase in problem dimension and yield dramatic savings.

The tradeoff for this savings the lack of an optimality guarantee as this algorithm can be trapped in a local minimum. However, there is a built-in local-minimum avoidance routine as the nodes checked around the local minimum are widely distributed throughout the problem space.

In more general terms the descent algorithm is:

1. Select the number of breakpoints, N, of the PLA.
2. Select a node (an intersection of N hyperplanes) and calculate the error using (28). Save this value as the minimum error.
3. For each intersection of N-1 of the hyperplanes forming the node perform the following:

   a. Determine the intersection with each of the K hyperplanes not part of the original node selection.
   b. Calculate the error of each intersection using (28).
   c. If the lowest error is less than the previously saved minimum error, replace the minimum error with this value and save the intersecting planes that produced this error.
4. If the minimum error is lower than the error from step 1, the solution has been improved and using the saved intersecting planes as the selected node, repeat step 1. If no improvement is made in error then this algorithm has found at least a local minimum error and the process is complete.

The end result of this algorithm is the discovery of a node that produces a lower error than any other node that exists on any hyperline emanating from the node. While this does not guarantee a global minimum, as does the exhaustive search algorithm described earlier, it dramatically reduces computational complexity. Additionally, several significant improvements over the gradient descent techniques used in backpropagation neural networks are offered:

- While it is possible for this method to become trapped at a local minimum, this method has an integral mechanism built right into the algorithm that searches both near and far from the node of interest for lower-error solutions. Unlike the backpropagation neural network, local minimum avoidance sub-algorithms (such as ‘shaking’ or the momentum coefficient) are not introduced.
- A neural network training scheme must continually examine a test data set in order to ensure the network is not overtraining on the training data. In contrast, the fuzzy system will continue to fit the training data until it has achieved the lowest error possible. The quality of the fit and any possibility of overtraining are determined solely by the number of breaks slected by the designer in the piecewise linear approximation of the input/output data set.

- The fuzzy system has a single adjustable parameter, the number of breaks (*N*). Neural networks have many adjustable parameters (learning coefficient, momentum coefficient, number of hidden layers, number of neurons in hidden layers, neuron transfer function, etc) and the proper determination of these parameters is critical to the convergence and performance of the network.

5.4 Comparison of a Fuzzy System with a Neural Network

The data used to train the systems was problem data taken from 0 to 4 inclusively every 0.25 units. The function output was the result of the Boolean expression ($x<1$) OR ($x>3$) perturbed by zero-mean normally-distributed noise with a standard deviation of 0.1. This data is shown in Figure 21.

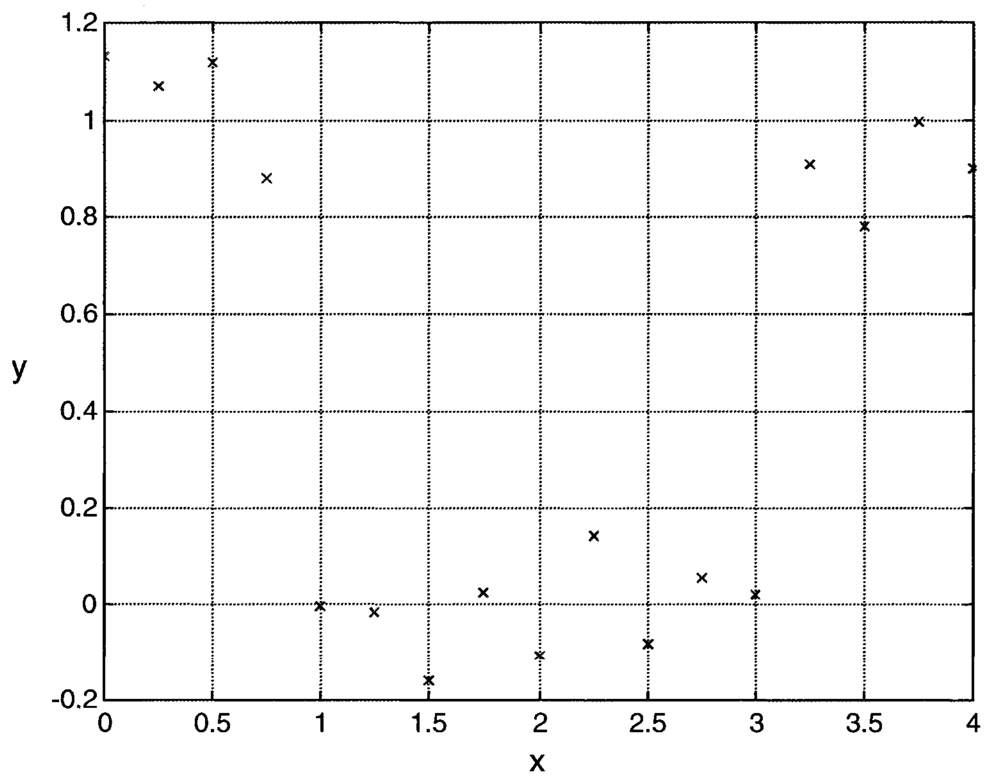

Figure 21 – Noise perturbed example data.

5.4.1 The Neural Network Results

A three-layer feedforward neural network was trained using the Levenburg-Marquardt technique to learn the pattern of the above data. A network with 7 hidden neurons was found to be sufficient for this problem and the network was trained for a thousand iterations. The result of this training was generated with the MATLAB code shown in Appendix C is shown in Figure 22. From this plot, it can be seen that the network has

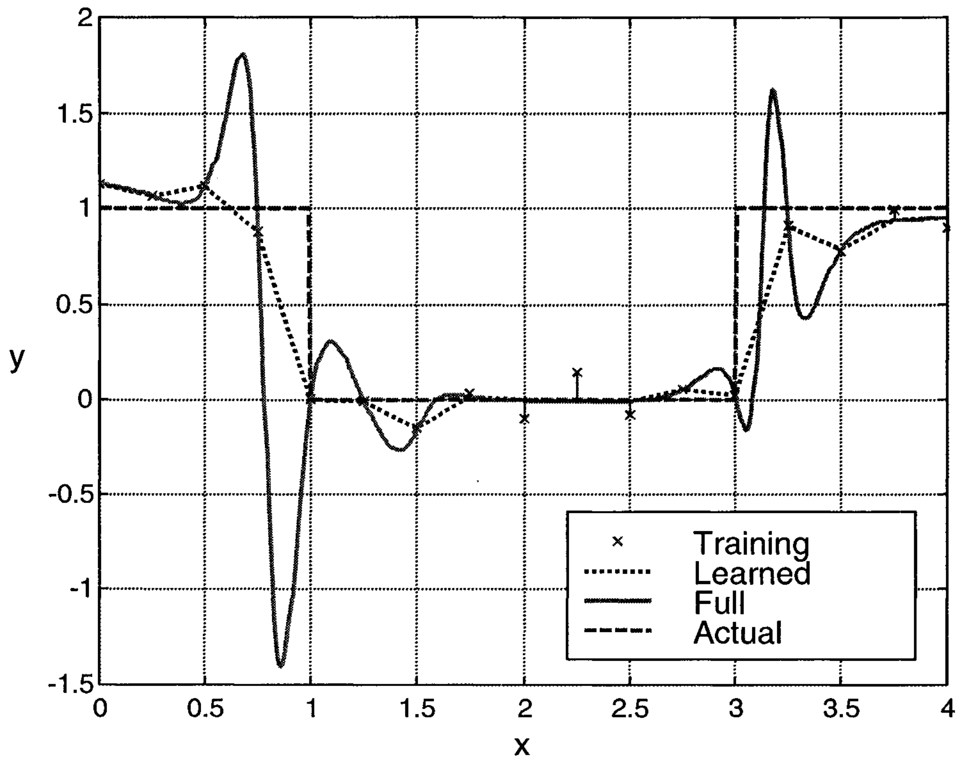

Figure 22 – Neural net approximation of data.

learned the training data well (dashed line) and there is only small error when looking only at the training data. However, the network has been over-trained on the training data and has lost much of it's ability to generalize the function (solid line) and varies considerably from the original function (long dashed line).

5.4.2 The Fuzzy System Results

A single-input fuzzy system was trained using the descent technique described in this section with 4 breakpoints. The end result of this algorithm is the determination of the unknown coefficients of a PLA in the form given by (25). While this form can be used directly to determine the output of this system, the ultimate goal of this work is to use the PLA to determine the parameters of a fuzzy system and there are several advantages for doing so:

1. Heuristic information can be extracted from the fuzzy system and used to gain deeper understanding of the underlying system.
2. The fuzzy system offers a significant computational reduction as only 2^N membership functions are active for any input datum whereas $M_1 \bullet M_2 \bullet \ldots \bullet M_N$ elements are active in the PLA case where M_n are the number of breakpoints in the n^{th} dimension.

While the existence of the mapping from PLA to fuzzy system was established earlier in this work, the actual mechanics of that transformation have yet to be discussed.

Between any two apexes of the fuzzy MFs d_n and d_{n+1} the output of the fuzzy system is given in (12). Since the activation strength of the two active MFs must sum to one, the resultant output of the fuzzy system for $d_n<x<d_{n+1}$ is:

$$y = z_{n+1}\frac{x-d_n}{d_{n+1}-d_n} + z_n\frac{d_{n+1}-x}{d_{n+1}-d_n} \tag{29}$$

Similarly, the PLA given by (25) where $d_n<x<d_{n+1}$ is equivalent to:

$$y = a_0 x + c + \sum_{n=1}^{n} a_n(x-d_n) - \sum_{n=n+1}^{N} a_n(d_n - x) \tag{30}$$

The algorithm has provided all the unknown coefficients in (30). The only remaining effort is to determine the z_n coefficients in (29).

At this point, it is important to note that it is possible that this algorithm returned breakpoints that are contained within the domain of our data set ($d_1 > x_0$ or $d_N < x_M$). For the fuzzy system to be effective all the way across our data set, we must add MFs to either end of the fuzzy set, creating a breakpoint at the minimum and maximum extent of the input data set. Furthermore, these breaks (d_0 and d_{N+1}) must be added without affecting the PLA solution. The solution to this problem is to add a break at the smallest value of the input data set and set it's corresponding *a* value to 0. Similarly a break is added at the largest input value of the training set with it's corresponding *a* set to 0. This has the desired effect of adding breaks at both extremities of the data and also not affecting the PLA solution (because the *a*'s are 0).

The first two MFs have apexes at d_0 and d_1 and corresponding output weights of z_0 and z_1. Since the fuzzy set equation for any $d_0 < x < d_1$ must be equivalent to the PLA equation over the same interval, the coefficient of *x* and the constant coefficient of both (29) and (30) must be equal yielding the following pair of equations:

$$\begin{aligned} a_0 - \sum_{n=1}^{N} a_n d_n &= \frac{-z_0}{d_1 - d_0} + \frac{z_1}{d_1 - d_0} \\ c + \sum_{n=1}^{N} a_n d_n &= \frac{z_0 d_0}{d_1 - d_0} - \frac{z_1 d_1}{d_1 - d_0} \end{aligned} \tag{31}$$

By definition, the breaks, d_n, are not permitted to be superimposed on one another and $d_0 \neq d_1$ and therefore (31) has unique solutions for z_0 and z_1. From here it is straightforward to continue down the chain and solve for all of the MF weights, z_n, completing the transformation of a PLA into a Fuzzy System. Consider the point where $d_n < x < d_{n+1}$ where $n>1$ and z_n has been determined (either through (31) or a previous recursion of this method). The *x*-coefficients of (29) and (30) can be equated yielding:

$$z_{n+1} = z_n + (d_{n+1} - d_n)\left[a_0 + \sum_{m=1}^{n} a_m d_m - \sum_{m=n+1}^{N} a_m d_m\right] \tag{32}$$

This method was encoded in MATLAB and included in Appendix D. The fuzzy system was trained to the data and was able to learn the pattern and the result is shown in Figure 23.

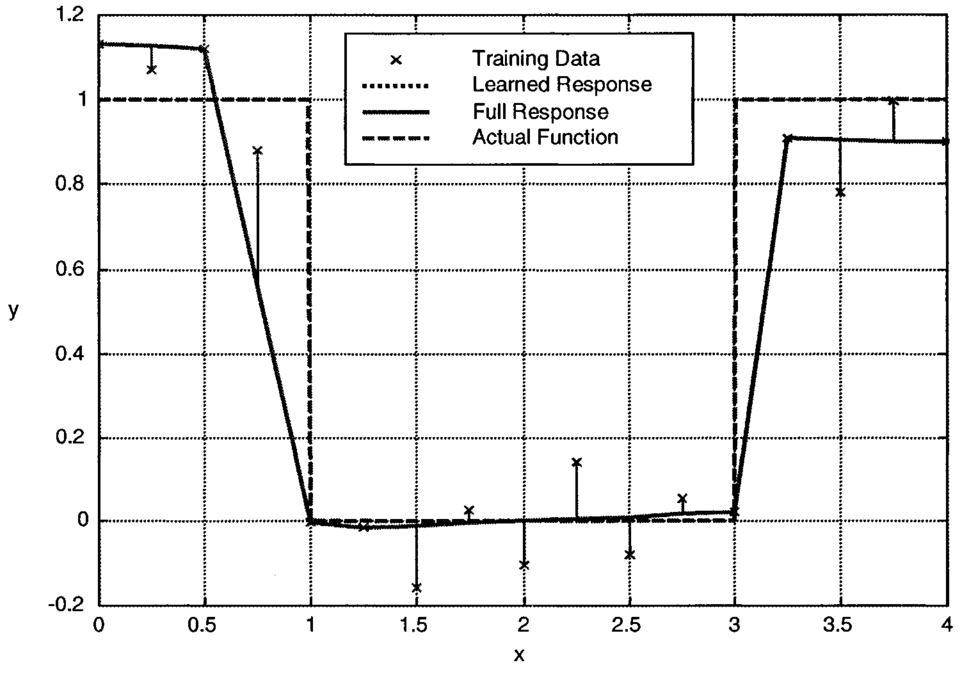

Figure 23 – Fuzzy system approximation of data.

The training data are indicated by x's, and the actual function is shown by a long dashed line. The response across the full domain is shown as a solid line. The vertical lines indicate the error between the approximation and the training data set. The very nature of this approximation technique ensures that a fuzzy system with an appropriate number of breaks cannot be overtrained. The approximation improves with each successive training iteration and finds a minimum at the 39th iteration. Figure 24 shows the history of the error over all the iterations.

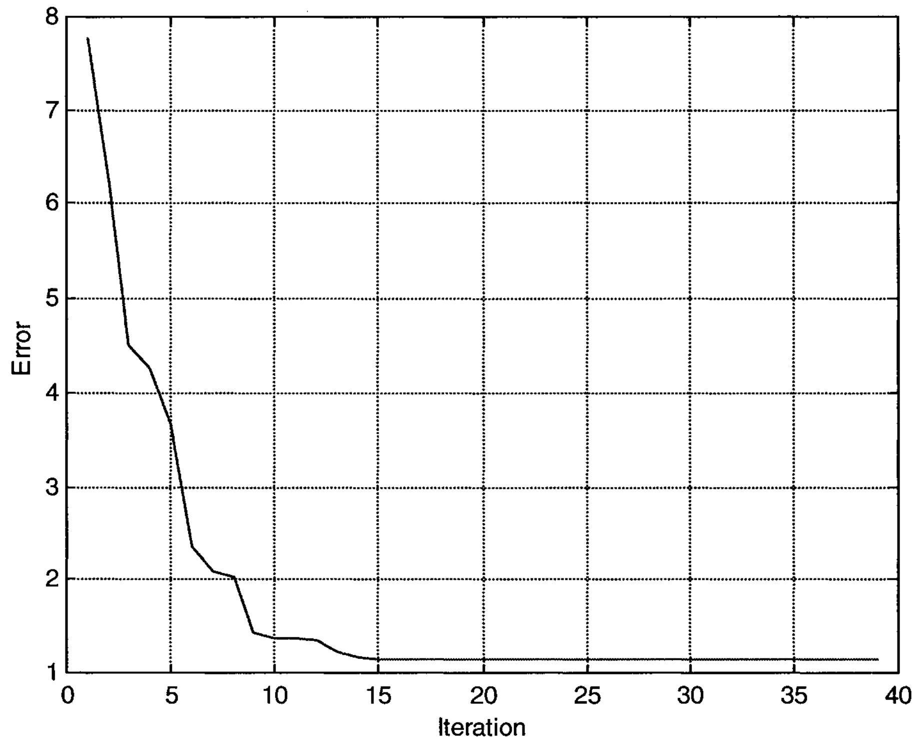

Figure 24 – Error vs. iteration for fuzzy training.

By the 15th iteration, the approximation has found a near-minimum point and continues for 24 additional iterations refining the solution.

5.5 Indirect Adaptive Control of a Wiener System

Consider the simple nonlinear, discrete-time system Wiener system shown in Figure 2. The goal is to design a control system that will provide a means to have the output of this system follow an input reference signal. For this example, an indirect adaptive control scheme is used to control this Wiener nonlinear plant. A diagram of the control scheme is shown in Figure 25.

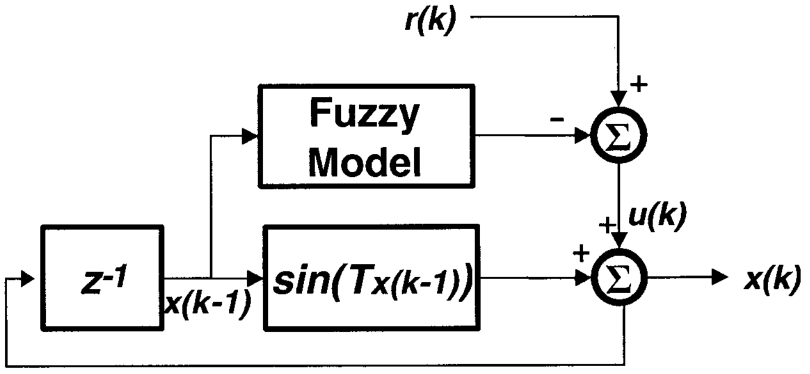

Figure 25 – Fuzzy IAC controller of a Wiener system

The first step in solving this problem is to train the fuzzy model to approximate the input-output pattern of the nonlinear plant using the descent method described earlier.

5.5.1 Training the Fuzzy System

The first step in the training process is to determine a suitable value of N (the number of breaks in the PLA that will approximate the system over a given region. For the purposes of this effort, it is assumed that $x(k)$ is bounded on $[-3, 3]$. Inspection of the input/output data set indicates that a reasonable selection for N might be as low as two. Increasing N can improve the approximation quality but the computational intensity of solving the fuzzy system will also be increased. The results for N=*2, 3, 4* are shown in Figure 26.

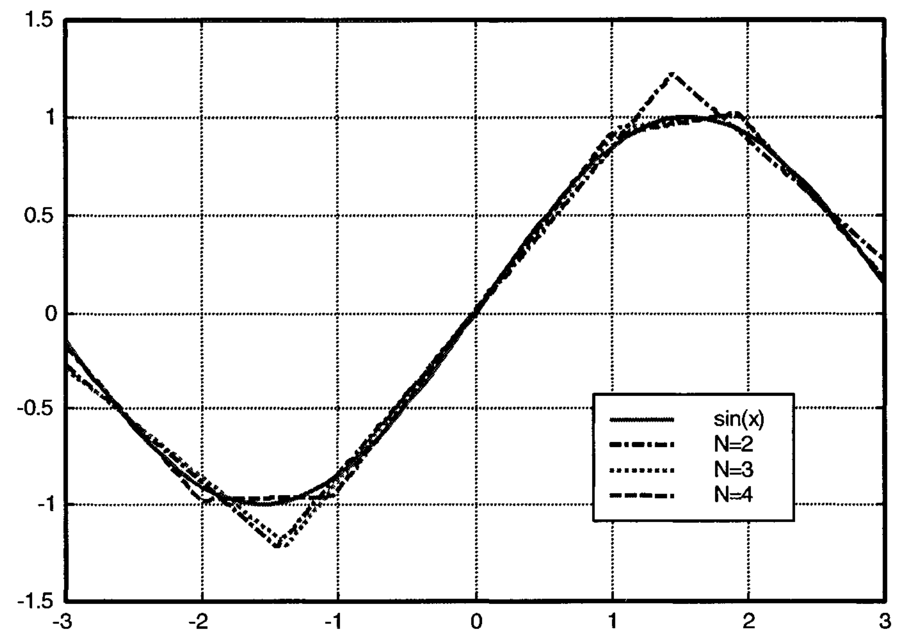

Figure 26 - Fuzzy approximation of sin function.

This figure demonstrates how the approximation is improved as N is increased. The approximation was performed for $N \in [1, 6]$ and several performance metrics were gathered.

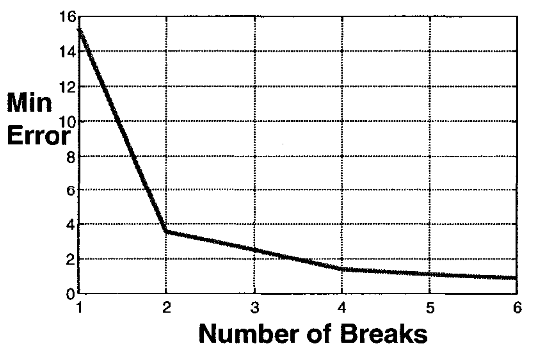

Figure 27 – Fuzzy approximation error.

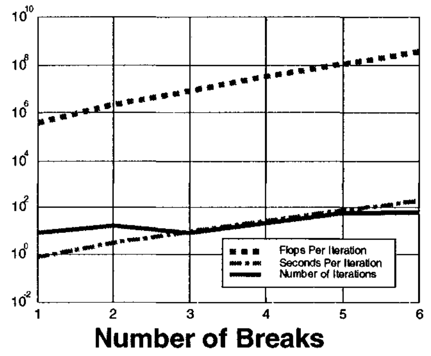

Figure 28 – Fuzzy performance parameters.

The total error of the approximated system for a given number of breaks calculated with (26) is shown in Figure 27. As expected, the error between the fuzzy system and the input/output data decreases as breaks are added. However, there is a cost-benefit tradeoff with adding breaks. The approximation improvement slows asymptotically to 0 whereas the computation cost grows exponentially. Additionally, adding excessive breaks is akin to overtraining a neural network system. With a large number of breaks, the PLA will too closely follow the training data and any noise contained therein.

Several performance statistics for the descent technique are shown in Figure 28. The horizontal axis is N as it increases from 1 to 6. The vertical axis represents the units appropriate to the data displayed. As predicted, the time and FLOP count increase exponentially relative to the number of breaks. This fairly simple problem with 30 input data points and 6 breaks took almost three hours to compute. Clearly significant algorithmic improvement is required before this method is useful for non-trivial problems. All runs were performed on a WindowsME PC running MATLAB 5.2 with 128M of RAM and a 500 MHz Pentium III processor.

The resultant error of the three approximations can also be compared with the upper bound given by Theorem 3. Because this method is not optimal, there is no guarantee that the resultant error between the approximation, s, and the original function, f, will be less than the bound. In each case, the order of the PLA (r) is 1 simplifying the argument to the modulus of continuity in Theorem 3 to $\omega_f(h)$.

Table 5 - Approximation error bound comparison

Number of breaks	h (largest distance between breaks)	$\omega_f(h)$ (Error bound)	**Maximum Error** ($\|f-s\|_\infty$)
2	2.9	2	0.5
3	2.4	1.9	0.4
4	2	1.7	0.2

Table 5 shows that in each case, the maximum error between the approximation and the function is better than the upper bound of the least maximum error. This provides an empirical indication that this algorithm is providing good results for this example.

5.5.2 Indirect Adaptive Control Results

Given the discrete-time Wiener nonlinear system:

$$x(k)=\sin(Tx(k-1))+u(k)$$

and a reference signal, $r(k)$, a control system is designed to permit the system to follow the reference signal. This is done by creating a control input such that:

$$u(k)=r(k)-m(x(k-1))$$

where $m(\bullet)$ is the fuzzy model of the nonlinear system.

The system was simulated with a unit step reference signal. The response of the system was simulated with both a linear model ($u(k)=r(k)-0.1x$) and the fuzzy model ($u(k)=r(k)-m(0.1x)$) and the results are shown in Figure 29.

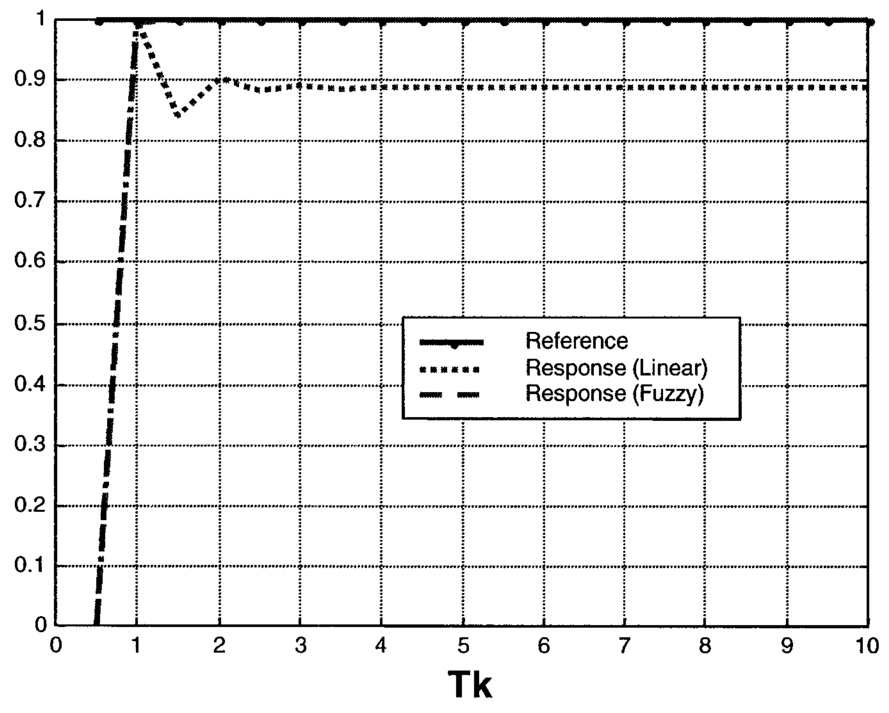

Figure 29 – Fuzzy IAC response to unit step.

The fuzzy system performance is excellent with a steady state error of less than 1%, whereas the linear model has an error exceeding 10%. While the performance of the fuzzy system was very good for this simple example, there remains a great deal of work, both in the refinement of an algorithm for the learning phase of the fuzzy system as well as methods for incorporating the fuzzy system into an adaptive control scheme. The excellent performance of the fuzzy system is not intended to be indicative of the performance of this type of fuzzy system in an indirect adaptive control scheme, but rather to demonstrate the potential benefits this method promises.

5.6 A Hammerstein System Identification Example

As discussed in Chapter 1, nonlinear systems can sometimes be separated into a linear and a static nonlinearity which is useful in analysis. Systems of the Hammerstein type shown in Figure 3 are among the most common mechanisms for modeling nonlinear systems. The Hammerstein sample system given in Chapter 1 where the linear portion of the system is given in (2), the nonlinear feedforward portion is given in (3) and the nonlinear feedback portion is given in (4) was solved using a technique discussed in [53]. This section will solve the same problem with the fuzzy system identification mechanism described in Chapter 4.

5.6.1 Preliminaries

It is assumed that the system input $u(k)$ and the system output $y(k)$ for $k=1..K$, where K is the number of data points available, are provided. The application of the fuzzy logic technique developed in this work to the Hammerstein system identification problem requires several steps.

1. Estimate the structure of the linear portion of the system, $G(z^{-1})$.
2. Estimate the output of the nonlinearity $f_0(\bullet)$, $u_n(k)$.
3. Using the input and output of the nonlinearity $f_0(\bullet)$, $u(k)$and $u_n(k)$ respectively, approximate it with the descent technique described earlier in this chapter.

Two approaches to the solution of this problem are discussed below. The first assumes the availability of an exact representation of the linear system (a completed step 1 above) and the second discusses a more complete Hammerstein problem solution.

5.6.2 Hammerstein System Solution with a Known Linear System

In a case where the linear portion of the Hammerstein system is predetermined either through another identification technique or a priori knowledge of the system the fuzzy logic solution to the nonlinearity is straightforward. As step 1 from the previous section is already completed, the next step is to estimate the output of the nonlinearity. A simple mechanism is to simply run the given output data, $y(k)$, through an inverse of the known linear system, $G(z^{-1})$. Zero-mean white Gaussian measurement noise with a magnitude of 0.1 was added to this output signal. Having acquired the nonlinearity output, the fuzzy logic descent algorithm presented earlier in this Chapter can be used to identify the nonlinear system which produces the signal $u_n(k)$ when provided $u(k)$. Given that a deadzone nonlinearity was expected, the number of breakpoints, N, was selected to be 2.

The results of this approximation are shown in Figure 30.

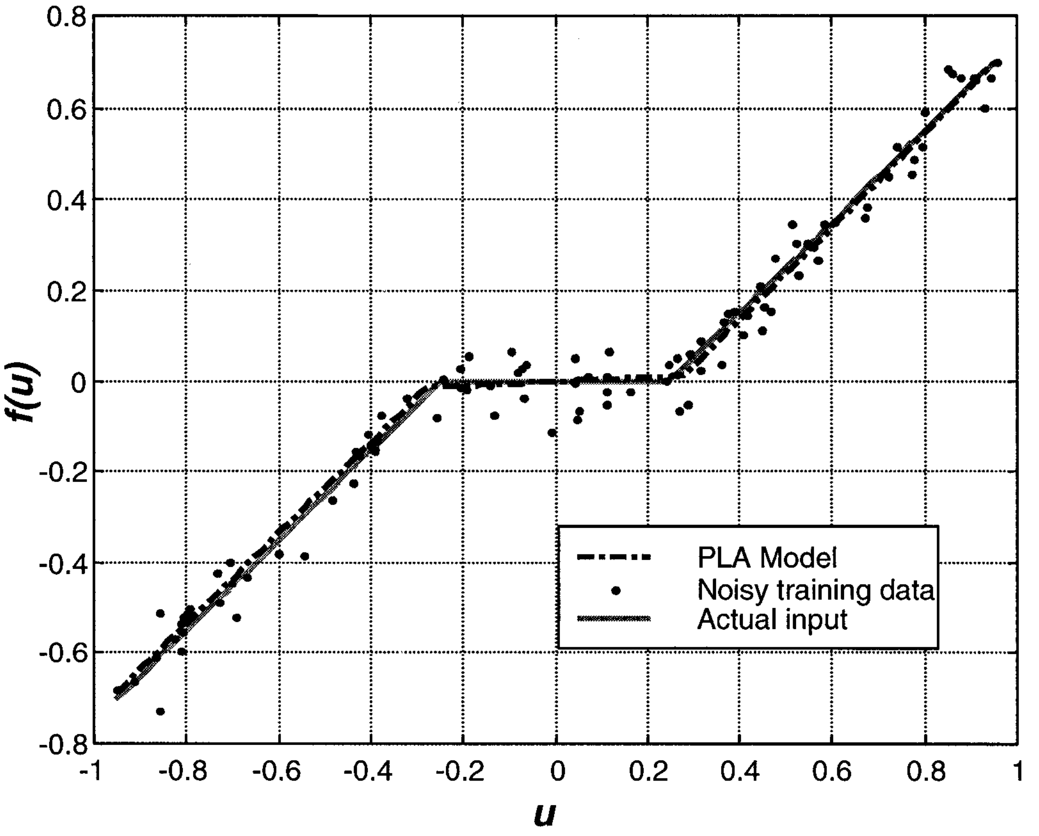

Figure 30 – Approximation of nonlinearity in a Hammerstein system.

The solid line is the actual response of the nonlinearity. The dots are the training data recovered from the original Hammerstein system with the measurement noise applied. The dashed line is the fuzzy approximation of the nonlinearity derived by the descent method described earlier. This figure demonstrates that the fuzzy system is able to very closely approximate a nonlinearity given in a Hammerstein system.

Often, the linear portion of the system will not be explicitly available but will have to be approximated. This situation is investigated in the next section.

5.6.3 Hammerstein System Solution with an Unknown Linear System

The section demonstrates a more complete application of the fuzzy logic technique developed in this work to the Hammerstein system identification problem. The previous section assumed that the linear structure of the Hammerstein model was known. This section develops a more complete identification method that includes identification of an unknown linear component.

For the purpose of this work, a least-squares identifier was used to estimate a linear system from the input/output data provided. The system output was perturbed by zero-mean Gaussian white noise of magnitude 0.01. Because there is a nonlinearity present, it is expected that the linear system will not be well identified. This identification was conducted and yielded:

$$G(z^{-1}) = \frac{0.398 + 0.3812z^{-1}}{1 - 1.689z^{-1} + 0.8399z^{-2}}$$

A plot of this linear estimate versus the original nonlinear system is shown in Figure 31.

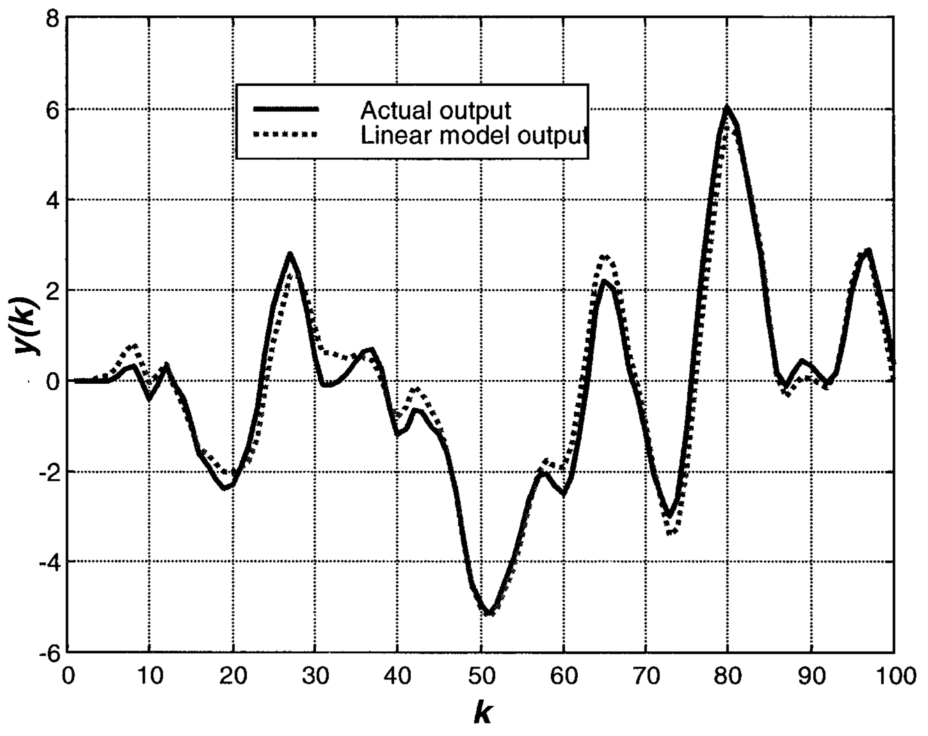

Figure 31 – Linear model versus original nonlinear Hammerstein system.

The linear model exhibits significant deviations from the nonlinear system and has a mean square error of 0.41.

Having derived a linear model, the next step is to estimate the output of the nonlinearity. This is accomplished in the same manner as the previous section, except using the linear model derived in the previous step to estimate the nonlinearity output. With this output estimated, the fuzzy system can be trained on the input/output signals. The plot of the fuzzy approximation of the dead zone nonlinearity is shown in Figure 32.

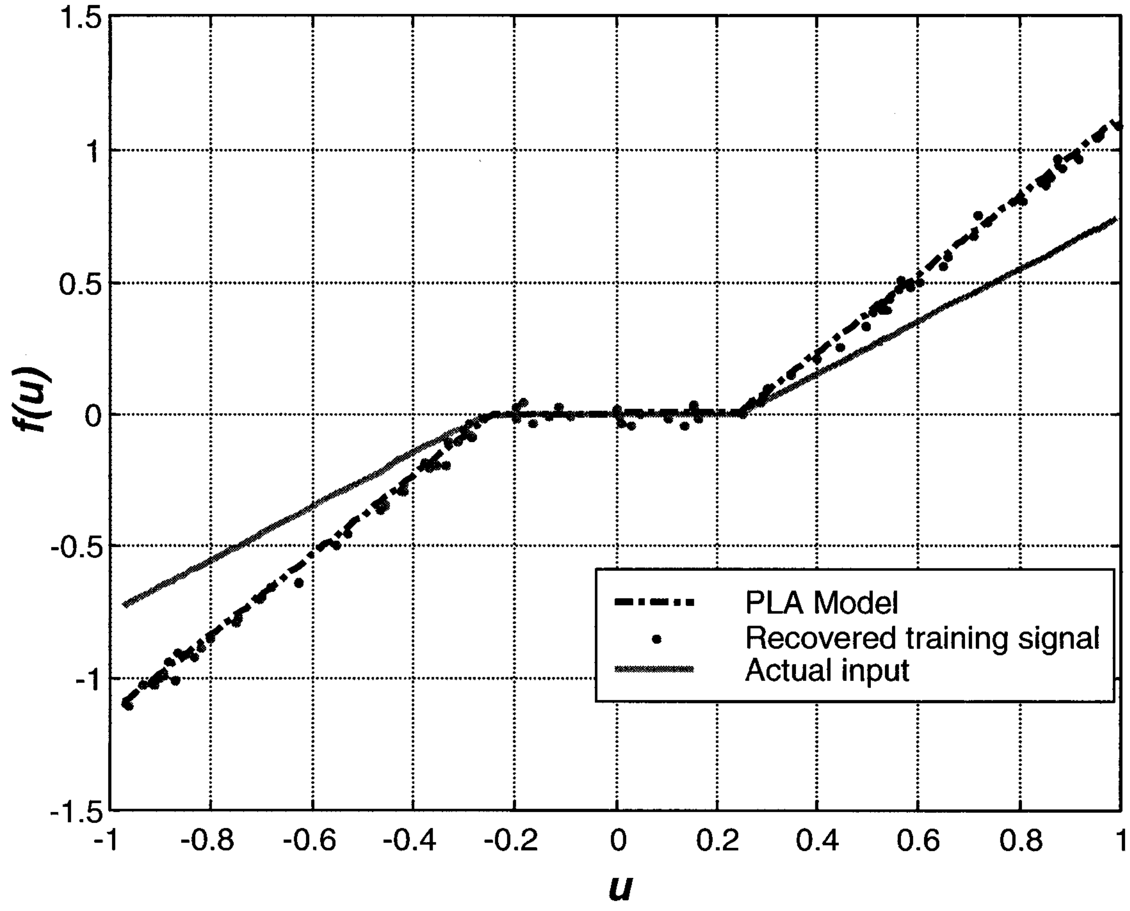

Figure 32 – Fuzzy approximation of deadzone nonlinearity.

The solid line represents the response of the original dead zone nonlinearity. The dots represent the estimated nonlinearity output as determined by running the original output signal through an inverse model of the estimated linear system. The dashed line represents the response of the fuzzy system approximation of the nonlinearity. While the recovered nonlinearity output signal does not match the actual nonlinearity, the fuzzy system is able to learn the recovered signal very well.

As a final check, the output of the initial system was compared to the output of the system with the fuzzy system model of the nonlinearity and the linear estimate of $G(z^{-1})$. The responses of these two systems are shown in Figure 33.

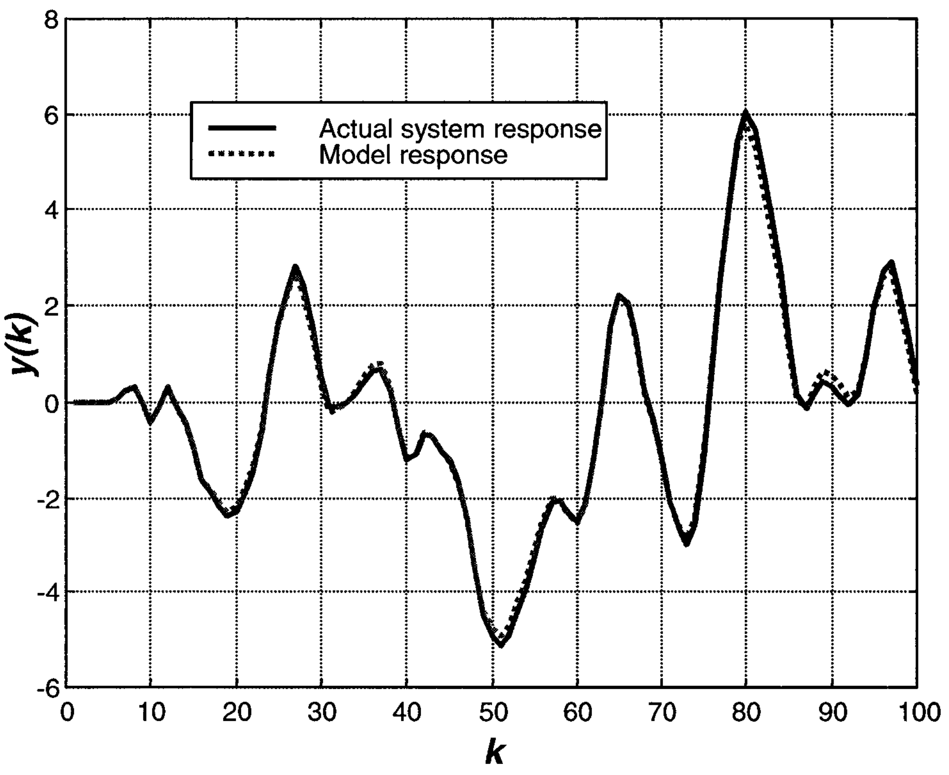

Figure 33 – Comparison of actual Hammerstein system with approximated system.

The solid line represents the response of the actual Hammerstein system whereas the dotted line shows the response of the fuzzy identifier. The result is a good approximation of the system, far superior to that shown in Figure 31. The mean square error of this approximation is 0.13. As a comparison, the solution used in [53] resulted in a mean square error of 0.82 against the training signal. Thus this PLA-based technique has shown almost an order of magnitude improvement over the original method.

5.7 Conclusion

This chapter examined several applications of the theory developed in Chapter 4. An initial implementation based upon an exhaustive search of the solution space was presented and shown to be computationally intractable in its current form. An algorithm based upon a descent technique was developed and applied to several different types of problems. The descent technique was compared to backpropagation neural networks and the relative strengths and weaknesses discussed.

The descent algorithm was also applied to both sample systems introduced in Chapter 1. The descent algorithm demonstrated the ability to approximate a simple nonlinear dynamic system and the ability to tune the quality of approximation with N, the number of breaks, was discussed.

An indirect adaptive control scheme was developed for the Wiener system example presented in the Introduction. It was shown that the fuzzy model of this system could be used to provide a satisfactory result in this IAC example. The maximum approximation error for this sample system was compared to the error bound provided by Theorem 3 and found to perform better than the bound.

Finally, a system identification algorithm was developed for a Hammerstein system and demonstrated that the fuzzy logic system could be trained to approximate a Hammerstein system.

Chief among the outstanding issues is the high computational complexity present in the algorithms developed above. Reducing this complexity is critical to further application of this method. Many applications involve multiple inputs but Chapter 4 addresses only single-input systems. The next chapter examines higher order systems and the issues involved with extending this theory to cover them.

6.0 FUTURE APPLICATIONS, RESEARCH, AND CONCLUSIONS

Virtually any aspect of fuzzy logic can draw from this work to help form and adapt fuzzy systems to meet specific requirements. The fuzzy system identification theory and methods described here can be used to support fuzzy control systems implementations, some of which have been introduced here. Furthermore, these techniques support the use of fuzzy systems in decision support and other artificial intelligence applications.

While the results of this work are powerful, additional research is required in order to more completely and fully utilize these techniques. In particular, research into multi-dimensional extensions, algorithmic improvements, and extensions to smooth membership functions is necessary.

6.1 Fuzzy System Development from Input Data

The simplest and most direct application of this work is to train a fuzzy system to learn the input-output mapping of a system. As demonstrated in the examples, the method described in this work is capable of learning a given input-output mapping to an arbitrary degree of closeness, controlled solely by the choice of N, the number of breaks.

Besides the superficial result of developing a system model, other advantages include the ability to build a fuzzy model of a system that provides a linguistic description of the operation of a system. This could be of great use in trying to understand the underlying principles of operation of complex multi-dimensional systems.

6.2 System Identification

Fuzzy logic has been applied to system identification in many different ways [21, 36, 38, 39, 52, 56]. In its most obvious manifestation, a fuzzy system could observe the inputs to an unknown system as well as an output and learn the mapping over time. However, in this work, the fuzzy learning mechanism is applied to a batch learning process where all the input-output data is provided at once.

6.3 Adaptive Control

A simple example of one-dimensional Indirect Adaptive Control was presented in Section 5.5. However, this barely scratches the surface of the potential applications of this method in the field of adaptive control. This method could also be applied to more sophisticated adaptation mechanisms and provide an analytic approach to the implementation of fuzzy logic in control systems.

6.4 Application to Surveys

Surveys are often used to gather information from people about a topic to help determine a thought process that led them to make a certain decision. Conventional statistical techniques are often used to process that data and they generally use a linear combination of some set of functions to determine how much different factors affect the ultimate decision. This technique offers the possibility of using an inherently nonlinear system to provide a mapping of this human decision system and provide a linguistic explanation of the decision process. In other words, besides just modeling a system, the fuzzy system offers insight into why the system is working the way it is.

6.5 Algorithmic Improvements

One of the most immediately important follow-up topics to this work is the improvement of the algorithm with which the fuzzy system is trained. Two algorithms were used in support of this work. The first was a 'brute-force' method that checked each vertex on the error surface and selected the best one. While this method provides a guaranteed optimal solution, the computational complexity involved is overwhelming for all but the most simple problem. The second method developed is a descent technique whereby a solution is iteratively improved until some local minimum is reached. While this method does not produce a guaranteed optimal solution, it reduces the computational requirements many orders of magnitude. Nonetheless, even the descent technique becomes too demanding before any truly significant problems can be tackled.

A complete and in-depth review of the mechanisms used to arrive at solutions with this method is required. There are probably techniques that will significantly reduce the computational complexity of both the full-optimization as well as the descent technique. Discovery and implementation of improved techniques will make this mechanism realizable in a meaningful application.

6.5.1 On-Line Learning

One possible avenue of improving algorithm performance is to implement some method of on-line learning.

6.6 Complexity Reduction

Besides strict algorithmic improvements, which are probably necessary for the long term usefulness of this technique, there are other mechanisms to reduce complexity and thereby improve performance.

One that has already been briefly examined is the partitioning of the solution space. Because the complexity of this process is exponential, dividing the problem into two pieces provides a drastic performance gain. For multidimensional problems, this gain becomes even greater if multiple dimensions can be partitioned. The challenge that arises in these cases is where to create the partition(s).

In some cases, the partition selection is natural. If a system has a set point around which it operates, that point is a natural one at which to insert a breakpoint and partition the problem into two halves.

Many control system problems offer another very natural way of partitioning a problem. Any system that is symmetric can immediately be partitioned around the axis of symmetry. Consider a two-input system where each input has five breakpoints. This fuzzy system will have 35 adjustable parameters. However, if the system is symmetric in both input dimensions, this number is reduced to 15 adjustable parameters. Considering that the number of vertices increases exponentially by the number of adjustable parameters, this improvement is tremendous.

6.7 Quantized Inputs

Some systems, particularly survey type questions, have inputs that are quantized. Typical of these are answers given on a scale of 1 through 5 or other multiple-choice questions often found on surveys. While this method will work well on results that are quantized over many different levels, inputs quantized over 5 or even ten levels will have a roughness that will not be easily learned by the fuzzy system. A fuzzy system that can handle the discontinuities present in quantized systems is required.

6.8 Alternate Input Membership Functions

The input MFs used in the development of this theory were restricted triangular membership functions that yielded the piecewise linear approximator necessary for the development of the Theorem 9. However, real systems are generally not discontinuous in the first derivative as are the PLAs. It would be desirable to extend this theory to a system that was smooth. There are several avenues available to investigate these features. The first would be to examine input MFs that were constructed of parabolic pieces vice linear pieces. These pieces can be used to construct a piecewise approximator whose pieces are made up of 2-degree curves vice the 1-degree curves used by the PLA.

A second method is to convert the triangular membership functions set by the methods described herein and use them to determine parameters of a similar, but curved MF (such as the exponential). While the optimality of the solution would be lost, some smoothness would be gained that might better approximate real systems.

6.9 Error Bound

As discussed in Theorem 3, the maximum error of a PLA is fairly conservatively bounded by a function relating to the modulus of continuity of the function to be approximated and the maximum space between breaks. A better error bound that relies primarily upon the number of breaks of the system needs to be developed.

6.10 Multi-Input Systems

Many applications of fuzzy logic will require systems with multi-dimensional inputs. This work applies immediately to single input systems. Further effort is required to extend this to multi-dimensional systems. A beginning of this effort is included as Appendix A.

6.11 Conclusions

This work was driven by the need to use a fuzzy system to approximate an arbitrary nonlinear function given a set of input-output data. While there exist a number of algorithms that can perform this function, this work focused on a method that was based on a technique to determine parameters of a piecewise linear approximator that would optimally minimize the error between the approximator and the given data. The end goal of this work was to apply this result to the identification of a Wiener and a Hammerstein nonlinear system.

In the course of this work several important results were uncovered:

1) A set of parameters exists that minimize the error between a piecewise linear approximator and an arbitrary set of input-output data points.

2) This set of optimal parameters exist among a finite set of possible parameters.

3) An algorithm was designed that takes advantage of this optimality result but trades the optimality guarantee for a massive decrease in computational complexity.

4) This algorithm was applied to the identification of a Wiener and a Hammerstein system and provided excellent results.

7.0 REFERENCES

[1] A. M. Ahmed, "BPD computation and model reference adaptive control (MRAC) of Hammerstein plants", *IEE Proceedings - Control Theory Appli*cations, vol. 142, pp. 475-485, 1995.

[2] Mark Akey, Kirk Dunkelberger, and Richard C. Erdman, "Case Studies in Tactical Decision Support Systems", *Artificial Intelligence and National Defense: Applications to C3I and Beyond*, AFCEA International Press, Washington DC, 1987.

[3] Edoardo Amaldi and Marco Matttavelli, *A Combinatorial Optimization Approach to Extract Piecewise Linear Structure from Nonlinear Data and an Application to Optical Flow Segmentation*, Swiss Federal Institute of Technology, Lausanne, 1997.

[4] Edoardo Amaldi, Marco Mattavelli, and Jean-Marc Vesin, *A Perceptron-Based Approach to Piecewise Linear Modeling with an Application to Time Series*, Swiss Federal Institute of Technology, Lausanne, 1995.

[5] Karl Johan Åström and Björn Wittenmark, *Adaptive Control*, Addison-Wesley Publishing Company, Reading, MA, 1989.

[6] S. A. Billings and W. S. F. Voon, "Piecewise Linear Identification of Nonlinear Systems", *Automatica*, vol. 46, pp. 215-235, 1987.

[7] E. W. Cheney, *Approximation Theory*, American Mathematical Society, Providence, RI, 1982.

[8] Kwang Bo Cho, Bo Hyeun Wang, "Radial basis function based adaptive fuzzy systems and their application to system identification and prediction", *Fuzzy Sets and Systems*, Vol. 83, No. 3, 1996.

[9] Leon O. Chua and Robin L. P. Ying, "Canonical Piecewise-Linear Analysis", *IEEE Transactions on Circuits and Systems*, vol. 30, pp. 125-140, 1983.

[10] Leon O. Chua and An-Chang Deng, "Canonical Piecewise-Linear Modeling", *IEEE Transactions on Circuits and Systems*, vol. 33, pp. 511-525, 1986.

[11] Leon O. Chua and An-Chang Deng, "Canonical Piecewise-Linear Representation", *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 101-111, 1988.

[12] H. A. E. de Bruin and B. Roffel, *A New Identification Method for Fuzzy Linear Models of Nonlinear Dynamic Systems*, *Journal of Process Control*, vol. 6, pp. 277-293, 1996.

[13] Carl DeBoor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[14] Paul Dierckx, *Curve and Surface Fitting with Splines*. Clarendon Press, Oxford, 1993.

[15] Thomas Guthrie Weidner Epperly, *Global Optimization of Nonconvex Nonlinear Programs Using Parallel Branch and Bound*, University of Wisconsin, Madison, 1995.

[16] J. W. Herrman, *A Genetic Algorithm for Minimax Optimization*, Institute for Systems Research, College Park, MD, 1997.

[17] Chin-Chih Hsu, et al, "Optimization of adaptive control rule of dead time system by genetic algorithm", *Proceedings of the IEEE International Symposium on Industrial Electronics*, 10-14 July 1995.

[18] Peter J. Huber, "Projection Pursuit", *The Annals of Statistics*, vol. 13, pp. 435-475, 1985.

[19] C. L. Hwang, "Nonlinear control design for a Hammerstein model system", *IEE Proceedings - Control Theory Applications*, vol. 142, pp. 277-285, 1995.

[20] Petros A. Ioannou and Jing Sun, *Robust Adaptive Control*, Prentice Hall, Inc., Upper Saddle River, NJ, 1996.

[21] S. Jagannathan, "Adaptive Fuzzy Control of Feedback Linearizable Discrete-Time Nonlinear Systems", *Proceedings of the 1996 International Symposium on Intelligent Control* (Dearborne, MI), pp. 133-138, September 1996.

[22] Anatoli Juditsky, et al, "Nonlinear Black-box Models in System Identification: Mathematical Foundations", *Automatica*, vol. 31, pp. 1725-1750, 1995.

[23] Claus Kahlert and Leon O. Chua, "A Generalized Canonical Piecewise-Linear Representation", *IEEE Transactions on Circuits and Systems*, vol. 37, pp. 373-383, 1990.

[24] Claus Kahlert and Leon O. Chua, "The Complete Canonical Piecewise-Linear Representation - Part I: The Geometry of the Domain Space", *IEEE Transactions on Circuits and Systems*, vol. 39, pp. 222-236, 1992.

[25] S. M. Kang and Leon O. Chua, "A Global Representation of Multidimensional Piecewise-Linear Functions with Linear Partitions", *IEEE Transactions on Circuits and Systems*, vol. 25, pp. 938-940, 1978.

[26] Bart Kosko, "Fuzzy Systems as Universal Approximators", *IEEE Transactions on Computers*, vol. 43, pp. 1329-1333, 1994.

[27] Andreas Kroll, "Partition Identification of Fuzzy Models using Objective Function Clustering Algorithms", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vancouver, BC, pp. 7-12, October 22-25, 1995.

[28] Domine M. W. Leenaerts and Wim M. G. van Bokhoven, *Piecewise Linear Modeling and Analysis*, Eindhoven University of Technology, Eindhoven.

[29] Wei Li, et al, "Automatic tuning of a fuzzy logic controller using neural network", *Proceedings of the IEEE International Conference on Industrial Technology*, 5-9 December, 1994.

[30] M. Locatelli, *Simulated Annealing Algorithms for Continuous Optimization*, Univesita di Torino, Torino, 2000.

[31] G. G. Lorentz, "The 13th Problem of Hilbert", *Mathematical Problems Arising From Hilbert Problems*, American Mathematical Society, Providence, RI, 1976.

[32] Ebrahim H. Mamdani, "Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis", *IEEE Transactions on Computers*, vol. 26, pp. 1182-1191, 1977.

[33] K. Z. Mao and S. A. Billings, "Algorithms For Minimal Model Structure Detection in Nonlinear Dynamic System Identification", *International Journal of Control*, vol. 68, pp. 311-330, 1997.

[34] Marco Mattavelli and Edoardo Amaldi, *Estimating Piecewise Linear Models Using Combinatorial Optimization Techniques*, Swiss Federal Institute of Technology, Lausanne, 1996.

[35] D. McAllester and others, *Three Cuts for Accelerated Interval Propagation*, MIT, Boston, 1995.

[36] Raúl Ordóñez, Jon Zumberge, Jeffrey T. Spooner and Kevin M. Passino, "Adaptive Fuzzy Control: Experiments and Comparative Analyses", *IEEE Transactions on Fuzzy Systems*, Vol 5, pp. 167-188, May 1997.

[37] G. A. Pajunen, "Recursive Identification of Wiener Type Nonlinear Systems", *Proceedings of the American Control Conference*, Boston, MA, pp. 1365-1370, 1985.

[38] Sireesh Kumar Pandey, *A Self Tuning Fuzzy Controller, Computational Intelligence: Theory and Applications: International Conference, 5th Fuzzy Days*, Dortmund Germany, 1997.

[39] Kevin M. Passino and Stephen Yukovich, *Fuzzy Control*, Addison Wesley Longman, Inc, Menlo Park, CA, 1998

[40] N. B. O. L. Petit, *Analysis of Piecewise Linear Dynamical Systems*, Wiley, New York, 1985.

[41] Marios M. Polycarpou and Mark J. Mears, "Stable Adaptive Tracking of Uncertain Systems Using Nonlinearly Parameterized On-line Approximators", *International Journal of Control*, vol. 70, pp. 363-384, 1998.

[42] M. J. D. Powell, *Approximation Theory and Methods*, Cambridge University Press, Cambridge, 1981.

[43] John R. Rice, "Multivariate Piecewise Polynomial Approximation", *Multivariate Approximation*, Edited by D. C. Handscomb, Academic Press, London, 1978.

[44] Riccardo Rovatti, "Fuzzy Piecewise Multilinear and Piecewise Linear Systems as Universal Approximators in Sobolev Norms", *IEEE Transactions on Fuzzy Systems*, vol. 6, pp. 235-249, 1998.

[45] Andrew P. Sage, "Knowledge Support Needs in C3I Systems", *Artificial Intelligence and National Defense: Applications to C3I and Beyond*, AFCEA International Press, Washington DC, 1987.

[46] Larry L. Schumaker, *Spline Functions: Basic Theory*, John Wiley & Sons, New York, 1981.

[47] Georgi E. Shilov, *Elementary Real and Complex Analysis*, Massachusetts Institute of Technology, Boston, 1973.

[48] J. Shi and H. H. Sun, "Nonlinear Systems Identification for Cascaded Block Model", *IEEE Transactions on Biomedical Engineering*, vol. 37, pp. 574-587, 1990.

[49] Yan Shi, et al, "A Learning Algorithm for Tuning Fuzzy Rules Based on the Gradient Descent Method", *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, September 8-11, 1996.

[50] Eduardo D. Sontag, "Nonlinear Regulation: The Piecewise Linear Approach", *IEEE Transactions on Automatic Control*, vol. 26, pp. 346-358, 1981.

[51] V. Staudt, "Compact Representation of Mathematical Functions for Control Applications by Piecewise Linear Approximations", *Electrical Engineering*, vol. 81, pp. 129-134, 1998.

[52] Jeffrey T. Spooner and Kevin M. Passino, "Stable Indirect Adaptive Control Using Fuzzy Systems and Neural Networks", *Proceedings of the 34th Conference on Decision and Control* (New Orleans, LA), pp. 243-248, December 1995.

[53] Tobin H. Van Pelt and Dennis S. Bernstein, "Non-linear system identification using Hammerstein and non-linear feedback models with piecewise linear static maps", *International Journal of Control*, vol. 74, pp. 1807-1823, 2001.

[54] H. Wang, et al, *Advanced Adaptive Control*, Elsevier Science Ltd., Oxford, 1995.

[55] Li-Xin Wang and Jerry M. Mendel, "Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least Squares Learning", *IEEE Transactions on Neural Networks*, vol. 3, pp. 807-814, 1992.

[56] Li-Xin Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, PTR Prentice Hall, Englewood Cliffs, NJ, 1994.

[57] D. G. Wilson, "Piecewise Linear Approximation Subroutine", *ACM Transactions on Mathematical Software*, 1976.

[58] Lotfi A. Zadeh, "Fuzzy Sets", *Information and Control*, Vol 8, pp. 338-353, 1965.

[59] Jacek M. Zurada, *Introduction to Artificial Neural Systems*, West Publishing Co., St. Paul, MN, 1992.

APPENDIX A TWO-DIMENSIONAL FUZZY SYSTEMS AS APPROXIMATORS

This Appendix extends the theory developed in Chapter 4 from single-dimensional fuzzy systems to two-dimensional systems. The primary challenge introduced is the inclusion of a fuzzy conjunction as part of the fuzzy consequence. Several different approaches to solving this problem are investigated and the difficulties are discussed. A more promising path is opened and discussed.

A.1 Two-Dimensional Fuzzy Systems

At first glance extending the results from one-dimensional to two-dimensional fuzzy systems would involve a straightforward extension of the mechanism worked out for the single dimensional case. However, two (and higher) dimensional fuzzy systems require an operation not required in the single dimension case; that of the fuzzy conjunction. A one-dimensional fuzzy if-then statement can be written as:

IF x THEN y

where x and y are a fuzzy input and output respectively. However two and higher dimensional system must be written as:

IF x_1 AND x_2 AND … AND x_N THEN y

where $x_1, x_2, \ldots, x_N$ are the fuzzy inputs, y is the fuzzy output, and N is the number of inputs to the system. In order to convert these into a higher order PLA, the fuzzy conjunction operator (AND) must be accounted for.

A.2 Two-Dimensional Fuzzy Systems with a Product Conjunction

Consider the two-dimensional fuzzy system constructed of two input fuzzy sets, each of the form shown in Figure 10. The vertices of the MFs of the first fuzzy set are given by $d_{11}, d_{12}, \ldots, d_{1N_1}$ where N_1 is the number of MFs in the first fuzzy set. Likewise the vertices of the second fuzzy set are given by $d_{21}, d_{22}, \ldots, d_{2N_2}$ where N_2 is the number of MFs in the second fuzzy set. The fuzzy consequence matrix contains members $c_{11}, \ldots, c_{N_1N_2}$ where member $c_{n_1n_2}$ is associated with the intersection of the MFs whose vertices are d_{n_1} and d_{n_2}.

For any $d_{1n_1} \le x_1 \le d_{1n_1+1}$ and $d_{2n_2} \le x_2 \le d_{2n_2+1}$ there are at most four consequences active: $c_{n_1n_2}$, $c_{n_1+1n_2}$, $c_{n_1n_2+1}$, and $c_{n_1+1n_2+1}$. By expanding the summation, (5) is reduced to:

$$y = \frac{c_{n_1n_2}a_{n_1n_2} + c_{n_1+1n_2}a_{n_1+1n_2} + c_{n_1n_2+1}a_{n_1n_2+1} + c_{n_1+1n_2+1}a_{n_1+1n_2+1}}{a_{n_1n_2} + a_{n_1+1n_2} + a_{n_1n_2+1} + a_{n_1+1n_2+1}} \tag{33}$$

$a_{n_1n_2}$ is the result of the fuzzy conjunction between the fuzzified value of x_1 and x_2. Using the product operator for the fuzzy conjunction yields:

$$a_{n_1n_2} = f_{1n_1}(x_1)f_{2n_2}(x_2) \tag{34}$$

The function $f_{1n_1}(x_1)$ is the value of the n_1th triangle where $d_{1n_1} \le x_1 \le d_{1n_1+1}$ and is equal to:

$$f_{1n_1}(x_1) = \frac{x_1 - d_{1n_1}}{d_{1n_1+1} - d_{1n_1}} \tag{35}$$

Likewise, the function $f_{2n_2}(x_2)$ is the value of the n_2th triangle where $d_{2n_2} \le x_2 \le d_{2n_2+1}$ and is equal to:

$$f_{2n_2}(x_2) = \frac{x_2 - d_{2n_2}}{d_{2n_2+1} - d_{2n_2}} \tag{36}$$

Just as in the one-dimensional case, with x_1 and x_2 bounded between two apexes the following is true:

$$f_{1n_1+1}(x_1) = 1 - f_{1n_1}(x_1)$$
$$f_{2n_2+1}(x_2) = 1 - f_{2n_2}(x_2)$$

Given this, and substituting the symbol f_1 for $f_{1n_1}(x_1)$ and f_2 for $f_{2n_2}(x_2)$ the denominator of (33) simplifies to:

$$\begin{aligned} &= f_1f_2 + (1-f_1)f_2 + f_1(1-f_2) + (1-f_1)(1-f_2) \\ &= f_1f_2 + f_2 - f_1f_2 + f_1 - f_1f_2 + 1 - f_1 - f_2 + f_1f_2 \\ &= 1 \end{aligned} \tag{37}$$

Thus (33) becomes:

$$y = c_{n_1n_2}a_{n_1n_2} + c_{n_1+1n_2}a_{n_1+1n_2} + c_{n_1n_2+1}a_{n_1n_2+1} + c_{n_1+1n_2+1}a_{n_1+1n_2+1} \tag{38}$$

Substituting (35) and (36) into (39) and further substituting this into (38) yields:

$$y = c_{n_1 n_2}\left(\frac{-x_1 + d_{1n_1+1}}{d_{1n_1+1} - d_{1n_1}}\right)\left(\frac{-x_2 + d_{2n_2+1}}{d_{2n_2+1} - d_{2n_2}}\right) + c_{n_1+1n_2}\left(\frac{x_1 - d_{1n_1}}{d_{1n_1+1} - d_{1n_1}}\right)\left(\frac{-x_2 + d_{2n_2+1}}{d_{2n_2+1} - d_{2n_2}}\right) +$$
$$c_{n_1n_2+1}\left(\frac{-x_1 + d_{1n_1+1}}{d_{1n_1+1} - d_{1n_1}}\right)\left(\frac{x_2 - d_{2n_2}}{d_{2n_2+1} - d_{2n_2}}\right) + c_{n_1+1n_2+1}\left(\frac{x_1 - d_{1n_1}}{d_{1n_1+1} - d_{1n_1}}\right)\left(\frac{x_2 - d_{2n_2}}{d_{2n_2+1} - d_{2n_2}}\right)$$

Expanding the products and gathering like terms yields:

$$y = \frac{1}{\left(d_{1n_1+1} - d_{1n_1}\right)\left(d_{2n_2+1} - d_{2n_2}\right)}\begin{bmatrix} \left(c_{n_1n_2} - c_{n_1+1n_2} - c_{n_1n_2+1} + c_{n_1+1n_2+1}\right)x_1x_2 + \\ \left(-c_{n_1n_2}d_{2n_2+1} + c_{n_1+1n_2}d_{2n_2+1} + c_{n_1n_2+1}d_{2n_2} - c_{n_1+1n_2+1}d_{2n_2}\right)x_1 + \\ \left(-c_{n_1n_2}d_{1n_1+1} + c_{n_1+1n_2}d_{1n_1} + c_{n_1n_2+1}d_{1n_1+1} - c_{n_1+1n_2+1}d_{1n_1}\right)x_2 + \\ c_{n_1n_2}d_{1n_1+1}d_{2n_2+1} - c_{n_1+1n_2}d_{1n_1}d_{2n_2+1} \\ - c_{n_1n_2+1}d_{1n_1+1}d_{2n_2} + c_{n_1+1n_2+1}d_{1n_1}d_{2n_2} \end{bmatrix} \tag{39}$$

This function is collinear in x_1 and x_2 and everywhere continuous and is therefore suitable for transformation into a form similar to that in the one-dimensional case.

A.3 Transformation of Two-Dimensional Fuzzy Systems into Piecewise Linear Approximators

The first step in the transformation is to determine a suitable form for the PLA system as was done in the one-dimensional case. The form is constrained in that it must be:

- Continuous,
- Linearizable,
- Capable of matching the fuzzy logic function above, and
- Containing the same number of independent variables (N_1N_2 c_{n1n2}, N_1 d_{1n1}, and N_2 d_{2n2} terms) as (39).

A function meeting these criteria is:

$$f(\cdot) = \sum_{m_1=1}^{N_1}\sum_{m_2=1}^{N_2} a_{m_1m_2}\left|x_1 - d_{1m_1}\right|\left|x_2 - d_{2m_2}\right| \tag{40}$$

For any given $d_{1n_1} \le x_1 \le d_{1n_1+1}$ and $d_{2n_2} \le x_2 \le d_{2n_2+1}$ the signs of $x_1 - d_{1m}$ and $x_2 - d_{2m}$ are known and (40) can be rewritten as:

$$\begin{aligned} f(\cdot) = &\sum_{m_1=1}^{n_1}\sum_{m_2=1}^{n_2} a_{m_1m_2}\left(x_1 - d_{1m_1}\right)\left(x_2 - d_{2m_2}\right) - \sum_{m_1=1}^{n_1}\sum_{m_2=n_2+1}^{N_2} a_{m_1m_2}\left(x_1 - d_{1m_1}\right)\left(x_2 - d_{2m_2}\right) \\ &- \sum_{m_1=n_1+1}^{N_1}\sum_{m_2=1}^{n_2} a_{m_1m_2}\left(x_1 - d_{1m_1}\right)\left(x_2 - d_{2m_2}\right) + \sum_{m_1=n_1+1}^{N_1}\sum_{m_2=n_2+1}^{N_2} a_{m_1m_2}\left(x_1 - d_{1m_1}\right)\left(x_2 - d_{2m_2}\right) \end{aligned}$$

Cross-multiplication yields:

$$\begin{aligned} f(\cdot) = &\sum_{m_1=1}^{n_1}\sum_{m_2=1}^{n_2} a_{m_1m_2}x_1x_2 - a_{m_1m_2}d_{2m_2}x_1 - a_{m_1m_2}d_{1m_1}x_2 + a_{m_1m_2}d_{1m_1}d_{2m_2} \\ &- \sum_{m_1=1}^{n_1}\sum_{m_2=n_2+1}^{N_2} a_{m_1m_2}x_1x_2 - a_{m_1m_2}d_{2m_2}x_1 - a_{m_1m_2}d_{1m_1}x_2 + a_{m_1m_2}d_{1m_1}d_{2m_2} \\ &- \sum_{m_1=n_1+1}^{N_1}\sum_{m_2=1}^{n_2} a_{m_1m_2}x_1x_2 - a_{m_1m_2}d_{2m_2}x_1 - a_{m_1m_2}d_{1m_1}x_2 + a_{m_1m_2}d_{1m_1}d_{2m_2} \\ &+ \sum_{m_1=n_1+1}^{N_1}\sum_{m_2=n_2+1}^{N_2} a_{m_1m_2}x_1x_2 - a_{m_1m_2}d_{2m_2}x_1 - a_{m_1m_2}d_{1m_1}x_2 + a_{m_1m_2}d_{1m_1}d_{2m_2} \end{aligned} \tag{41}$$

In order to apply Theorem 9 to this system, (41) must be expressed as a system linear in it's adjustable parameters and the number of parameters in the linearized system must be the same as in (40) which has $N_1+N_2+N_1 \bullet N_2$ adjustable parameters.

Herein lies the primary obstacle to solving this problem. Some sort of transformation must be performed on this equation (or some other equation suitable for representing the 2 dimensional fuzzy system) to allow it to be described with the appropriate number of parameters.

A.4 Two-Dimensional Fuzzy Systems with a Min Conjunction

An alternative to the product fuzzy-AND operator is the min(•) function. Instead of (39) being used for the activation function, the following equation is used:

$$a_{n_1n_2} = \min\left(f_{1n_1}(x_1), f_{2n_2}(x_2)\right) \tag{42}$$

Unlike the product case discussed above, the min(•) function does not reduce the denominator of the centroid calculation (33) to one. Instead, the denominator is a nonlinear function of x_1 and x_2 given below.

$$\begin{aligned} denom &= a_{n_1 n_2} + a_{n_1 n_2+1} + a_{n_1+1 n_2} + a_{n_1+1 n_2+1} \\ &= \min\left(f_{1n_1}(x_1), f_{2n_2}(x_2)\right) + \min\left(f_{1n_1}(x_1), f_{2n_2+1}(x_2)\right) \\ &\quad + \min\left(f_{1n_1+1}(x_1), f_{2n_2}(x_2)\right) + \min\left(f_{1n_1+1}(x_1), f_{2n_2+1}(x_2)\right) \end{aligned}$$

A plot of this denominator function for a typical two-dimensional system is shown in Figure 34.

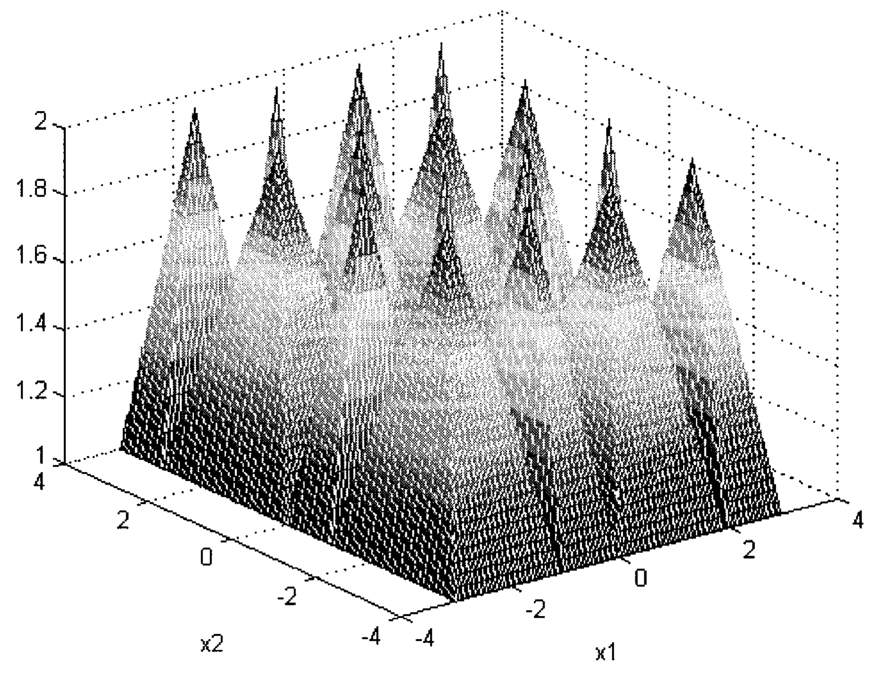

Figure 34 – Denominator of centroid calculation of 2D fuzzy system with min(•) fuzzy-AND.

The presence of this non-constant denominator term greatly complicates analysis. In order to simplify the system, a brief study in the feasibility of developing an alternate defuzzification equation that did not yield a denominator was conducted. The first case examined was one in which the denominator of (33) was set to one (mirroring what occurs with product fuzzy-AND defuzzifier discussed above). This modification resulted in a significant impairment in the ability of the system to model given data. A plot of the response of a fuzzy system with a min(•) fuzzy AND operation is shown in Figure 35 both with and without the denominator term from the defuzzifier. The artifacts from the missing denominator are clearly apparent. Furthermore, the magnitude of the artifacts grows as min(x_1, x_2) gets further from zero. Essentially, this simplified system has lost its ability to approximate generalized systems. In order to counteract this loss, an examination was made as to whether an approximation could be made of these artifacts that would not include the complexity of the variables in the denominator.

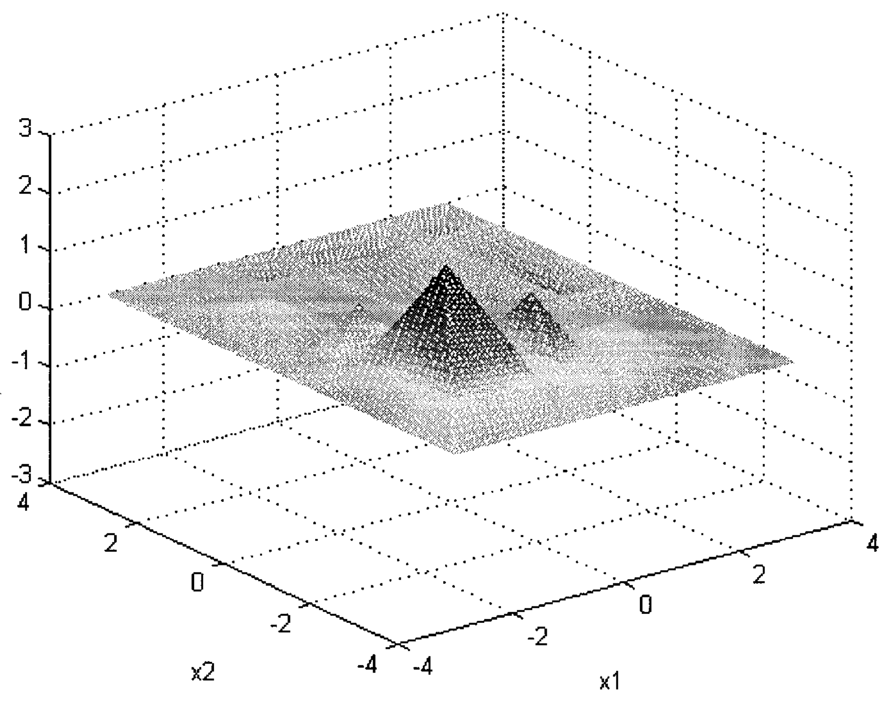

Figure 35 – Difference between min-fuzzy systems with and without the denominator.

It is noted that wherever $x_1=d_{1n1}$ or $x_2=d_{2n2}$, the denominator of the defuzzification equation (33) is one and therefore the difference between the standard and the simplified methods is zero. The error is at a maximum wherever both $x_1$and x_2 are equidistant from two adjacent breakpoints and is equal to:

$$Error = \frac{c_{n_1n_2}a_{n_1n_2} + c_{n_1n_2+1}a_{n_1n_2+1} + c_{n_1+1n_2}a_{n_1+1n_2} + c_{n_1+1n_2+1}a_{n_1+1n_2+1}}{2} \qquad (43)$$

At a point equidistant between two breakpoints:

$$a_{n_1n_2} = a_{n_1n_2+1} = a_{n_1+1n_2} = a_{n_1+1n_2+1} = 0.5 \qquad (44)$$

Substituting (44) into (43) yields:

$$Error_{\max} = 0.25\left(c_{n_1n_2} + c_{n_1n_2+1} + c_{n_1+1n_2} + c_{n_1+1n_2+1}\right) \qquad (45)$$

Therefore, the error in the region $d_{1n1} \leq x_1 \leq d_{1n1+1}$, $d_{2n2} \leq x_2 \leq d_{2n2+1}$ varies from 0 where $x_1=d_{1n1}$, $x_1=d_{1n1+1}$, $x_2=d_{2n2}$, *or* $x_2=d_{2n2+1}$ to (45) where $x_1=(d_{1n1+1}-d_{1n1})/2$ and $x_2=(d_{2n2+1}-d_{2n2})/2$. A model of this error is:

$$e_m = 0.25(c_{n_1n_2} + c_{n_1n_2+1} + c_{n_1+1n_2} + c_{n_1+1n_2+1})(a_{n_1n_2} + a_{n_1n_2+1} + a_{n_1+1n_2} + a_{n_1+1n_2+1} - 1) \qquad (46)$$

This model perfectly approximates the error between the simplified model and the original defuzzification equation at its minimum ($x_1=d_{1n1}$, $x_1=d_{1n1+1}$, $x_2=d_{2n2}$, *or* $x_2=d_{2n2+1}$) and at its maximum ($x_1=(d_{1n1+1}-d_{1n1})/2$ and $x_2=(d_{2n2+1}-d_{2n2})/2$) and is a linear interpolation between them. The simplified defuzzification equation resulting from subtracting (46) from the simplified defuzzification equation (33) modified with its denominator set to one) is:

$$\begin{aligned} y = {} & c_{n_1n_2}a_{n_1n_2} + c_{n_1n_2+1}a_{n_1n_2+1} + c_{n_1+1n_2}a_{n_1+1n_2} + c_{n_1+1n_2+1}a_{n_1+1n_2+1} - \\ & 0.25(c_{n_1n_2} + c_{n_1n_2+1} + c_{n_1+1n_2} + c_{n_1+1n_2+1})(a_{n_1n_2} + a_{n_1n_2+1} + a_{n_1+1n_2} + a_{n_1+1n_2+1} - 1) \end{aligned}$$

Expanding the product and gathering the a_{n1n2} terms yields:

$$\begin{aligned} y = {} & \left(0.75c_{n_1n_2} - 0.25c_{n_1n_2+1} - 0.25c_{n_1+1n_2} - 0.25c_{n_1+1n_2+1}\right)a_{n_1n_2} + \\ & \left(-0.25c_{n_1n_2} + 0.75c_{n_1n_2+1} - 0.25c_{n_1+1n_2} - 0.25c_{n_1+1n_2+1}\right)a_{n_1n_2+1} + \\ & \left(-0.25c_{n_1n_2} - 0.25c_{n_1n_2+1} + 0.75c_{n_1+1n_2} - 0.25c_{n_1+1n_2+1}\right)a_{n_1+1n_2} + \\ & \left(-0.25c_{n_1n_2} - 0.25c_{n_1n_2+1} - 0.25c_{n_1+1n_2} + 0.75c_{n_1+1n_2+1}\right)a_{n_1+1n_2+1} + \\ & 0.25(c_{n_1n_2} + c_{n_1n_2+1} + c_{n_1+1n_2} + c_{n_1+1n_2+1}) \end{aligned}$$

Substituting (42) yields:

$$\begin{aligned}
y = &\left(0.75c_{n_1n_2} - 0.25c_{n_1n_2+1} - 0.25c_{n_1+1n_2} - 0.25c_{n_1+1n_2+1}\right)\min\left(f_{1n_1}(x_1), f_{2n_2}(x_2)\right) + \\
&\left(-0.25c_{n_1n_2} + 0.75c_{n_1n_2+1} - 0.25c_{n_1+1n_2} - 0.25c_{n_1+1n_2+1}\right)\min\left(f_{1n_1}(x_1), f_{2n_2+1}(x_2)\right) + \\
&\left(-0.25c_{n_1n_2} - 0.25c_{n_1n_2+1} + 0.75c_{n_1+1n_2} - 0.25c_{n_1+1n_2+1}\right)\min\left(f_{1n_1+1}(x_1), f_{2n_2}(x_2)\right) + \\
&\left(-0.25c_{n_1n_2} - 0.25c_{n_1n_2+1} - 0.25c_{n_1+1n_2} + 0.75c_{n_1+1n_2+1}\right)\min\left(f_{1n_1+1}(x_1), f_{2n_2+1}(x_2)\right) + \\
&0.25(c_{n_1n_2} + c_{n_1n_2+1} + c_{n_1+1n_2} + c_{n_1+1n_2+1})
\end{aligned} \tag{47}$$

The arguments of the min function above always produce an affine result in both x_1 or x_2 as can be seen from the definitions of the component functions given in (35) and (36). Therefore, the results of this simplified defuzzification function provide both a reasonable approximation and one that is piecewise-linear.

The difference in output of the original min-inference given by equation (33) and that of the simplified model is shown in Figure 36.

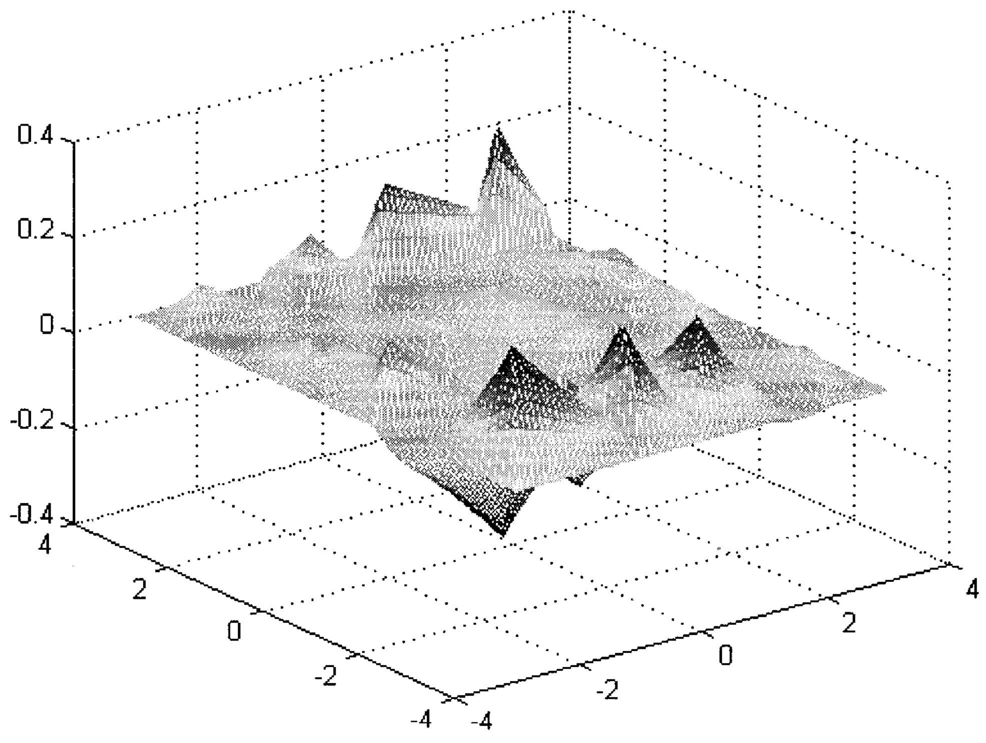

Figure 36 – Difference between original and simplified min-inference results.

An interesting result is that the simplified min-inference model more closely approximates the product-inference defuzzifier than it does the original min-inference defuzzifier. The difference between the product-inference defuzzifier and the simplified min-inference defuzzifier is shown in Figure 37.

It is clear that the product-inference defuzzifier is more closely approximated than the min-inference defuzzifier. This result is applicable to this particular fuzzy system and further examination is required before it can be applied to fuzzy systems in general.

The code used to develop these plots is included as Appendix E.

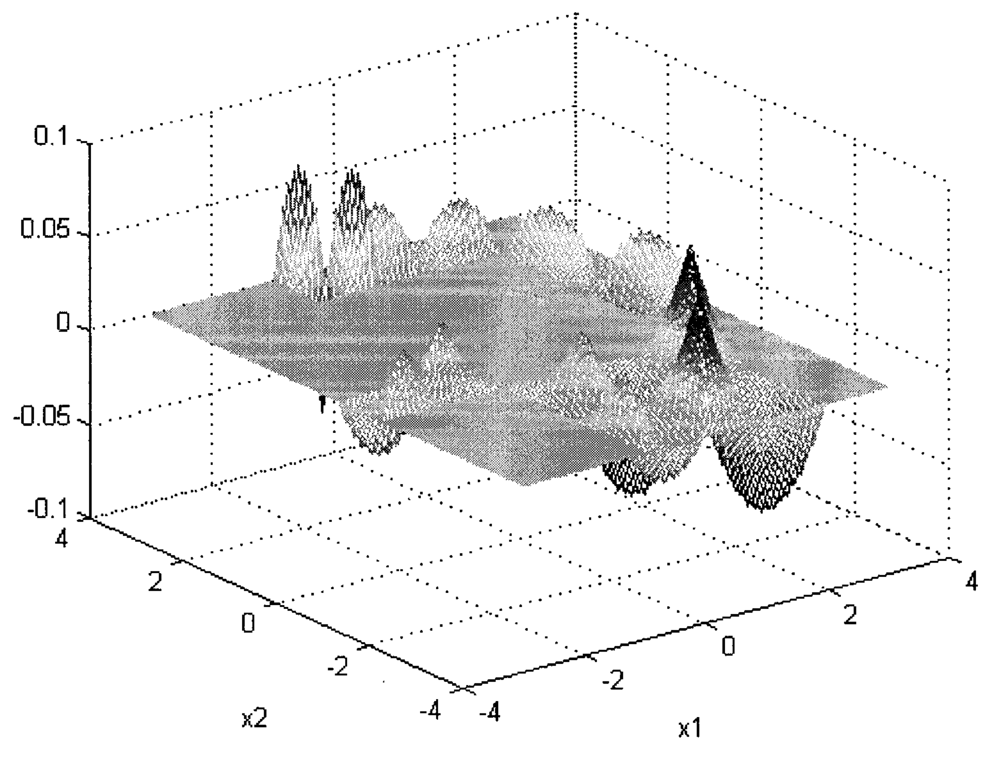

Figure 37 - Difference between simplified min-inference and product inference defuzzifiers

A.5 Implications and Applications

The most important revelation from the previous section is that the 2 dimensional fuzzy system can be constructed to yield a piecewise linear error function. This allows application of Theorem 9 to prove that the error must occur at one of a finite number points in the solution space.

Rewriting the equation for a 2-dimensional PLA given in (47) by substituting (35) and (36) yields:

$$\begin{aligned}
&f(x_{m1},x_{m2},d_{1n_1},d_{1n_1+1},d_{2n_2},d_{2n_2+1},c_{n_1n_2},c_{n_1+1n_2},c_{n_1n_2+1},c_{n_1+1n_2+1})=\\
&\quad\left(0.75c_{n_1n_2}-0.25c_{n_1n_2+1}-0.25c_{n_1+1n_2}-0.25c_{n_1+1n_2+1}\right)\min\left(\frac{x_1-d_{1n_1}}{d_{1n_1+1}-d_{1n_1}},\frac{x_2-d_{2n_2}}{d_{2n_2+1}-d_{2n_2}}\right)+\\
&\quad\left(-0.25c_{n_1n_2}+0.75c_{n_1n_2+1}-0.25c_{n_1+1n_2}-0.25c_{n_1+1n_2+1}\right)\min\left(\frac{x_1-d_{1n_1}}{d_{1n_1+1}-d_{1n_1}},\frac{-x_2+d_{2n_2+1}}{d_{2n_2+1}-d_{2n_2}}\right)+\\
&\quad\left(-0.25c_{n_1n_2}-0.25c_{n_1n_2+1}+0.75c_{n_1+1n_2}-0.25c_{n_1+1n_2+1}\right)\min\left(\frac{-x_1+d_{1n_1+1}}{d_{1n_1+1}-d_{1n_1}},\frac{x_2-d_{2n_2}}{d_{2n_2+1}-d_{2n_2}}\right)+\\
&\quad\left(-0.25c_{n_1n_2}-0.25c_{n_1n_2+1}-0.25c_{n_1+1n_2}+0.75c_{n_1+1n_2+1}\right)\min\left(\frac{-x_1+d_{1n_1+1}}{d_{1n_1+1}-d_{1n_1}},\frac{-x_2+d_{2n_2+1}}{d_{2n_2+1}-d_{2n_2}}\right)+\\
&\quad 0.25(c_{n_1n_2}+c_{n_1n_2+1}+c_{n_1+1n_2}+c_{n_1+1n_2+1})
\end{aligned}$$

The error function for a two dimensional PLA is defined as:

$$e=\sum_{m=1}^{M}\left|y_m-\sum_{n_1=1}^{N_1}\sum_{n_2=1}^{N_2}f(x_{m1},x_{m2},d_{1n_1},d_{1n_1+1},d_{2n_2},d_{2n_2+1},c_{n_1n_2},c_{n_1+1n_2},c_{n_1n_2+1},c_{n_1+1n_2+1})\right| \qquad (48)$$

where $f(\bullet)$ is the function given in (47) substituting (35) and (36).

Extending the techniques developed in Chapter 5 to work on higher order system requires further research. Those techniques took advantage of particular structures of the solution, most notably the intersecting hyperplanes to develop the descent algorithm. The 2-dimensional case also results in intersecting hyperplanes, but also has a series of hypersegments that are not accounted for in the original descent algorithm. Further work is required to resolve this and develop a workable algorithm for this and higher-order cases.

A.6 Conclusion

This Chapter demonstrated that 2 dimensional fuzzy system can be expressed in a manner such that a piecewise linear error function can be developed. This opens this class of fuzzy systems to analysis using the tools developed in Chapter 4. However, applying these systems to practical problems still requires significant further research.

APPENDIX B MATLAB PROGRAM TO OPTIMALLY DETERMINE A PLA FROM DATA

```
x=-5:2:5;
y=sin(x)+sin(2*x);
y=(x-1).^2;
y=[5 3 1 1 1 3];
y=[16 10 4 6 8 14];
x=-3:2:5;
y=[3 1 1 1 4]%+.1*randn(1, length(x));
N=2; %N is number of breaks
M=length(x); %M is the number of data points
A=[];
B=[];
%Form hyperplanes bounding linear regions
for n=1:N
   Apre=zeros(1, 2*(n-1));
   Apost=zeros(1, 2*(N-n));
   for m=1:M
      A=[A; 0 0 Apre x(m) -1 Apost];
      B=[B; 0];
   end
end
signs=[];
for n=1:2^N
   signs=[signs; de2bi(n-1,N)];
end
tempsigns=[];
for n1=1:N
   for n2=1:2
      tempsigns=[tempsigns signs(:,n1)];
   end
end
signs=tempsigns;
signs=2*signs-1;
signs=[ones(2^N,2) signs];
for m=1:M
   tempx=[x(m) 1];
   for n=1:N
      tempx=[tempx x(m) -1];
   end
   tempxm=[];
   for n=1:2^N
      tempxm=[tempxm; tempx];
   end
   A=[A; tempxm.*signs];
   B=[B; ones(2^N,1)*y(m)];
end

%Solve for vertices
NumBounds=length(A)
NumVertices=nchoosek(NumBounds,N*2+2)
```

```
eqs=nchoosek(1:NumBounds,N*2+2);
BestErr=1e100;
SingSolns=0;
NonFeasibleSolns=0;
BadlyOrderedSolns=0;
for k=1:length(eqs)
   tempeqs=eqs(k,:);
   TempA=[];
   TempB=[];
   for k1=1:length(tempeqs)
      TempA=[TempA; A(tempeqs(k1),:)];
      TempB=[TempB; B(tempeqs(k1))];
   end
   if det(TempA)==0
      SingSolns=SingSolns+1;
   else
      soln=inv(TempA)*TempB;
      a=[];
      b=[];
      for n=1:N+1
         if n==1
            a0=soln(1);
            c=soln(2);
         else
            b=[b soln(2*n)];
            a=[a soln(2*n-1)];
         end
      end
      if min(abs(a))>0
         a1=a(1:N-1);
         a2=a(2:N);
         if min(a2-a1)<0 & N>1
            BadlyOrderedSolns=BadlyOrderedSolns+1;
         else
            d=b./a;
            err=0;
            for m=1:M
               temperr=a0*x(m)+c;
               for n=1:N
                  temperr=temperr+a(n)*abs(x(m)-d(n));
               end
               err=err+abs(temperr-y(m));
            end
            if err<BestErr
               [tempeqs a d c err]
               BestErr=err;
               Besta0=a0;
               Bestc=c;
               Besta=a;
               Bestd=d;
            end
         end
      else
         NonFeasibleSolns=NonFeasibleSolns+1;
      end
   end
end
```

```
Besta0
Bestc
Besta
Bestd
BestErr
%Percentage of poor solns
(SingSolns+NonFeasibleSolns+BadlyOrderedSolns)/length(eqs)
xt=[];
yt=[];
for xtemp=x(1):.1:x(end)
   ytemp=Besta0*xtemp+Bestc;
   for n=1:N
      ytemp=ytemp+Besta(n)*abs(xtemp-Bestd(n));
   end
   xt=[xt xtemp];
   yt=[yt ytemp];
end
plot(x,y,'r',xt,yt,'b')
```

APPENDIX C MATLAB PROGRAM TO TRAIN A NEURAL NETWORK TO A GIVEN SYSTEM

```
% Set up the problem
clear;
nntwarn off
load xordata
Hidden = 7;

% Initialize the network
[w1,b1,w2,b2] = initff(x,Hidden,'tansig',y,'purelin');

%  Train the network using Levenburg-Marquardt
df = 10;   % Frequency of progress displays (in epochs).
me = 1000; % Maximum number of epochs to train.
eg = 0.01; % Sum-squared error goal.
tp = [df me eg];
[w1,b1,w2,b2,ep,tr] = trainlm(w1,b1,'tansig',w2,b2,'purelin',x,y,tp);

% Plot the results
PlotNNXor
```

APPENDIX D MATLAB PROGRAM TO TRAIN A PLA-LIKE FUZZY SYSTEM TO GIVEN DATA

```
% Initialize the data
%x=0:.25:4;  % Non-noisy xor
%y=x>3 | x<1;
hold off
load xordata
N=4; %N is number of breaks
M=length(x); %M is the number of data points
A=[];
B=[];
ErrHist=[];
ElapsedFlops=[];
ElapsedSecs=[];
flops(0);

%Form hyperplanes bounding linear regions
for n=1:N
   Apre=zeros(1, 2*(n-1));
   Apost=zeros(1, 2*(N-n));
   for m=1:M
      A=[A; 0 0 Apre x(m) -1 Apost];
      B=[B; 0];
   end
end
signs=[];
for n=1:2^N
   signs=[signs; de2bi(n-1,N)];
end
tempsigns=[];
for n1=1:N
   for n2=1:2
      tempsigns=[tempsigns signs(:,n1)];
   end
end
signs=tempsigns;
signs=2*signs-1;
signs=[ones(2^N,2) signs];
for m=1:M
   tempx=[x(m) 1];
   for n=1:N
      tempx=[tempx x(m) -1];
   end
   tempxm=[];
   for n=1:2^N
      tempxm=[tempxm; tempx];
   end
   A=[A; tempxm.*signs];
   B=[B; ones(2^N,1)*y(m)];
end
```

```
%Find initial feasible solution
NumBounds=length(A)
Dimension=N*2+2;
NumVertices=nchoosek(NumBounds,Dimension)
while Dimension>0
   tempeqs=fix(rand([1 Dimension])*NumBounds)+1;
   TempA=[];
   TempB=[];
   for k1=1:length(tempeqs)
      TempA=[TempA; A(tempeqs(k1),:)];
      TempB=[TempB; B(tempeqs(k1))];
   end
   [a0, c, a, d, success]=solvesystem(TempA, TempB, N);
   if success>0
      break
   end
end

% Find the best a0, c, a, and d
BestErr=CalcGeneralizedError(a0, c, a, d, x, y);
BreakSet=0;
NewEqs=1:NumBounds;
NewEqs=NewEqs';
Temp1=ones([NumBounds 1]);
BestEqsNew=tempeqs;
Passes=0;
StartTime=now;
flops(0);
while BreakSet==0
   Passes=Passes+1;
   BreakSet=1;
   ImprovedError=0;
   NumEqs=rows(BestEqsNew);
   if NumEqs>5
      TempEqs=BestEqsNew(1,:);
      for EqNum=2:5
         TempEqs=[TempEqs; BestEqsNew((EqNum-1)*round(NumEqs/5),:)];
      end
      BestEqsNew=TempEqs;
      NumEqs=5;
   end
   [length(ErrHist) BestErr rows(BestEqsNew)]
   if BestErr<inf
      ErrHist=[ErrHist; BestErr];
   end
   BestEqs=BestEqsNew;
   for EqNum=1:NumEqs
      BestEqs1=BestEqs(EqNum,:);
      TempEqsExtended=Temp1*BestEqs1;
      for dim=1:Dimension
         TempNewEqs=[TempEqsExtended(:,1:dim-1) NewEqs
TempEqsExtended(:,dim+1:end)];
         for eqnum=1:NumBounds
            tempeqs=TempNewEqs(eqnum,:);
            if sum(tempeqs~=BestEqs1)>0
               TempA=[];
               TempB=[];
               for k1=1:length(tempeqs)
                  TempA=[TempA; A(tempeqs(k1),:)];
                  TempB=[TempB; B(tempeqs(k1))];
               end
               [a0, c, a, d, success]=solvesystem(TempA, TempB, N);
               if success>0
                  TempErr=CalcGeneralizedError(a0, c, a, d, x, y);
                  if TempErr<BestErr
                     BreakSet=0;
                     ImprovedError=1;
                     BestEqsNew=tempeqs;
                     BestErr=TempErr;
                     Besta0=a0;
                     Bestc=c;
                     Besta=a;
                     Bestd=d;
                  end
                   if TempErr>BestErr & TempErr<BestErr*1.00001 %Handle
error plateaus
                     Duplicate=0;
                     for k=1:rows(BestEqsNew)
                        if sum(tempeqs~=BestEqsNew(k,:))==0
                           Duplicate=1;
                        end
                     end
                     if Duplicate==0
                        BreakSet=0;
                        BestEqsNew=[BestEqsNew; tempeqs];
                     end
                  end
              end
            end
         end
      end
      plotxor
      pause(1)
   end
   ElapsedFlops=[ElapsedFlops flops];
   ElapsedSecs=[ElapsedSecs (now-StartTime)*86400];
end

% Gather the data and plot
ElapsedTime=(now-StartTime)*86400
SecsPerIteration=ElapsedTime/length(ErrHist)
Besta0
Bestc
Besta
Bestd
BestErr

PlotXor
```

APPENDIX E MATLAB PROGRAM TO COMPARE DIFFERENT 2-DIMENSIONAL FUZZY SYSTEM IMPLEMENTATIONS

```
d1=[-3 -1 0 2 3]; %x-axis breakpoints
d2=[-4 -1 0 3 4]; %y-axis breakpoints
c=[-4 -2 -1 -.5 0; %output fuzzy singletons
   -3 -1  -.5 0 .5;
   -2 -.5 0 .5 2;
   -.5 0 .5 1 3;
   0 .5 1 2 4];
step=.05;
x1=-3:step:3;
x2=-4:step:4;
%Prepare the activation functions for x1
for k1=1:length(x1)
   n1=1;
   while x1(k1)>d1(n1) & n1<length(d1)-1
      n1=n1+1;
   end
   if x1(k1)<d1(n1)
      n1=n1-1;
   end
   N1(k1)=n1;
   a11(k1)=(x1(k1)-d1(n1))/(d1(n1+1)-d1(n1));
   a12(k1)=(d1(n1+1)-x1(k1))/(d1(n1+1)-d1(n1));
end
%Prepare the activation functions for x2
for k2=1:length(x2)
   n2=1;
   while x2(k2)>d2(n2) & n2<length(d2)-1
      n2=n2+1;
   end
   if x2(k2)<d2(n2)
      n2=n2-1;
   end
   N2(k2)=n2;
   a21(k2)=(x2(k2)-d2(n2))/(d2(n2+1)-d2(n2));
   a22(k2)=(d2(n2+1)-x2(k2))/(d2(n2+1)-d2(n2));
end
%Defuzzify to find y using 4 different methods.
%  y1: min-conjunction with centroidal defuzzification
y1=sum(c_ij*min(a1_i,a2_j))/sum(min(a1_i,a2_j))
%  y2: min-conjunction with simplified centroidal defuzzification (no
denominator) y2=sum(c_ij*min(a1_i,a2_j))
%  y3: product-conjunction with centroidal defuzzification
y3=sum(c_ij*a1_i*a2_j))/sum(a1_i*a2_j)
%  y4: min-conjunction with adjusted simplified centroidal
defuzzification y4=sum(c_ij*min(a1_i,a2_j))-
.25*sum(c_ij)*sum(min(a1_i,a2_j))
for k1=1:length(x1)
   for k2=1:length(x2)
      n1=N1(k1);
      n2=N2(k2);
```

```
y2(k1,k2)=c(n1,n2)*min(a12(k1),a22(k2))+c(n1+1,n2)*min(a11(k1),a22(k2))
+c(n1,n2+1)*min(a12(k1),a21(k2))+c(n1+1,n2+1)*min(a11(k1),a21(k2));

y3(k1,k2)=c(n1,n2)*a12(k1)*a22(k2)+c(n1+1,n2)*a11(k1)*a22(k2)+c(n1,n2+1
)*a12(k1)*a21(k2)+c(n1+1,n2+1)*a11(k1)*a21(k2);

actsum(k1,k2)=min(a11(k1),a21(k2))+min(a12(k1),a21(k2))+min(a11(k1),a22
(k2))+min(a12(k1),a22(k2));

adjust(k1,k2)=.25*(c(n1,n2)+c(n1+1,n2)+c(n1,n2+1)+c(n1+1,n2+1))*(actsum
(k1,k2)-1);
   end
end
y1=y2./actsum;
y4=y2-adjust;
mesh(x1,x2,y3')
xlabel('x1')
ylabel('x2')
title('a) Plot of product-conjunction with centroid defuzz');
pause
mesh(x1,x2,actsum')
xlabel('x1')
ylabel('x2')
title('b) Plot of denominator of centroid defuzz');
pause
mesh(x1,x2,y1')
xlabel('x1')
ylabel('x2')
title('c) Plot of min-conjunction with centroid defuzz');
pause
mesh(x1,x2,y2')
xlabel('x1')
ylabel('x2')
title('d) Plot of min-conjunction with simplified defuzz');
pause
mesh(x1,x2,(y1-y2)')
xlabel('x1')
ylabel('x2')
title('e) Plot of difference between min-conjunction defuzz methods');
pause
mesh(x1,x2,-adjust')
xlabel('x1')
ylabel('x2')
title('f) Plot of adjustment between defuzz methods');
pause
mesh(x1,x2,y1-y2-adjust')
xlabel('x1')
ylabel('x2')
title('f1) Plot of difference between adjustment and error');
pause
mesh(x1,x2,y4')
xlabel('x1')
ylabel('x2')
title('g) Plot of min-conjunction with simplified defuzz (adjusted)');
pause
mesh(x1,x2,(y3-y4)')
```

```
xlabel('x1')
ylabel('x2')
title('h) Plot of difference between adjusted method and product-
conjunction');
pause
xlabel('x1')
ylabel('x2')
mesh(x1,x2,(y1-y4)')
title('i) Plot of difference between adjusted method and min-
conjunction');
```

CURRICULUM VITA

for

Raymond Scott Starsman

DEGREES:

Doctor of Philosophy (Electrical and Computer Engineering), Old Dominion University, Norfolk, VA, May 2003

Master of Science (Electrical Engineering), Naval Postgraduate School, Monterey, CA, December 1991

Bachelor of Science (Electrical Engineering), Naval Postgraduate School, Monterey, CA, December 1991

Bachelor of Science (Systems Engineering), United States Naval Academy, Annapolis, MD, May 1986

PROFESSIONAL CHRONOLOGY:

United States Navy

Naval Officer, May 1986 – Present

SCIENTIFIC AND PROFESSIONAL SOCIETIES MEMBERSHIP:

Institute of Electrical and Electronic Engineers

United States Naval Institute

Armed Forces Communications and Electronics Association

HONORS AND AWARDS:

1991 - Electrical Engineering Honors Award from the Naval Postgraduate School

1986 – WHP Blandy Award from the United States Naval Academy