APPLICATION OF FUZZY THEORY

TO

PATTERN RECOGNITION

by

David Usechak

Dissertation submitted to the Faculty of the Graduate School
of New Jersey Institute of Technology in partial fulfillment
of the requirements for the degree of
Doctor of Engineering Science
1983

APPROVAL OF DISSERTATION

Application of Fuzzy Theory to Pattern Recognition

by David Usechak

for Department of Electrical Engineering

New Jersey Institute of Technology

Approved: ______________________________ Chairman

____

____

____

Newark, N.J.

April 1983

VITA

Name: David Bungar Usechak

Degree and date to be conferred: D.Engr. Sc., 1983

Secondary education: Slippery Rock High School, June 1963

Collegiate Institutions	Date	Degree	Date of Degree
PMC Colleges	63-67	BSEE	67
FDU	69-71	MSEE	71
Princeton U	72-75	MSAMS	75
NJIT	78-83	D.Engr.Sc.	83

Major: Electrical Engineering

Position held: Elec. Engr., U.S. Army Ft. Monmouth, N.J. 07703

ABSTRACT

Title of Dissertation: Application of Fuzzy Theory to Pattern Recognition

David Usechak, Doctor of Engineering Science, 1983

Dissertation directed by: Dr. Stanley S. Reisman
Associate Professor
Department of Electrical Engineering
New Jersey Institute of Technology

Piecewise curve approximation is used to describe boundaries of objects in pictures and waveforms. The method consists of linear and quadratic piecewise polynomial approximations in which the error must not exceed a predetermined cost threshold. A fuzzy Bayes model is used to determine if a breakpoint exists within an interval I_n or between intervals I_n and I_{n+1} and to determine whether these intervals can be merged for data compaction reasons. In order to achieve these objectives a new fast algorithm has been proposed which gives good curve/object fitting. This algorithm uses a technique for generating generalized inverse matrices once an initial generalized inverse matrix has been determined. The continuity requirements at the breakpoints are relaxed such that the only requirement is that the data point for interval I_{n-1} is the starting point for interval I_n. Results of computer experiments with graphic outlines and radar data are reported.

ACKNOWLEDGMENT

I am grateful to my advisor, Dr. Stanley Reisman, for his guidance and encouragement and to the members of my committee, Dr. Denis Blackmore, Dr. Joseph Frank, and Dr. Solomon Rosenstark, for their constructive comments on this work.

TABLE OF CONTENTS

TABLE OF CONTENTS(Cont.)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES(Cont.)

LIST OF FIGURES(Cont.)

CHAPTER I

A PATTERN RECOGNITION PROBLEM

1.0 Introduction

The purpose of this research is to apply the concepts and techniques of fuzzy set theory to the detection of contours or boundaries of signals for pattern recognition of objects. To do this, we will use fuzzy set theory in the decision-making portion of the problem as it will shorten the decision time required for a solution. Since fuzzy theory is based largely on subjective information it can aid a process under consideration by supplying initial input data or data updates. For most pattern recognition schemes an initial training period is required to teach the algorithm the input data space characteristics. This method works well if the input data space is large. For the small input data space, subjective inputs are particularly helpful as they can teach the algorithm until a sufficient data base has been established. Problems also exist when the benefits derived from an algorithm training period are minimal because of the nature of the input data and/or problem under investigation. For example, in the design of new automobile bodies the contour has a very small or nonexistent sample space because the body shape is determined by such factors as new engines, and materials, etc.. The same is true in other design areas where new objects with different shapes and contours are being developed. One technique which has been used extensively for the description of object boundaries with variable knots, specifically for automobile body design, is spline theory [55,58]. A knot is defined as a point where the describing polynomial changes slope. Spline theory imposes a slope continuity requirement at the knots. However, for pattern recognition this requirement is not always relevant because of the possibility of boundary discontinuities i.e. sharp corners. Also, the computational effort required when the continuity requirement is imposed is greater than when it is absent [39]. Therefore, if the local continuity requirement could be ignored in the search for an approximating polynomial on a subinterval and then adjusted globably, we could achieve a near optimum solution with reduced computational time.

A key feature in the recognition of an object is its shape. The perceived shape of an object as suggested by Attneane [6] depends upon the boundary's points of maximum curvature. The exact location of the maximum curvature can be determined from the second derivative of the polynomial which describes the contour precisely. However, from a practical point of view, exact contour descriptions are not necessary because approximating polynomials will yield good results with small cost(error) and the computational complexity involved is reduced. Approximating polynomials will give the approximate location of the maximum contour curvature. Such maxima in general will be close to the actual maxima of the second derivative of a polynomial which describes the contour exactly.

The problem of shape and edge detection is one of the central issues in pattern recognition and as such it has received considerable attention. The importance of this topic is reflected in the immense amount of literature dealing with recognition of characters, waveforms, cells, machine parts, etc. [14, 16, 39]. In this research we will restrict ourselves to the study of plane objects; i.e. we will deal only with waveforms, and 2-dimensional segmented pictures. We will not draw any conclusions or inferences to higher dimensional figures. Also, we will consider objects which have only closed external boundaries, i.e. they do not contain internal edges or boundaries. In order to decrease computational time, the boundaries for this study will be described by using piecewise linear or quadratic polynomial approximations. The term "piecewise" refers to approximations on subintervals within a closed interval [a,b]. Futhermore, these approximations could be composed of combinations of linear or quadratic polynomials on the interval [a,b] . The approximating polynomials will be found by using fuzzy decision-making along with the least squares technique.

1.1 Background

The fundamental concepts of pattern recognition have been applied across many different fronts, from data classification to motion recognition, and in all cases they have several characteristics in common. Some of the major common characteristics within the various areas of pattern recognition are: the solutions are very problem oriented, perfect recognition is very difficult to achieve, and the computational time can be enormous if the cost function requirement is too strict. Thus, in a global sense, a generalized theory within pattern recognition will probably never exist because of the many different types of objects (patterns) which can be contained within a scene.

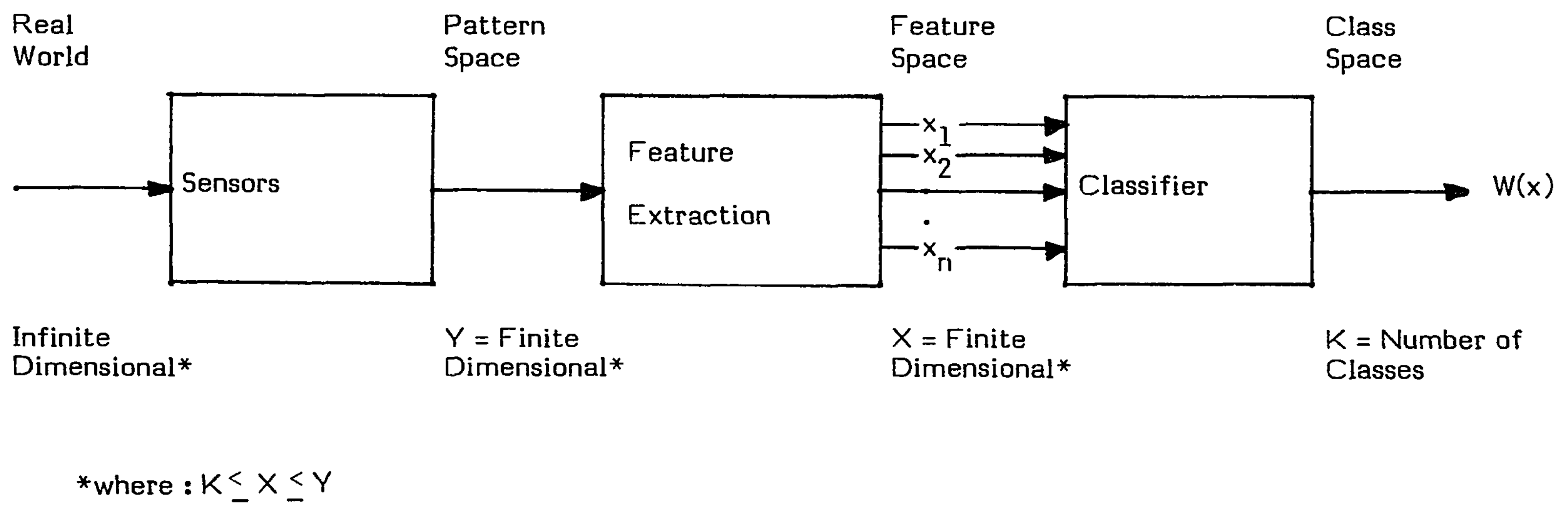
One of the fundamental attributes of a human being is his ability to recognize objects. Given a complex environment human beings perform pattern recognition with fuzzy or ill defined object characteristics and they do this task very well with a minimum amount of effort. One reason why pattern recognition can be considered a fuzzy process is that the boundaries between different patterns within a scene are not well defined. One method of analyzing patterns in scenes is to use statistical decision and estimation techniques for their classification . There is a great deal of information on the application of statistical decision-making to pattern recognition. However, if the amount of information of the sample patterns is small or non-existent, then this approach is not very helpful in the pattern decision process. For situations where the sample space is small and vague, the fuzzy decision process is helpful in the classification of objects because of the admissibility of subjective input data. Zadeh [58], who developed the theory of fuzzy sets, has studied the uses of fuzzy sets in engineering systems (and also algorithms associated with those systems) because initially in most engineering systems the specifications are ill-defined or vague. This then permits a much larger design and decision space. In order to reduce this space to a manageable size the system design parameters must be assigned to groups and subgroups within the system design-decision space. To accomplish this on a small and vague input space, Zadeh and others have used one of the attractive features of fuzzy set theory which is the notion of grades of membership or the grade of belonging. Imprecision, fuzziness, relates not to randomness but to a lack of a clearly defined membership in a class of objects. In classical mathematics, as for example in abstract algebra, an object either belongs to a specific class or it does not. However, in the real world a class of objects may belong to several categories in varying degrees or grades of membership. This is a fundamental concept of fuzzy sets. For example, the classification of smells is based upon subjective information which can be represented by grade of membership of all smells. Imprecise descriptions and the relaxing of the requirement of a numerical input for decision analysis has been discussed by Watson [55].

The aim of this dissertation will be to use the attractive features of fuzzy set theory and the least-squares technique to detect edges and breakpoints within noisy data. The approach will be to use a piecewise polynomial approximation on subsets of the set of all input data. In particular, for shape detection, we are interested in detecting breakpoints, given optimum or near optimum piecewise linear or quadratic polynomial fits, and minimizing the computational time required for a complete solution on a given data set. An attribute of shape detection is the curvature of the boundary and a primary characteristic of curvatures is a point of infinite curvature, i.e. a corner given a continuous curve. The above applies for continuous data but it can also be extended for discrete data which will be shown later in the text. The cost function for both the continuous and discrete cases for evaluating goodness of fit of the approximations will be the l_2-norm, l_{max}, and slope. These cost functions will be defined later in the text.

1.2 Statement of the Problem

A short description of the entire pattern recognition process will now be provided , because this process can be decomposed into several unique processes, and will identify the area in which the problem under consideration occurs. From this discussion the problem under investigation will be defined by presenting, without proof, the mathematical approach to be taken in this study. The results of this discussion will permit us to precisely define the problem to be investigated.

One of the problems of pattern recognition is that the environment can provide large volumes of data to a pattern recognition system such that the system could be overloaded to the point where it would hardly produce an output. Figure 1.2.1 represents an overview of a typical pattern recognition problem. In this figure sensors provide information about the environment, which is of infinite dimension, to a pattern recognition system. The sensors, which do not have infinite sensing capabilities, will provide a limited set of information to the recognition system. The feature extraction process will eliminate or filter the sensor information which a priori does not contribute to the process under consideration. The feature space, output from the extraction process, is normally less than the input space. The feature space information(data) is further processed by a classifier to determine the data class. The objective is to achieve, depending on the problem, either data reduction (compaction) or classification of the original data. The problem we are interested in is data compaction or the representation of the input data in a shorter form. Also, we are interested in fitting a polynomial to a given set of data. One of the methods for accomplishing this efficiently is the least squares technique. In general the least squares technique is a method of polynomial approximation; that is a function $f(x)$ is approximated by a polynomial function $F_n(x)$ of degree n on an



Overview of a Pattern Recognition Problem

Figure 1.2.1

interval $a \le x \le b$. For approximating continuous functions by the least squares technique, we are interested in determining a polynomial of the form

$$F_n(x) = c_0 + c_1x^1 + c_2x^2 + \text{------} + c_nx^n \qquad 1.2.1$$

which minimizes the mean squared error, or l_2 norm,

$$||f - F_n|| = \left(\int_a^b [f(x)-F_n(x)]^2 dx \right)^{\frac{1}{2}}, \qquad 1.2.2$$

between the two functions . The norm, $||g||$, satisfies the following:

(a) $||g|| \ge 0$,

(b) $||g|| = 0$ if and only if $g \equiv 0$,

(c) $||Kg|| = ||K|| \cdot ||g||$ for any constant K,

(d) $||g+h|| \le ||g|| + ||h||$, the triangle inequality.

A function, ψ (c), can be defined for n+1 variables by using equation 1.2.1 and any function f(x) as follows:

$$\psi(c_0, c_1, \ldots, c_n) \equiv \int_a^b [f(x)-F_n(x)]^2 dx. \qquad 1.2.3$$

Note that by minimizing $\psi(c)$ this will also minimize $||f(x) - F_n(x)||$.

In order to find the polynomial that minimizes ψ (c) in equation 1.2.3 we must find a coefficient $\bar{c} \equiv (c_0, c_1, \ldots\ldots, c_n)$. The coefficient "$\bar{c}$" defines a point in the n+1 dimensional space for which ψ (c) is a minimum. We start our solution by expanding equation 1.2.3 which gives:

$$\psi(c_0, c_1, \ldots, c_n) = \int_a^b f^2(x)dx - 2\sum_{i=o}^{n} c_i \int_a^b x^i f(x)dx + \sum_{i=o}^{n} \sum_{j=o}^{n} c_i c_j \int_a^b x^{i+j} dx, \qquad 1.2.4$$

where ψ is a quadratic function in c_i. To find the minimum of ψ we take the derivative with respect to the coefficients, c_l and set the derivative equal to zero . This can be written as:

$$\left. \frac{\partial \psi(c_0, c_1, \ldots\ldots, c_n)}{\partial c_l} \right|_{c=\bar{c}} = 0, \qquad 1.2.5$$

Substituting equation 1.2.4 into equation 1.2.5 we get:

$$\left.\frac{\partial\psi}{\partial c_l}\right|_{\bar{c}} = 0-2\int_a^b x^l f(x)dx + \sum_{i-o}^{n} \bar{c}_i \int_a^b x^{i+l}dx + \sum_{j=o}^{n} \bar{c}_j \int_a^b x^{l+j}dx=0,$$

$$\left.\frac{\partial\psi}{\partial c_l}\right|_{\bar{c}} = 2(\sum_{i=o}^{n} \bar{c}_i \int_a^b x^{i+l}dx-\int_a^b x^l f(x)dx) = 0, \qquad 1.2.6$$

where $l = 0,1,2,\dots,n$.

Equation 1.2.6 is known as the normal equation which is defined for a system of n+1 linear equations. The normal equation, 1.2.6 which defines a system of n+1 linear equations, can be written as

$$\begin{bmatrix} \int_a^b x^{i+j}dx \end{bmatrix} \begin{bmatrix} c_o \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} \int_a^b x^o f(x)dx \\ \vdots \\ \int_a^b x^n f(x)dx \end{bmatrix} .$$

Since almost all pattern recognition systems used today are digital, and ours is no exception, the above equations must be rewritten for the discrete case. Starting with equation 1.2.2 we can write

$$||f-F_n|| = (\sum_{i=o}^{n} \{f(x_i) - F_n(x_i)\}^2)^{\frac{1}{2}} \qquad 1.2.7$$

Equation 1.2.7 holds for a fixed set of data points. That is we assume that we are given an input data set $(x_o,x_1,\dots,x_n)$. Also, the constraint $n \leq N$ will be imposed where N is the number of equations. Continuing with equations, 1.2.3 thru 1.2.6 we can now write

$$\psi(c_o,c_1,\dots,c_n) \equiv ||f-F_n||^2. \qquad 1.2.8$$

Expanding equation 1.2.8 for the discrete condition and using the same notation as given above, we get

$$\psi(c_o,c_1,\ldots\ldots,c_n) = \sum_{i=o}^{N} f^2(x_i) - 2\sum_{l=o}^{n} c_l \sum_{i=1}^{N} x_i^l f(x_i)$$

$$+ \sum_{i=o}^{n} \sum_{j=o}^{n} c_i c_j \sum_{i=o}^{N} x_i^{l+j} \quad . \qquad 1.2.9$$

A necessary condition for ψ to have a minimum is given by the normal equation, i.e.

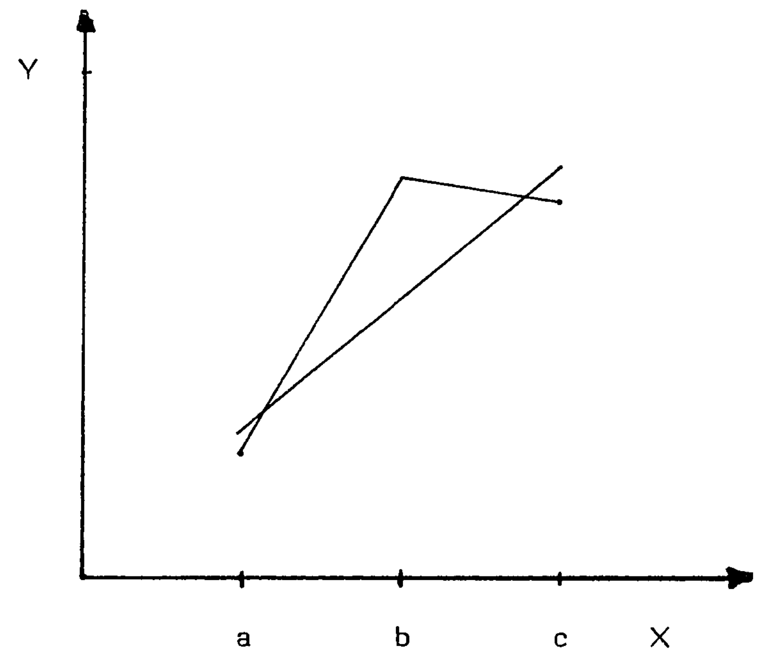
$$\left.\frac{\partial\psi(c_o,c_1,\ldots\ldots,c_n)}{\partial c_l}\right|_{c=\bar{c}} = 0, \qquad 1.2.10$$

where $l = 0,1,2,\ldots,n$.

Equation 1.2.10 will yield the discrete form of the normal equation 1.2.6 which can be written as

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^{l+j} \end{bmatrix} \begin{bmatrix} c_o \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{bmatrix} = \begin{bmatrix} \Sigma x_1^o f(x_1) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \Sigma x_i^n f(x_i) \end{bmatrix} .$$

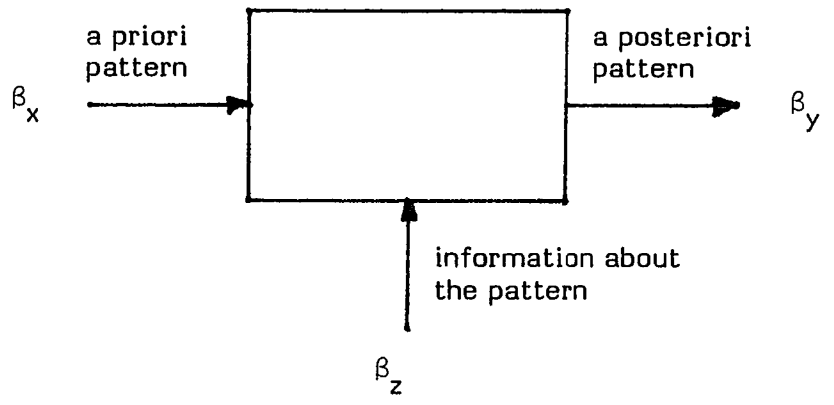
Using the above equation, an approximating polynomial(s) can be found, via the least squares method, over intervals [a,b] and [b,c] - see Figure 1.2.2 . The question to be asked is if an approximating polynomial over the interval [a,c], as given in Figure 1.2.2, can be determined with reduced computational effort? In order to answer this question, we need to predict a priori whether an approximating polynomial, over the interval [a,c], exists within a specified threshold value(cost). The following agument contributes to the solution of this problem. The process of edge and pattern detection done by human beings, which by the way is very effective, is not a precise process; that is, it is a fuzzy process. Precise edges and patterns are not necessarily a primary variable when humans perform these processes. Thus the process, when implemented by a machine, should allow for imprecise edge or corner descriptions, but at the same time not miss significant object features in the pattern. This can be accomplished by using fuzzy sets, as mentioned above, with an interval of evaluation defined as [0,1]. A fuzzy model which can provide answers to the question of polynomial approximations across two



Two edges represented by another edge.

Figure 1.2.2

intervals is called a fuzzy Bayes' model. This model, which we will develop later in Chapter III, can be represented as



Fuzzy Bayes' Model

Figure 1.2.3

Mathematically we can write the fuzzy Bayes' equation as

$$\tilde{\int}_H \beta_y(G|H) \circ f_x = \tilde{\int}_G \beta_z(H|Y) \circ f_x, \qquad 1.2.13$$

where

$\tilde{\int}$ represents a fuzzy integral

and

o represents the composition of two functions.

The a posteriori fuzzy measure, $\beta_y(G|x)$, can be determined from equation 1.2.13 for each interval. From this a final fuzzy value can be calculated to show if a corner exists between these two intervals. For this problem $\beta_y(G|x_i)$ represents the grade of fuzziness of the statement that "a corner of some magnitude exists in interval I_n" which has been subjectively scored from some criterion y_i. This final fuzzy value will then be compared against some empirical threshold to determine if a fit is possible. The initial threshold value must be approximated, and then for every iteration thereafter it shall be adaptively adjusted to give a near optimum solution. The term "near optimum" is used because we are not interested in the exact optimum solution for either the linear or quadratic case. Instead, we want a solution which describes the boundaries or contours of an object well enough, and with a minimum amount of processing, so that it can be easily recognized by a human. Thus, the contour/boundary detection and pattern recognition problem can be stated as:

Given a finite set of discrete input data on an object's contour, find and describe the edges or boundaries such that the description is smooth and continuous. Futhermore, this is to be accomplished in minimum time and with minimum number of describing polynomials to meet a given error criterion.

The approach which will be followed to achieve the above objective will be to use a combination of linear and quadratic piecewise polynomial approximations with fuzzy decision-making on the amount of polynomial compaction. That is, a decision is made within a subinterval or between two contiguous subintervals to determine if a breakpoint exists. Also, fuzzy linguistic descriptive values will be used to vary the goodness of the approximation on the description of the object's contour. The above investigation will provide information on a unique way to describe contours.

Linear contour approximations have been investigated and are reported in [33] and [39], cubic approximations in [39], and higher order approximations in [45]. However, to date no work has been reported on using the quadratic polynomial or a combination of approximating polynomials. Linear approximation is very attractive because it is simple and computationally very fast. However, it does not provide smooth traces. Cubic and higher order approximations provide good edge detection and object representation but they are computationally complex. Thus, the attempt in this paper will be to show reasonable edge detection and object representation by using piecewise linear and quadratic polynomials with fuzzy decision-making.

1.3 Dissertation Outline

A summary of the details involved in the solution to the above stated problem are presented in the following discussion. This summary also contains the contributions of this paper to the problem of pattern recognition of contours and boundaries.

Chapter II deals with the matrix formulation of the problem. The contents of Chapter II include:

(a) The least square solution using generalized inverse and

(b) The optimum breakpoint location within an interval.

Chapter III deals with the fuzzy decision model formulation of the problem. The contents of Chapter III include:

(a) Presentation of fuzzy concepts

(b) Presentation of fuzzy integrals with proofs and

(c) Presentation of fuzzy Bayes' Model.

Chapter IV deals with the application of Chapters II and III to the edge detection problem. The contents of Chapter IV include:

(a) Presentation of an adaptive threshold scheme and

(b) Presentation of a physical application of the fuzzy model .

Chapter V deals with the algorithm and its differences from other algorithms. The contents of Chapter V include:

(a) Applying the algorithm to real world data and

(b) A comparison of the new algorithm with other edge detection algorithms of the same class.

Chapter VI deals with the computational complexity of the algorithm. In particular it details the computational time of the algorithm developed in this study and compares this computational time with that of other algorithms of the same class.

Five appendices are provided which show either detailed mathematical proofs of equations given in the text or results of the algorithm as applied to the detection of different types of contours. Appendix A shows a detailed proof of the optimum breakpoint location within an interval[a,b]. A discussion and proof of the optimum polynomial for data fitting using least-square techniques is given in Appendix B. Appendix C presents the results of the algorithm's ability to detect the contours of various objects. A detailed numerical example of the short matrix method, as presented in the text, is presented in Appendix D. Appendix E presents the computer program listing of the algorithm developed in this study.

The contributions of this paper to pattern recognition of edges/contours include:

(a) A unique adaptive thresholding scheme for contour fitting on an interval and for the merging of several intervals.

(b) The application of a fuzzy Bayes model for the detection and decision making of whether breakpoints exist within the given intervals.

(c) A unique inverse matrix computational method for reducing the algorithm's processing time.

(d) The admissibility of heuristic inputs to aid the algorithm in the detection and processing of breakpoints for a given contour input.

CHAPTER II

MATRIX THEORY

2.0 Introduction

As presented in Chapter I, equation 1.2.7, the discrete least squares method can be written in matrix form. In this chapter we will examine the conditions which will permit a solution of equation 1.2.7. The primary reason for formulating the problem as a set of matrices is that we want to solve the contour problem via a digital computer. This formulation is primarily motivated by Pavlidis [39] and McClure [33] because contour approximating on large amounts of data is very expensive computationally. As stated earlier the investigation will consider only linear and quadratic polynomial approximations of data; i.e. a function f(x) can be approximated by a function F(x) which can be expressed for the linear case as

$$F_l(x) = c_0\,\psi_o(x) + c_1\,\psi_1(x)$$

and for the quadratic case as

$$F_q(x) = c_0\psi_o(x) + c_1\,\psi_1(x) + c_2\,\psi_2(x)$$

where $F_l(x)$ and $F_q(x)$ are the linear and quadratic approximating functions respectively. In this paper the ψ 's are defined as :

$$\psi_o(x) = 1,$$

$$\psi_1(x) = x,$$

and

$$\psi_2(x) = x^2 .$$

The functions ψ_o, ψ_1 and ψ_2 are chosen in advance and the coefficients c_0, c_1, and c_2, are to be determined. To determine the coefficients such that $F_q(x) \cong f(x)$, a set of linear system equations can be written in the following form:

$$c_0\psi_o(x_0) + c_1\,\psi_1(x_0) + c_2\psi_2(x_0) = f(x_0)$$

$$c_0\psi_o(x_1) + c_1\,\psi_1(x_1) + c_2\psi_2(x_1) = f(x_1) \qquad 2.0.2$$

$$c_0\psi_0(x_2) + c_1\psi_1(x_2) + c_2\psi_2(x_2) = f(x_2).$$

For linear approximations equation 2.0.2 will take the following form

$$c_0\psi_0(x_0) + c_1\psi_1(x_0) = f(x_0)$$

$$c_0\psi_0(x_1) + c_1\psi_1(x_1) = f(x_1).$$

2.1 Matrix Attributes

A discussion on some key matrix attributes and the associated vector space is presented in the following sequence along with a least squares geometrical interpretation of equation 1.2.7. We start our discussion by defining the matrix equation 2.0.2 as

$$\psi \bar{C} = \bar{b} \qquad 2.1.1$$

where is a square matrix consisting of $\psi_i(x_j)$, $\bar{C}$ is a column vector composed of $[c_0,c_1,\dots,c_n]$, and $\bar{b}$ is a column vector composed of$[f(x_0),\dots,f(x_n)]$. That is, a solution to equation 2.1.1 must lie in a plane spanned by the column vectors ofψ. This column space is usually referred to as the range of ψ ,and written as $R(\psi)$. Further, the dimension of the column space $R(\psi)$ is defined in this paper to be equal to the rank, which is also equal to the dimension of the row space. The rank of a matrix is equal to the number of nonzero pivots in the elimination process. Furthermore, equation 2.1.1 could have been defined in a row space by transposing the matrices in equation 2.1.1. This equation, 2.1.1, is the matrix form of equation 1.2.7. The polynomial, as given in equation 1.2.1, may be viewed as a coordinate vector in a "function space" of all polynomials of degree n . The actual function, f(x), and the approximating function, F(x), have been defined to lie in a vector space . Thus a determination of the amount of separation between these two functions can be made. This distance, in the least squares sense, will be the error between the actual function, f(x), and the approximating function, F(x). A detailed discussion of the error vector will be given in Chapter IV .

In the linear space of two or three dimensions, known as "Euclidean", the use of a norm can be defined for the general case as

$$||g|| = (\sum_{i=1}^{n} |x_i|^p)^{1/p}, \qquad 2.1.2$$

where $p \geq 1$.

This is popularly known as the $l_{p\text{-norm}}$. We are interested in two particular norms because of the amount of information they contain. The two norms of interest are the l_2-norm, which is defined by equation 1.2.2, and the $l_{max\text{-norm}}$ defined as

$$l_{max} = \max_{x\epsilon[a,b]} |x_i| \qquad 2.1.3$$

over a closed interval [a,b]. The above Euclidean norms, l_2 and l_{max}, must satisfy the properties as given in section 1.2 on a given linear space.

A geometrical interpretation of the solution to polynomial approximations, when using the Euclidean norm in two and three dimensions, is the shortest distance between a point and a linear subspace. For the problem under consideration this means that the error vector, $E(x) = f(x) - F_n(x)$, must be perpendicular to the subspace spanned by the vectors ψ_o, ψ_1, ψ_2 if the norm of E is to attain a minimum .

2.2 Matrices

The system $\psi \bar{C} = \bar{b}$, equation 2.1.1, of n equations and m unknowns is popularly known as the normal equation(s) within the least squares technique. Upon investigation of equation 2.1.1, we find that there are certain conditions which must be examined before a solution is possible. Otherwise, errors are introduced which make the system, $\psi \bar{C} = \bar{b}$, give invalid results. There are three conditions which we must consider in order to avoid solution and computional errors. These conditions are listed as follows:

(a) A determination of the linearly dependent and independent system vectors which can be defined in either the column or row space of ψ,

(b) A determination of whether the system is orthogonal, and

(c) A determination of whether the system is consistent or inconsistent.

Since we are interested in solutions to systems of linear equations, a brief examination of the above conditions where solutions exist is in order. Starting with condition (a) we investigate the homogeneous solution of equation 2.1.1 where the ψ_i are column vectors of ψ are said to be linearly independent if the linear equation $c_0 \psi_0 + c_1 \psi_1 + \ldots\ldots + c_n \psi_n = 0$ is satisfied only if the coefficients $c_i = 0$. Otherwise there exists at least one combination of vectors which is linearly dependent. The maximal number of linearly independent rows or columns i.e. the basis of the row or column space of a matrix is referred to as the rank. We will have more to say about the rank of a matrix in Chapter IV. Let $\Omega_0, \Omega_1, \ldots \Omega_n$ be a basis for a column space; then the column space has an orthonormal basis consisting of n vectors. This leads to condition (b) where, if the columns are assumed to be orthonormal, we can then form a diagonally dominant matrix. A diagonally dominant matrix is one which has its diagonal elements, a_{ii} , greater than a radius r_i, where radius is defined as $r_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} [a_{ij}]$ with its center located at the element entry location a_{ii}. A diagodominantminant matrix has some very desirable computational attributes which will be described in Chapter IV when we try to solve the system equation 1.2.6. In order to construct a diagonally dominant coefficient matrix for the continuous function $F_n(x)$, equation 1.2.1, we start by writing the function as

$$F_n(x) = \sum_{i=0}^{n} c_i G_i(x)$$

where $G_i(x)$ is a set of orthonormal polynomials of degree i. Substituting the above equation into equation 1.2.3 and solving for the normal equations, as shown in equation 1.2.6, we get:

$$\int_a^b G_j(x) f(x) dx = \sum_{i=1}^{n} c_j \int_a^b G_j(x)\ G_i(x) dx \qquad 2.2.1$$

where $j = 0,1,2,\ldots,n$.

As $\{G_i(x)\}$ is an orthonormal set it satisfies:

$$\int_a^b G_j(x)G_i(x)dx = \delta_{ji}, \qquad 2.2.2$$

where δ_{ji} is the Kronecker delta, which then gives the coefficients as

$$c_i = \int_a^b G_j(x)f(x)dx. \qquad 2.2.3$$

The construction of an orthonormal set of polynomials can be accomplished by using the Gram-Schmidt process (see reference [44]) . We may assume that the interval $[a,b] = [-1,1]$ since this can be accomplished by a linear change of variables. Now for example, we can use the Chebyshev, Legendre, or Lagrange polynomials. The important point is that all of these methods yield a diagonally dominant coefficient matrix in the transformed space. Once a solution has been determined in the x_i' space, a transformation into the x_i space can be accomplished by a linear change of variables.

For condition (c) we must consider whether the system $\Psi\bar{C} = \bar{b}$ is inconsistent, i.e. b is not in the column space of R(C). Where the column space of $\bar{C}$ is often referred to as the range of $\bar{C}$. The range R(C) of a matrix is the same as the range of the linear function $L(x) = \Psi x$. That is, if L(x) is given, then x represents the domain and the value of L(x) represents the range.

The last matrix concept we will cover briefly is that of inverses. A matrix inverse is defined for a nonsingular square matrix; it is used in the solution of equation 2.1.1 where the number of equations equals the number of unknowns. However, real world data which describe a system could yield a nonsquare matrix and the technique used to solve for this system of equations is referred to as the pseudoinverse or generalized inverse - see [7]. If a matrix A has more rows than columns, the generalized inverse can be defined as

$$A^+ = (A^TA)^{-1}A^T$$

for a nonsingular (A^TA). A solution of equation 2.1.1 under these conditions is called the over determined case; there are more equations than unknowns. The resulting solution can be expressed as the column vector $\overline{C}$, composed of the coefficients $c_0, c_1, \ldots, c_n$, which equals the generalized inverse of the square matrix ψ^+ times the column vector $\overline{b}$. Written mathematically we have

$$\overline{C} = \psi^+ \overline{b},$$

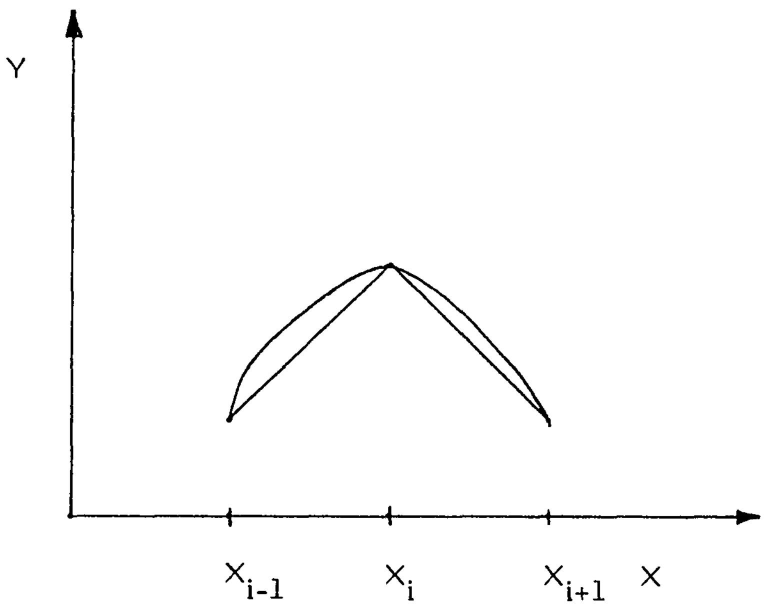
is the minimum in the least square sense. See Appendix B for a proof that $\overline{C} = \psi^+ \overline{b}$ is the optimum least square solution. If we have more unknowns than equations, this condition is referred to as the under determined case. This gives an infinite number of least square solutions. The generalized inverse for the under determined case (more columns than rows) is written as

$$A^+ = A^T (AA)^{-1}.$$

In this paper we will restrict ourselves to the over determined case. A detailed discussion of the properties of the generalized inverse can be found in [7,9].

2.3 Optimum Breakpoint Location

Up until now we have presented a discussion of least square polynomial approximation on a set of data in [a,b]. In order to avoid confusion all input data will be normalized onto the -1 to 1 interval. The concept of using an approximating function to describe a contour fails if the resulting solution exceeds a predetermined error threshold $E_T = \| f - F_n \|$. When this happens it indicates that a corner or breakpoint exists within the interval [a,b] and we need to use a piecewise polynomial approach to find the location of the breakpoint(s) within [a,b]. A breakpoint, in this paper, is defined as a change in the slope of the given contour which causes the error threshold to be exceeded. Or said another way, we would like to find the location of the breakpoint without using a trial and error approach. A technique which is widely used [45]where approximating piecewise polynomials are used to accomplish the above is called variable breakpoint location. For example, in design applications where an interactive graphics system is used, it is possible to specify the breakpoint locations which minimize either the number of describing polynomials or the error caused by theapproximations. Thus, it is desirable to have a technique for finding the optimum breakpoint location within an interval. As shown above, the partial derivatives of equation 1.2.4, with respect to the coefficients c_i, give the normal equation 1.2.6. For optimum breakpoint location the partial derivative of E_T , with respect to x in equation 1.2.9 as shown in Figure 2.3.1, yields



Optimun Breakpoint Location

Figure 2.3.1

$$e_i^2(x_i) - e_{i+1}^2(x_i) = 0 \qquad 2.3.1$$

where e_i is the error associated with F_i at location x_i and e_{i+1} is the error associated with F_{i+1} at x_i and $i = 1,2,\ldots,n-1$. Equation 2.3.1 shows that for optimum breakpoint location the absolute values of the pointwise errors from the left and right must be equal. In order to find the minimum, i.e. the optimum breakpoint location of equation 2.3.1, the matrix of the second derivatives $\frac{\partial^2 E_T}{\partial x^2}$ must be positive definite. A detailed discussion of positive definite matrices can be found in any text on linear algebra [18]. Pavlidis [39] has proved conditions for the matrix required for the equation

$$\frac{\partial^2 E_T}{\partial x_i} = [f(x_i) - F_i(x_i)]^2 - [f(x_i) - F_{i+1}(x_i)]$$

to be positive definite, see Appendix A for the detailed calculations. Pavlidis's proof shows the necessary and sufficient conditions for a positive definite matrix of the above equation to be:

$$\frac{\partial^2 E_T}{\partial x_{i-1} \partial x_i} = \frac{2Z_2 e_i(x)_{i-1} e_i(x_i)}{x_i - x_{i-1}}, \qquad 2.3.2$$

$$\frac{\partial^2 E_T}{\partial_{x_i}^{\ 2}} = 2e_i(x_i)\dot{e}_i(x_i) - 2e_{i+1}(x_i)\dot{e}_{i+1} - \frac{2Z_1 e_i^2(x_i)}{x_i - x_{i-1}} - \frac{2Z_1 e_{i+1}^2(x_i)}{x_{i+1} - x_i} \qquad 2.3.3$$

$$\frac{\partial^2 E_T}{\partial x_i \partial x_{i+1}} = \frac{2Z_2 e_{i+1}(x_i) e_{i+1}(x_{i+1})}{x_{i+2} - x_i} \qquad 2.3.4$$

$$\frac{\partial^2 E_T}{\partial x_i \partial x_K} = 0 \qquad 2.3.5$$

where

$z_1 = n^2$,

$z_2 = (-1)^{n-1} n$,

$e_i(x_i)$ = pointwise error at x_i,

$\dot{e}_i(x_i)$ = derivative of the pointwise error,

and

n = number of terms.

There are two conditions which will exist at the breakpoint x_i which are of interest. First, if the solution is symmetrical, that is $e_i(x_i) = -e_{i+1}(x_i)$, then the pointwise error can be defined as

$$\frac{\partial^2 E_T}{\partial x_i^2} = 2e_i(x_i)[\dot{e}_i(x_i) + \dot{e}_{i+1}(x_i)]. \qquad 2.3.6$$

Second, if the solution is continuous, i.e. $e_i(x_i) = e_{i+1}(x_i)$, we have

$$\frac{\partial^2 E_T}{\partial x_i^2} = 2e_i(x_i)[\dot{e}_i(x_i) - \dot{e}_{i+1}(x_i)]. \qquad 2.3.7$$

Equation 2.3.6 says that the optimum breakpoint location is where the approximation from the left and right is symmetrical and where the slope of f(x) minus the average slope of the approximation, $F_n(x)$ must have the same sign as the pointwise error. Equation 2.3.7 says that the slope of the approximation on one side of the breakpoint minus the slope of other side has the same sign as the pointwise error.

CHAPTER III

FUZZY THEORY

3.0 Introduction

In this chapter we will consider fuzzy sets and show a fuzzy Bayes model for use in decision making. The problem we are interested in is the ability of an edge detection algorithm's decision making process, that is to make decisions against an error threshold, given some a priori data. This effort is motivated by Zadeh [58], Terano and Sugeno [53] because for curve detection the precise description of a contour is not absolutely required. As described in Chapter I, we are interested in the magnitude of change of the slope at a corner within the interval $[a,b]$, where corner has a fuzzy definition. Before presenting the development of the fuzzy Bayes model, a brief description of fuzzy mathematical concepts will be presented in the next section. But, first we will present some set definitions which will be used through out the discussion. If we are given a space U, which contains a well defined set A, then we can determine whether each element u_i in U belongs to A. This can be written as $u_i \varepsilon A$ if u_i is an element in A. If u_i does not belong to A this can be written as $u_i \notin A$. If we are given two sets in U, namely A and B, then we can make the statement that A is contained in B if every element of A is contained in B which can be written symbolically as $A \subset B$ or $B \supset A$.

3.1 Fuzzy Sets

Fuzzy sets in this paper will be represented by $\tilde{A}$ and non fuzzy sets by A without the $\sim$ above the symbol.

A fuzzy subset $\tilde{A}$ of a set of discourse E is defined by the membership function $\mu_{\tilde{A}}: E \rightarrow [0,1]$ which associates with each element x_i of E a number $\mu(x_i)$ in the interval $[0,1]$. Thus, the fuzzy subset $\tilde{A}$ of E can be denoted as

$$\tilde{A} \subset E$$

and the membership of elements within a fuzzy subset can be denoted as

$$x \underset{.1}{\varepsilon} \tilde{A},\ y \underset{1}{\varepsilon} \tilde{A},\ \text{and}\ z \underset{.5}{\varepsilon} \tilde{A}$$

where the number below the symbol represents the grade(membership) of the element(s) within the specified fuzzy set; i.e., $\mu(x) = .1$, $\mu\ (y) = 1$, and $\mu\,(z) = .5$.

Definition 3.1.1

Given two fuzzy sets $\tilde{A}$ and $\tilde{B}$ over E, with membership functions $\mu_A(x)$ and $\mu_B(x)$ respectively, then one can introduce the following:

1. The union of $\tilde{A}$ and $\tilde{B}$ symbolized by $\tilde{A} \cup \tilde{B}$ is defined as the smallest fuzzy subset that contains both $\tilde{A}$ and $\tilde{B}$. That is to say,

   $$\forall x \,\varepsilon\, E: \mu_{\tilde{A} \cup \tilde{B}}(x) = \vee[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)]$$

   where v represents the maximum value with respect to the variable x.

2. The intersection of $\tilde{A}$ and $\tilde{B}$ symbolized by $\tilde{A} \cap \tilde{B}$ is defined as the largest fuzzy subset containing $\tilde{A}$ and $\tilde{B}$ simultaneously. That is to say,

   $$\forall x \,\varepsilon\, E: \mu_{\tilde{A} \cap \tilde{B}}(x) = \wedge[\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)]$$

   where $\wedge$ represents the minimum value with respect to the variable x.

3. The negation $\underline{\tilde{B}}$ of a fuzzy set $\tilde{A}$ of E is defined as

   $$\forall x \,\varepsilon\, E: \mu_{\tilde{B}}(x) = 1 - \mu_{\tilde{A}}(x).$$

   This can also be denoted as

   $$\underline{\tilde{B}} = \tilde{A}$$

   or $\underline{\tilde{A}} = \tilde{B}.$

   In some texts negation is called complementation.

Definition 3.1.2

The α level set of a fuzzy subset $\tilde{A}$ of E is a nonfuzzy subset of E denoted by A_α and defined as

$$A_\alpha = (x \mid \mu_{\tilde{A}}(x) \geq \alpha, x \in E).$$

That is A_α is a subset of E whose members all have a grade of membership in A greater than or equal to α .

Definition 3.1.3

The disjunctive sum of two fuzzy sets $\tilde{A}$ and $\tilde{B}$ is defined in terms of unions and intersections and can be expressed as

$$\tilde{A} + \tilde{B} = (\tilde{A} \cap \underline{\tilde{B}}) \cup (\underline{\tilde{A}} \cap \tilde{B}).$$

The theory of fuzzy sets is based upon nonstatistical concepts and yet many readers think that probability theory provides all the necessary concepts and techniques for solving vague or ill defined problems. The latter is not true because probability is concerned with the random occurrence of an event whereas fuzzy set theory deals with the situation where the object(event) is imprecise. That is to say, fuzzy set theory assigns to each element of a set a value of the membership function, $\mu: E \rightarrow [0,1]$. Probability theory assigns a number $p \in [0,1]$ to an element which constitutes a probability of an occurrence. We can assign a probability value to a fuzzy set, however we cannot do the reverse because we would violate the fundamental axioms of probability as given in [38] . Furthermore, probability theory is based upon the theory of distributive and complemented lattice or Boolean lattice whereas fuzzy theory is based upon the theory of vector lattice [28] . The fundamental differences between vector and Boolean lattices are

(a) The vector lattice is a totally ordered product set and

(b) the Boolean lattice has the least and greatest elements 0 and 1.

Also, for the Boolean lattice the condition $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ holds and for any "a" there exists an element a^1 in the lattice such that $a^\vee a^1 =$

1 and $a \wedge a^1 = 0$. A detailed discussion of fuzzy subset theory and its operations can be found in [22 28, 58].

3.2 Fuzzy Measures

As mentioned in Chapter I we will use a fuzzy Bayes' model to aid in the detection of edges. The development of this model requires us to investigate measure theory which also forms the fundamental base for probability theory. We start our development by discussing some of the key attributes of measure theory which will lead to the definition of a fuzzy measure space. The difference between probability measure space and fuzzy measure space is shown in the sequel.

We will start the development of fuzzy measures by considering only functions $\nu: K \rightarrow R^*$, where K is a non-empty class of sets and R^* is the extended real number system which is simply ordered if we put $-\infty < x < +\infty$ for all $x \varepsilon R$ where R denotes the real numbers. In the above, ν defines for each $E \varepsilon K$ a unique value which is either a real number or $\pm\infty$. It is assumed that the empty set $\emptyset$ is always contained within K.

We will define a space as having a structure which is described in terms of a class of subsets called "open". One way of obtaining this class of open sets is to define a distance between a pair of points located within the space. Thus a non-empty set E, together with a function $\nu : E \times E \rightarrow R$, which is now defined as a distance function, will form a metric space if

(a) $\nu(x,y) = \nu(y,x) \geq 0$ for all $x,y \varepsilon E$

(b) $\nu(x,y) = 0$ if and only if $x = y$

and

(c) $\nu(x,y) \leq \nu(x,z) + \nu(z,y)$ for all $x,y,z \varepsilon E$.

If we think of $\nu(x, y)$ as the distance between the points x and y, then (c) above can be called the triangle inequality.

A set function $\nu : K \to R^*$ can be defined as being finitely additive

(d) if $\nu(\emptyset) = 0$ and

(e) for every finite disjoint collections of sets $E_1, E_2, \ldots\ldots, E_n$ in K whose union is also in K, we have

$$\nu\left(\bigcup_{i=1}^{n} E_i\right) = \sum_{i=1}^{n} \nu(E_i). \qquad 3.2.1$$

A ring is any non-empty class, say K, of subsets which are closed under the operations of union, intersection, difference, and $\emptyset \varepsilon K$. For any ring K the set function $\nu: K \to R^*$ is defined to be additive if, and only if, $\nu(0) = 0$ where

$$E, F \varepsilon K;$$

$$E \cap F = \emptyset \qquad 3.2.2$$

and

$$\nu(E \cap F) = \nu(E) + \nu(F). \qquad 3.2.3$$

From these conditions we can now obtain equation 3.2.1. A σ additive set function ν $: K \to R^*$ is defined to be additive

(f) if $\nu(\emptyset) = 0$, and

(g) for any sequence $E_1, E_2, ----, E_n$ which are disjointed sets in K such that

$$E = \bigcup_{i=1}^{n} E_i \varepsilon K,$$

and

$$\nu(E) = \sum_{i=1}^{\infty} \nu(E_i).$$

Now a measure can be defined as a non-negative set function $\nu: K \to R^+$ which is σ additive on K; where

$$R^+ = \{x \varepsilon R^* : x \geq 0\} .$$

A standard way of defining a measure on a subset of R is to consider R as a line of infinite length which is composed of an infinite number of intervals. Then we can define a length $\nu(n)$ of an interval as the difference between its end points. The domain can be defined as a collection of all intervals which are defined by a set function. This set function assigns an extended real number, $\nu(n) \leq \pm \infty$, to each set in a class. This same idea can be extended to area, volume, and higher dimensional geometry.

Theorem 3.2.1 [23]

If ν is a measure on a ring K, and if $\{E_n\}$ where $E_1 \subset E_2 \subset \ldots \subset \lim_n E_n = \bigcup_{K=1}^{\infty} E_k$ is an increasing sequence of sets in K for which $\lim_n E_n \varepsilon K$ then

$\nu(\lim_n E_n) = \lim_n \nu(E_n)$.

Theorem 3.2.2 [23]

If ν is a measure on a ring K, and if $\{E_n\}$ where $E_1 \subset E_2 \subset \ldots\ldots \subset \lim_n E_n = \bigcap_{K=1}^{n} E_k$ is a decreasing sequence of sets in K of which at least one has finite measure and for which $\lim_n E_n \varepsilon K$, then $\nu(\lim_n E_n) = \lim_n \nu(E_n)$.

Now we can state that ν, provided it is a measure as defined in Theorems 3.2.1 and 3.2.2, is continuous from above and from below E in the $\lim_n E_n$. Up until now we have discussed the fundamental theory behind Lebesgue measure. If more detail is desired, the reader should consult [24]. Lebesgue measures consist of a set function with monotonicity and σ additivity. However, for fuzzy set measures, the additivity requirement can be relaxed to the point where it applies only some of the time. In the human thought processes additivity has not been proved a necessary or a sufficient requirement for information processing, as it is for example in probability measures. Fuzzy set theory uses the term "membership" or "grade" to define elements belonging to a set whereas probability theory uses the term "probability of occurrence" or "randomness of an element" to define which elements which occur given an experimental outcome. For the definition of fuzzy measure, in this paper, we will use the term ϕ for the grade of an element(s) belonging to a given set. We start by defining an arbritrary set U as the universe of discourse and an empty set as ϕ.

We are now in a position, after some brief definitions on measure theory, to define a fuzzy measure space. A function, $\nu:\omega \rightarrow R^*$, is defined to be B-measurable if and only if

$$\nu^{-1}(A) \varepsilon B$$

for every $A \varepsilon a^*$. B-measurable refers to a Borel set [23,38] and a^* is defined as the class of Borel sets in R^*. Now, using Terano's and Sugeno's [53] definitions, we define a fuzzy measure space as a triplet (X,B,ϕ) where B is the Borel field. If we let $h:X \rightarrow [0,1]$ and $F_\alpha = \{X \mid h(X) \geq \alpha\}$ and if $F_\alpha \subset \tilde{B}$, then h can be called B-measurable. α is defined as a level or threshold value and h as a mapping from X to [0,1]. This leads to the following fuzzy measure definition which can be found in [53]

.

Definition 3.2.1

A set function $\phi(X)$ with the following properties is called a fuzzy measure:

(h) $\phi(\emptyset) = 0$, $\phi(X) = 1$

(i) $A,B \varepsilon \tilde{B}$ and $A \subset \tilde{B}$ then $\phi(A) \leq \phi(B)$

(j) $F_n \varepsilon \tilde{B}$ and F_n is a montone sequence, then $\lim_{n\rightarrow\infty} \phi(F_n) = \phi(\lim_{n\rightarrow\infty} F_n)$.

From Definition 3.2.1 we see that (h) means boundedness and non-negativity, (i) means monotonicity, and (j) means continuity. Condition (j) can be dropped if our set X is finite.

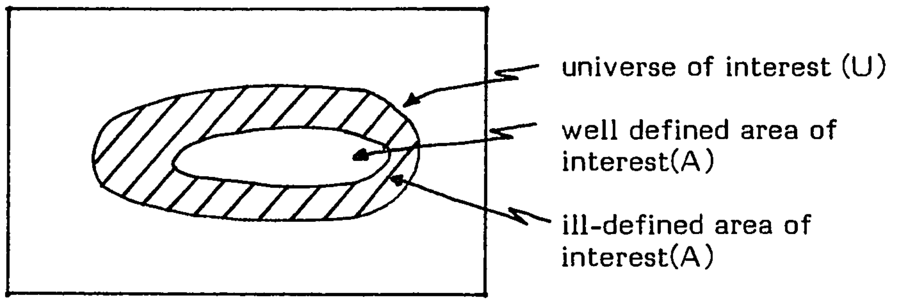
As a comparison to probability space, we know from [38] that the triplet (X, B, P) has the following properties:

(k) $P(0) = \emptyset$, $P(X) = 1$

(l) $A,B \varepsilon C$ and $A \subset B$ then $P(A) \leq P(B)$

(m) $F_n \varepsilon C$ then $\lim_{n\rightarrow\infty}(F_n) = P(\lim_{n\rightarrow\infty} F_n)$.

What we have shown in (h) through (m) is that probability measures are a subset of fuzzy measures. This is also supported by the fundamental definition of fuzzy subset theory as defined in [58], that is $A \subset \tilde{A}$. Or, from another point of view, probability measures are more restrictive than fuzzy measures. In [28]and [58] the authors define fuzzy sets with respect to ordinary set theory. Basically fuzzy sets cover the entire universe of interest, including regions which are ill-defined, whereas ordinary set theory covers well defined regions as illustrated in Figure 3.2.1.


universe of interest (U)
well defined area of interest(A)
ill-defined area of interest(A)

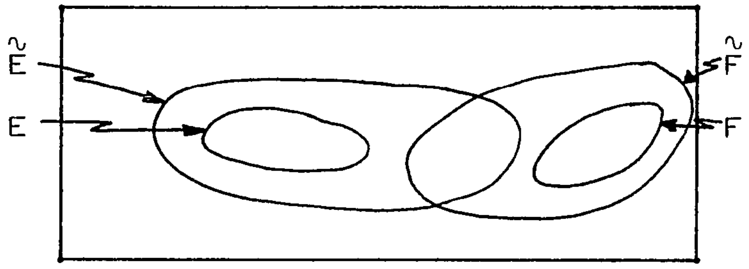
Definition of Fuzzy and Ordinary Regions

Figure 3.2.1

As can be seen in Figure 3.2.1, an ordinary set A is a subset of the fuzzy set $\tilde{A}$ which can be written as $A \subset \tilde{A}$.

We can measure fuzzy occurrence by using probability measures with additivity. Therefore we can write $P(\tilde{A})$, which represents the randomness of a fuzzy event. However, we cannot do the reverse since it has not been proven that additivity exists within fuzzy theory. This is because of the human subjective input. Thus, a good assumption would be to throw away the additivity requirement in the fuzzy measure as Terano and Sugeno have done. What this means is that equations 3.2.2 and 3.2.3 do not necessarily hold. For example, in Figure 3.2.2 we are given two sets E and F in which $\tilde{E} \cap \tilde{F} \neq 0$ but $E \cap F = \emptyset$. That is, for a given problem space the quantities $\tilde{E}$ and $\tilde{F}$ could be ill-defined where the condition $\tilde{E} \cap \tilde{F} \neq 0$ exists. This indicates that the additivity term is empty. But for well defined quantities, where probability measures hold, the additivity term must be empty i.e.

$E \cap F = \phi$. We will return to Figure 3.2.2 in order to give a complete description of fuzzy measures after we have developed some fundamental fuzzy integral definitions.



Additivity Fails

Figure 3.2.2

3.3 Fuzzy Integrals

The next step in the development of our model will be to define an integral of a function on the fuzzy measure space (X, B, ϕ) where X is a space, B a σ field of subsets X, and ϕ a measure on B . Since fuzzy set theory covers a more general domain than ordinary set theory, as shown in Figure 3.2.1, we can now write fuzzy functions which look a lot like ordinary set functions. Thus, we are now in a position to define a fuzzy integral which is very similar to the Lebesgue integral as defined in [38]. We start by defining a simple function as

$$f(x) = \sum_{i=1}^{n} \beta_i \underline{\overline{X}}_{E_i}(x), \qquad 3.3.1$$

where β_i is a finite set of real numbers and $\underline{\overline{X}}_E(x)$ is the characteristic function of a measurable set E. The characteristic function is defined as

$$\underline{\overline{X}}_{E_i}(x) = 1 \text{ if } x \in E_i$$

or

$$= 0 \text{ if } x \notin E_i$$

Thus,

$$f(x) = \begin{cases} \beta_i \text{ if } x\varepsilon E_i=1,\dots n, \\ 0 \text{ if } x \notin E_i \cup \dots\dots \cup E_n \end{cases}.$$

The simple function, equation 3.3.1, has been defined on a measurable space which is composed of disjoint measurable sets such that

$$E_i \cap E_j = \emptyset \text{ for } i \neq j \qquad 3.3.2$$

and

$$\underline{\overline{X}}_{E_i}(x) = \bigcup_{i=1}^{n} \underline{\overline{X}}_{E_i}$$

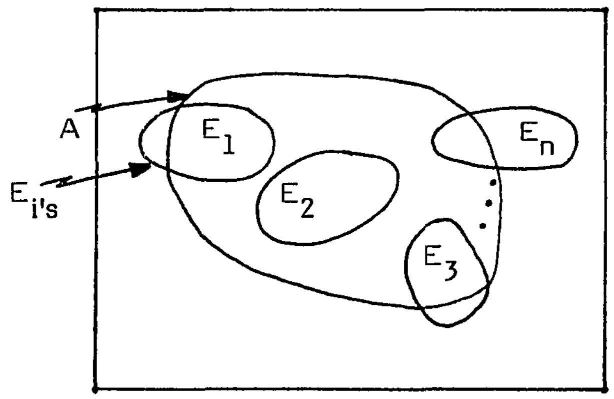
The integral of the non-negative simple function, equation 3.3.1, can be expressed as

$$\int f(\phi) d\underline{\overline{X}}(\phi) = \sum_{i=1}^{n} \beta_i \phi(E_i)$$

where the E_i's are measurable sets in the space x and $\beta_i \geq 0$ for i = 1, 2, ..., n. The above integral amounts to averaging the function over the given space. In order to show the similarity between Lebesgue integrals and the fuzzy Lebesgue integrals, we will define very briefly the Lebesgue integral. First let $\underline{\overline{X}}$ be a Lebesgue measure on the σ field B of Lebesgue measurable sets . Then the Lebesgue integral, on the measure space $(x, B, \underline{\overline{X}})$ of f over E, is defined as

$$\int_E f(\underline{\overline{X}}) d\underline{\overline{X}} = \sum_{i=1}^{n} \beta_i \phi(A \cap E_i) \qquad 3.3.3$$

where E is a Lebesgue measurable set in x. We have assumed that $0 \leq \beta_i \leq 1$ for $1 \leq i \leq n$ and that β_i is an ordered monotone increasing sequence. Graphically, the right



Lebesgue Integral Defined Over A

Figure 3.3.1

side of equation 3.3.3 is shown in Figure 3.3.1 where equation 3.3.2 holds and

$$\underline{X}_{E_i} = 1 \quad x \in E_i$$
$$= 0 \quad x \notin E_i .$$

Now equation 3.3.1 can be written as

$$f(x) = \bigvee_{i=1}^{n} [\beta_i \wedge \underline{X}_{E_i}(X)] . \qquad 3.3.4$$

This means equation 3.3.1 can also now be written in a fuzzy form similar to equation 3.3.4. We start by writing a simple fuzzy function as

$$\tilde{f}(x) = \sum_{i=1}^{n} \lambda_i \underline{X}_{F_i}(X).$$

The above fuzzy function can be expressed in the same form as in equation 3.3.4 namely

$$f(x) = \bigvee_{i=1}^{n} [\lambda_i \wedge \underline{X}_{F_i}(x)] . \qquad 3.3.5$$

Thus, we can see the similarity between equations 3.3.4 and 3.3.5. The fuzzy integral, as defined in [53] over A εB of f(x) with respect to a fuzzy measure $\phi(x)$ can be defined as

$$\tilde{\int_A} f(x) \circ \phi(x) = \bigvee_{\lambda \varepsilon [0,1]} [\lambda \wedge \phi(A \cap F_\lambda)] \qquad 3.3.6$$

where

$$F_\lambda = \{x \mid \tilde{f}(x) \geq \lambda\}.$$

Equation 3.3.6 can be rewritten as

$$\tilde{\int_A} \tilde{f}(x) \circ \phi(x) = \bigvee_{i=1}^{n} [f(x_i) \wedge \phi(A \cap F_i)]$$
$$= \bigvee_{i=1}^{n} [\tilde{f}(x_i) \wedge \phi(F_i)]$$

if $\tilde{f}(x_i)$ is arranged in increasing order and if we let $F_i = \{x_i, x_{i+i}, \text{-----}, x_n\}$.

Then another way of writing equation 3.3.6 is as follows:

$$\int f d\underline{x} = \bigvee \tilde{\int} \phi \, d\underline{x}$$
$$= \bigvee [\sum_{i=1}^{n} \lambda_i \phi(A \cap E_i)]$$

$$= \bigvee_{i=1}^{n} [\lambda_i \wedge \phi(A \cap E_i)].$$

The symbol for the fuzzy integral will be written as $\tilde{\int}$.

In the above we refer to $F_\lambda \subset \tilde{B}$ which represents a class of all fuzzy subsets of A with B- measurable membership functions. As defined earlier, $B \subset \tilde{B}$ can be viewed as an extension of B into $\tilde{B}$ which preserves all the properties of B. These properties are defined in detail in [53]. As pointed out in [26]and[53]these fuzzy integrals are also called fuzzy expectations and can be compared to the probabilistic expectations as defined in [38]. In [26] the authors point out that the difference between the probabilistic expected value and the fuzzy expected value when defined with respect to p is

$$|\Delta| = |\int f(x)dp - \int f(x) \circ p \; (F_\lambda)| \leq ¼.$$

That is by using subjective information via the fuzzy membership value the average difference between the fuzzy expected value and the probabilistic expected value could be as large as 25%. Take for example five people whose monthly incomes are respectively \$2200,\$2500,\$2700,\$3500, and \$10000. In order to compute the fuzzy expected value(FEV) we need to form the fuzzy measure, which acts as a standard probabilistic measure [53], from

$$\mu_j(\tau_\lambda) = \mu[y|z(y) \geq \lambda] = g_p(\lambda)$$

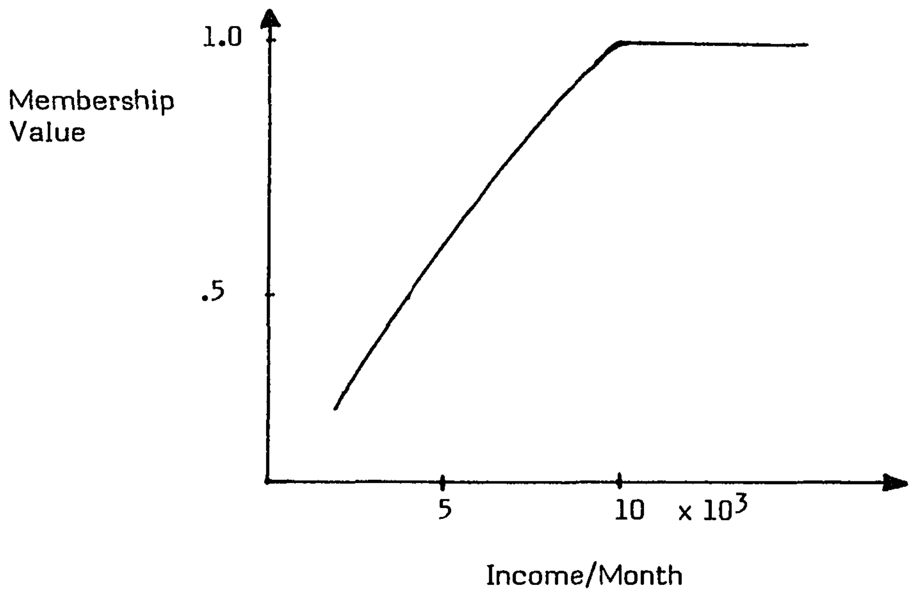
where

$$g_p(\lambda) = \frac{|\lambda|}{P}$$

and

P = total population.

Next the fuzzy density, $\tilde{f}_{x_i}$, must be formed. In this paper the fuzzy density values are derived from a subjective rating curve as shown in Figure 3.3.2. Forming the union of the fuzzy density and fuzzy measure we get a set of 2n + 1 elements. This set of 2n + 1 elements is arranged in increasing order because of the Max-Min comparison operations. That is, we will take the minimum of every comparison. The results must lie to the left of the element position n+1. Next we take the maximum of these minimum values, which obviously is the value in position n+1, because the array is ordered. This number is the mean of the array. Using the subjective rating values from Figure 3.3.2 the fuzzy densities are



Subjective Rating
Figure 3.3.2

$2200 = .25

$2500 = .30

$2700 = .35

$3500 = .45

$10000 = 1.00

Next we compute the fuzzy measure $\mu_j(\tau_\lambda)$ according to the above defined F_i sequence. This computation yields:

$\mu_1(\tau_\lambda) = .8$

$\mu_2(\tau_\lambda) = .6$

$\mu_3(\tau_\lambda) = .4$

$\mu_4(\tau_\lambda) = .2$

where

$j = i-1.$

The fuzzy density values, which include 0 and 1, give different thresholds within the interval [0,1] . Forming the union of the fuzzy measure and fuzzy density values we get .2,.25,.3,.35,.4,.45,.6,.8, and 1.0. The FEV as discussed above will be found in position (n+1)/2 and for this example the value is .4. The probabilistic expected value(mean) of the subjective ratings on the [0,1] interval is .47. The difference between the two methods of computing the expected value is .07 or 7%. If we change our subjective rating to

$2200 = .10

$2500 = .12

$2700 = .35

$3500 = .98

$10000 = 1.0

then the probabilistic expected value is .51. The difference between the FEV value and the mean value for the second case is .11. We see that a change did occur between the probabilistic expected value and the FEV. It is possible to have a difference between the mean and the FEV greater than .25. However, the data in that case would not make any sense; i.e. the data would not be realistic. As another example, if we changed the subjective rating of the above case to

$2200 = .10

$2500 = .12

$2700 = .70

$3500 = .98

$10000 = 1.00

the probabilistic expected value becomes .54 but the FEV becomes .6. We see from the above examlpes that the FEV varies according to the subjective rating scheme. This comes as no surprise because fuzzy set theory allows for subjective selection of the membership function.

Since fuzzy subset theory is based upon a membership function, we can now let $f_y(x)$ be a membership function of a set Y . The fuzzy measure of Y can be defined as

$$\tilde{\phi}(y) = \tilde{\int} f_y(x) \circ \phi(x), \qquad 3.3.7$$

and the fuzzy integral over Y as

$$\tilde{\int}_y f(x) \circ \phi(x) = \tilde{\int} [f_y(x) \wedge f(x)] \circ \phi(x).$$

From the above definitions we can see that fuzzy measures include probability measures as a special case. However, fuzzy measures cannot be used in a probabilistic setting because of the subjective nature of fuzzy subset theory.

The additivity of Lebesgue measures is a fundamental concept in measure theory. This can be seen if we let α be a Lebesgue measure, as defined in [24]. Then it follows that

(a) if $A \cap B = 0$ then 3.3.8

$$\int_{AUB} h(x)d\alpha = \int_A h(x)d\alpha + \int_B h(x)d\alpha, \qquad 3.3.9$$

(b) 3.3.10

$$\int_X (h_1(x) + h_2(x))d\alpha = \int_X h_1(x)d\alpha + \int_X h_2(x)d\alpha$$

However, for fuzzy integrals, monotonicity is the fundamental concept which gives

(c) if $A \subset B$ 3.3.11

then

$$\tilde{\int}_A h(x) \circ \phi(x) \leq \tilde{\int}_B h(x) \circ \phi(x) \qquad 3.3.12$$

(d) if $h_1 \leq h_2$ 3.3.13

then

$$\tilde{\int}_X h_1(x) \circ \phi(x) \leq \tilde{\int}_X h_2(x) \circ \phi(x). \qquad 3.3.14$$

If equations 3.3.8 thru 3.3.14 are viewed from the concept of functionals then the Lebesgue integrals (equations 3.3.8 thru 3.3.10) are linear functionals and the fuzzy integrals (equations 3.3.11 thru 3.3.14) are non-linear functionals where a functional is an operation which assigns a number to a function.

Now we can define fuzzy conditional measures which are similar to probability conditional measures which possess the following properties. We start by letting (X, B_X, ϕ_x) be a fuzzy measure space and (y, B_y) be a Borel measurable space. Also, let a fuzzy conditional measure with respect to x be written as $\xi_y(F|x)$ with the following conditions:

(a) For a fixed $F \varepsilon B_y$, $\xi(F|x)$ is, as a function of x, a B_X - measurable function.

(b) For a fixed $x \varepsilon B, \xi_y(F|x)$ is a fuzzy measure of (y, B_y).

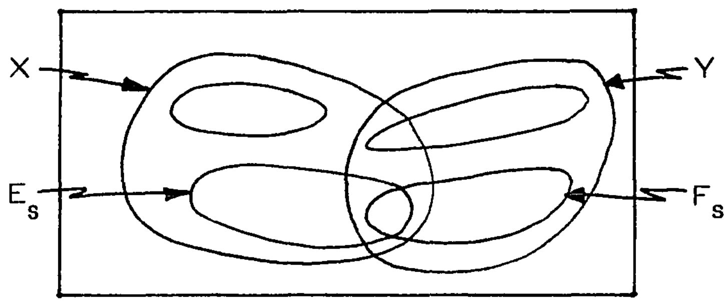
A fuzzy measure of Y can be written in a form similar to equation 3.3.6 by using the above defintions in (a) and (b). Writing this equation, we have

$$\phi_y(F) = \tilde{\int}_x \tilde{\int}_y \xi(F|x) \circ \phi_x \ (E), \qquad 3.3.15$$

or written another way,

$$\tilde{\int}_y h(y) \circ \phi_y(F) = \tilde{\int}_x [\ \tilde{\int}_y h(y) \circ \xi_y(F|x)] \circ \phi_x(E).$$

See Figure 3.3.3 for a graphical definition of F.



E⊂X and F ⊂ Y and the Conditionals are Defined on Their Respective Sets Within X and Y.

Figure 3.3.3

The expression for $\xi_x(E|y)$can be written as

$$\phi_x(E) = \tilde{\int}_x \xi_x(E|y) \circ \phi_y(F) \qquad 3.3.16$$

or

$$\tilde{\int}_x h(x) \circ \phi_x(E) = \tilde{\int}_y [\tilde{\int}_x h(x) \circ \xi_x(E|y) \circ \phi_y(F).$$

See Figure 3.3.3 for a definition of E. Now from equations 3.3.15 and 3.3.16 we can write an expression showing the relationship between $\xi_y(F|x)$and $\xi_x(E|y)$. However, before the relationship can be expressed the sets E and F, as shown in Figure 3.3.3, must be defined as E⊂X and F⊂Y. Futher, if we let$\phi_x(E) = \phi_y(F)$ then the relationship can be expressed as

$$\tilde{\int}_F \xi_x(E|y) \circ \phi_y(F) = \tilde{\int}_E \xi_y(F|x) \circ \phi_x(E). \qquad 3.3.17$$

As can be seen, equation 3.3.17 is Bayes' formula written in fuzzy terms, see [38], for the probabilistic expression of Bayes' formula. In equation 3.3.17,$\phi_x(E)$is called the a priori fuzzy measure and $\xi_x(E|y)$ is called the a posteriori fuzzy measure. Equation 3.3.17 can be rewritten as

$$\xi_x(E|y) = \frac{\xi_y(F|x) \circ \phi_x(E)}{\int \phi_y(F)}$$

$$= \frac{\xi_y(F|x) \circ \phi_x(E)}{\tilde{\int_F} \xi_y(F|x) \circ \phi(E).} \qquad 3.3.18$$

As can be seen, equation 3.3.18 represents the continuous case for fuzzy measures.

The generation of fuzzy measures as described below is according to Terano and Sugeno [53]. They define a finite set X as $\{x_1, \dots, x_n\}$ and then consider how to generate a fuzzy measure of a fuzzy measure space $(x, 2^x, \phi)$. First they start by letting $0 \le \phi^i \le 1$ for $1 \le i \le n$ where ϕ^i is defined as the fuzzy density. Next[53] defines a λ rule as

$$\lambda = \prod_{i=1}^{n} (1+\lambda\phi^i) - 1 \qquad 3.3.19$$

where

$$-1 < \lambda < \infty .$$

Now for a set $E \subset X$, as defined earlier, we have

$$\phi_\lambda(E) = \frac{1}{\lambda}\left[\prod_{i=1}^{n} (1+\lambda\phi^i) - 1\right]$$

where all i's ε E. Thus with the aid of the above definition, we can obtain $\phi_\lambda[\{X_i\}] = \phi^i$ for $1 \le i \le n$ and if $A \cap B \ne 0$, then

$$\phi_\lambda(A \cup B) = \phi_\lambda(A) + \phi_\lambda(B) + \lambda\phi_\lambda(A)\phi_\lambda(B). \qquad 3.3.20$$

If $\lambda = 0$, then $\phi_\lambda(A \cup B)$ becomes a probability measure. From equation 3.3.20, if $\lambda > 0$ then

$$\phi_\lambda(A \cup B) > \phi_\lambda(A) + \phi_\lambda(B)$$

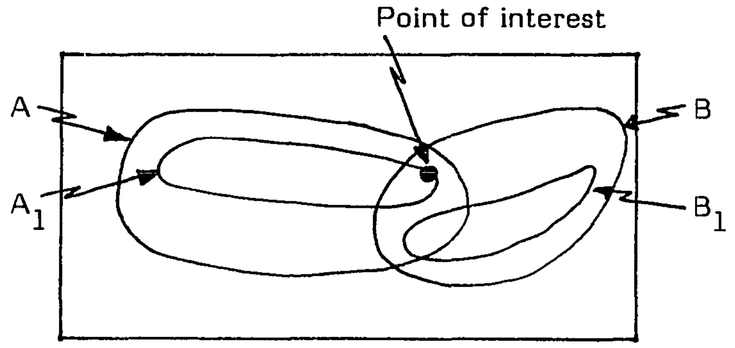
and if $\lambda \le 0$ then

$$\phi_\lambda(A \cup B) \le \phi_\lambda(A) + \phi_\lambda(B).$$

The lambda rule is a technique to incorporate additivity into fuzzy measure theory. For example, as shown in Figure 3.3.4, if the subspaces A_1 and B_1 exhibit $A_1 \cap B_1 = 0$ then what is the additivity i.e. ϕ_λ $(A \cup B)$, or what is the solution to equation 3.3.20? If the point of interest occurs in A_1, as shown in Figure 3.3.4, the effect can be determined by solving equation 3.3.19 for λ . Thus, given a fuzzy density of ϕ^i, lambda will give a measure of spaces or subspace intersections, i.e. a grade of relationship of spaces or subspaces with respect to one another. For this example we have $\phi_A^i=1$ and $\phi_B^i=0$. Substituting these values into equation 3.3.19 and solving for lambda we get λ =0. Thus, equation 3.3.20 reduces to

$$\phi_\lambda(A \cup B) = \phi_\lambda(A) + \phi_\lambda(B)$$

as expected because of the condition $A_1 \cap B_1 = 0$.



Space and Subspace Intersections

Figure 3.3.4

CHAPTER IV

CORNER/EDGE DETECTION CRITERION

4.0 Introduction

In this chapter we are concerned with the application of the above techniques to corner and edge detection within a noisy environment. The technique of describing an object's boundaries by using piecewise linear approximating polynomials was motivated by Albano [4], McClure [33], and Pavlidis [39]. Also, this motivation is driven by a need to develop an algorithm which is computationally fast and by the desire for data compaction. It has been shown elsewhere, [14, 33, 45], that approximating polynomials of order $n \geq 3$ are computationally complex and time consuming. Higher order approximating polynomials give a smooth continuous contour for curve fitting on the given data space. However, these polynomials normally have a strong continuity requirement which is not necessarily a requirement for corner and edge detection because of a possibility of discontinuities in the pattern. Pavlidis [39] has shown a number of ways to approach the contour approximating polynomial problem when a continuity requirement is desired. For linear polynomial approximations Pavlidis [39] has shown, corners are likely to occur near a maxima of the second derivative of the approximating polynomial and, by minimizing the least square error, the location of the maxima of curvature can be determined. The question to be answered next is what is the definition of a corner or maximum of curvature. This question will be answered in the following sequence.

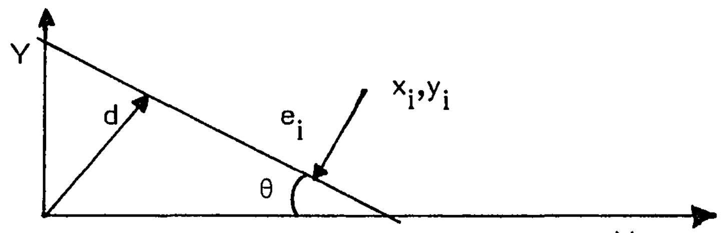
4.1 Error Criterion

Since our technique can be used against both wave forms and two dimensional curves, we must define the error criterion and the computional method for both cases. For wave forms we can define the point wise errors as

$$e_i = | f - F_n |. \qquad 4.1.1$$

The error e_i is computed along the coordinate axis which is perpendicular to the time axis. However, this is not the case for a two dimensional curve because equation 4.1.1 will not give meaningful results with respect to the approximating curve. The Euclidean distance from a point (x_i, y_i) to the approximating curve will be ill-defined because of the increase of the problem dimensional space, i.e. from one to two. Therefore, we will define an expression e_i which gives the normal Euclidean distance between a point, (x_i, y_i), and the approximating curve $F_n(x_i)$. Perkins in [43] shows an equation of the form

$$x_i \cos\theta + y_i \sin\theta - d \qquad 4.1.2$$



Definition of Terms Used in Equation 4.1.2

Figure 4.1.1

for determinimg the error between $f(x_i)$ and $F_n(x_i)$ where d is the distance from the origin to the approximating curve. The first two terms in equation 4.1.2 give the distance from a point to the origin where θ is the angle between the distance vectors and the x axis. The error, e_i, can then be computed by taking the magnitude of equation 4.1.2 which can be written as

$$e_i = | x_i \cos\theta + y_i \sin\theta - d |. \qquad 4.1.3$$

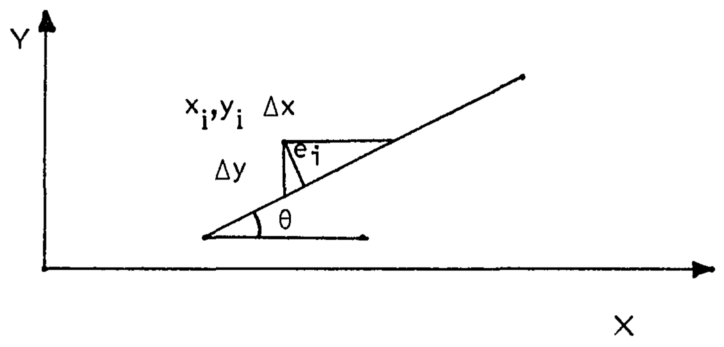
Equation 4.1.3 is one method of solving for equation 4.1.1 if no constraints are imposed upon the input data. In the following sequence we will show another method for solving equation 4.1.1. In this problem we will solve for e_i by using the constraint that all input data be spaced along a coordinate direction at regular intervals. This requirement is easy to achieve because we can control the input data to the feature extractor as shown in Figure 1.2.1. The following conditions, which are shown in Figures 4.1.2 and 4.1.3, must be accounted for:

(a) The data point is located above the approximating curve

(b) The data point is located below the approximating curve

(c) The data point is located such that the x_i or y_i component is zero. This is shown in Figure 4.1.4.

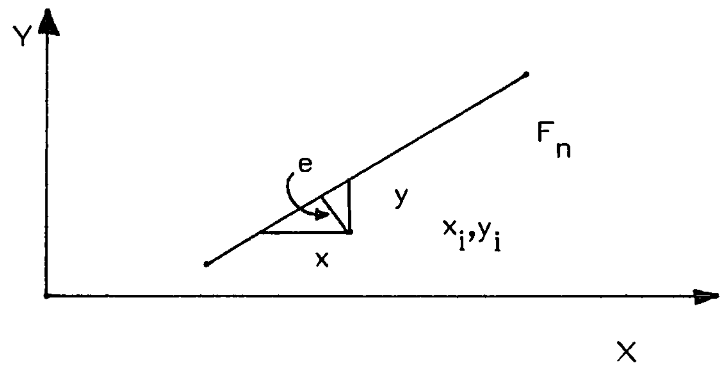
As can be seen from Figures 4.1.2 through 4.1.4 , e_i is given as $e_i = |\Delta x \sin(\tan^{-1} \frac{\Delta y}{\Delta x})|$ if the data point x_i, y_i is located below $F_n(x)$ as shown in Figure 4.1.3 or $e_i = |\Delta y \sin(90 - \tan^{-1} \frac{\Delta y}{\Delta x})$ if the condition shown in Figure 4.1.2 exists. For the condition shown in Figure 4.1.5, if either the Δx or Δy component is zero then e_i equals the non zero component. Using either equation 4.1.3 or 4.1.4 we can determine the error norm over the data set, D_j , of interest.



Data Point Above Approximating Curve

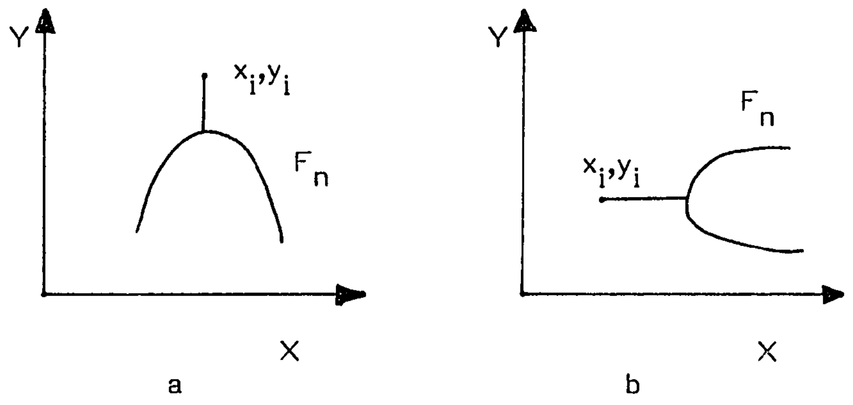
Figure 4.1.2

The optimum piecewise linear polynomial approximation over an interval[a,b] can be determined by one of two methods: 1) find the polynomial of the lowest order n, in this case n equals either 1 or 2, where the error e_i is equal to or less than some given threshold, or 2) for a given order, n, minimize the error.



Data Point Below Approximating Curve

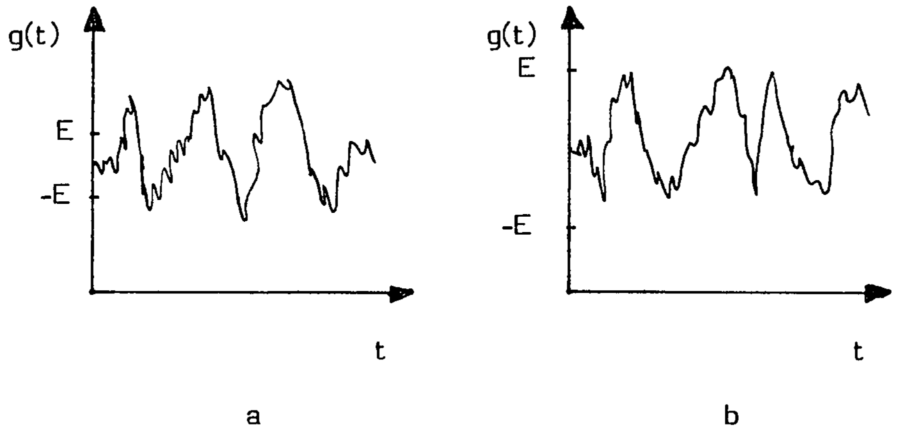
Figure 4.1.3



x_i or y_i Component is Zero

Figure 4.1.4

In this study we will use the first method because computationally it is easier and the accuracy of the describing contour can be varied. The second method requires fitting a polynomial or set of polynomials of a given order n on the interval [a,b] within the specified error bound. Potentially the second method could give a more accurate description of the contour by decomposing the interval [a,b] into smaller subintervals in order to meet the error threshold. Computionally this method is expensive and it is not absolutely clear that that much effort is necessary. The first method works well on an interval [a,b] no matter what the input data characteristics are even if the input data has been corrupted by noise. Consider the situation where the input data has been corrupted by noise as shown in Figure 4.1.5. If the error bound, E, has been set low as in Figure 4.1.5a then the algorithm would report that a corner exists within the data. Conversely, if the error bound is high Figure 4.1.5b the system would report that there were no corners. This situation is a dilemma because the system could miss or could interpret the existence of false corners, caused by noise, as actual object breakpoints and thus give false reports.



Noisy Input Signal

Figure 4.1.5.

One cannot completely avoid such situations, but by experimenting to determine an initial error bound one can minimize false detections. We are concerned with two different types of error bounds within this paper. The first one has to do with error on an interval, or a segment of the interval [a,b], where the error norm is compared against a specified error threshold. This can be expressed as $e_i \leq E_i'$ where E_i' represents the specified interval error . The second error threshold is concerned with error over the continuum of two segments(intervals). That is if I_n and I_{n+1} are contiguous intervals, then the maximum error in, $I_n \cup I_{n+1}$, cannot exceed E_i. E_i is the adaptive upper error bound over n continuous intervals and it adjusts according to the signal's characteristics. Experimentally it was determined that for good contour descriptions the adaptive error threshold can be expressed as:

$$E_i = E_{i-1} + \rho_i(S/N) + e^{\pm\rho_i} \qquad 4.1.4$$

where

E_0	=	initially estimated or experimentally determined threshold
ρ_i	=	fuzzy value determined from the model(will be discussed in Section 4.2)
E_{i-1}	=	error computed over interval I_{i-1} and
S/N	=	signal to noise power ratio.

The last term in expression 4.1.4 was experimentally determined to require the following conditions:

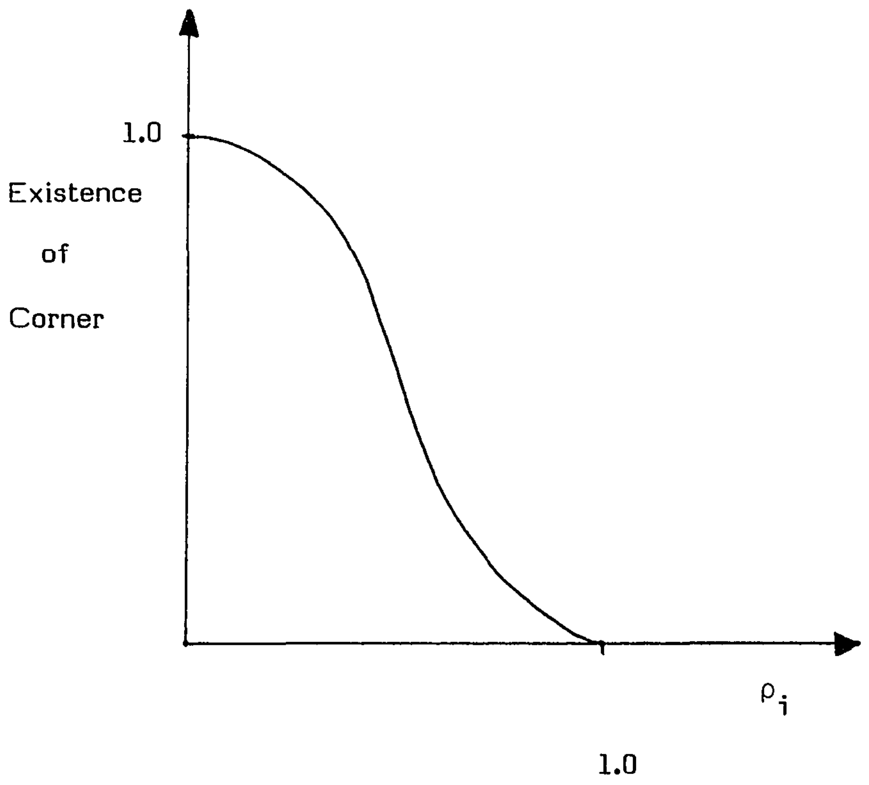
(a) use $+\rho_i$ to determine the upper error bound for merging intervals I_{n-1} and I_n.

(b) use $-\rho_i$ to determine the upper error bound on an interval I_n.

The $+\rho_i$ term, for the upper error bound, has been designed to allow the error criterion to increase or decrease according to the intervals I_{n-1} and I_n fuzzy value. This term also has the effect of smoothing the contour when merging several continuous intervals and, in addition, it controls the magnitude of the upper error bound. The $-\rho_i$ term, for the lower error bound, has been designed in such a way as to limit the amount of error, E_i', per interval I_n . The error limit per interval is directly dependent upon the fuzzy value ρ_i. Thus the ρ_i term will guarantee that the computed error over n continuous segments will not exceed the error bound. This is significant because we can now achieve data compaction by minimizing the number of describing polynomials for the object. This result is also heavily dependent upon our fuzzy definition of a corner which will be presented later.

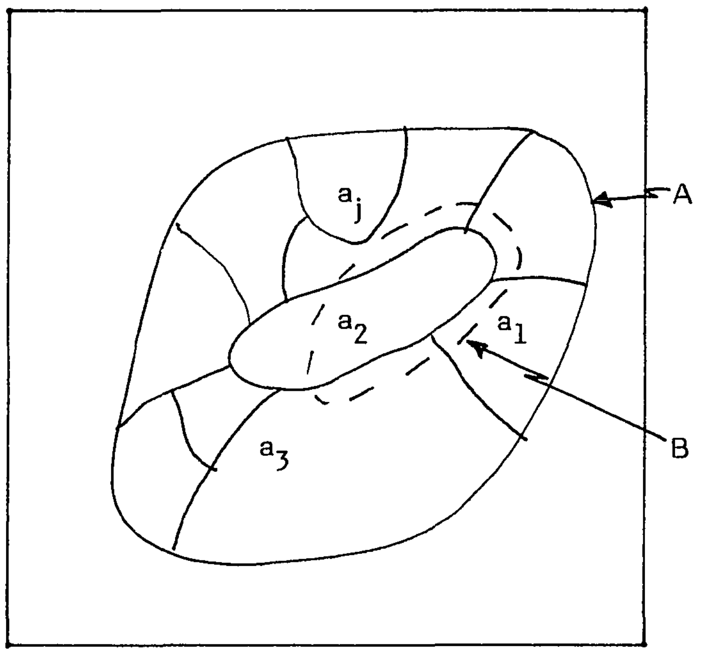
4.2 Fuzzy Criterion

The use of fuzzy sets in this study is a way of evaluating the existence of a corner within an interval I_n or between intervals I_n and I_{n+1} for purposes of merging. As mentioned earlier, an element of a set is assigned a grade of belonging to that set by the fuzzy membership function. The membership function is assessed subjectively and this type of assessment has received little attention in the literature. As Zadeh points out in [48], it is not in keeping with the spirit of the fuzzy set approach to be too concerned about the precision of this number. Also, as mentioned earlier, we are not too concerned with the precise detection of edges or boundaries. For our purposes it was decided that as ρ_i, the membership value, approaches one the presence of a corner is diminished to the point where no corner exists when $\rho_i = 1$. Figure 4.2.1 shows the relationship of existence of corners versus ρ_i. We could have chosen the reverse logic, i.e. when $\rho_i = 0$ no corners exist. The only criterion for selecting any membership function like the one shown in Figure 4.2.1 is that it represent, although not precisely, the desired effect. As can be seen, this has been accomplished by our chosen membership function. What we are interested in is making a decision about whether the approximating polynomial over interval I_n is good given some input data and some other information gathered from the process being studied. The fuzzy Bayes' model as discussed earlier will supply a fuzzy output measure on an interval I_n. In order to discuss this model with respect to the multi-variable problem under investigation, we start by considering Figure 4.2.2. This Figure can be viewed as showing how much B belongs to the various a_j's. A is considered to be the space which can be described by several attributes, a_j's, where some are well defined and others are ill-defined. For the fuzzy Bayes' model, Figure 1.2.3, a_j could represent the a priori subjective input and B would then represent the information about the process, that is $\beta_z(y|x)$ The subjective inputs for our problem are related to three information elements: l_2, l_{max}, and slope. These will be discussed later in the text. Membership values are assigned according to these information elements on each interval I_n. We will assign higher membership values, subjectively, to parameters which provide a significant amount of information about the process being studied. For this study the significant information parameters are l_2 and slope(θ). The l_2 value, when computed over an interval [a,b], can be viewed as



Existence of Corners Versus ρ_i

Figure 4.2.1



Venn Diagram for a Multi-Variable Problem

Figure 4.2.2

providing the average error between the actual and approximating function across the interval. The slope parameter gives information as to the amount of direction change an approximating polynomial makes. For merging intervals a comparison can determine the amount of change between the two intervals and a decision can be made if a corner exists between these two intervals. Based upon the above statements that l_2 and slope provide a significant amount of information about the process, we will subjectively(initially) select high fuzzy density values for these terms. The initially selected values were $\tilde{f}_{x l_2} = \tilde{f}_{x\,\theta} = .5$. Since l_{max} does not provide a great deal of information about the process, we initially selected the a priori fuzzy density value as $\tilde{f}_{x_{max}} = .1$. After a training period as described in what follows which is necessary to determine the best possible set of values, our initial values could change. We will provide a discussion in the sequence below on the effect training has on these information elements. The information term, $\beta_z(y|x)$, will be defined as:

$$\xi_1(\{y_1\}|x_i) = \xi_1(\{\frac{e_i}{l_2}\}|x_i),$$

$$\xi_2(\{y_2\}|x_i) = \xi_2(\{\frac{e_i}{l_{max}}\}|x_i,$$

and

$$\xi_3(\{y_3\}|x_i) = \xi_3(\{\theta_i\}|x_i,$$

where e_i is as defined in section 4.1 , x_i is defined as the input data position within an interval I_n, and $\xi_j(\{y_j\}|x_i)$ is as defined in section 3.3. In a fuzzy multi-parameter problem, humans tend to assimilate information for the determination of a near optimum or optimum solution with little or no effort. Problems of this type, which represent nearly every problem processed by humans, have certain ill-defined parameters which are dependent upon the occurrence of other parameters as described by the fuzzy model given above.

For most pattern recognition systems a period of training is necessary to teach the algorithm on a representative set of data. The algorithm involved with edge and corner detection in this paper can also go through a learning period, although it is not necessary. The learning process of the fuzzy measure term $(\tilde{f}_{x_i})$, which is defined in this problem to be in terms of l_2, l_{max}, and θ , can be accomplished by iterating over a sample data set until the results converge to a constant or near constant value. Here, "near constant" implies that the final values of $\tilde{f}_{x_i}$ could oscillate about a constant value. Thus, an expression is needed which will limit each of the $\tilde{f}_{x_i}$ terms in the model. That is, the $\tilde{f}_{x_i}$ values must be allowed to vary over a bounded range which will yield an acceptable solution. In order to simplify the a priori fuzzy measure, an iterative learning process was used. The following learning rule[52]was applied in order to achieve a good initial a priori fuzzy measure value $\tilde{f}_{x_i}$ for the model . The rule is as follows:

$$\tilde{f}_{x_i} = \Lambda f_{x_{i-1}} (1+\Lambda)\tilde{\xi}_y(A|x_{i-1}) \quad \text{for } i = 1,2,\ldots,l \qquad 4.2.1$$

$$\tilde{f}_{x_i} = \Lambda \tilde{f}_{x_{i-1}} \quad \text{for } i = l+1,l+2,\ldots,n. \qquad 4.2.2$$

where $\Lambda \equiv 0 \leq \Lambda \leq 1$. Λ is a multiplying term which affects the rate of convergence of $\tilde{f}_{x_i}$. To determine the behavior of $\tilde{f}_{x_i}$ we need to give the decision maker information on a representative, training, set of data. We start by presenting to the decision maker the a posteriori fuzzy measure data,$\xi_j(\{y_j\}|x_i)$, as shown in Table 4.2.1 . The fuzzy measure of y can be determined from

$$\tilde{f}_y(E) = \int_x \tilde{\xi}_y(\{y_j\}|x_i) \circ f_{x_i}. \qquad 4.2.3$$

The data given in Table 4.2.1 is an example of a fuzzy measure process where the data is not related to any specific contour detection problem in this paper. This example is intended to show how we can use ill-defined(fuzzy) information to make a decision. In the example, the x's represent a system state. As shown we have three states, and the y's are functions which are evaluated given a system state. Thus, we have three functions which have been evaluated subjectively given the three system states. The subjective evaluation could also have been performed by using functions which have been subjectively selected for the process being evaluated. The concept is to allow decision making, given some ill-defined terms and subjective inputs, in a fuzzy environment.

$\xi_j(\{y_j\}|x_i)$	Y_1	Y_2	Y_3
X_1	.78	1.0	.55
X_2	.60	.84	.55
X_3	.56	.77	.55

A Posteriori Fuzzy Measure Data

Table 4.2.1

We start by solving for $\xi_y(\tilde{A}|x_i)$ and l in equation 4.2.1. In order to find $\xi_y(\tilde{A}|x_i)$ in equation 4.2.1 we must first define a fuzzy expression for the output information. This expression can be written as

$$\tilde{f}_y(A) = \tilde{\int_y} I_n(y) o \tilde{f}_y(E) \qquad 4.2.4$$

where

$I_n(y)$ represents the menbership function of the fuzzy set $\tilde{A}$ which contains the ill-defined elements.

If we substitute equation 4.2.3 into the above expression and solve we get

$$\begin{aligned}\tilde{f}_y(A) &= \tilde{\int_y} I_n(y) o \{\tilde{\int_x} \xi_y(\{y_j\}|x_i) o \tilde{f}_{x_i}\} \\ &= \tilde{\int_x} \xi_y(\tilde{A}|x_i) o \tilde{f}_{x_i}\end{aligned} \qquad 4.2.5$$

where

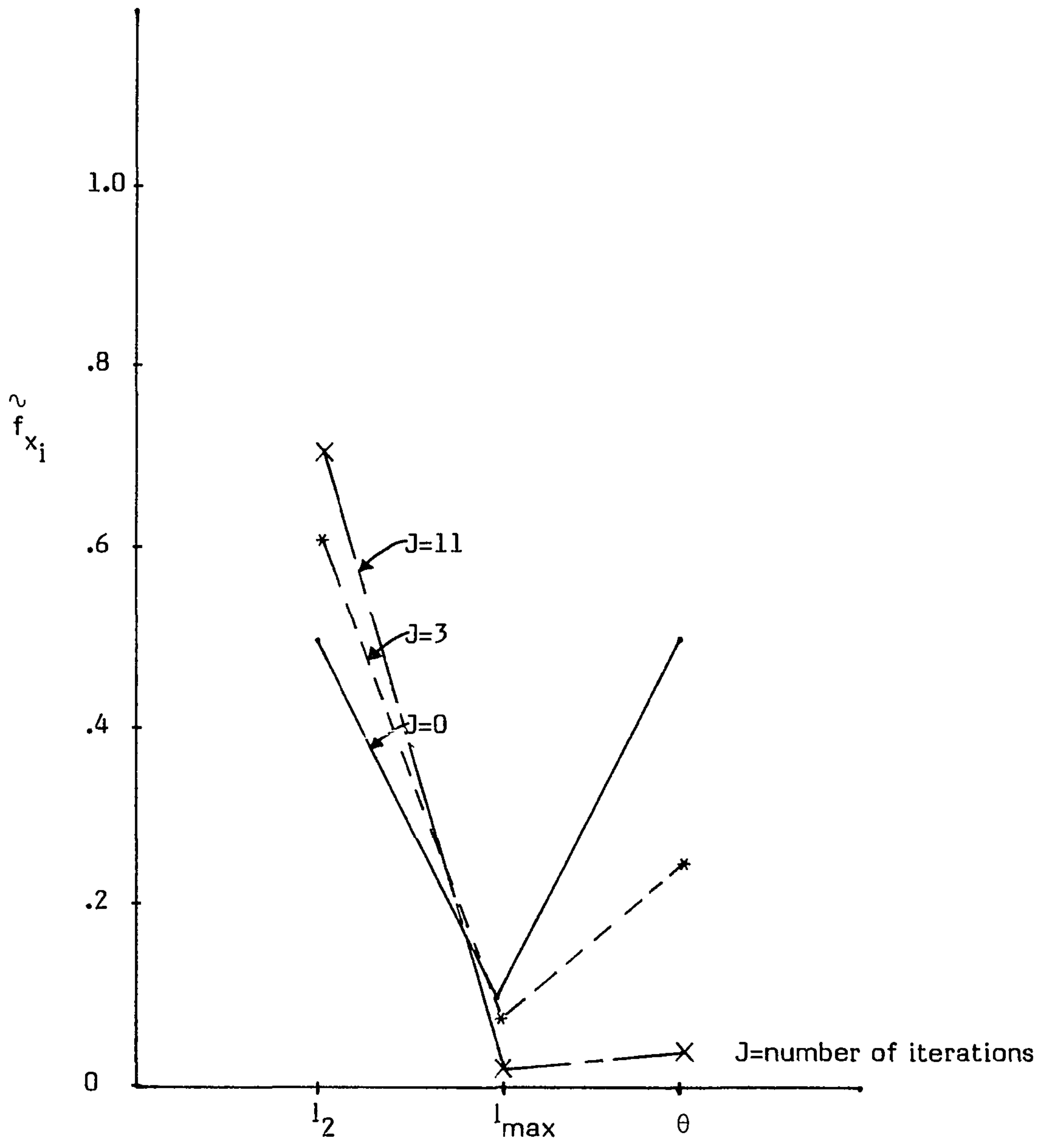
$$\tilde{\xi}_y(A|x_i) \overset{\Delta}{=} \tilde{\int_y} I_n(y) o\ \xi_y(\{y_j\}|x_i). \qquad 4.2.6$$

The grade of fuzziness of the information $\tilde{f}_y(A)$ must be evaluated in order to determine if the a priori fuzzy density values are reasonable. These fuzzy densities are later described in terms of linguistic descriptive values for use by humans. However, through this example we are able to determine the initial numerical fuzzy densities. These initial densities will be correlated with the linguistic descriptive terms to determine an initial starting value. Solving equation 4.2.5 we get

$$\tilde{f}_y(A) = \bigvee_{i=1}^{n} [\xi_y(\tilde{A}|x_i) \wedge \tilde{f}_x(\{x_1, x_2, \ldots, x_i\})]. \qquad 4.2.7$$

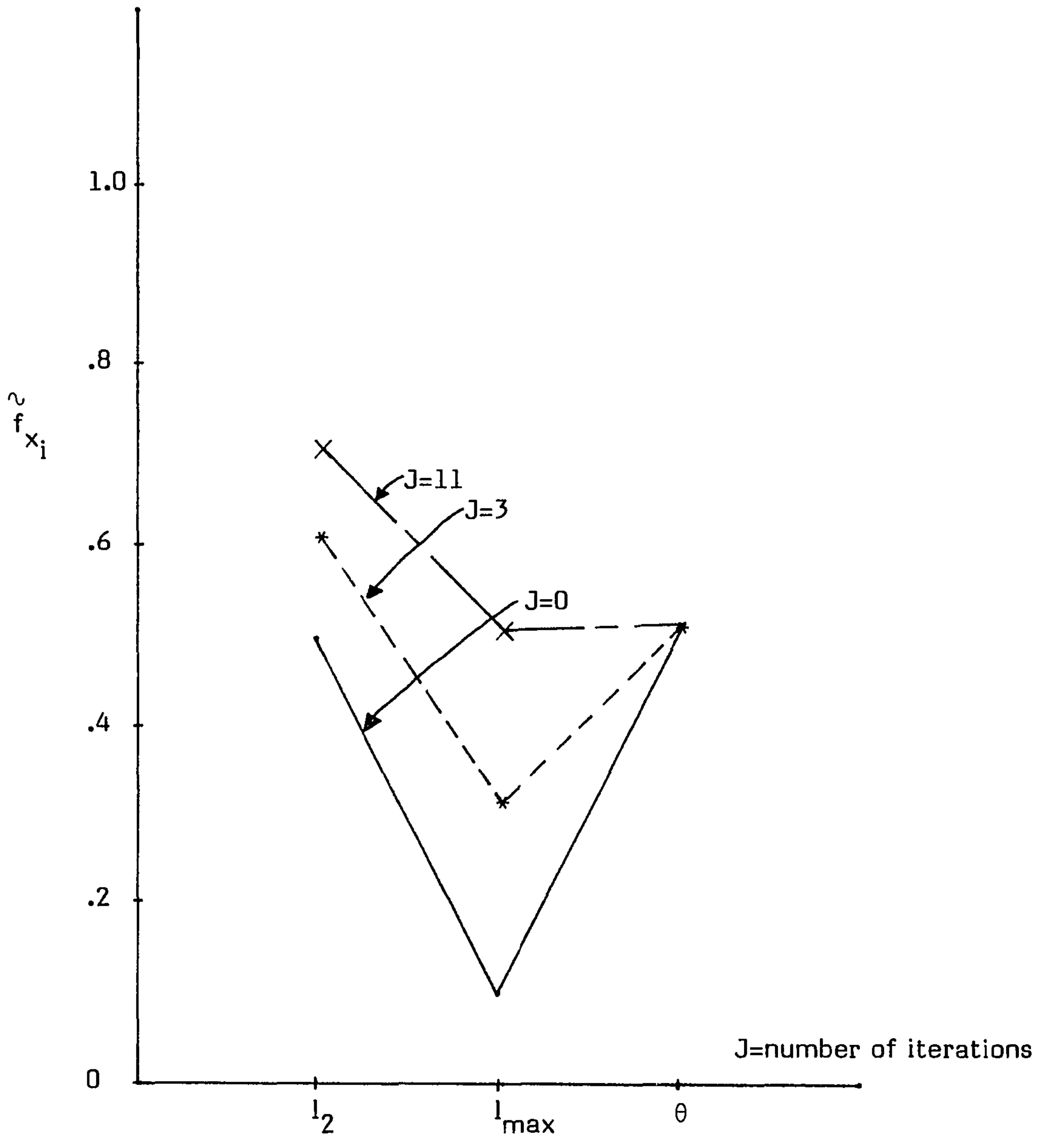
From equation 4.2.7 we can define l as the intersection of $\xi_x(\tilde{A}|x_i)$ and $\tilde{f}_x(\{x_1, x_2, \ldots\ldots, x_i\})$. This definition was discussed in section 3.3 of this paper. The learning rule objective is to decrease or eliminate the a priori fuzzy grade. This is accomplished by iterating with equations 4.2.1 and 4.2.2 to increase the value of

$\tilde{f}_y(A)$. That is,a small a priori value indicates that an event has occurred and in our problem this means that the information indicates the nonexistence of a corner in the interval I_n. Some typical cases of the above procedure are presented in order to show the effect of different types of information on the a posteriori values. For our example we will choose three kinds of information conditions as shown in Table 4.2.2. The data presented in Table 4.2.2 represents the objective information which the decision maker holds a priori on the states W_1 thru W_3. For condition one, data entry W_1, the decision maker indicates that it has occurred and the remaining data has not. Condition two represents data which has occurred in a decreasing order. In condition three, the occurrence of W_1 thru W_3 is ill-defined such that the decision maker has assigned a value of .5 indicating that all events are weighted equally. Results of the learning process on $\tilde{f}_{x_i}$ are discussed and shown below. The decision maker was presented the information as shown in Tables 4.2.1 and 4.2.2 against which the learning rule, equations 4.2.1 and 4.2.2, was iterated until $\tilde{f}_{x_i}$ converged.The reason for this training period was to determine some initial $\tilde{f}_{x_i}$ values given three different types of input information about the process. That is, if the decision maker is given some deterministic information, as in condition one, event one, then we should expect the $\tilde{f}_{x_i}$ value associated with that event to be high as shown in Figure 4.2.3. When the information is fuzzy then as expected the curves will converge to a final value more slowly than for the deterministic case. This can be seen in Figure 4.2.4. The results of iterating the learning rule are shown in Figures 4.2.3 through 4.2.5. When the decision maker is presented information which is fuzzy, it will take longer for the process to converge to a steady state as shown in Figure 4.2.4. This can be observed by comparing the results as shown in Figures 4.2.3 and 4.2.4. In particular, even after 11 iterations, state 1_{max} (shown in Figure 4.2.4) is still converging to its steady state value of .6 as shown in Figure 4.2.3. If the information is so fuzzy that the best we can do is



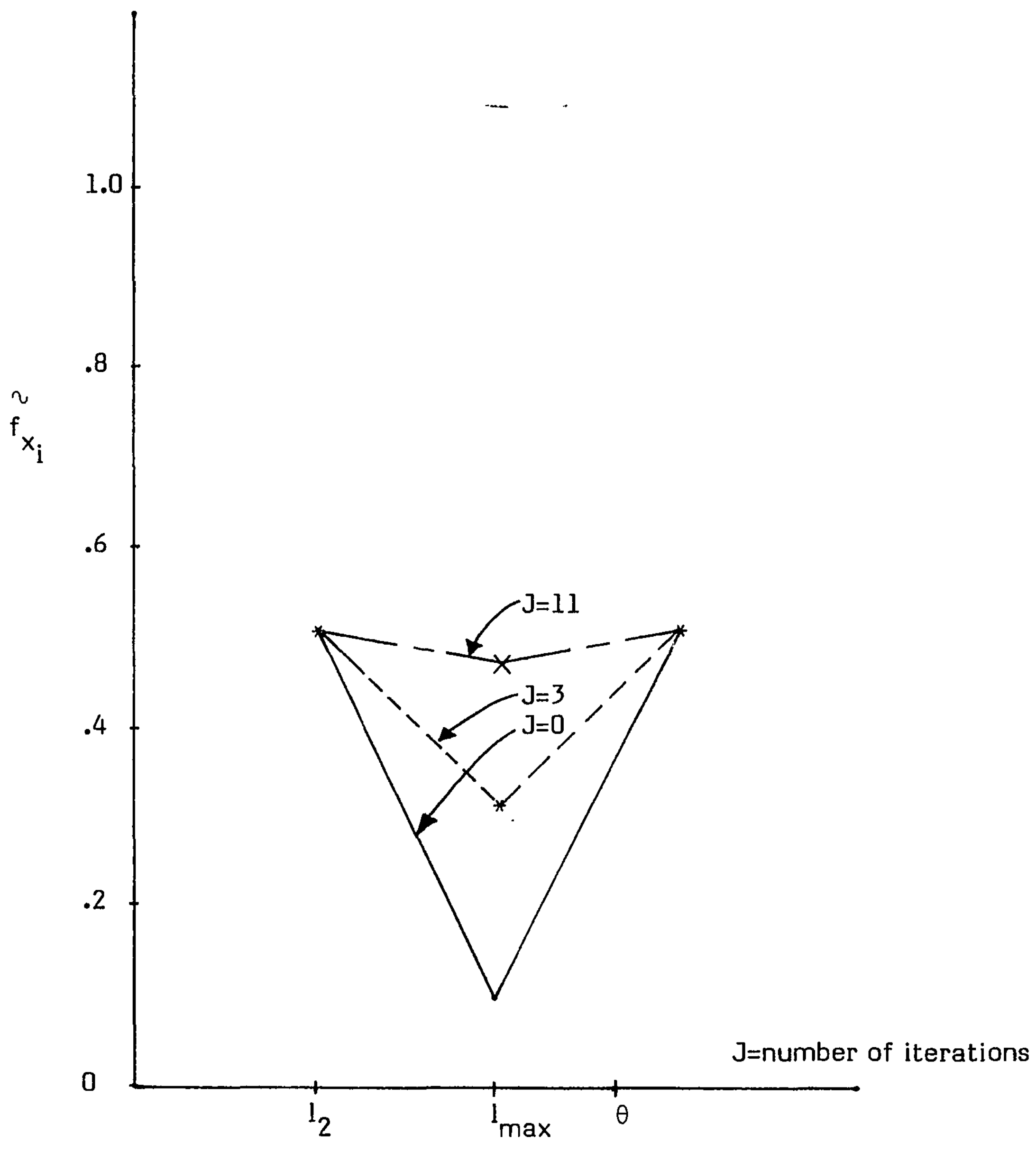
Non Fuzzy Information Given To Decision Maker

Figure 4.2.3



Fuzzy Information Given To Decision Maker

Figure 4.2.4



Constant Fuzzy Information Given To Decision Maker

Figure 4.2.5

Conditions \ $I_n(y)$	W_1	W_2	W_3
1	1	0	0
2	.8	.6	.5
3	.5	.5	.5

Objective Information

Table 4.2.2

to assume a constant, across all y's, the process will coverge to the constant value. This is demonstrated in Figure 4.2.5. As we can see and intuitively expect, if the information is known then a solution should be readily obtained. However, if the information is vague, as shown in Figure 4.2.4, then the convergence time to a solution will be longer. Finally, if the information is very vague, then the results will become vague as shown in Figure 4.2.5. In the above calculations Λ was assumed to be .5 . The effect Λ has on the entire process is on the rate of convergence; that is Λ acts like a damping term. Another point to be made is that when $I_n(y)$ and $\tilde{f}_{x_i}$ values are large the convergence is rapid. This means that the event measured by that parameter has occurred. This is shown by the parameter I_2 in Figure 4.2.3. The above process works well if we have good numerical values to give to the decision maker. However, fuzzy set theory is based upon a subjective input made by humans and humans do not use numbers effectively when describing a process. They tend to use adjectival descriptions which somehow humans seem to understand better than numerical descriptions. Thus, if we use linguistic descriptive values as a means of input, then a fuzzy measure can be easily described in terms which humans can understand. A detailed discussion of descriptive linguistic or syntactical pattern recognition can be found in [19,54]. The linguistic descriptive terms used in this paper are based upon the following criterion. The descriptive terms used must allow for the subjective rating of each term as it relates to the problem under investigation. For example, in our problem this applies to the subjective rating of importance or unimportance of the information terms l_2, l_{max}, and θ. Thus, the linguistic descriptive terms used in this paper relate the importance of the information terms to the detection of edges and corners. The linguistic terms are defined as

(a) l_2 important = l_2i

(b) l_{max} less important = $l_{max}l_i$

and

(c) slope important = si.

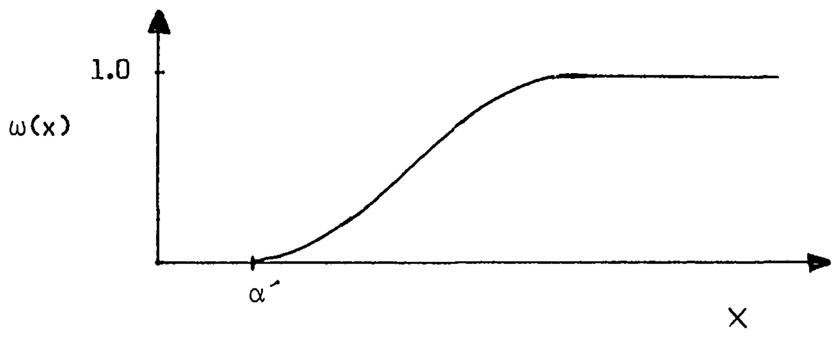
A linguistic descriptive membership function (shown in Figure 4.2.6) was generated in order to define the membership values. The shape of this membership function was chosen such that it would allow for a smooth continuous transition from one membership value to the next and at the same time it would fall off or rise rapidly on either side of $\omega(x) = .5$. Thus, the membership function $\omega(x)$ for each linguistic descriptive term can be determined by using equation 4.2.8 which can be written as

$$\omega(x) = \frac{K(x-\alpha')^2}{1+K(x-\alpha')^2}, \quad \alpha' \leq x \leq \infty \qquad 4.2.8$$

and

$$\omega(x) = 0, \quad 0 \leq x \leq \alpha'.$$

Graphically equation 4.2.8. can be shown as



Membership Function of Equation 4.2.8

Figure 4.2.6

For our problem we will define "important" as having a membership value of .5 and "less important" as having a membership value of .1. These definitions will then define k and α' in equation 4.2.8 as .04938 and .5 respectively. Based upon Zadeh's [59] work we will define the following linguistic variables for our problem as

(a) much/very $X \equiv (X)^2$ 4.2.9

(b) medium $X \equiv (X)^{1.5}$ 4.2.10

(c) not $X \equiv \neg(X) = (1-X)$ 4.2.11

where $\neg \equiv$ not.

These terms will allow us to subjectively change, during the training period, the a priori fuzzy density term $\tilde{f}_{x_i}$. For example, applying the term "very important" to equation 4.2.8 we obtain

$$\text{very important} = \frac{K((x-\alpha')^2)^2}{1+K((x-\alpha')^2)^2},$$

and applying the term "not very important" we have

$$\text{not very important} = \neg(\text{very important}).$$

With the above, a human can subjectively vary the results according to some criterion which is germane to the particular problem under investigation.

Bayes fuzzy model as used in this problem will generate three outputs per interval, I_n (l_2, l_{max}, θ), from which we must determine the best possible solution. It has been experimentally determined that the fuzzy disjunctive sum operation will yield a term which best describes the interval's characteristics. The disjunctive sum can be written as follows:

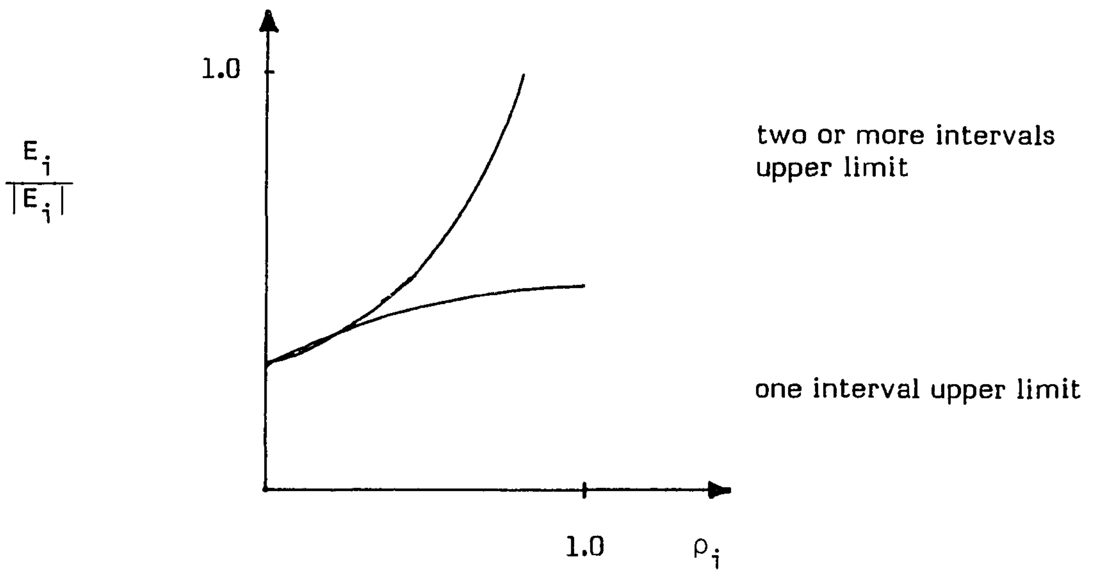
$$\tilde{C}_a = (l_2 \wedge \underline{l}_{max}) \vee (\underline{l}_2 \wedge l_{max}) \qquad 4.2.12$$

$$\tilde{C}_n = (\tilde{C}_a \wedge \underline{\theta}) \vee (\underline{\tilde{C}}_a \wedge \theta) \qquad 4.2.13$$

where $\underline{l}_2$, $\underline{l}_{max}$, and $\underline{\theta}$ are the complements of their respective parameters. By using equation 4.2.13 we find a term which represents the characteristics of interval I_n. This value $\tilde{C}_n$, along with the value $\tilde{C}_{n-1}$ from interval I_{n-1}, will be used in equation 4.2.14

$$D = (\tilde{C}_n \wedge \underline{\tilde{C}}_{n-1}) \vee (\underline{\tilde{C}}_n \wedge \tilde{C}_{n-1}) \qquad 4.2.14$$

to determine if a corner exists between these two intervals. The determination is made by comparing D against an experimentally derived threshold value E_i to see if the criterion has been satisfied. The results of the threshold value have been summarized in Figure 4.2.7 for one and two or more intervals respectively.



Piecewise Decision Space

Figure 4.2.7

4.3 Matrix Criterion

As discussed in Chapter II, we are interested in solving the system equation $\psi C = b$, equation 2.1.1, as efficiently as possible. "Efficiently" as used in this study has two meanings; they are

(1) efficient computationally

(2) and efficient solutions, i.e. optimum or near optimum.

The above two meanings are areas which have received a great deal of attention [16, 19, 39, 54]by investigators in the field of pattern recognition. In Chapter II we discussed ways of reducing a system matrix to a diagonally dominant matrix. These methods work reasonably well on one set of data. However, given a data space as defined for this problem, there exist two methods for enhancing the solution(s) to equation 2.1.1. First, for a solution on an interval I_n, we know that all input data will be spaced one unit apart in the dominant coordinate direction. Knowing this fact we can quickly reduce the C matrix, in equation 2.1.1, to a diagonally dominant matrix as discussed in Chapter II. Another method of accomplishing this is to reduce the C matrix by using a multiplying factor after transforming the data into the -1 to 1 space. A discussion on the determination of the multiplying factor δ will be presented in Chapter V. This multiplying factor, once determined, will allow us to compute C^+ without actually performing any of the required matrix operations. For example, given five consecutive input data points in one coordinate direction and after transformation, the linear C matrix becomes

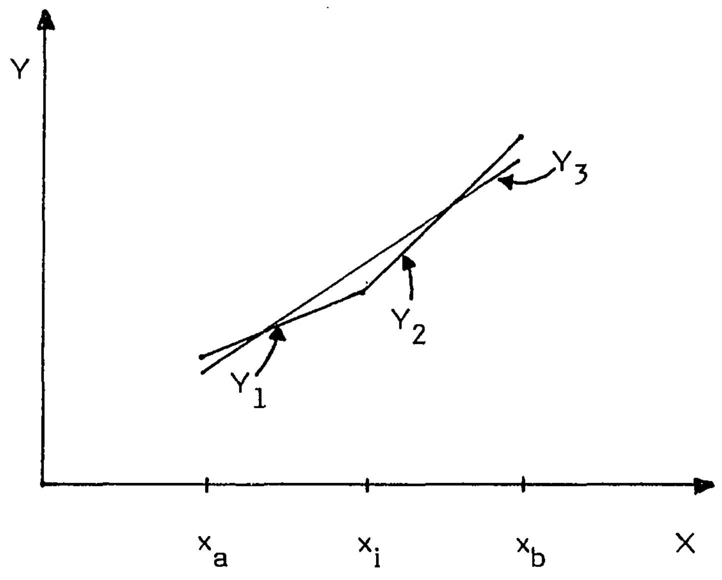
$$C = \begin{bmatrix} 5 & 0 \\ 0 & 2.5 \end{bmatrix} \qquad 4.3.1$$

Multiplying C above by δ , the multiplying factor, gives C^+ as

$$C^+ = \begin{bmatrix} .2 & 0 \\ 0 & .4 \end{bmatrix} . \qquad 4.3.2$$

We can verify the results of equation 4.3.2 by using the techniques as discussed earlier. If the above technique works well on one interval of data, can we find a method for applying it to n consecutive intervals as shown in Figure 4.3.1? The answer is yes, as discussed below.

We are now interested in forming the sum of two generalized inverse matrices in order to shorten the computional method of finding the generalized inverse over two consecutive intervals. That is to find



Optimum Curve Y_3 was Found, Given Y_1 and Y_2

Figure 4.3.1

$$C_3^+ = (C_1 + C_2)^+. \qquad 4.3.3$$

Before we show the proof of equation 4.3.3 for the generalized inverse case, we will start by first considering only square invertible matrices. We know from matrix theory how to find the inverse of a matrix and, if given two nonsingular matrices, we can find the inverse of C_1C_2 as $C_2^{-1}C_1^{-1}$. The problem we are interested in is to find C_3^{-1}, where $C_3^{-1} = (C_1 + C_2)^{-1}$, in terms of C_1^{-1} because we have already computed C_1^{-1}.

We start by defining C_3^{-1} as

$$C_3^{-1} = (C_1 + C_2)^{-1} = C_1^{-1} - \frac{1}{1+a} C_1^{-1} C_2 C_1^{-1} \qquad 4.3.4$$

where $a = \operatorname{tr} C_2C_1^{-1}$ and we are assumming that the inverse exists. Equation 4.3.4 is known as the Sherman-Morrison Equation [12]. If we consider a square matrix H of rank one, then all eigenvalues, except possibly one, are zero. We know from matrix theory that the sum of the eigenvalues is the trace. Thus for such a matrix, H, the eigenvalue is tr H. A matrix of rank one can be formed from two nonzero column vectors b and d of the same dimension. The matrix $H - db^T$ has rank one because b and d were given as having rank one. From [23] a matrix H of rank one is composed of nonzero vectors b and d such that $H = db^T$. Now if we are given any square matrix C and a square matrix H, as defined above, then we can define HCH as

$$HCH = db^T c db^T = \Omega\, db^T = \Omega\, H \qquad 4.3.5$$

where

= bilinear form $b^T cd$.

Thus, we can write the matrix $C_1 + H$ as

$$C_1 + H = (I + HC_1^{-1})C_1 \qquad 4.3.8$$

and, since C_1 is nonsingular, the matrix HC_1^{-1} will have a rank of one. One eigenvalue of $I + HC_1^{-1}$ is $1 + \operatorname{tr} HC_1^{-1}$ and the remaining eigenvalues are one. The matrix $C_1 + H$ is nonsingular if and only if $\operatorname{tr} HC^{-1} \neq -1$.

In order to write the right hand side of equation 4.3.3 it is convenient to consider a function f(x). Let f(x) be differentiable on an interval $a \leq X \leq b$. Using the law of the mean 29 , we can define an f(x) as

$$f(x) = f(a) + \dot{f}(\Omega b)x \qquad 4.3.9$$

where

$$0 < \Omega < 1.$$

If we now let $f(x) = (x + b)^{-1}$, then we can write

$$(x + b)^{-1} = b^{-1} - (1 + \Omega)^{-2} b^{-1} x b^{-1} \qquad 4.3.10$$

where a = 0.

Equation 4.3.10 can be rewritten as

$$(x + b)^{-1} = b^{-1} - \tau b^{-1} x b^{-1} \qquad 4.3.11$$

where

$$\tau = (1 + \Omega)^{-2}.$$

Now if we identify b with A_1 and x with H we get

$$(H + C_1)^{-1} = C_1^{-1} - \tau C_1^{-1} H C_1^{-1}. \qquad 4.3.12$$

If equation 4.3.12 is the inverse of $H + C_1$, then their product must be the identity matrix I. To prove this, start by forming the product as

$$(H + C_1)(C_1^{-1} - \tau C_1^{-1} H C_1^{-1}) = I - \tau H C_1^{-1} + H C_1^{-1}$$
$$- \tau H C_1^{-1} H C_1^{-1}. \qquad 4.3.13$$

Solving the right hand side of equation 4.3.13 for gives

$$\tau H C_1^{-1} - H C_1^{-1} + \tau (H C_1^{-1})^2 = 0 \qquad 4.3.14$$

From equation 4.3.7 we find $(HC^{-1})^2 = \Omega HC$, and by substituting this into equation 4.3.14 we get

$$(\tau + l + \tau\Omega)\ HC^{-1} = 0 \qquad 4.3.15$$

Solving equation 4.3.15 for τ we find

$$\tau - 1 + \tau\Omega = 0,$$

or $$\tau = \frac{1}{1-\Omega},$$

or

$$\tau = \frac{1}{1+\mathrm{tr}HC} \qquad 4.3.16$$

We have now shown that given two matrices, one square and the other of rank one, the inverse of there sum can be found by equation 4.3.4. What we will show next is that, given two square nonsingular matrices of rank r, equation 4.3.4 will still hold. To start, assume that matrix C_2 in equation 4.3.4 is of rank r, which can be written as

$$C_2 = H_1 + H_2 + \dots\dots + H_r \qquad 4.3.17$$

where each H_i, $1 \leq i \leq H_r$, has rank one [23]. Now equation 4.3.4 can be solved recursively but before we can do that we must show that C_2, which now is written as equation 4.3.17, is nonsingular. As given above, the eigenvalue of $HC^{-1} \neq 1$ allows $C_1 + C_2 = (I + C_2C_1^{-1})\, C_1$ to be nonsingular. If we let Z be a nonsingular matrix where $J = Z^{-1}(C_2C_1^{-1})Z$, this is the Jordan form of $C_2C_1^{-1}$; and if $\bar{J} = \bar{J}_1 + \bar{J}_2 + \dots\dots + \bar{J}_r$ where the kth row of $\bar{J}_k$ is the kth row of $\bar{J}$, then the remaining rows of $\bar{J}_k$ are zero. This gives the matrix J_j a rank of one which then gives $I + \bar{J}_1 + \bar{J}_2 + \dots\dots + \bar{J}_k$ as a nonsingular matrix for $k = 1,2,\dots\dots,r$. Now we can write

$$\begin{aligned}(I + Z^{-1}\bar{J}Z)C_1 &= (I + \bar{J}_1 + \bar{J}_2 + \dots\dots + \bar{J}_k)Z)C_1 \\ &= C_1 + Z^{-1}\bar{J}ZC_1 + \dots\dots + Z^{-1}\bar{J}_kC_1\end{aligned}$$

which is nonsingular for $k=1,\dots\dots,r$. Since $Z^{-1}\bar{J}_kZC_1$ has a rank of one, we can let $C_2 = Z^{-1}\bar{J}_kZC_1$. The above can now be stated as

"Given two square matrices, C_1 and C_2, we can form a matrix C_3^{-1} as

$$C_3^{-1} = (C_1 + C_2)^{-1} = C_1^{-1} - \frac{1}{1+a}\, C_1^{-1}C_2C_1^{-1}$$

where

$$a = \frac{1}{1+\mathrm{tr}C_2C_1^{-1}}\,.$$

That is, the inverse of the sum of two nonsingular matrices can be found by a straight forward solution as given above. Now, returning to the generalized inverse case, the proof of equation 4.3.4 depends upon the existence of the product of

$$(AB)^+ = B^+A^+ \qquad 4.3.18$$

because according to Greville [7] this situation does not always exist. Greville shows in [7] a proof which says that the conditions $R(A^*AB) \subset R(B)$ and $R(BB^*A^*) \subset R(A^*)$ must exist in order for equation 4.3.18 to hold where * denotes the conjugate transpose of A. R(B) and $R(A^*)$ denote the range of B and A^*. Also, the following equations must be satisfied before equation 4.3.18 is true:

1) $A^+AB(AB)^* = B(AB)^*$

2) $BB^+A^*AB = A^*AB$

Or, stated another way, the rank of B^+A^+ must equal the rank of AB which means that $B^+A^+ \varepsilon AB$. That is if rank x = rank A, then R(xA) = R(x) because these two subspaces are identical if, and only if, they have the same dimensions. We know that the rank of any matrix is the dimension of its range [23]. Therefore we can write $R(xA) \subset R(x)$. Also, nullity of any matrix is the number of columns minus the rank [23]. So N(A) = N(xA) if, and only if, A and Ax have the same null space. They do because they have the same rank, i.e. the same number of columns. By applying Penrose equations [7] it can be demonstrated that equation 4.3.18 does exist. Thus, if these conditions are satisfied, we can then write equation 4.3.4 as

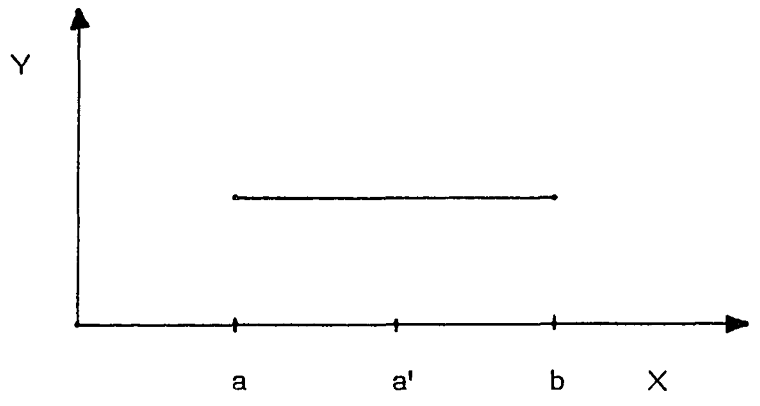
$$C_3^+ = (C_1 + C_2)^+ = C_1^+ - \frac{1}{1+a} C_1^+ C_2 C_1^+ . \qquad 4.3.19$$

CHAPTER V

ALGORITHM

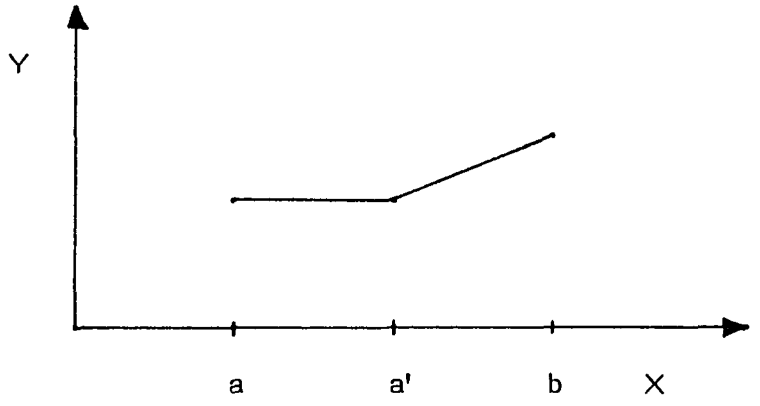
5.0 Introduction

Up to this point, we have discussed the theory behind approximating contours by using linear and/or quadratic polynomials. Also, as mentioned earlier, we are concerned with the closeness of fit of polynomials at corners (breakpoints) within a given interval or between intervals. In this chapter we will present an algorithm which uses the fuzzy Bayes model to ascertain if a corner exists within an interval I_n. Results of the algorithm are presented and are compared to results from other algorithms of the same class on the same set of data. Before starting our discussion, we will establish the conditions/constraints of the input data and the constraints of the approximating polynomials at breakpoints. Before the algorithm will process the data; all data presented to the algorithm must be equally spaced in a given coordinate direction throughout the entire data space. For example, if the input data is a time varying signal, then the time coordinate data must be equally spaced. If the input data is a 2-dimensional figure, then either the x or y-coordinate data must have uniform spacing. The second constraint on the algorithm is that the ending point for interval I_n is the starting point for interval I_{n+1}; that is $I_{nj} \cap I_{(n+1)j} \neq 0$. This constraint will give an approximation which is continuous at either a breakpoint or at the end of an interval. This condition is demonstrated in Figure 5.0.1. This last constraint is significant because now a contour or boundary of an object is closed. Pavlidis[39] uses a symmetrical constraint which does not give a continuous representation of a contour; instead gaps can exist within the object's describing contour. For example, in Figure 5.0.2 we see a symmetrical condition and note the gap in the contour. The constraints which Pavlidis uses are



Continuous Condition in an Interval

(a)

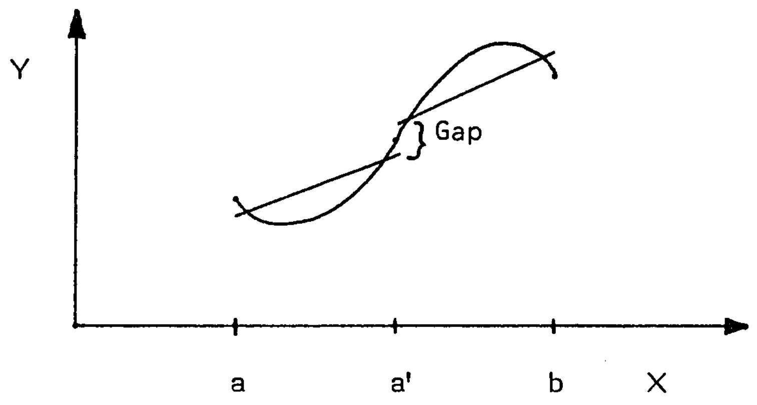


Continuous Condition at a Breakpoint

(b)

Figure 5.0.1

valid for linear polynomial approximation, but the overall boundary description is not very good because of these gaps. In our investigation we are concerned with good boundary descriptions but we must bear in mind the computational time penatly one pays if good contour description is desired. A discussion on computational times will be given in Chapter VI. In addition to the above



Symmertical Constraint Can Give a Gap

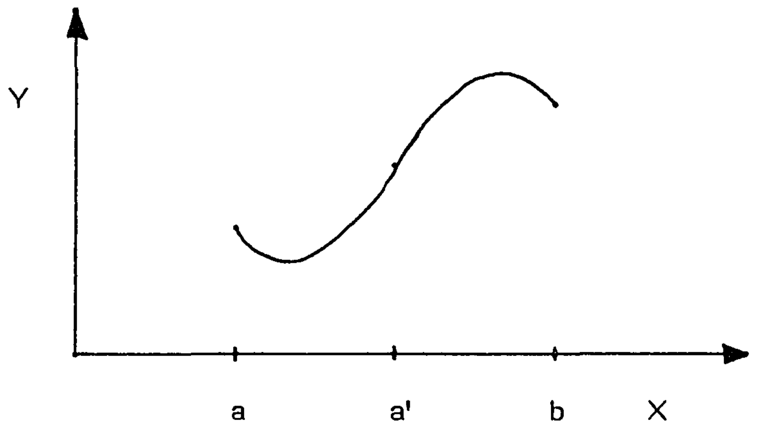
Figure 5.0.2

conditions, the algorithm being presented will also use a quadratic polynomial to aid in contour description. For example, in Figure 5.0.2 the algorithm would use two quadratics to describe the given curve as shown in Figure 5.0.3. The difference between our algorithm, which uses linear and quadratic approximating polynomials, and algorithms which use linear approximating polynomials is that we could have more describing polynomials per object. However, this is not necessarily true for all cases because quadratic polynomials will give a certain amount of data compaction on describing an object's boundary.

5.1 Matrix Operations

In Chapter IV, a method was presented for computing the sum of two generalized inverse matrices. It was noted that one of the major drawbacks with the computation of equation 4.3.19 was the time required to obtain a final solution and, as discussed in Chapter I, this is one of the major problem areas of pattern recognition. In order to reduce the above matrix computional time we will now show a unique scalar multiplying coefficient which will yield the left side of equation 4.3.19. If we rewrite equation 4.3.19 as

$$A^{+}_{i+1} = A_i^{+} - \Gamma A_i^{+} B_i A_i^{+}, \qquad 5.1.1$$



Quadratic Fit of the Curve Given in Figure 5.0.2

Figure 5.0.3

where B_i represents A_2 and $\Gamma_i = \frac{1}{1+a_i}$, then a result can be stated such that the left side of equation 5.1.1 can be found by using a scalar multiplying term. That is A^+_{i+1} can now be found by the expression

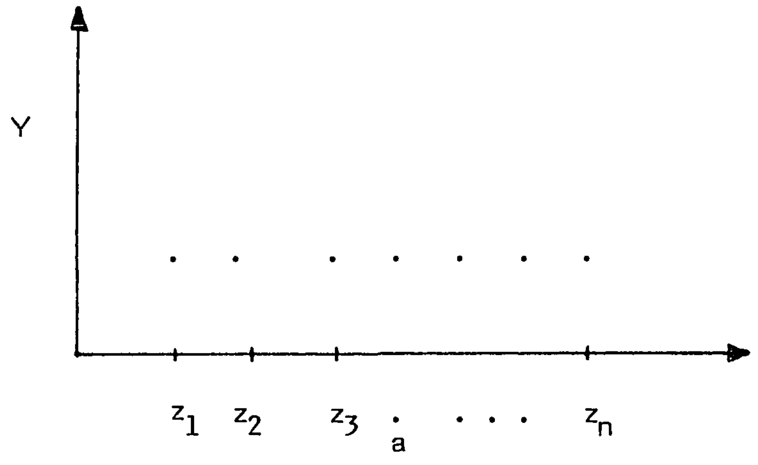
$$A^+_{i+1} = \delta_i (A^+_{i-1} + A_i^+) \qquad 5.1.2$$

where $A^+_{-1} = 0$ for i = 0.

Theorem 5.1: Given a set, $\underline{\overline{X}}^+$, of ordered data points which are equally spaced (in distance between each point) along an axis, as shown in Figure 5.1.1a, and if $\underline{\overline{X}}^+$ is partitioned into ordered subsets $x_i < \underline{\overline{X}}^+$,as shown in Figure 5.1.1b, then a set of scalar multiplying δ_i values can be determined as

$$\delta_i = A^+_{i+1}A_{i-1}(I + A_i^+A_{i-1}). \qquad 5.1.3$$

Such that $A^+_{i+1} = \delta_i (A_{i-1}^+ + A_i^+)$ where $\underline{\overline{X}}^+$ represents the set of all natural numbers. Now we can determine the generalized inverse matrix for $x_i \cup x_j$ by equation 5.1.1. Computing the generalized inverse matrix A^+_{i+1} by using the Gaussian elimination method on intervals $x_j \cup x_i \cup x_{i+1}$, where interval $x_j \cup x_i$ has already been found, is not a very efficient process. That is, the matrix



$X_1 = \{z_1, z_2, z_3\}$, $X_2 = \{z_3, z_4, z_5\}$, ...
$X_n = \{z_{n-2}, z_{n-1}, z_n\}$

b

Ordered Data and Sets

Figure 5.1.1

computational time involved when using the Gaussian elimination method is too large when compared to scalar multiplication. The scalar multiplying term δ_i , as given in equation 5.1.3, can be determined a priori and stored within the algorithm. The right side of equation 5.1.1 can be computed without having to perform any matrix operations other than the operations indicated by equation 5.1.2. Writing the new and short form of equation 5.1.1 we have

$$A^{+}_{i+1}=\delta_i(A^{+}_{i-1}+A^{+}_{i}). \qquad 5.1.4$$

Appendix D contains a detailed example showing the results of equations 5.1.1 and 5.1.4. By preprocessing we can determine as many δ_i terms as desired. These values can then be stored in a look up table within the contour's algorithm. When the contour algorithm determines that two intervals can be combined,(this is one form of data compaction) it will do so by choosing the appropriate δ_i value and it will then perform the calculations as indicated by equation 5.1.4. It must be noted that we refer to the combining of two intervals in which the first interval, represented by A^+_{i-1} in equation 5.1.4, could be composed of several combined intervals. That is $a_i \subseteq A^+_{i-1}$ where $a_i(i=1,.....,n)$ represents intervals which are subsets of A^+_{i-1}. Table 5.1.1 contains the set of δ_i values used in this study for contour approximating polynomials after the input data has been transformed onto the -1 to 1 interval. The table contains two sets of values; one for the linear case and one for the quadratic case. Since all input interval data has been normalized between -1 and 1, the resulting matrices are diagonally dominant as discussed earlier. It should be pointed out that these δ_i values will work for any object/signal as long as the conditions as stated in this paper are satisfied. The values contained in Table 5.1.1, which have been plotted on a semi-logarithmic scale in Figure 5.1.2, show a tendency to approach zero as the number of merged intervals approaches a large number, that is, in the limit as $n \to \infty$, $\delta_i \to 0$. But large values of n, for instance those greater than 10, are not practical because real world data is not well behaved. However, if the data is well behaved, then the number of points per interval should be increased to keep the number of δ_i terms small. This increase in points per interval will not change the δ_i values as long as the number of points per interval remains constant. Appendix D shows the calculations of some of the values which are contained in Table 5.1.1.

5.2 Fuzzy Decision Values

Earlier we discussed using a fuzzy Bayes' model as an aid in determining if we have encountered a corner/breakpoint within an interval. We also discussed the subjective importance of certain parameters whose values contain significant information on the data contained within the interval I_n. We defined these information parameters as l_2, l_{max}, and slope (θ). We will now present the effect

Matrix element

Position multiplying

coefficients

Intervals Merged	a_{11}	a_{22}
1	.2777	.3333
2	.1282	.1648
3	.0735	.0980
4	.0476	.0649
5	.0333	.0461
6	.0246	.0344

Linear Case

(a)

Matrix element

Position multiplying

coefficients

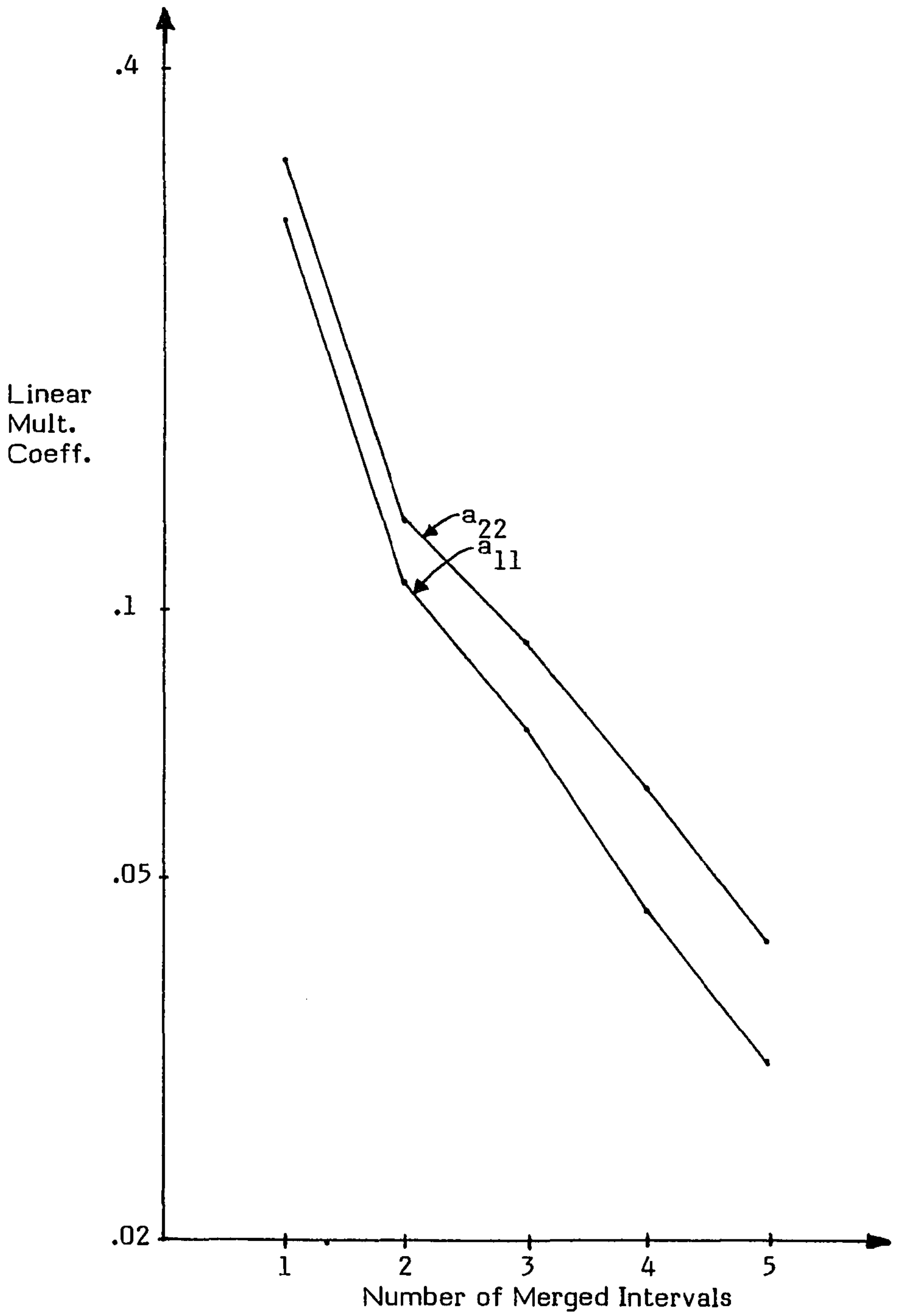
Intervals merged	a_{11}	$a_{13}=a_{31}$	a_{22}	a_{33}
1	.2629	.3030	.3333	.3636
2	.1199	.1468	.1648	.1888
3	.0685	.0866	.0980	.1155
4	.0442	.0572	.0649	.0780
5	.0309	.0405	.0461	.0561
6	.0228	.0302	.0344	.0423

Quadratic Case

(b)

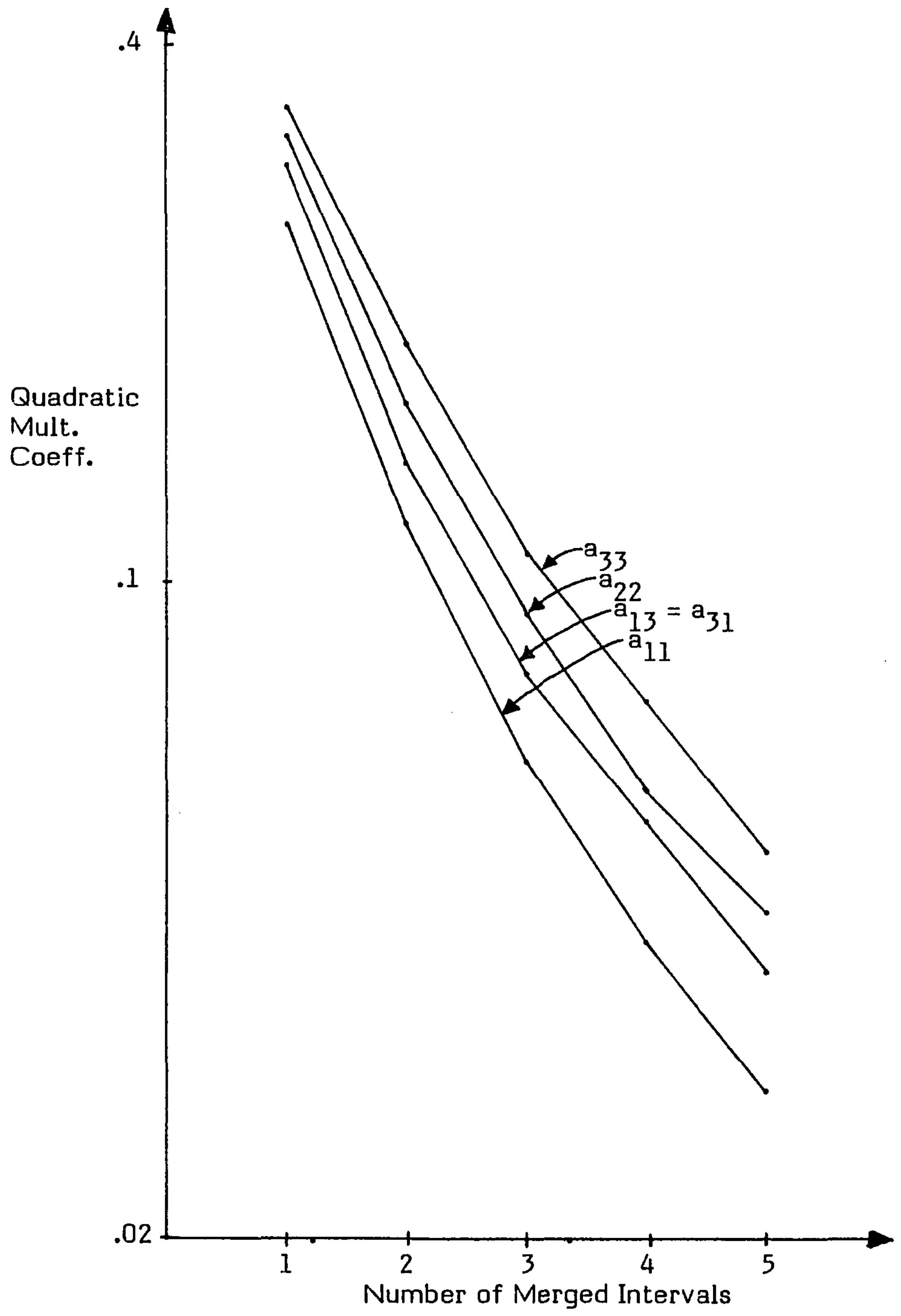
Multiplying Coefficients

Table 5.1.1



Multiplying Coefficients

(a)



Multiplying Coefficients

(b)

Figure 5.1.2

of various linguistic descriptive values have on the algorithm's ability to detect a corner. The test data presented to the algorithm consisted of two intervals, each containing five data points. The first interval was a straight line and, in all cases considered below, the S/N = 0. The second interval contained data which started as a straight line, $l_2 = l_{max} = \theta = 0$, and was varied so that a breakpoint was observed at $I_{nr} \cap I_{(n+1)r}$ where r represents the data point location within an interval. The results of the second interval for these test cases are shown in Table 5.2.1 where we can observe the effect the linguistic descriptive values have on the outcome. The linguistic descriptive terms used in this study are shown below for both the linear and quadratic approximations. They are defined as:

LI = large important

LS = large small

LVI = large very important

LMI = large medium important

LMS = large medium small

LNMI = large not medium important.

Further, the numerical values for the above linguistic terms are defined on the [0,1] interval with the subintervals defined as ranging over the following:

Small = 0 to .3

Medium = .4 to .6

Large = .7 to 1.0.

The breakpoints introduced between intervals I_1 and I_2 in the above Table have angular differences of 0^o,30^o, and 45^o for test cases 1,2, and 3 respectively. That is, the slope for interval I_2 was varied in order to cause a breakpoint to appear at the intersection of the intervals I_1 and I_2. The results shown for test case number one in Tables 5.2.1 and 5.2.2 are also the same as the results for interval I_1. Because of the non-existence of a breakpoint at $I_1 \cap I_2$. For the remaining test

Test Cases	1	2	3
l_2	0	.0153	0
l_{max}	0	.0219	0
θ	0	.3346	.5
ρ_i	0	.900	.5
E_T	0	.984	0*

Linguistic Descriptive Values:

$l_2 = \theta$ = LI and

l_{max} = LS

(a)

Test Cases	1	2	3
l_2	0	.015	0
l_{max}	0	.021	0
θ	0	.334	.5
ρ_i	0	.665	.499
E_T	0	.984	0*

Linguistic Descriptive Values:

$l_2 = \theta$ = LVI and

l_{max} = LNVS

(b)

Test Cases	1	2	3
l_2	0	.015	0
l_{max}	0	.021	0
θ	0	.334	.5
ρ_i	0	.857	.499
E_T	0	.984	0*

Linguistic Descriptive Values:

$l_2 = \theta$ = LMI and

l_{max} = LMS

(c)

Test Cases	1	2	3
l_2	0	.015	0
l_{max}	0	.021	0
θ	0	.334	.5
ρ_i	0	.90	.818
E_T	0	.98	0

Linguistic Descriptive Values:

$l_2 = \theta$ = LNMI and

l_{max} = LS

(d)

*Did not merge intervals I_1 and I_2. Thus, E_T represents the error over the last interval, i.e. I_2.

Linear Approximation Case

Table 5.2.1

cases the results of l_2, l_{max}, θ ,and ρ_i shown in the tables are for interval I_2. The E_T term represents the computed error for the merged intervals I_1 and I_2 except where noted. The key term in these tables is ρ_i for the following two reasons: first, we compute $|\rho_i-\rho_{i-1}|$ and if this difference is less than the largest linguistic value then a merging is possible; and second, the difference is substituted into equation 4.1.4 so that an upper error bound can be established. Since the S/N = 0 the initial error term, E_0, in equation 4.1.4 was set equal to zero. Note that if a merge fails then the resulting E_T is for the last interval, which in this example is interval I_2. A further interpretation of the results as presented in Table 5.2.1 is as follows. If for example a corner/breakpoint of significance is defined to be greater than 50 degrees, when measured from the horizontal, then all of the test cases as given in Table 5.2.1 with there associated linguistic descriptive values are acceptable. That is, given the linguistic terms for the various test conditions, the resulting ρ_i values are less than the corner/breakpoint threshold values derived from the linguistic descriptive equation 4.2.8. Or stated differently, corners which are less than 50 degrees are considered to be insignificant(not important). The corners which are less than 50 degrees do not contain any significant amount of information about the object's contour and therefore the object's contour over that interval will be approximated by a single polynomial. However, if a corner is defined to be 40^o, or greater, then the linguistic terms given in Table 5.2.1 a, b, and c will indicate the absence of a corner. That is,for the linguistic values as given in Table 5.2.1 the resulting ρ_i values must be greater than .6 for merging to occur. Observe that this is true for the conditions given in Table 5.2.1 a,b,c, but not d because of the linguistic values used and thus the resulting ρ_i failed the test i.e. $\rho_i \leq \omega$ the significant linguistic descriptive value. These results occur because

Test Cases	1	2	3
l_2	0	.68	.88
l_{max}	0	.62	.69
θ	8	9.9	6.5
ρ_i	0	.09	.09
E_T	0	1.62	1.16

Linguistic Descriptive values:

$l_2 = \theta =$ LI and

$l_{max} =$ LS

(a)

Test Cases	1	2	3
l_2	0	.68	.89
l_{max}	0	.62	.69
θ	8	7.7	6.5
ρ_i	0	-.8	-.8
E_T	0	.68*	.88*

Linguistice Descriptive values:

$l_2 = \theta =$ LVI and

$l_{max} =$ LNVS

(b)

Test Cases	1	2	3
l_2	0	.68	.89
l_{max}	0	.62	.69
θ	8	7.7	6.5
ρ_i	0	.85	.14
E_T	0	1.61	1.16

Linguistic Descriptive values:

$l_2 = \theta =$ LMS and

$l_{max} =$ LMS

(c)

Test Cases	1	2	3
l_2	0	.68	.89
l_{max}	0	.62	.69
θ	8	7.7	6.5
ρ_i	0	.9	.09
E_T	0	1.6	1.16

Linguistic Descriptive values:

$l_2 = \theta =$ LNMI and

$l_{max} =$ LS

(d)

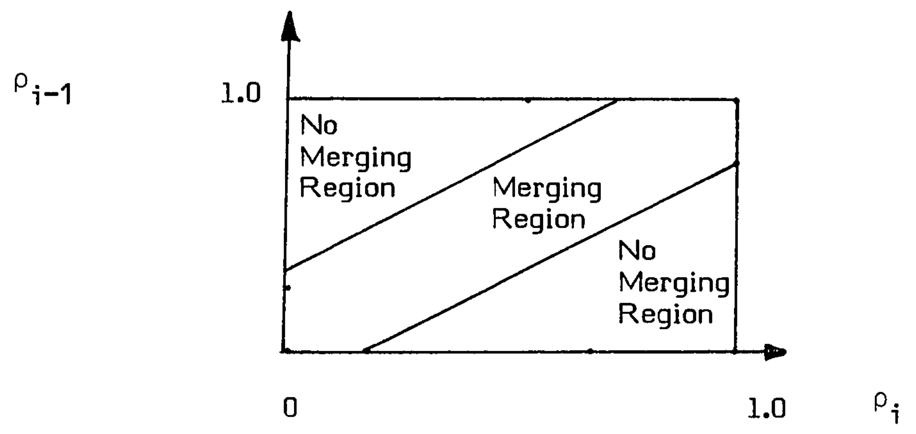
*Did not merge intervals I_1 and I_2. Thus, E_T represents the error over the last interval,i.e. I_2.

Quadratic Approximation Case

Table 5.2.2

the corners/breakpoints are not properly detected by the I_{max} term as discussed in Chapter IV. Table 5.2.1d shows the effect of a poor selection of linguistic descriptive terms for corner/breakpoint detection. For the quadratic case we find that the linguistic terms as given above give good results as can be seen in Table 5.2.2. The quadratic test case was composed of two intervals in which the first interval was held constant and the second was varied in order to cause a breakpoint at $I_{nr} \cap I_{(n+1)r}$. Also, observe that in general the linguistic descriptive terms for the quadratic case are lower than for the linear case. This fact is shown in Table 5.2.2 b test case 3 where the algorithm did not merge the intervals because of a stringent corner requirement. The lower linguistic descriptive values for the quadratic approximation can be attributed to the fact that there are more describing terms for quadratic polynomials than for linear polynomials.

From the above discussion we can draw the following conclusion about the merging of intervals for data compaction. If good edge and contour detection is desired then higher linguistic values are required for the linear case than for the quadratic case where the values can be a little lower. The converse of the last statement is also true. For data compaction, the difference between ρ_{i-1} and ρ_i must be less than or equal to the ω values as determined by equation 4.2.8. The conditions as to when to merge and when not to merge intervals I_n and I_{n+1} are represented in Figure 5.2.1. As shown in Figure 5.2.1, if we change the linguistic value then we get a corresponding shift, as expected, in the allowed merging region. The allowable merging region is determined by forming the difference between ρ_{i-1} and ρ_i and comparing this value aganist the ω value as computed in equation 4.2.8. If the difference is less than or equal to ω the algorithm will then



Merging Regions

Figure 5.2.1

merge intervals I_n and I_{n-1}. Figure 5.2.1 applies for both the linear and quadratic cases.

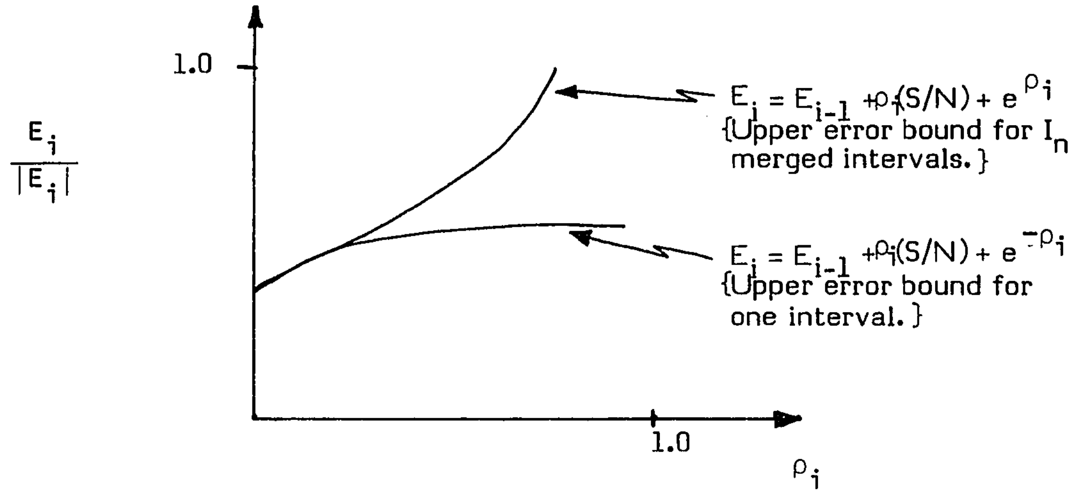
5.3 Error Threshold

The error criterion as presented in Chapter IV, equation 4.1.7, is dependent upon several values. The most critical one is the initial threshold term because two of the terms in equation 4.1.4 are dependent upon that value. The penalty for an initial value which is too large is poor edge detection and, conversely, if the initial threshold is too small, the algorithm will take a considerable amount of time to process the data . This is normally undesirable. Therefore, we must be careful in our selection of an initial error threshold value . The threshold constraint across intervals I_n and I_{n+1} for polynomial approximation can be stated in the following definition.

Definition 5.3.1:

Let I_n and I_{n+1} be two contiguous intervals such that the maximum error of approximation on each interval does not exceed E_i. Then the maximum error on $I_n \cup I_{n+1}$ is bounded from above by

$$E_i = E_{i-1} + \rho_i (S/N) + e^{\pm\rho_i} .$$



Error Threshold for One and n Intervals

Figure 5.3.1

$$E_i = E_{i-1} + \rho_i(S/N) + e^{\pm\rho_i} . \qquad 5.3.1$$

Equation 5.3.1, which is the same as equation 4.1.4, defines the maximum allowable upper error bound which permits the merging of intervals I_n and I_{n+1}. Figure 5.3.1 shows the upper error bound for one interval and for I_n merged intervals. The second term in equation 5.3.1 will move the curves either up or down but it will not change the shape of these curves.

5.4 Algorithm

The algorithm developed for this study was motivated by Albano [4] and Pavlidis [39] where speed of execution and minimum amount of required storage are of importance. We will defer the discussion on speed of execution until Chapter VI. In the following sequence we will provide a discussion on the algorithm's processing scheme and will show some results against two different types of input patterns. Pavlidis[39] discusses the advantages and disadvantages of splitting and merging algorithms used for contour approximations. If the split-and-merge techniques are implemented separately then the developed algorithms will not produce the best contour descriptions, nor can data compaction be achieved.

However, if these two techniques are combined [4,39] , then it is possible to achieve good contour descriptions and data compaction. The algorithm considered in this study is of the split-and-merge type with some significant differences over past algorithms of the same class. The significant differences are:

(1) two small input buffers instead of one large buffer,

(2) addition of fuzzy corner descriptions, and

(3) shorter matrix computations then in the previous algorithms.

The algorithm was designed to have small input buffers and thus they do not permit the storage of all the input data. In a real time signal processing situation, the a priori storage requirements are unknown; thus we cannot design a system to store all the input data. The algorithm was designed to process the input signal's data in real time with a minimum amount of storage.

The following is a discussion of how the algorithm works.

<u>Algorithm:</u> Fuzzy split-and-merge approximation algorithm.

Input : Boundary or contour points X_i and Y_i, $i=1,2,\ldots,\psi$

Output: Segmented piecewise polynomial approximation of the given object.

Step 1: Fill a buffer with n_p input data points and start processing points where $\eta \leq n_p$. First check to see if $e \leq E_i$ for the linear case. If it isn't, then try a quadratic fit and, if this check fails, then split η in half. Repeat the linear and quadratic approximations until the test $e \leq E_i$ does not fail. The algorithm will sense the η input data points to determine if they are perpendicular to the primary coordinate axis.The x-axis was chosen as the primary coordinate for this study. If such a determination is made, then a 90^0 data rotation is performed on the η points before processing continues.

Step 2: Process the next η points according to Step 1.

Step 3: Compute and compare fuzzy value ρ over intervals I_n and I_{n+1} and if $\rho \leq \omega$ then merge these intervals.

Step 4: Compute the optimum piecewise polynomial approximation for merging intervals I_n and I_{n+1}.

Step 5: Return to Step 1 if $n_p < \psi$. Otherwise return to Step 2 and continue processing. If $n_p = \psi$, terminate the processing.

The following is a detailed discussion on each of the above steps of the algorithm.

Remarks: In Step 1 the buffer size, n_p, was set to 5 and the interval, , to 5 although they could have been set to any value as long as $\eta \leq n_p$. If interval I_n fails the error test and must be split,then the next interval, I_{n+1} , will start with the last point of the completed interval I_n. After the processing of Step 1 is completed, the algorithm then proceeds to Step 3. In Step 3 the fuzzy decision values for interval I_n are computed and compared against the values from interval I_{n-1}. If the computed fuzzy values from intervals I_n and I_{n-1} pass the decision threshold test, the processing continues to Step 4. Otherwise it returns to Step 2.

In Step 4, the computation for the linear or quadratic optimum approximation across intervals I_n and I_{n-1} is accomplished. Note that interval I_{n-1} is the running polynomial approximation over n contiguous intervals which have been successfully merged.

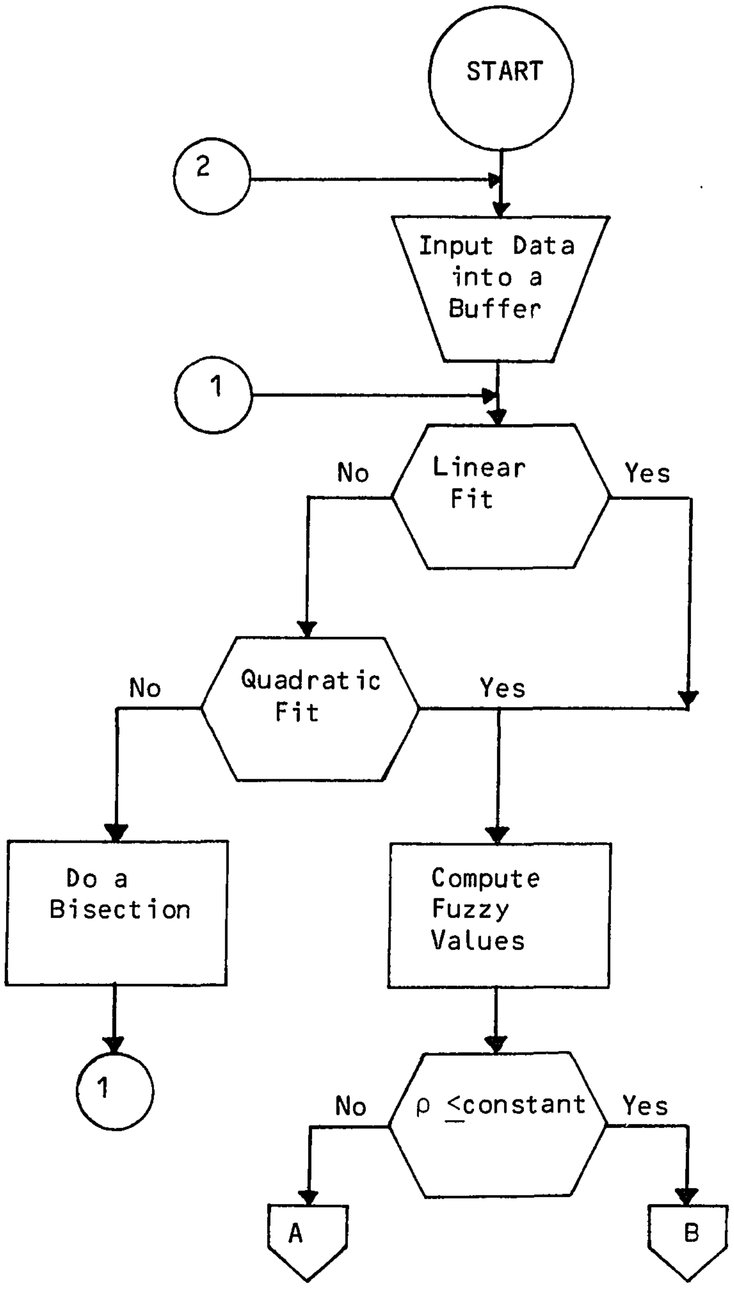
Step 5 returns to Step 1 or Step 2 to continue processing, or it terminates the algorithm if $n_p = \psi$. The above steps are graphically represented in Figure 5.4.1 and, as can be seen, the algorithm is quite simple and will not iterate on the data space like other algorithms of the same class.

We should emphasize that the above algorithm has two objectives. First it must find a local approximating polynomial on a small subset of the input space which satisfies the error threshold as discussed earlier. Second it must determine if a global approximating polynomial(s), which passes the error threshold for multiple intervals, is possible.

Theorem 5.4: The algorithm, as discussed, will terminate after local and global error thresholds are reached and all the input data has been processed.

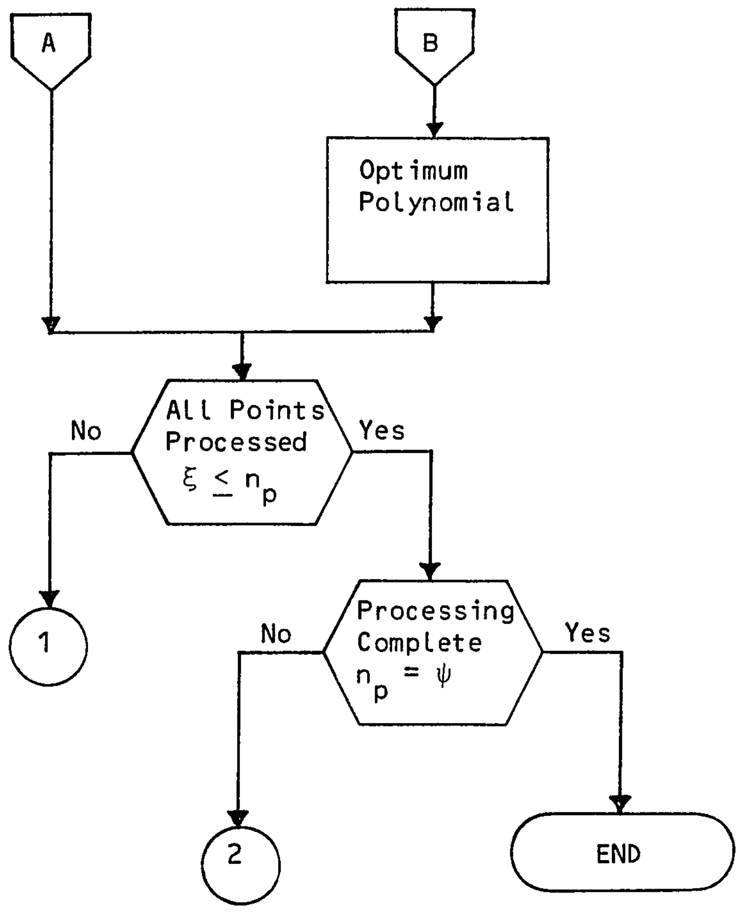
Plausibility argument for Theorem 5.4: The error threshold criterion will always be met because, after a sufficient number of split operations, the computed error will become less than the threshold. Observe that the algorithm can fit either a linear or quadratic polynomial through m points and the smallest number of points where a good fit can be achieved is three because we can always fit a quadratic through three points. Once the algorithm has satisfied the error threshold inequality, it cannot reverse itself. Thus, we see that the algorithm as given above will terminate. In addition to the above, the algorithm will continuously process the data, in real time, by using small aggregates of the input data. This is because of the input buffer size. The algorithm will find the optimum solution(s) without the requirement of storing all the data .This is because of its "local fitting" feature.

The algorithm has been tested on several different types of one and two-dimensional data sets with considerable success. This particular algorithm was implemented in Fortran V on an Interdata 7/32 digital computer. Data used to test the algorithm was obtained from [39] as shown in Figures 5.4.2, 5.4.3 and some synthetically generated objects as shown in Appendix C. Figure 5.4.4 shows the results of the algorithm's ability to process an object, in this case a cell, as shown in Figure 5.4.2. It can be observed that the algorithm will give an excellent piecewise polynomial description of the object's contour. Further, observe that when the data becomes perpendicular to the primary axis the algorithm will detect this and process accordingly. Figure 5.4.5, which is an EKG signal as shown in



Overall Algorithm

Figure 5.4.1


A
B
Optimum Polynomial
All Points Processed $\xi \leq n_p$
No
Yes
1
Processing Complete $n_p = \psi$
No
Yes
2
END

Overall Algorithm(Cont.)

Figure 5.4.1

Input Point Number	Describing Polynomial
1-5	$Y = 4.0$
5-17	$Y = -.2315X + 5.414$
17-21	$X = .1764Y + 17.411$
21-25	$Y = 5.999$
25-29	$Y = .8X - 6.2$
29-33	$Y = .699X - 4.4$
33-37	$Y = 1.5X - 24.5$
37-41	$X = .199Y + 24.8$
41-45	$X = .199Y + 25.0$
45-50	$Y = -.478X + 41.77$
50-55	$X = .299Y + 17.89$
55-59	$Y = 1.346X - 10.76$
55-63	$Y = .5X + 7.6$
63-67	$Y = 17.0$
67-71	$X = -.1764Y + 16.411$
71-75	$Y = .4411X + 14.73$
75-79	$Y = 1.038X + 4.923$
79-83	$X = -.199Y + 24.199$
83-91	$Y = -.333X + 36.06$
91-95	$Y = .5999X + 26.2$
95-99	$Y = 1.56X + 20.81$
96-100	$Y = 2.0X + 19.00$
100-105	$X = .199Y - 2.199$
105-109	$Y = -1.0X + 22.0$
109-113	$X = -.199Y + 9.599$
113-117	$Y = 1.176X + 3.176$
117-121	$Y = .911X + 4.794$
118-122	$Y = 1.562X + 3.50$

Cell Describing Polynomials

(a)

Input Point Number	Describing Polynomial
1-5	$Y = .25X - 2.39$
5-9	$Y = .599X - 4.59$
9-13	$Y = .899X - 19.79$
13-17	$Y = .783X + 17.30$
17-29	$Y = .03X - 8.18$
29-33	$Y = .30X^2 - 34.97X + 1002.02$
33-35	$Y = 6.75X - 425.16$
35-37	$Y = -1.0X^2 + 145.0X - 5202.0$
37-39	$Y = -.99X^2 + 143.99X - 5129.89$
39-41	$Y = -.5X^2 + 65X - 2014$
41-43	$Y = 1.99X^2 - 331.99X + 13745.92$
43-45	$Y = 1.99X - 197.99$
50-55	$Y = .29X - 39.19$
55-57	$Y = .25X - 33.833$
57-59	$Y = -.49X^2 + 114.99X - 6613.90$
59-63	$Y = -.05X + 3.8$
63-67	$Y = .11X^2 - 27.22X + 1726.57$
67-75	$Y = .22X - 31.03$
75-79	$Y = .16X^2 - 48.50X + 3660.85$
79-83	$Y = .29X - 41.8$
83-87	$Y = .09X^2 - 29.54X + 2452.46$
87-100	$Y = .04X + 7.0$
101-113	$Y = -.59X + 129.10$
113-117	$Y = .1X - 26.8$
117-121	$Y = -.319$

EKG Describing Polynomials

(b)

Describing Polynomials

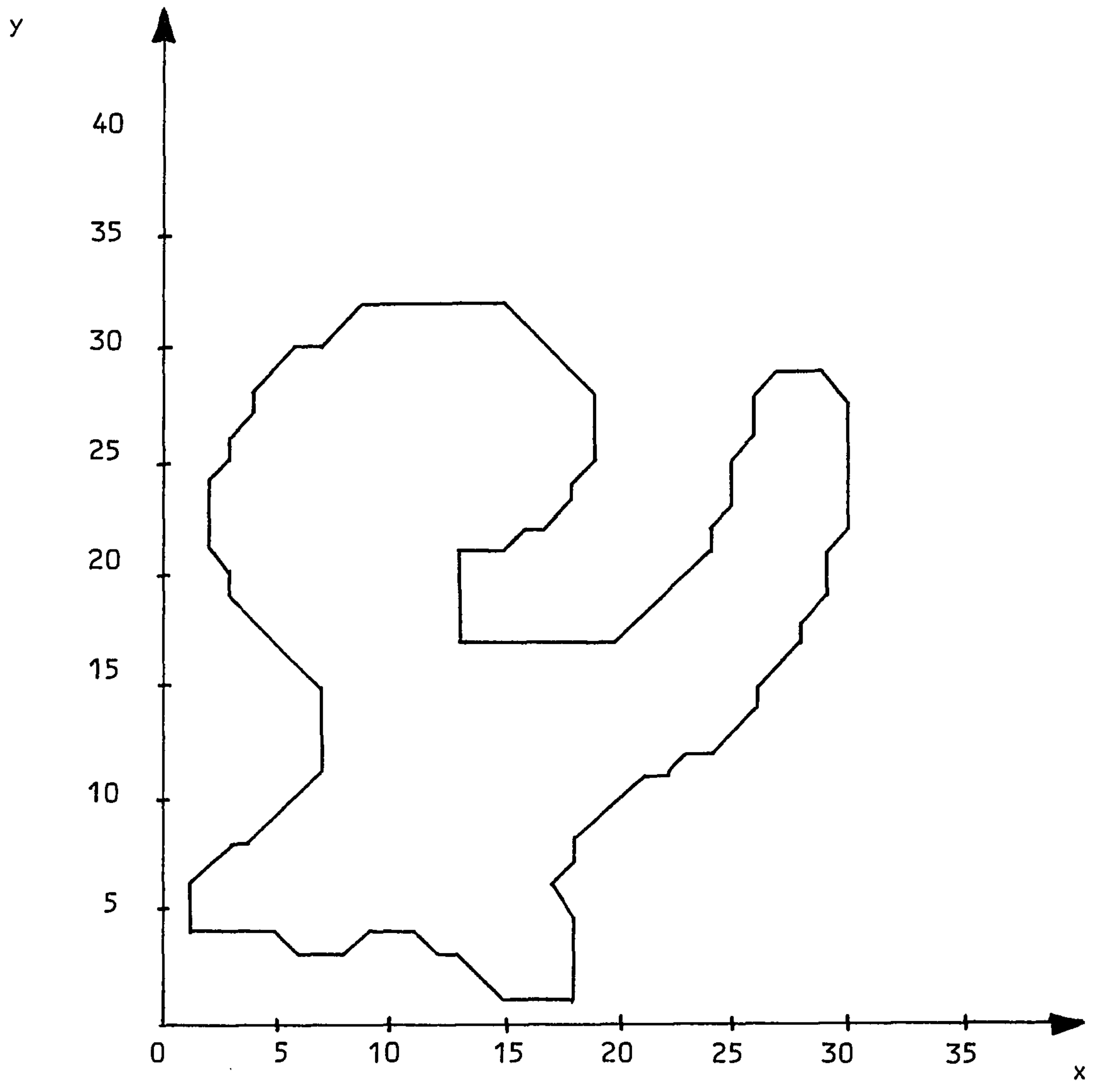
Table 5.4.1

Figure 5.4.3, demonstrates the algorithm's ability to use both linear and quadratic approximations on a given data set. Table 5.4.1 gives the set of linear piecewise describing polynomials for Figures 5.4.4 and 5.4.5.

5.5 Other Algorithms

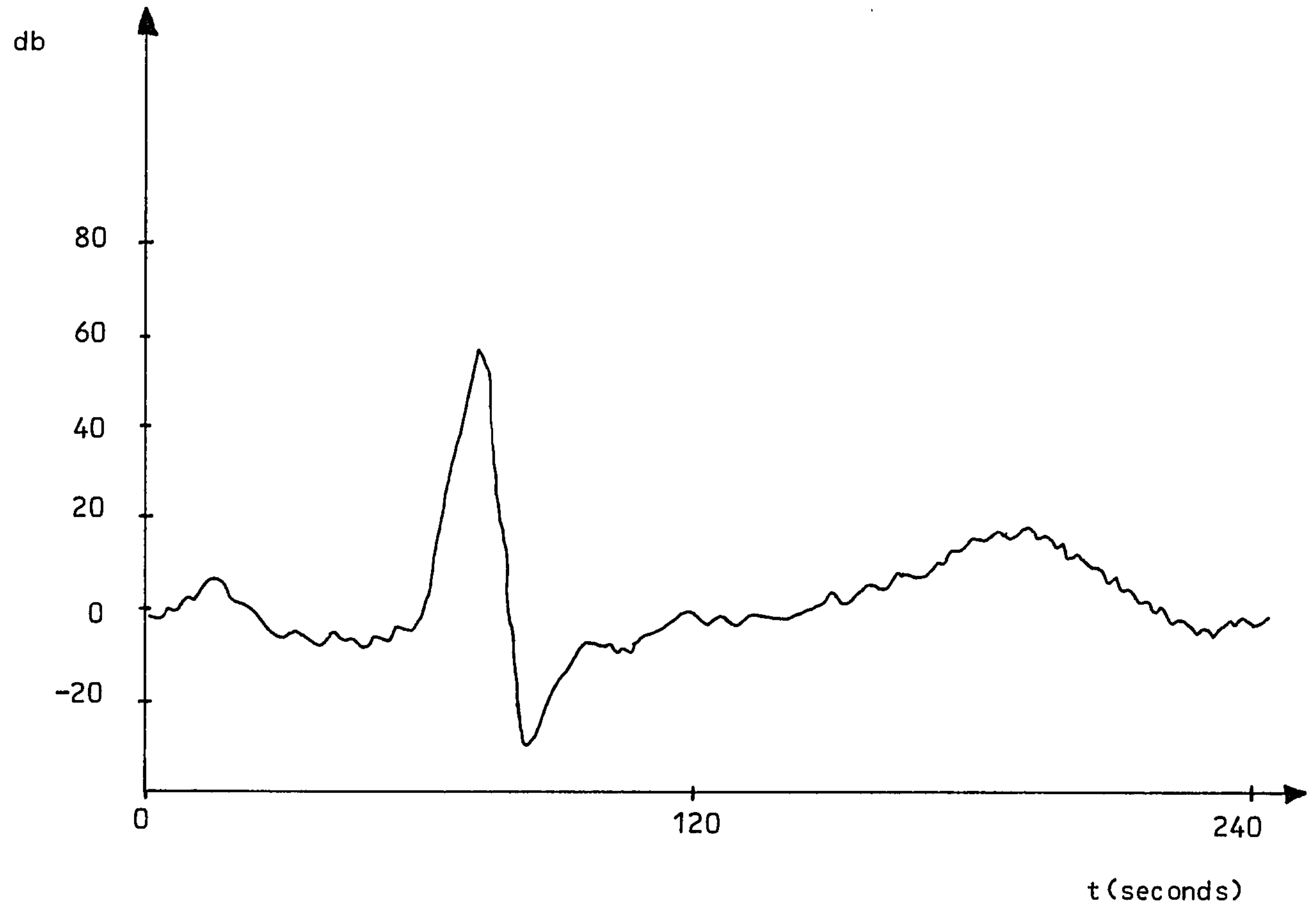
We will now compare our algorithm to other algorithms of the same class and will note the differences in both techniques. As discussed earlier, our algorithm does not iterate over the given data space, but the algorithms presented in [4,39] do iterate over the given data space. This is one major difference between these algorithms. Another major difference is that the latter algorithms use either a linear piecewise approximation or approximating polynomials of order three or higher. In some cases higher order approximating polynomials are justified, for example in automobile body design [45], but if a fast solution is desired then the use of such approximating polynomials is not yet practical. Our algorithm uses a combination of both the linear and quadratic approximating polynomals for boundary detection and data compaction. Thus, we are not restricted to linear approximations nor are we confined to the complex higher order mathematical approximating polynomials.

Figure 5.5.1 shows the results of Pavlidis' split-and-merge algorithm [39] and, when this is compared to Figure 5.4.4, we can observe a distinct improvement in our algorithm's ability to detect and describe boundaries. That is, it can be observed between these two figures that our algorithm gives a better description of the object's contour/boundary. In Figure 5.5.1, gaps in the contour can be observed and a significant amount of contour smoothing has occurred. However in Figure 5.4.4, which is the output from our algorithm, there are no gaps and the smoothing was greatly minimized.



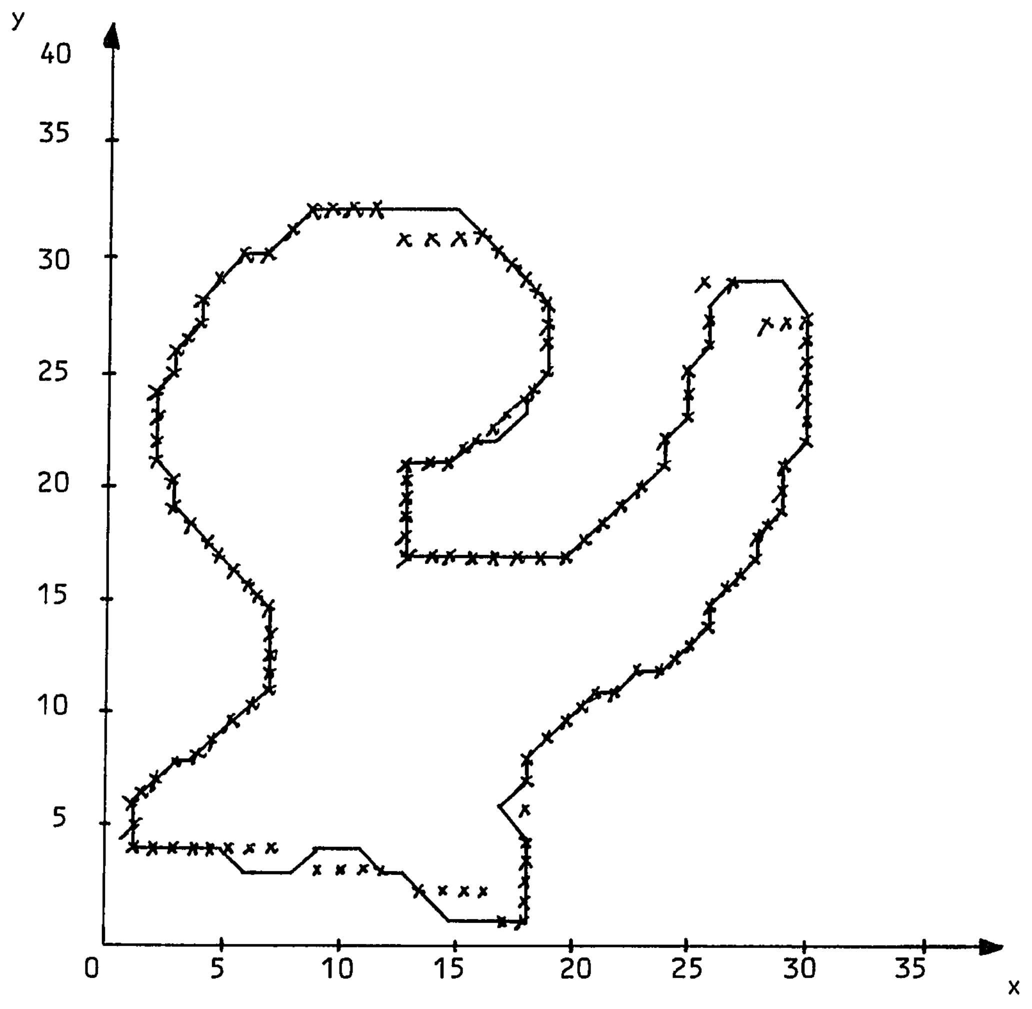
Cell Outline

Figure 5.4.2



Electrocardiogram

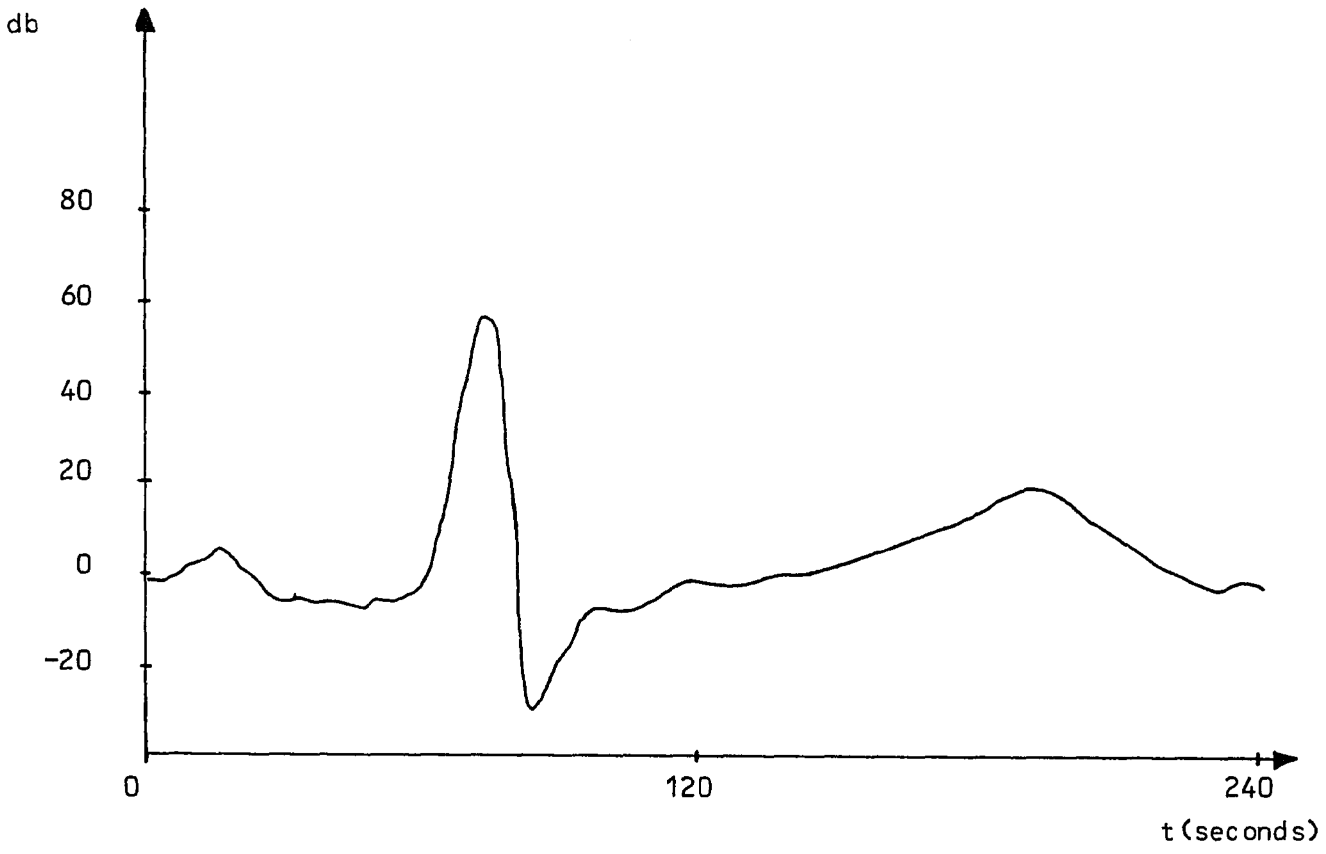
Figure 5.4.3



Piecewise Polynomial Approximation of a Cell Using

Fuzzy Variables LI and LS.

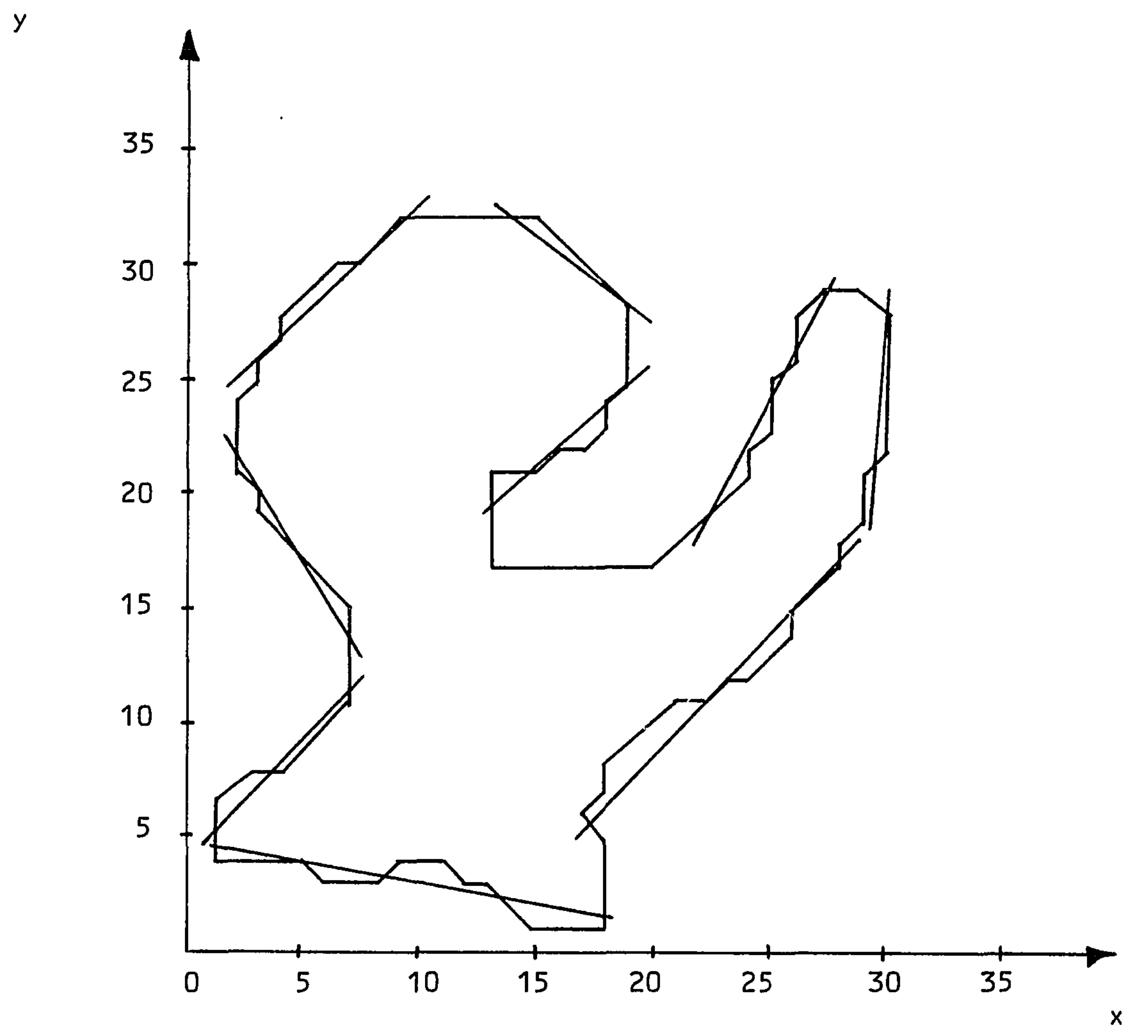
Figure 5.4.4



Piecewise Polynomial Approximation of a

Electrocardiogram

Figure 5.4.5



Piecewise Linear Approximation of a Cell 39

Figure 5.5.1

CHAPTER VI

COMPUTATIONAL COMPLEXITY

6.0 Introduction

As stated earlier, we are interested in the computational time required to solve the boundary/contour detection and description problem. The computational capabilities of the algorithm presented in this paper will be provided in this chapter. An algorithm's processing capabilities, as presented by computational complexity theory, are primarily concerned with the time and space requirements. There are other possible computational measures, such as the number of iterations required for a solution. Computational complexity theory, which has opened up a relatively new field in algorithm computational techniques, has been applied to problems such as the traveling salesman's schedule, the vertex coloring, and maximum independent sets of vertices in graphs to determine if an optimum solution exists. Stated another way, the theory is interested in polynomial-time algorithms which are guaranteed to yield optimum solutions. The above mentioned problems have been termed NP hard, that is to say, despite the intensive efforts of many workers, optimum solutions for these problems have not been found. Furthermore, if such polynomial-time algorithms exist, then this implies, according to the theory, that algorithms for a much larger class exist [11]. We will not consider NP problems. Instead we will draw from the theory behind NP (computationally complex) problems. In our analysis of boundary/contour algorithms a distinction between the worst case and expected time behavior will be made. This distinction is made because certain boundary/contour algorithms may require an enormous amount of processing time on a given data set. From a practical point of view, since the a priori data set cannot be determined, the average behavior is the more significant feature of an algorithm's ability to find a solution in time.

6.1 General Framework

We will begin by discussing a computational task in general terms. Let K be a computational problem; that is a series of computational tasks each of which we can call a procedure of K. A procedure of K is composed of n operations which are the minimum number required to perform the intended function. Now the K problem can be formed by $K_1 \cup K_2 \cup \ldots . \cup K_i$, $i=1, \ldots ., m$ procedures where each K_i has n operations. In the problem K we are primarly interested in two parameters. They are time and size, or the amount of memory required. The time and size required for a computational task depends upon the K_i computational characteristics. For example, in a K_i searching task there will be time required for query and for storage. These can be represented respectively as $Q(N) = \log N$ and $S(N) = N$ where $Q(N)$ is the query processing time, $S(N)$ the storage processing time and N equals the number of processing operations [49] . The total time required for task K_i can then be written as $T(N) = Q(N) + S(N)$ or $T(N) = \log N + N$. Thus, in order to evaluate computational algorithms of the same class, they must first be decomposed into tasks of K_i such that the parameters of interest can be easily evaluated. Within a given algorithm class there may be differences in processing time and storage because of the method or methods used to solve the particular task. For example, consider the product of two complex numbers, a + jb and c + jd, which can be found in either of the following two ways. In the first method the real part of the product can be formed as $X = ac - bd$ and the imaginary part as $Y = ad + bc$, which requires four multiplications, one addition and one subtraction. In the second method the product can be formed as $x = (a + b)(c - d)$, $y = ad$ and $z = bc$ where the final product is found by computing the real part $X = x + y - z$ and the imaginary part $Y = y + z$. This method requires three additions, two subtractions, and three multiplications. Now it is known that addition and subtraction operations on many computing machines take less time than multiplication/division. Thus, the second method is faster than the first method.

The above discussion was intended to:

(a) bound the following discussion on computations and

(b) suggest a way algorithms can be decomposed for comparison purposes.

However, it is not the intent in this paper to prove or state conditions on computational complexity. These and associated items are left to the theory of computational complexities [17, 49].

6.2 Computations of the Algorithm

Polynomial approximation algorithms of the edge/boundary class can be decomposed into tasks called procedures for computional evaluation. The algorithm developed in this paper, as stated earlier, uses input data which is uniformly spaced along the prime coordinate axes. The algorithm as designed has two input buffers of finite length in which data is initially stored. The size of these buffers was initially set to the length of the number of data points in an interval. These two buffers were designed to work like a flip-flop, i.e. after the first buffer has been filled with data the algorithm starts processing while the second buffer has data read into it. Then when the processing of data from the first buffer is completed the algorithm will read the contents from buffer number two while buffer number one reads more input data. This technique greatly reduces the amount of storage space required for input data. The input buffer length can be adjusted according to the object or signal characteristics. For example, if the training data has many corners or the signal is noisy, the input buffer length could be made smaller and, conversely, if the signal has few corners or very little noise the input buffer length could be longer. Futhermore, this technique of adjustable input buffer lengths has been shown experimentally to reduce the number of splits per interval which can be encountered during processing. The process of splitting an interval is computationally costly, as pointed out in [39], because it requires more computational time for a solution. At this point the algorithm must find a solution to equation 1.2.6 which requires a solution of equation 2.1.1. The solution to equation 2.1.1 requires matrix multiplication and inversion. That is the algorithm must form the matrix $C = A^T A$, then form C^{-1}, and finally form $D = C^{-1}A^T$. The time required to perform a square matrix multiplication is the time to perform n^3 multiplications and $n^2(n-1)$ additions for a total computational time of $T(n) = n^3 + n^2\ (n-1)$ [56]. Thus, the time needed to solve matrix equation 2.1.1 using the Gaussian elimination method [56] requires $1/3(n^3+3n^2-n)$ multiplications and $1/6(2n^3+3n^2+n)$ additions for a total of

$$T_m(n) = (2/3)n^3+(3/2)n^2-(1/6)n$$
$$= .16667n(4.019n^2+8.998n-1). \qquad 6.2.1$$

The fuzzy decision portion of this algorithm, as presented in Chapter IV, requires 4 multiplications, 12 comparisons , and 12 additions for a total time of $T_F(n) = 4m+12g+12c$; where m,g, and c equal the computing machine's multipy, add, and compare times respectively. From the above we see that the total computational effort required for this algorithm, using standard techniques, is

$$T_T(n) = T_m(n) + T_F(n)$$

or

$$T_T(n) = (2/3)n^3 + (3/2)n^2 - (1/6)n + 4m +12g + 12c. \qquad 6.2.2$$

The $T_m(n)$ term in equation 6.2.1 must be recomputed each time a determination is made that intervals I_n and I_{n+1} can be merged because of the standard matrix technique employed. Therefore the number of terms required to form the elements of A_i in equation 2.1.1 will increase, thus requiring more computation time. However, as pointed out in Chapter V, we can compute A_i^+ by using a method which circumvents a significant amount of the required computation. That is, for a

2x2 generalized inverse matrix the algorithm requires 2 multiplies and 2 additions or $T_2 = 2m + 2g$. For a 3x3 generalized inverse matrix we have $T_3 = 4m + 4g$. This gives the total computational time required, using the shorter matrix method, as

$$T_T(n) = I_n\,(pm + qg) + I_n\,T_F(n) \qquad 6.2.3$$

where

I_n = number of intervals merged

and

$p=q$ which equals 2 for a 2x2 matrix or

3 for a 3x3 matrix.

Also, note that $T_m(n)$ in the above equation has now become a constant term because there is no longer a requirement to recompute the elements for the matrix A_i in equation 2.1.1. For example, given a straight contour in a 2-dimensional space of 100 points, where the contour is a line parallel to one of the axes, the time required to process twenty intervals would be

$$T_T(n) = 20\,(2p + 2g) + 20\,T_F(n).$$

Applying this equation, along with the characteristics of the computer used in this study, the time required is approximately $T_T(n) = 7.03 \times 10^{-6}$ seconds . The actual measured processing time was $T_T(n) = 7.11 \times 10^{-6}$ seconds . It can be observed that the analytical computational time expression gives good results when compared to the measured values. The time required to process the above 100 point contour by using the method contained in equation 6.2.1 is

$$T_m(n) = I_n T_m(n) + I_n T_f(n)$$

or

$$T_m(n) = 10.21 \times 10^{-6} \text{ seconds} . \qquad 6.2.4$$

Thus, we observe that the method introduced in this paper is indeed shorter when compared to standard polynomial processing techniques.

6.3 Computations of Other Algorithms

We will briefly look at other boundary/contour algorithms and compare them computationally to the algorithm developed in this paper. First it should be pointed out that for a true comparision all algorithms being compared should be tested against the same data set and run on the same computer. This is not feasible in most cases because of time and/or equipment constraints. The comparisons made in this paper were made using the same data base but due to equipment constraints we were unable to use the same type of computer. A discussion on the effect of using two different types of computers and the results are presented below. We start by comparing the approach given in [39, 42] on the set of data as given by Figure 5.4.2. Pavlidis and Horowitz [42] showed that the execution time on an IBM 360-91 for a piecewise linear fit of a cell outline was approximately .35 seconds. Their results are shown in Figure 5.5.1 . By comparison our results are shown in Figure 5.4.4 which took about 1.2 seconds on a mini-computer. The time comparison between these two methods can only be approximate because of the different computers used. However, we can project the mini-computer's execution time onto the IBM 360-91 by a dividing factor of about 2 to 3. Therefore, we see that our algorithm is comparable time wise with the algorithm given in [39, 42]. One major difference between these two algorithms, given the time as computed above, is the capability of the algorithm presented in this paper to accurately describe an object's contour. This can be observed by viewing Figures 5.4.4 and 5.4.5 . Appendix C contains contours of other objects where results are presented in the form of Figures in which the fuzzy decision space was varied. Figure 5.5.1 contains results which are very similiar to the results of the cell outline as given in [39]. The execution time for Figure 5.5.1 was .9 seconds. Using the execution assumptions as given above, we see that the algorithm as given in this paper is faster than the one given in [39, 42]. Chang and Pavlidis [10] used a fuzzy characteristic function to determine the angle at each knot in order to decide if adjacent intervals could be approximated by a single higher order polynomial. Their fuzzy characteristic function depends upon computing the slope between each point contained on the contour. Computationally this is expensive because of the large number of multiplies required for a solution.

CHAPTER VII

CONCLUSIONS and RECOMMENDATIONS

7.0 Introduction

The observed results and areas for future investigation are presented in the following two sections.

7.1 Conclusions

A new technique has been introduced which can determine the description of any object's contour by using combinations of linear and quadratic approximating polynomials. This technique incorporates into an algorithm the use of fuzzy linguistic descriptors and fuzzy decision making for the detections and descriptions of object contours. The fuzzy linguistic descriptors allow for dynamic error threshold adjustment which in turn controls the goodness of an object's contour description and the associated processing time. A fuzzy decision space was generated by using a fuzzy Bayes model which produced values on the interval $[0,1]$. From these fuzzy values a decision was made as to the existence of a breakpoint within an interval $[I_{n-1}, I_n]$. This new method greatly improved the merging of intervals for overall object contour descriptions and data compaction. That is, based upon the output from the fuzzy model, a fuzzy decision was made on whether a merge of intervals I_{n-1} and I_n could occur within the computed error threshold. The fuzzy model was integrated into a polynomial approximation algorithm which used a combination of piecewise linear and quadratic polynomials to describe object contours. This technique of using a combination of low order approximating polynomials to describe contours is new and unique. It is unique for two reasons : first because it uses a quadratic polynomial for contour approximation, and second because it uses a combination of approximating polynomials to describe a contour. This method proved to be highly successful, as shown by this study, in describing a contour while keeping the computational complexity to a minimum. Futhermore with this technique we can describe, up to the optimum mathematical approximation limit, any contour without using higher order approximating polynomials and without the associated computational complexity. This scheme demonstrated that good contour descriptions can be achieved for the linear case if the fuzzy value is less than .3 and for the quadratic case if the fuzzy value is less than .35. The quadratic polynomial approximation technique was shown to give a smoother describing contour than the linear approximation technique.

One of the most important attributes of the fuzzy model, as applied in this paper, is the admissibility of heuristic inputs. These inputs in the form of linguistic parameters greatly affect both object description and processing time. Given a set of linguistic parameters, the algorithm will find the near optimum or optimum solution with minimal computational time. Near optimum solutions were defined, in the least squares sense, to be no greater than 2% of the optimum error value. The fuzzy model improved object contour detection when compared to the results from other algorithms of the same class. Also, the fuzzy model allows for breakpoint adjustment by using fuzzy linguistic descriptive values which aid in the control of merging intervals. This method is further enhanced if an interactive display is used because the object and the approximating contour can be displayed simultaneously and a judicious selection of the linguistic parameter can then be made.

It was demonstrated that the computational time required for interval merging can be reduced by using a unique matrix operation as introduced in this paper. This method uses a scalar multiplying term along with matrix addition to produce a new generalized inverse matrix for interval merging. The determination of the scalar multiplying term can be decided on in advance and then stored in the algorithm's look up table routine. Using this scheme it was shown that the matrix operations associated with interval merging can be significantly reduced. Futhermore, the matrix muliplying term did not vary per interval unless the number of data points per interval varied or the spacing between data points varied.

The contour detection algorithm as given in this paper can operate on noisy signals with good results. Also, it was shown that the algorithm was faster in processing noisy signals than other algorithms of the same class.
same class.

7.2 Recommendations

This research has suggested that more work is required in the area of fuzzy model development and in the application of these models to the pattern recognition process. That is, the fuzzy modeling process needs to be refined and extended to where ill-defined processes can be easily modeled to a level where the ill-defined process can be defined. Once we understand a particular process we can continue to investigate the process and(or), refine the model, by using classical mathematical modeling techniques. Along those lines an investigation into extending the fuzzy model presented in this paper to higher order approximating polynomials should be made. In particular the breakpoint adjustment process for higher order polynomials should be studied when several breakpoints occur within an interval. Also, this process should be investigated where the computional complexity is kept to a minimum. One immediate area for investigation is the extension of the fuzzy Bayes model's ability to predict the use of a conic approximating polynomial over I_n past intervals where computational complexity is kept to a minimum.

APPENDIX A

OPTIMUM BREAKPOINT LOCATION

In this appendix the derivation of equations 2.3.2 thru 2.3.5 will be shown. These equations show the optimum location of breakpoints within an interval where the absolute values of the pointwise errors from the right and left of a breakpoint x_i, as shown in Figure A.1, are equal in an interval$[x_{i-1}, x_{i+1}]$. A partition of[a,b] is defined as a collection of subintervals which are ordered along an axis as follows

$$a = x_0 < x_1 < \ldots\ldots\ldots\ldots x_{n-1} < x_n = b.$$

Within the subinterval $[x_{i-1}, x_i]$ let f(t) be a continuous differentiable function which is to be approximated by $\psi_i(x)$. We can express the approximating polynomial as

$$\psi_i(x) = C_{oi}\theta_{oi}(x) + C_{1i}\theta_{1i} + \ldots\ldots + C_{mi}\theta_{mi}(x)$$

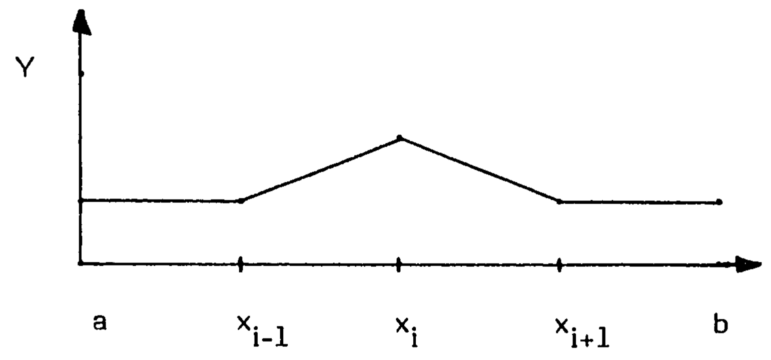
where

$$0<m<n.$$

Here $\theta_{oi},\theta_{1i},\ldots,\theta_{mi}$ are orthonormal polynomials on $[x_{i-1}, x_i]$ which satisfy

$$\int_a^b \theta_i(x)\theta_j(x)dx = \delta_{ij}$$

where δ_{ij} is the Kronecker delta, over the range $[a,b]$. We will use these orthonormal polynomials in order to simplify the computations, thereby reducing round-off errors.



Breakpoint Located At x_i

Figure A.1

The following notation will be used within this derivation.

$$\theta_i(x) = [\theta_{oi}(x), \theta_{1i}(x), \ldots\ldots, \theta_{mi}(x)]^T$$

$$C_i = [c_{0i}, c_{1i}, \ldots\ldots\ldots, c_{m,i}]^T$$

$$F_i = \int_{x_{i-1}}^{x_i} f(x)\theta_i(x)dx, \qquad \text{A.1}$$

$$G_i = \int_{x_{i-1}}^{x_i} \theta_i(x)\theta_i^T(x)dx, \qquad \text{A.2}$$

where

$\theta_i(x)$ is a vector which represents the approximating polynomial of degree $m(m = 1, \ldots\ldots, n-1)$ on $[x_{i-1}, x_i]$,

C_i is the vector of coefficients of the approximating polynomial on $[x_{i-1}, x_i]$,

F_i is a vector associated with the actual function,

G_i is the Gram matrix, which is positive definite and symmetric [13],

and

the superscript T denotes the transpose of a vector . The derivation starts by defining the integral square error equation in the least square polynomial approximation sense as

$$E_i(C_i) = \int_{x_{i-1}}^{x_i} [f - \psi_i]^2 dx \qquad \text{A.3}$$

and

$$E = \sum_{i=1}^{n} E_i \qquad \text{A.3'}$$

where $f(x)$ is the actual function and $\psi_i(x)$ is the approximating polynomial on $[x_{i-1}, x_i]$. Using the least squares criterion, the best approximation ψ_i to f is found on each interval to obtain E. The normal equations corresponding to A.3 can be written as

$$G_iC_i = F_i\ , \quad i = 1,2,\ldots\ldots,n \qquad \text{A.4}$$

The minimum error E_i, in A.3, is found by taking the partial derivative of E_i with respect to the C_i and setting the expressions equal to zero. In order to dertemine the results of A.4 we start by expanding the definition A.3 which yields

$$E_i = \int_{x_{i-1}}^{x_i} [f^2 - 2\psi_i f + \psi_i^2]dx$$

$$= \int_{x_{i-1}}^{x_i} f^2 dx - 2\int_{x_{i-1}}^{x_i} \psi_i f dx + \int_{x_{i-1}}^{x} \psi_i^2 dx \qquad \text{A.5}$$

Substituting

$$\psi_i(x) = \sum_{l=o}^{m} C_{li}\theta_l(x) \qquad \text{A.6}$$

into A.5 gives

$$E_i = \int_{x_{i-1}}^{x_i} f^2dx - 2\sum_{l=o}^{m} C_{li} \int_{x_{i-1}}^{x_i} f(x)\theta_l(x)dx$$

$$+ \sum_{l=o}^{m} \sum_{K=o}^{m} C_{li}C_{Ki} \int_{x_{i-1}}^{x_i} \theta_l(x)\theta_K(x)dx$$

which can now be differentiated, as defined above, to yield the normal equations. Since each G_i is the identity matrix A.4 can be solved for the coefficients as follows

$$C_i = F_i.$$

Taking the partial derivatives of the above equation yields

$$\frac{\partial C_i}{\partial x_K} = \frac{\partial F_i}{\partial x_K}$$

Substituting these coefficients into A.6 will yield the optimum approximating polynomial across each of the intervals $[x_{i-1},x_i]$ in the least squares sense. Next we will take the partial derivatives of E with respect to the x_i's and set the results equal to zero. This process will yield the optimum breakpoint location(s) within the given interval[a,b] . Differentiating C_i with respect to the location x_j yields

$$\frac{\partial C_i}{\partial x_j} = \frac{\partial F_i}{\partial x_j} \,. \qquad A.7$$

We note the following formulas which will be used in the sequel. These equations were obtained by differentiating equation A.1 which yields

$$\frac{\partial F_i}{\partial x_i} = f(x_i)\ \theta\ (x_i) + \int_{x_{i-1}}^{x_i} f\frac{\partial}{\partial x}\ \theta\ dx\,,$$

$$\frac{\partial F_i}{\partial x_{i-1}} = f(x_{i-1})\ \theta\ (x_{i-1}) + \int_{x_{i-1}}^{x_i} f\frac{\partial}{\partial x}\ \theta\ dx\,.$$

In the above equations the second term is dependent upon the end points which evaluate to zero. Since θ is orthornomal the results of differentiating equation A.2 will yield the identity matrix. Solving equation A.4 for C_i and differentiating with respect to x_i we get

$$\frac{\partial C_i}{\partial x_j} = \frac{\partial F_i}{\partial x_j} \qquad A.8$$

Substituting the results from $\frac{\partial F_i}{\partial x_i}$ into the above equation will give

$$\frac{\partial C_i}{\partial x_i} = -F_i - f(x_i)\theta_i(x_i)$$

then substituting $C_i = F_i$ into the above equation yields

$$\frac{\partial C_i}{\partial x_i} = -(C_i - f(x_i)\theta_i(x_i))$$
$$= \theta_i(x_i)e_i(x_i)$$

where

$$e_i(x_i) = C_i - f(x_i)\theta_i(x_i)\,.$$

Solving for x_{i-1} we get

$$\frac{\partial C_i}{\partial x_{i-1}} = \theta_i(x_{i-1})e_i(x_{i-1})\,.$$

From 12 equation A.3' can be written as

$$E = \int_a^b f(t)^2 dt - \sum_{i=1}^{n} F_i^T C_i . \qquad \text{A.9}$$

Differentiating equation A.9 with respect to x_j and substituting $\frac{\partial F_i^T}{\partial x_j}$ and $\frac{\partial C_i}{\partial x_j}$ from above will give

$$\frac{\partial E}{\partial x_j} = - \sum_{i=1}^{n} [\frac{\partial F_i^T}{\partial x_j} c_i + F_i^T \frac{\partial c_i}{\partial x_j}]$$

$$= -f(x_j)c_j + f(x_j)c_{j-1} - F_j^T [c_j - f(x_j)]$$

$$+ F_{j-1}^T [c_j - f(x_j)] .$$

Since $C_i = F_i$ and after substituting into the above equation we get

$$\frac{\partial E}{\partial x_j} = f(x_j)[c_{j-1} - c_j] + c_{j-1}[f(x_j) - c_{j-1}] - c_j [f(x_j) - c_j]$$

which reduces to

$$\frac{\partial E}{\partial x_j} = 2f(x_j)c_{j-1} - 2f(x_j)c_j - c_{j-1}^2 + c_j^2 . \qquad \text{A.10}$$

Since

$$e_i = f(x_i) - C_i \qquad \text{A.11}$$

we can solve for C_i and C_{i-1} which will yield

$$C_i = f(x_i) - e_i(x_i) \qquad \text{A.12}$$

and

$$C_{i-1} = f(x_i) - e_{i-1}(x_i) \qquad \text{A.13}$$

Substituting equations A.12 and A.13 into A.10 gives

$$\frac{\partial E}{\partial x_j} = e_j^2(x_j) - e_{j-1}^2(x_j) .$$

Differentiating this with respect to x_j gives

$$\frac{\partial^2 E}{\partial x_j \partial x_l} = 2e_j(x_j)\frac{\partial e_j(x_j)}{\partial x_l} - 2e_{j-1}(x_j)\frac{\partial e_{j-1}(x_j)}{\partial x_l} \qquad \text{A.14}$$

Differentiating equation A.11 with respect to x_k gives

$$\frac{\partial e_j(x_j)}{\partial x_k} = \frac{\partial f(x_j)}{\partial x_k} - \frac{\partial C_j}{\partial x_k}$$

and if j = k then

$$\dot{e}_j(x_j) = \frac{df}{dx_j} .$$

Solving $\frac{\partial e_j(x_j)}{\partial x_K}$ where k=j gives

$$\frac{\partial e_j(x_j)}{\partial x_j} = \dot{e}_j(x_j) - C_j e_j(x_j) \qquad \text{A.15}$$

$$\frac{\partial e_j(x_j)}{\partial x_{j-1}} = C_j e_j(x_{j-1}) \qquad \text{A.16}$$

and for k = j-1 we get

$$\frac{\partial e_{j-1}(x_j)}{\partial x_{j-1}} = C_{j-1} e_j(x_{j-1}) \qquad \text{A.17}$$

$$\frac{\partial e_{j-1}(x_j)}{\partial x_j} = \dot{e}_j(x_i) - C_{j-1} e_{j-1}(x_j) \qquad \text{A.18}$$

We now complete the proof by selecting particular families of orthonormal polynomials. Pavlidis in personnel communications suggested that the above equations can be simplified by introducting

$$\Gamma(p,q,r,s) \equiv \theta^T(p)c^{-1}\theta(x) = \frac{B}{r-q} \qquad \text{A.19}$$

where

c is the Gram matrix defined on the interval q,r and B

does not depend upon r or q .

We will obtain the orthonormal families of polynomials by starting with the Legendre polynomials $P_0, P_1, \ldots, P_m$ which are orthogonal on [-1,1]. The Legendre polynomials are defined as

$$\theta_m(x) = \sqrt{\frac{2n+1}{2}}\, P_m$$

with the following properties

$$P_m(1) = 1$$

$$P_m(-1) = (-1)^m$$

over the interval -1 to 1. We wish to find $\theta_o, \theta_1, \ldots \theta_m$ orthonormal on the interval [r,q] . It can be seen that with a change of variables the polynomials will become

$$\theta_m(x) = \sqrt{\frac{2n+1}{r+q}}\, P_m\left(\frac{2x-(r+q)}{r-q}\right).$$

If we let $x_j = r$ then

$$\theta_j(r) = \sqrt{\frac{2n+1}{r-q}}\, P_n(1)$$

which in turn give

$$\theta^T c^{-1}\theta = \sum_{j=o}^{n} \frac{2n+1}{r-q} \,. \qquad \text{A.20}$$

Expanding the right side of equation A.20 we get

$$\sum_{j=o}^{n-1} 2n+1 = 1+3+5+\ldots\ldots+(2n-1)$$

or

$$2s = 2(n)(n)$$

or

$$s = n^2$$

Equation A.20 can now be written as $\theta^T c^{-1}\theta = \dfrac{n^2}{r-q}$.

For terms of the form $\theta^T(x_j)c^{-1}\theta(x_{m-1})$ we get a different result. Writing

$$S = \sum_{j=o}^{n} \theta_m(x_j)\theta_n(x_{j-1})$$

$$= \sum_{j=o}^{n} \frac{2n+1}{r-q}\{P_n(1)P_n(-1)\}$$

where

$$x_{j-1} = q$$

$$x_j = r$$

Since $P_n(-1) = (-1)^n$ S can be expanded to give

$$s = (1-3+5-7+\ldots\ldots(-1)^n(2n-1))/(r-q) \,.$$

Now if we let

$$s_1 = 1-3+5\ldots\ldots+(-1)^{n-1}(2n-1)$$

and

$$s_2 = n^2 = 1+3+5\ldots\ldots\ldots+(2n-1)$$

we can then sum the above two equations as

$s_1 + s_2 = 2(1+5+9+\ldots\ldots\ 2n-1)$ if n is even

$= 2(1+5+9+\ldots\ldots\ (2(n-1)-1))$ if n is odd

Writing s in general terms we have

$$s = \sum_{0}^{n-1}(-1)^n(2n+1)$$

If N is even the last term is negative, as shown below

$$s = 1+5+9\ldots\ldots(2n-3)-(3+7+11+\ldots\ldots+(2n-1))$$

$$= s_1 + s_2$$

$$2s_1 = (1+2n-3) + (5+2n-7)\ldots\ldots N/2$$

$$= (2n-2)n/2$$

$$s_1 = (2n-2)n/4$$

$$s_2 = (2n+2)n/2$$

Solving for s we get

$$S = -n$$

If N is odd the last term is plus as shown below

$$s_1 = 1+5+9\ldots\ldots\ 2n-1$$

$$s_2 = 3+7+11\ldots\ldots 2n-3$$

$$2s_1 = ((n+1)/2)2n$$

$$s_1 = n(n+1)/2$$

$$s_2 = n(n-1)/2$$

$$S = s_1 + s_2 = \tfrac{1}{2}(n(n+1-m+1))$$

$$S = n$$

Thus the general expression for S can now be written as

$$S = (-1)^{n+1}n \qquad \text{A.21}$$

which upon substitution into A.20 gives

$$\theta^T(x_j)c^{-1}\theta(x_{j-1}) = \frac{(-1)^{n+1}n}{r-q}$$

We can now write equations A.20 and A.21 in terms used in Chapter Two of this paper. Writing we have

$$Z_1 = n^2,$$

and

$$Z_2 = (-1)^{n+1} n,$$

where

n = number of terms.

Since $\theta^T(p)c^{-1}\theta(s) = \frac{Z}{r-q}$ equations A.15 thru A.18 can be written as

$$\frac{\partial e_j(x_j)}{\partial x_j} = \dot{e}_j(x) - \frac{Z_1 e_j(x_j)}{x_j - x_{j-1}} \qquad \text{A.22}$$

$$\frac{\partial e_j(x_j)}{\partial x_{j-1}} = \frac{Z_2 e_j(x_{j-1})}{x_j - x_{j-1}} \qquad \text{A.23}$$

$$\frac{\partial e_{j+1}(x_j)}{\partial x_{j+1}} = \frac{Z_2 e_{j+1}(x_{j+1})}{x_{j+1} - x_j} \qquad \text{A.24}$$

$$\frac{\partial e_{j+1}(x_j)}{\partial x_j} = \dot{e}_{j+1}(x_j) + \frac{Z_1 e_{j+1}(x_j)}{x_{j+1} - x_j} \qquad \text{A.25}$$

Substituting equations A.22 thru A.25 into equation A.14 we get

$$\frac{\partial^2 E}{\partial x_{j-1} \partial x_j} = \frac{2Z_2 e_j(x_{j-1}) e_j(x_j)}{x_j - x_{j-1}}$$

$$\frac{\partial^2 E}{\partial x_j^2} = 2e_j(x_j)\dot{e}_j(x_j) + 2e_{j+1}(x_j)\dot{e}_{j+1}(x_j) - \frac{2Z_1 e_j^2(x_j)}{x_j - x_{j-1}} - \frac{2Z_1 e_{j+1}^2(x_j)}{x_{j+1} - x_j}$$

$$\frac{\partial^2 E}{\partial x_j \partial x_{j+1}} = \frac{2Z_2 e_{j+1}(x_j) e_{j+1}(x_{j+1})}{x_{j+1} - x_j}$$

$$\frac{\partial^2 E}{\partial x_j \partial x_l} = 0$$

The above four equations are the same as given in equations 2.3.2 thru 2.3.5 of this paper.

APPENDIX B

OPTIMUM LEAST SQUARES SOLUTION

It will be shown in this appendix that the least-square technique gives the best polynomial of order $n \leq m$ given some input data space. From the theroy of approximating functions it is known that a function can be uniquely approximated by an m^{th} degree polynomial which is equal to f at m data points. Because of computational reasons we are not interested in such functions. Instead we are interested in functions of order less than or equal to m. The primary reason for this lack of interest in polynomials of order m is the computational costs involved. Furthermore it has been shown in [7, 20] that a function, f(x), can be adequately described by a polynomial of order less than m.

Assume that x is an approximate solution of the equation $Ax = b$ which is the minimum-norm least square solution i.e. a solution of the normal equation. It is known that the error vector E is the length of the vector Ax-b between two points or functions. One way of solving for the minimum of E is to find the first derivative of E; that is $\frac{\partial E}{\partial x_i}$, set it equal to zero, and then to apply a second derivative test. Another way is to note that x is a solution of the normal equation, which can be written as $A^TAx = A^Tb$. Any other vector y can be written in the form $y = x + w$ then the square of the error is

$$||Ay-b||^2 = ||(Ax-b)+Aw||^2 \qquad \text{B.1}$$

Expanding equation B.1 gives

$$||Ay-b||^2 = ||Ax-b||^2+2(Aw)^T(Ax-b)+||Aw||^2 \qquad \text{B.2}$$

The error vector $E = Ax - b$ must be perpendicular to a column space which contains the projection of b onto that space as shown in Figure B.1. Thus the vector Aw in equation B.1 must be in the column space of A which is composed of a linear combination of the columns by the components $w_1, \ldots\ldots, w_n$. The error vector of equation B.1 must be perpendicular to all vectors contained in the plane, i.e.

$$(Aw)^T(Ax - b) = 0$$

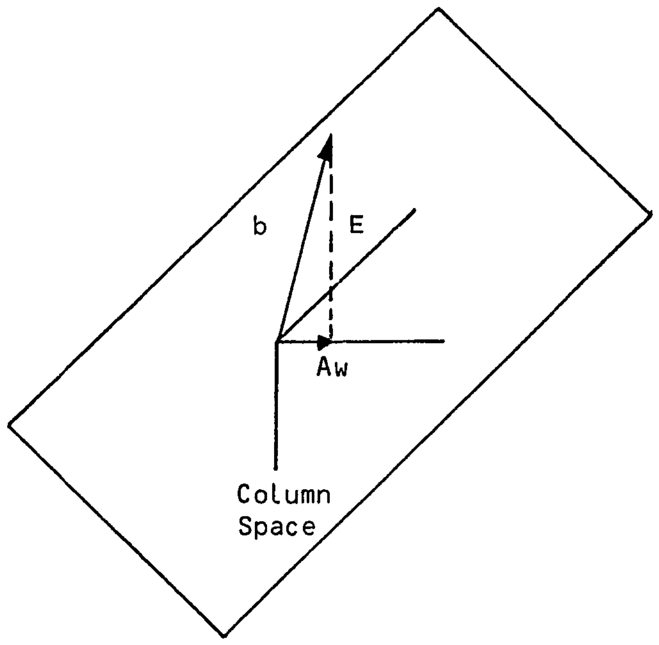
or

$$w^T(A^TAx - A^Tb) = 0.$$

It follows that

$$||Ay - b||^2 = ||Ax - b||^2 + ||Aw||^2 \geq ||Ax - b||^2$$

with equality if and only if $Aw = 0$. Hence x minimizes the error.



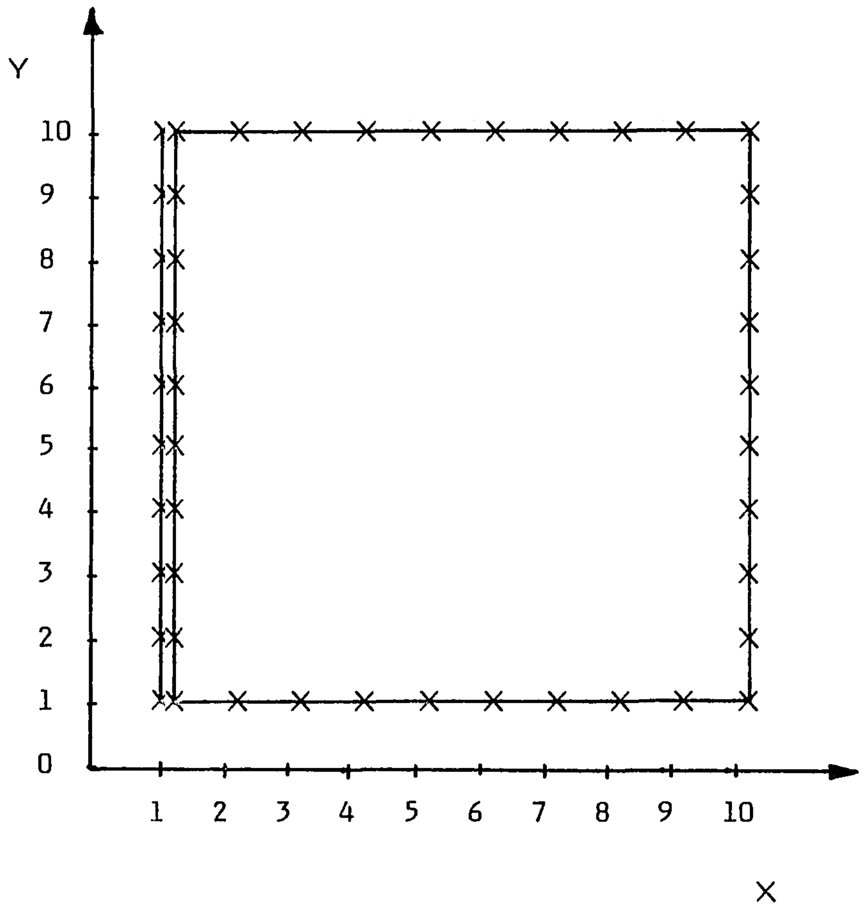
Projection of b Onto a Column Space.

Figure B.1

APPENDIX C

CONTOUR EXAMPLES

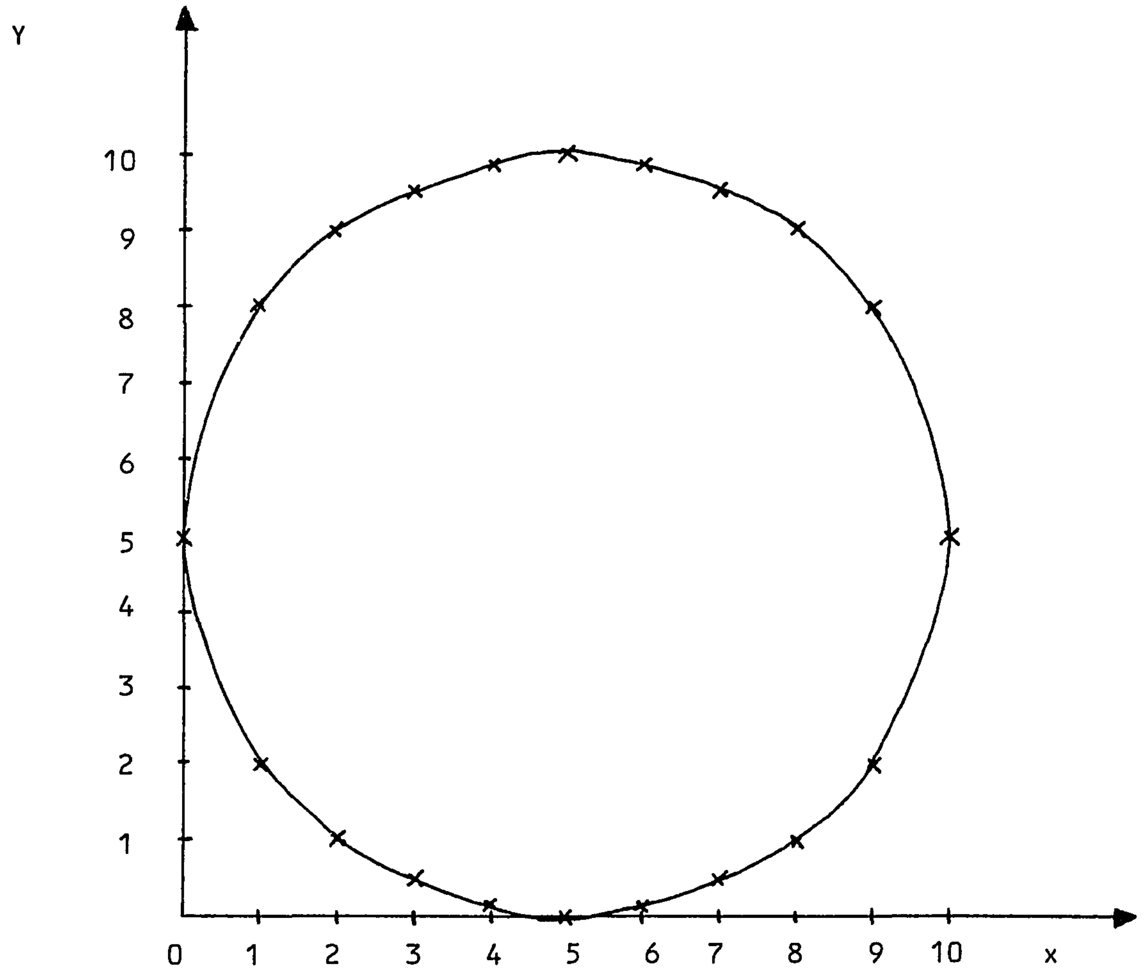
In this appendix we will present the results of the algorithm's contour detection capabilities on some objects. Figures C.1 and C.2 have been discussed in the literature, see [39] , as being difficult to detect and describe. The reason the author in [39] says that these objects are difficult to describe is because of the size of the processing interval. That is, given a processing interval which is too large, the algorithm will find a describing polynomial which passes the error threshold but gives a poor contour description. The algorithm presented in this study does not have that problem and thus, as shown in these figures, can give a good description of an object's contour. The input data for all of the Figures in this appendix will be represented by a dot and the algorithm's output data will be represented by an x. Figure C.3 is presented to show the algorithm's ability to detect and describe a sine wave. A higher order function, as shown in Figure C.4, was presented to the algorithm and, as can be seen, the detection and description results are excellent. The results presented in Figure C.5 were an attempt to reproduce the results as reported in [39] by using our algorithm. The original results from [39] are shown in Figure 5.5.1. As can be seen if we use the linguistic descriptive terms LMS and LS, the algorithm's output will almost match those given in Figure 5.5.1. However, as discussed in Chapter V, our algorithm can give better results without a loss in performance. Figure C.6 is a radar return signal of a corner reflector with a S/N = 24 db and Figure C.7 is the algorithm's output of that signal. The input signal in Figure C.8, which is the same as shown in Figure C.1, was corrupted by a random noise signal. The input noise signal was 50% of the actual input signal. As can be seen in Figure C.8 the algorithm can detect the desired object once the appropriate fuzzy variables are selected. In both test cases the error criterion was held constant in order to demonstrate the effect changing the fuzzy variables would have.



Piecewise Polynomial Approximation of a

Square Using Fuzzy Variables LI and LS.

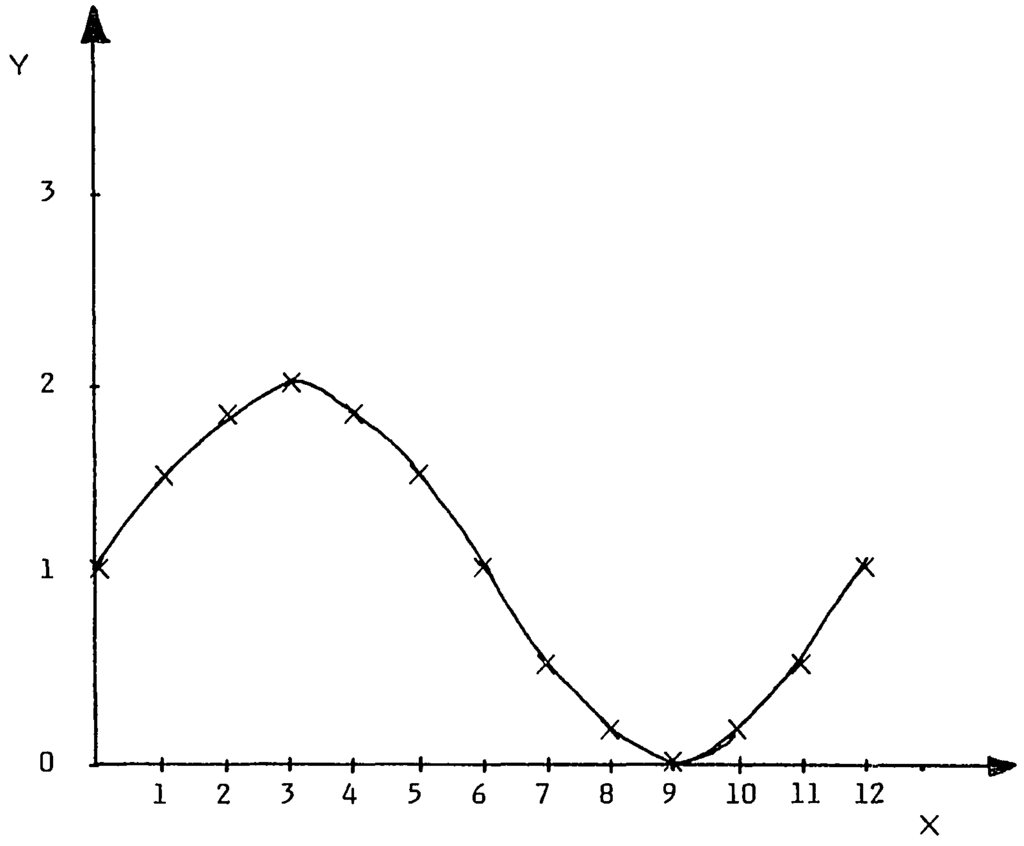
Figure C.1



Piecewise Polynomial Approximation of a

Circle Using Fuzzy Variables LI and LS.

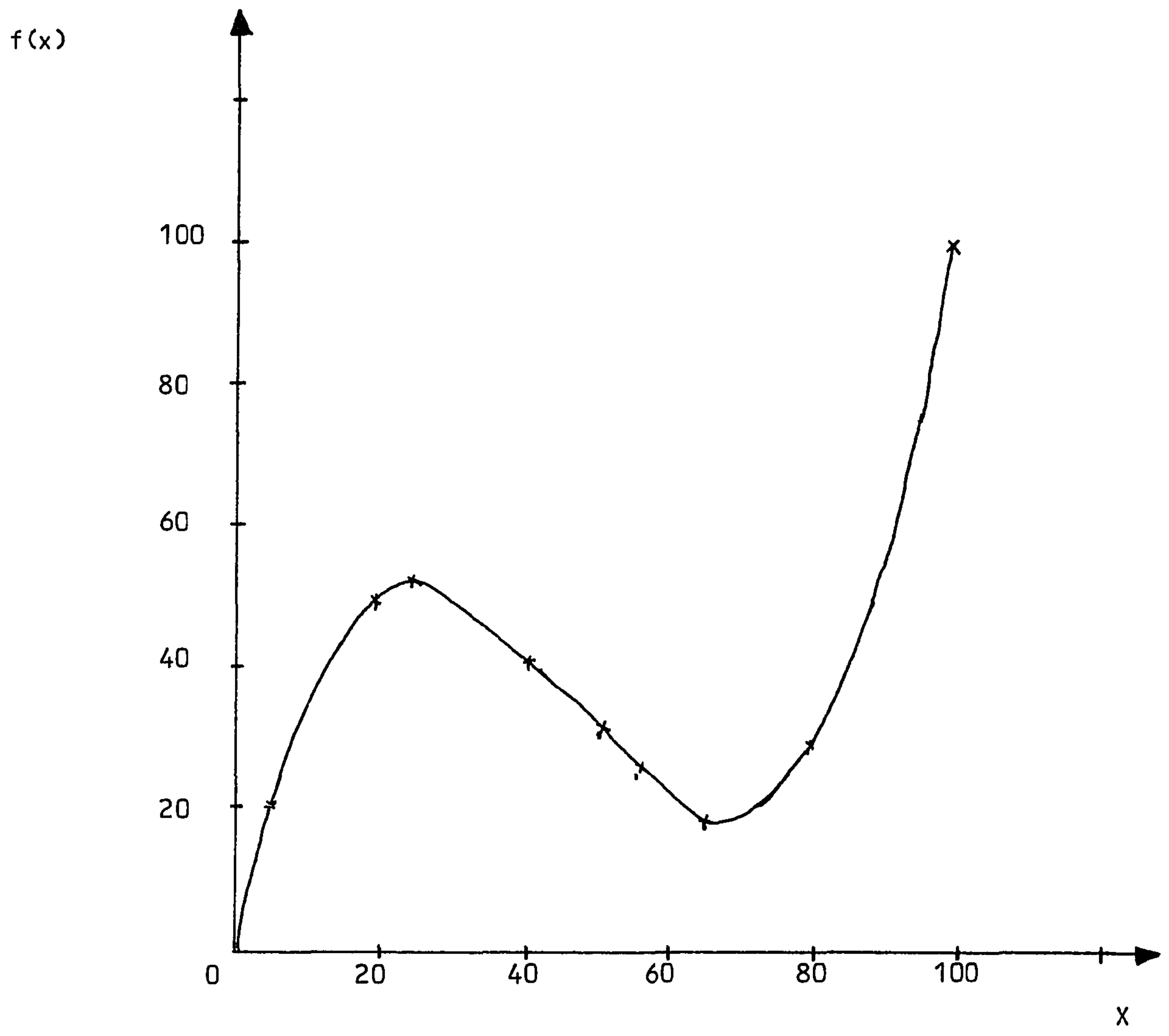
Figure C.2



Piecewise Polynomial Approximation of a

Sine Wave Using Fuzzy Variables LI and LS.

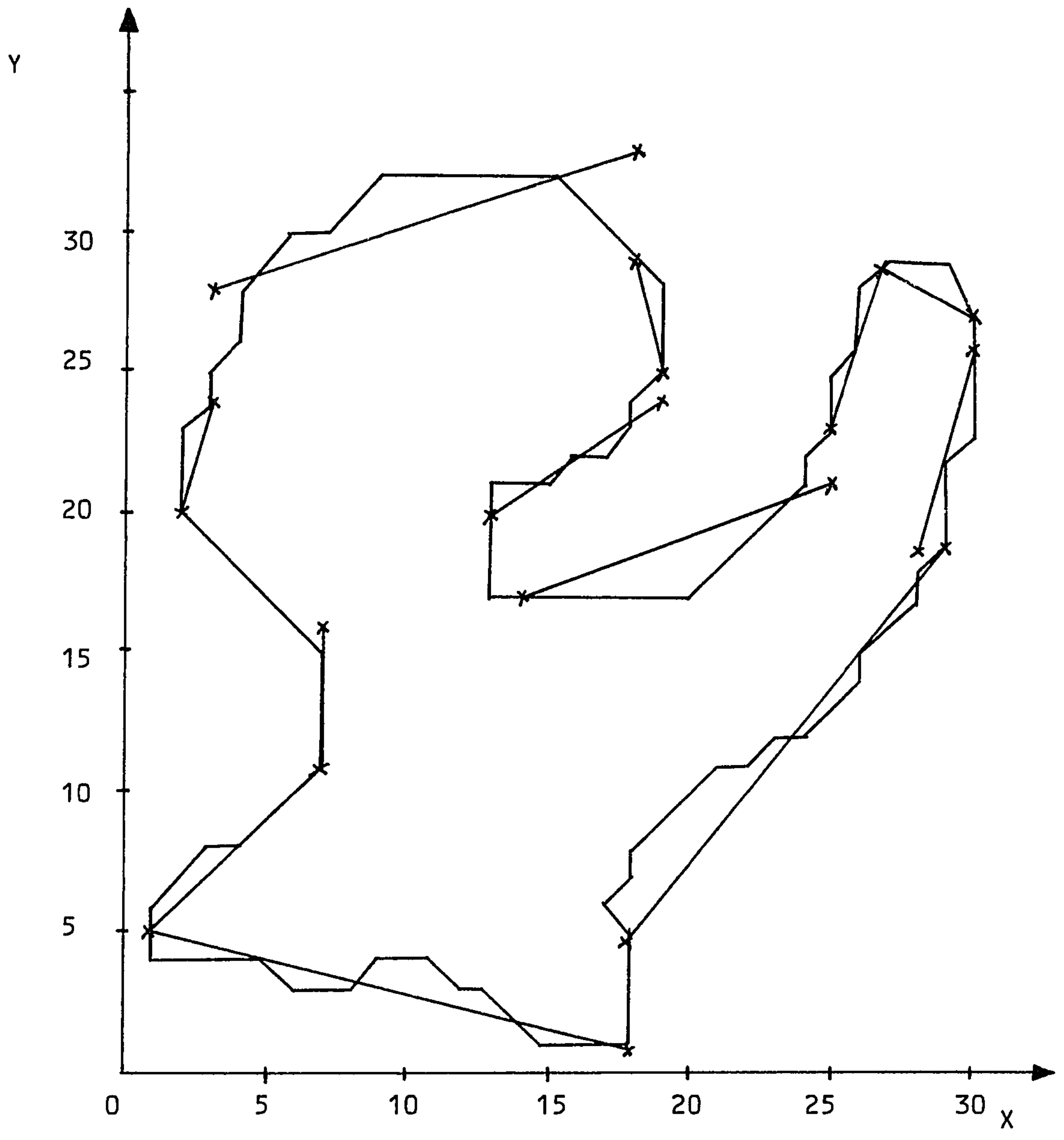
Figure C.3



Piecewise Quadratic Approximation of a

Cubic Function Using Fuzzy Variables LI and LS.

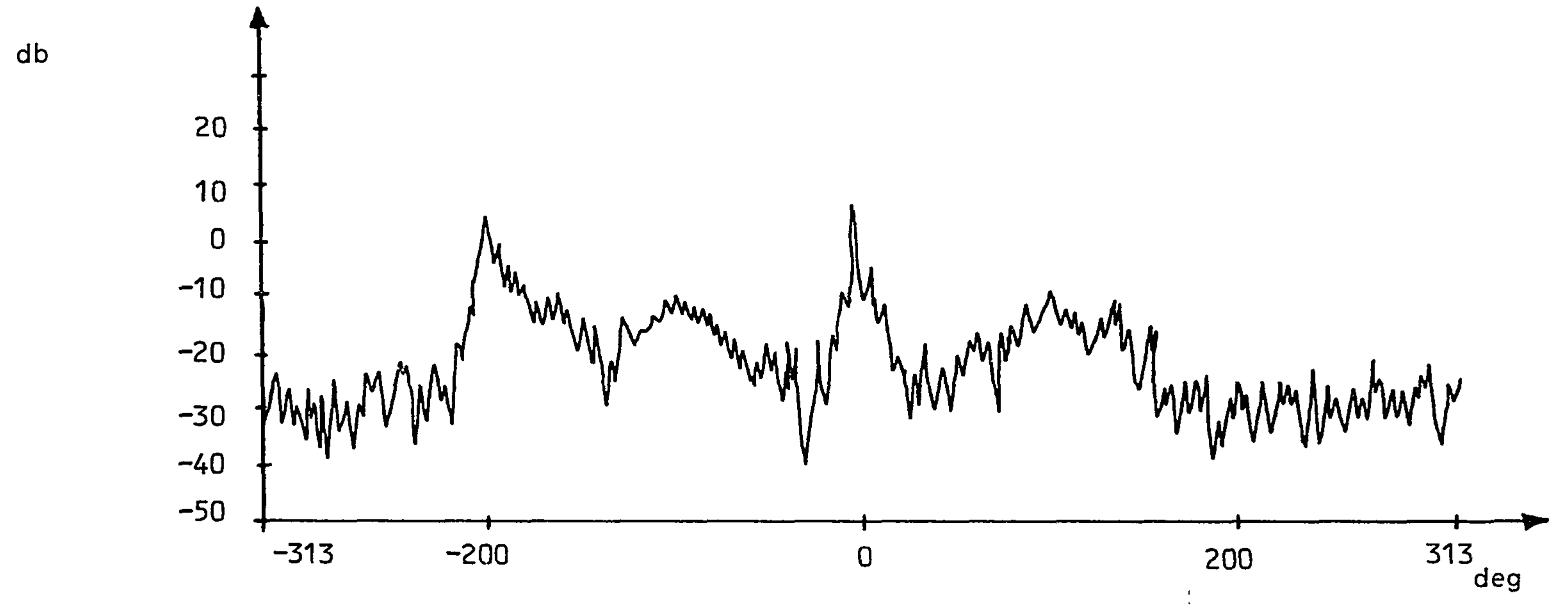
Figure C.4



Piecewise Approximation of a Cell Using

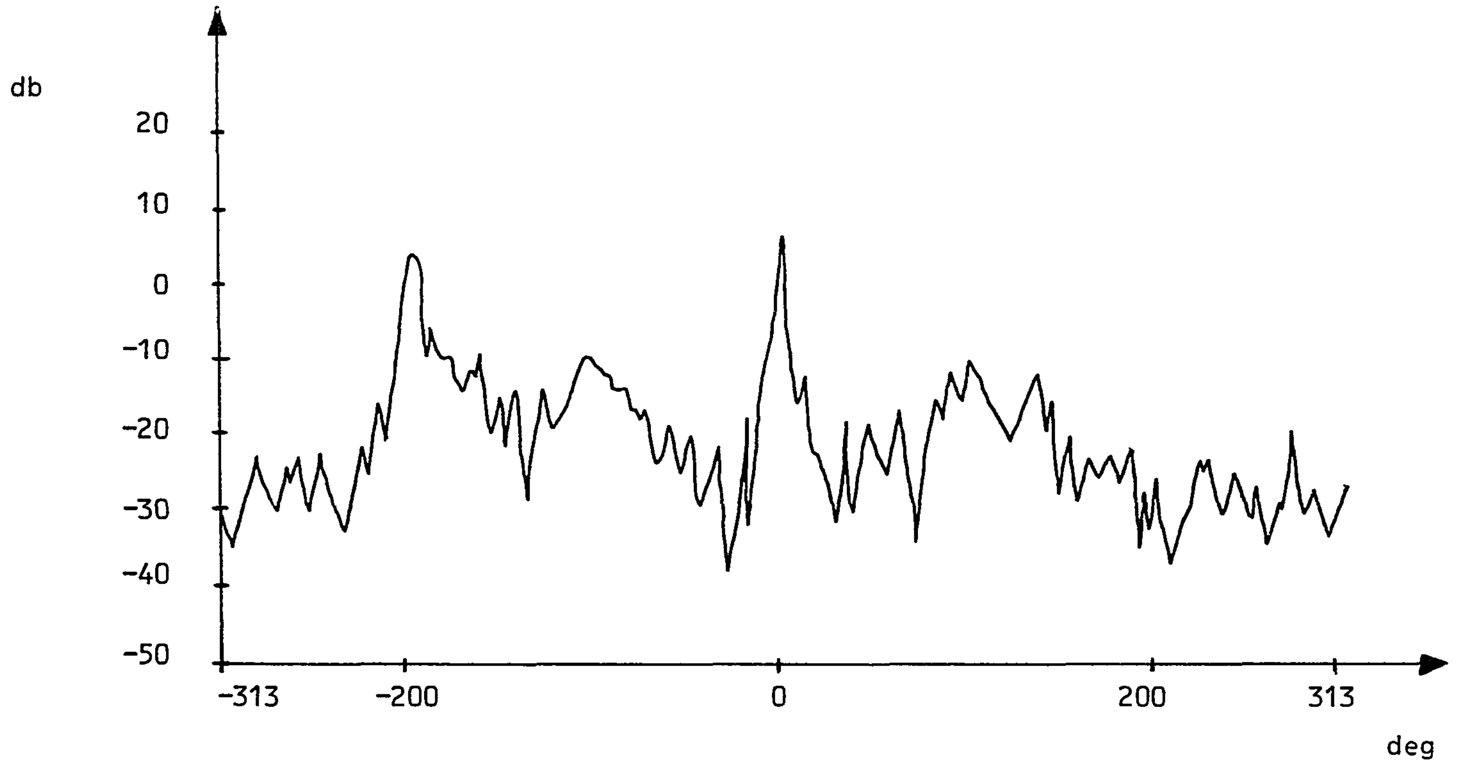
Fuzzy Variables LMS and LS.

Figure C.5



Radar Return Signal From a Corner Reflector

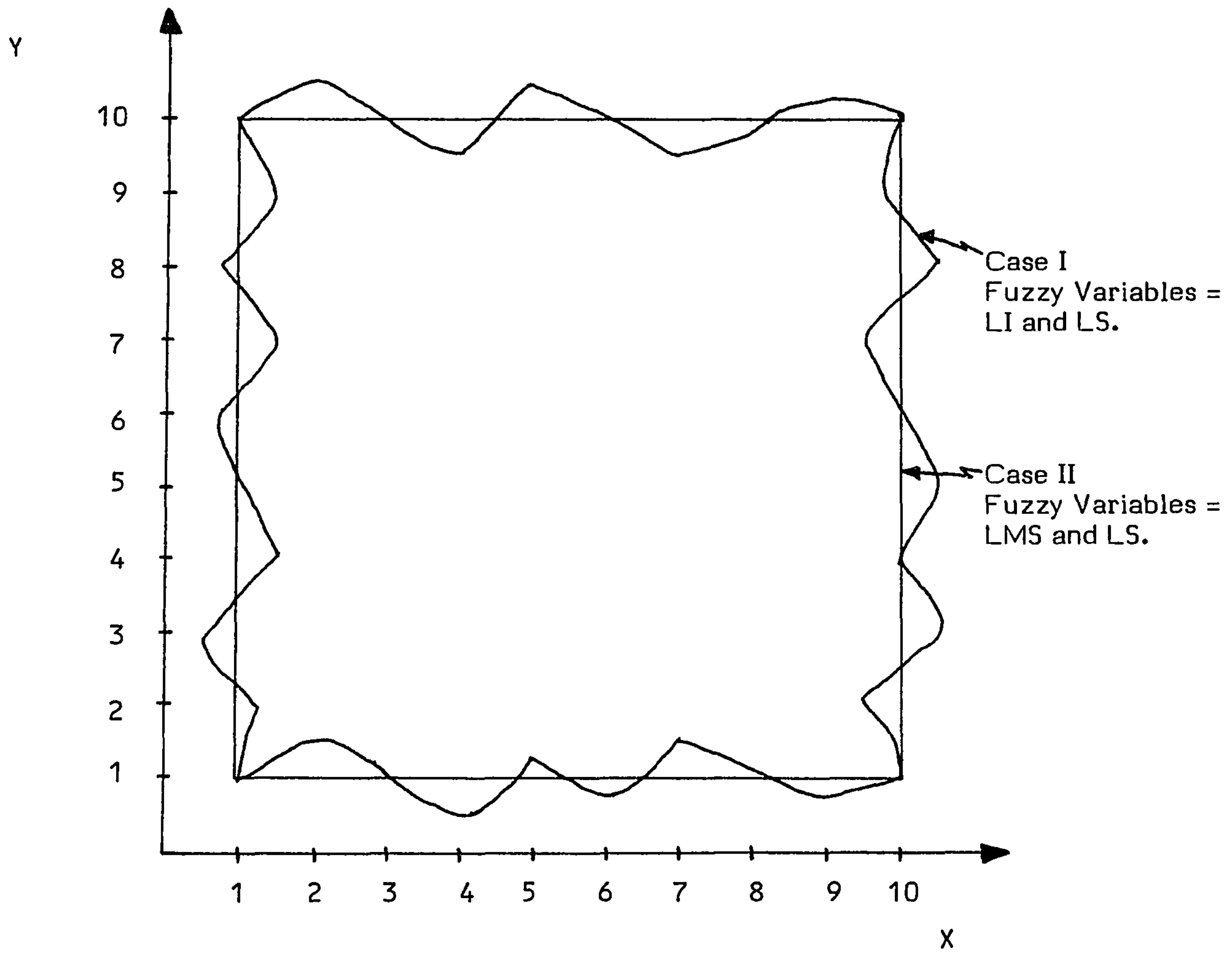
Figure C.6



Piecewise Approximation of Figure C.6 Using Fuzzy

Variables LI and LS.

Figure C.7



Detection of a Boundary Within a Noisy Signal

Figure C.8

APPENDIX D

NUMERICAL EXAMPLE

This appendix presents a detailed numerical example of interval merging where the computations and results of two different matrix methods are shown. The two methods which will be presented are

(a) a standard inverse matrix computation

and

(b) the short matrix method as presented in this paper.

These methods are concerned with computing A^{-1}. We start by defining the input data, x_i, as being equally spaced along a coordinate axis. For this example the number of x_is per interval will be five, however there is no limit on the number of points per interval other than from a computational limit. The other requirement is that the number of x_is per interval be held constant. In order to keep the example simple we will not transform the data onto the -1 to 1 space. Instead we will work in the natural number space. To futher simplify the problem we will only consider a contour/boundary which is parallel to the x-axis, i.e. a straight line. We will first present method (a) above in order to establish a basis from which comparisons can then be made. First we form the A matrix in which its elements are derived from the input data raised to the appropriate power and then summed. For the linear case, which is written in general terms, yields

$$A = \begin{bmatrix} \#\text{ of points} & x \\ x & x^2 \end{bmatrix} .$$

Next the inverse matrix of the above A matrix is formed. For the example problem, as shown in Figure D.1 and Table D.1, the A_1 and A_1^{-1} matrices are as follows

$$A_1 = \begin{bmatrix} 5 & 15 \\ 15 & 55 \end{bmatrix},$$

and

$$A_1^{-1} = \begin{bmatrix} 1.1 & -.3 \\ -.3 & .1 \end{bmatrix},$$

where the subscript indicates the interval number. Solving for the second interval in the same manner we have

$$A_2 = \begin{bmatrix} 5 & 35 \\ 35 & 255 \end{bmatrix},$$

and

$$A_2^{-1} = \begin{bmatrix} 5.1 & -.7 \\ -.7 & .1 \end{bmatrix}.$$

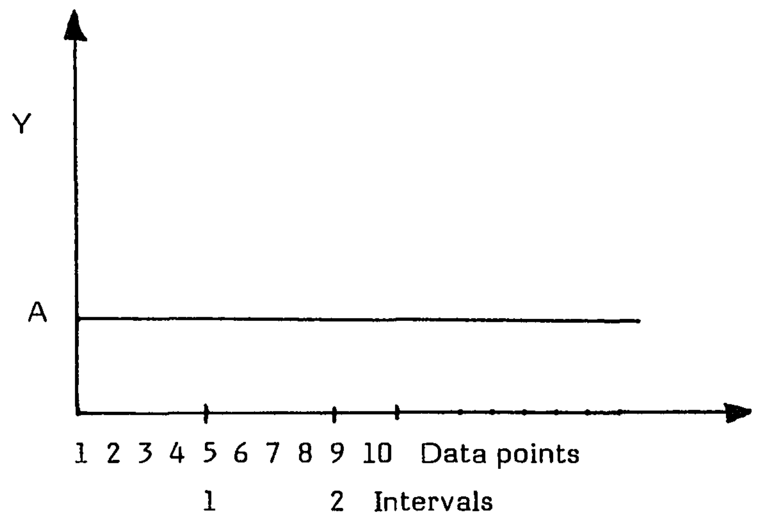
The fuzzy model when given the above data will make a decision that the first two intervals can be merged. Thus, the expression which must be found is A_3^{-1} or the matrix inverse of all the data in $I_1 \cup I_2$. By method (a) above the first step which must be accomplished is the formulation of the sum of A_1 and A_2. Since the matrices A_1 and A_2 already exist within the computer's memory, the sum can be found easily. The inverse matrix A_3^{-1} can be found by method (a) which gives

$$A_3 = \begin{bmatrix} 10 & 50 \\ 50 & 310 \end{bmatrix},$$

and

$$A_3^{-1} = \begin{bmatrix} .516 & -.083 \\ -.083 & .0166 \end{bmatrix}.$$

By method (b), A_3^{-1} can be computed by first forming $A_3 = A_1^{-1} + A_2^{-1}$. The inverse matrices A_1^{-1} and A_2^{-1} already exist because of the computations performed for intervals one and two. Next find the inverse matrix A_3^{-1} by using the multiplying coefficient as discussed in Chapter IV. This can be written as $A_3^{-1} = \delta_i A_3$ which for



Straight Contour

Figure D.1

Interval	1		2	
Matrix	A_1		A_2	
Input Data	x	x^2	x	x^2
	1	1	5	25
	2	4	6	36
	3	9	7	49
	4	16	8	64
	5	25	9	81
Total	15	55	35	255

Matrix Input Data For Intervals 1 and 2.

Table D.1

this example will give the following:

$$A_3 = \begin{bmatrix} 6.2 & -1.0 \\ -1.0 & .2 \end{bmatrix},$$

and

$$A_3^{-1} = \begin{bmatrix} .516 & -.083 \\ -.083 & .0166 \end{bmatrix},$$

where the multplying coefficient δ_i was set equal to .0833 as shown in Chapter V. As can be seen, method (b) is indeed shorter than method (a) because method (b) circumvents an inverse matrix computation. This becomes significant as the number of intervals which will be merged continues to grow.

The following discussion is a demonstration, using the above example, on solving for δ_i. As pointed out in Chapter V we can find δ_i as

$$\delta_i = A_{i+1}^{+} + A_{i-1}(I + A_i^{+} A_{i-1})^{-1} \qquad \text{D.1}$$

where i = 2,3, ...,n.

Solving for δ_i where i = 2 we get

$$\delta_1 = A_3^{+} A_1(I + A_2^{+} A_1)^{-1}$$

$$\delta_1 = \begin{bmatrix} 1.33 & 3.16 \\ -.16 & -.33 \end{bmatrix} \begin{bmatrix} -.33 & -3.16 \\ .16 & 1.33 \end{bmatrix}$$

$$\delta_1 = \begin{bmatrix} .083 & -.0001 \\ -.0001 & .083 \end{bmatrix}. \qquad \text{D.2}$$

Performing the check we have

$$A_3^{+} = (A_1^{+} + A_2^{+})$$

$$A_3^{+} = \begin{bmatrix} .083 & -.0001 \\ -.0001 & .083 \end{bmatrix} \begin{bmatrix} 6.2 & -1.0 \\ -1.0 & .2 \end{bmatrix}$$

$$A_3^{+} = \begin{bmatrix} .516 & -.083 \\ -.083 & .016 \end{bmatrix}. \qquad \text{D.3}$$

As can be seen equation D.3 does agree with the results as given above. The solutions given in equations D.2 and D.3 were performed in the data space [a,b]. This space works well if the input data set is small both in quantity and magnitude. The reason for this is that computer round-off error greatly affects the solution. To generalize this procedure, as discussed in Chapter V, we transform the input data onto the -1 to 1 interval space. The following discussion is on the data for the above problem but in the transformed space. Solving we get

$$A_1 = \begin{bmatrix} 5 & 0 \\ 0 & 2.5 \end{bmatrix} \text{ and } A_2 = \begin{bmatrix} 5 & 0 \\ 0 & 2.5 \end{bmatrix}$$

the resulting generalized inverses are

$$A_1^+ = \begin{bmatrix} .2 & 0 \\ 0 & .4 \end{bmatrix} \text{ and } A_2^+ = \begin{bmatrix} .4 & 0 \\ 0 & .4 \end{bmatrix}.$$

Solving for δ_1 using D.1 we get

$$\delta_1 = \begin{bmatrix} .11 & 0 \\ 0 & .27 \end{bmatrix} \left[\begin{bmatrix} 5 & 0 \\ 0 & 2.5 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} .2 & 0 \\ 0 & .4 \end{bmatrix} \begin{bmatrix} 5 & 0 \\ 0 & 2.5 \end{bmatrix} \right]^{-1}$$

$$\delta_1 = \begin{bmatrix} .55 & 0 \\ 0 & .66 \end{bmatrix} \begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}$$

$$\delta_1 = \begin{bmatrix} .27 & 0 \\ 0 & .33 \end{bmatrix}. \qquad \text{D.4}$$

The δ_1 multiplying factor as given in equation D.4 is the same as displayed in Table 5.1.1. The results from equation D.4 must then be retransformed onto the original data space which will yield D.2. By using the transformed data space we have avoided the round-off error as discussed above.

APPENDIX E

COMPUTER PROGRAM LISTING

In this appendix we present a computer generated listing of the algorithm developed for this study. The listing contains testing comment statements which were inserted during the initial development and testing phase. These comment statements have not been deleted since the completion of the algorithm. However, the executable code as given is the final code for the algorithm.

```
$BATCH
$TITL                                       LSQ
C***************************************************************
      COMMON
     X   /MAIN     /XIN(50),     YIN(50),     A(5,5),     X(5),
     X      N,       M,       EVAL,      ESQVAL,      Y(50),     DNORM,
     X   XINI(50), ,   YINI(50)
     X   /FUZZY     /SIG(50),     EMAX,     ICOUNL,     ICOUNQ,
     X   FUZLL,     FUZLH,     F1,     F2,     F3,
     X   FUZQL,     FUZQH
     X   /YOPT     /XX(5),     ISUML,     ISUMQ,     C3(2),
     X   INTL,     INTQ,     MM,     RHO1,     SNR,     RHO
     X   /SCALE     /C,     D,     IL
     X   /SOLIN     /E(5),     ISKIP,     FSOLIN(3)
     X   /OPTSOL    /MSTAR,     MFIN,     E2,     MFOPT,
     X   FYOPT(50,8),     NY
      DIMENSION IAR(4),IT(4)
      REAL NN,IY
      DATA NN,IY,IB,IBB/1HN,1HY,1H ,4H    /
      DATA LI,LS,LVI,LNVS/2HLI,2HLS,3HLVI,4HLNVS/
      DATA LMI,LMS,LNVI,LNMI/3HLMI,3HLMS,4HLNVI,4HLNMI/
C                READ INPUT DATA FROM EITHER THE CARD READER OR FROM WITHIN TH
C        PROGRAM.THE PROGRAM DATA IS EITHER A LINEAR CASE OR QUADRATIC CASE
C        OF KNOWN SOLUTIONS. THESE ARE THE TEST POINTS FOR THE ALGORITHMS.
C
C             IPASS IS USED TO INITIALIZE ICOUNT IN
C             FUZZY INORDER TO KEEP TRACK OF THE INTERVAL
C             WE ARE SOLVING I.E.(1 OR 2).
C
10    CONTINUE
         DO 2 I=1,2
2        C3(I) = 0.0
      ISUML = 0
      ISUMQ = 0
      ICOUNL = 0
      ICOUNQ = 0
      INTL = 0
      INTQ = 0
      ISEC1 = 0
      ISEC2 = 0
      ANS = IB
      MFOPT = 0
      POWER = 2.0
      CONK = .04938
      ALPHA = .5
      IANS1 = IIB
      IANS2 = IIB
      IANS3 = IIB
      WRITE(1,1050)
      READ(1,2000) IANS1
      WRITE(1,1060)
      READ(1,2000) IANS2
      WRITE(1,1070)
      READ(1,2000) IANS3
      DO 4 I=1,50
         DO 6 J=1,8
              FYOPT(I,J) = 0.0
6        CONTINUE
4     CONTINUE
      F5 = (5.0-ALPHA)**POWER
      F6 = CONK*F5
      F7 = 1.0+F6
      F1 = F6/F7
      F2 = F1
      F8 = (2.0-ALPHA)**POWER
      F9 = CONK*F8
      F10 = 1.0+F9
      F3 = F9/F10
      IF(IANS1 .NE. LI) GO TO 7
      IF(IANS2 .NE. LI) GO TO 7
      IF(IANS3 .NE. LS) GO TO 7
      GO TO 11
7     CONTINUE
      IF(IANS1 .NE. LVI) GO TO 8
      IF(IANS2 .NE. LVI) GO TO 8
      IF(IANS3 .NE. LNVS) GO TO 8
      POWER = 4.0
      F5 = (5.0-ALPHA)**POWER
      F6 = CONK*F5
      F7 = 1.0+F6
      F1 = F6/F7
      F2 = F1
      F8 = (2.0-ALPHA)**POWER
      F9 = CONK*F8
      F10 = 1.0+F9
      F3 = 1.0-(F9/F10)
      GO TO 11
8     CONTINUE
      IF(IANS1 .NE. LMI) GO TO 9
      IF(IANS2 .NE. LMI) GO TO 9
      IF(IANS3 .NE. LMS) GO TO 9
      POWER = 3.0
      F5 = (5.0-ALPHA)**POWER
      F6 = CONK*F5
      F7 = 1.0+F6
      F1 = F6/F7
      F2 = F1
      F8 = (2.0-ALPHA)**POWER
      F9 = CONK*F8
      F10 = 1.0+F9
      F3 = F9/F10
      GO TO 11
9     CONTINUE
      IF(IANS1 .NE. LNVI) GO TO 13
      IF(IANS2 .NE. LNVI) GO TO 13
      IF(IANS3 .NE. LS) GO TO 13
      POWER = 4.0
      F5 = (5.0-ALPHA)**POWER
      F6 = CONK*F5
      F7 = 1.0+F6
      F11 = F6/F7
      F1 = 1.0-F11
      F2 = F1
      GO TO 11
13    CONTINUE
      IF(IANS1 .NE. LNMI) GO TO 11
      IF(IANS2 .NE. LNMI) GO TO 11
      IF(IANS3 .NE. LS) GO TO 11
      POWER = 3.0
      F5 = (5.0-ALPHA)**POWER
      F6 = CONK*F5
      F7 = 1.0+F6
      F11 = F6/F7
      F1 = 1.0-F11
      F2 = F1
11    CONTINUE
```

```
      WRITE(6,3000) F1,F2,F3
C
C               THIS FLAG IS USED TO SKIP RESCALE WHEN COMPUTING
C               YOPT.
C
      ISKIP = 0
C
C        THESE INDICATE THE ORDER OF THE POLY.
C
      N = 0
C
C        M = NUMBER OF POINTS
C
      M = 5
C
C             INITIALIZE THE GLOBEL VARIABLES
C
C
C             MM = A VARIABLE USED TO DETERMINE WHEN WE ARE FINISHED
C
      MM = 5
C
C             IQ= NUMBER TO INDICATE IF WE ARE COMPUTING LINEAR OR
C             QUADRATIC COEFFICIENTS INITIALLY SET = 0.
C
      IQ = 0
C
C             E2 = RMS VALUE PER INTERVAL INITIALLY = 10.
C
      E2 = 10.0
C
C             EMAX =ERROR VALUE OVER THE ENTIRE INTERVAL (A,B)
C             INITIALLY = 100.0.
C
      EMAX = 100.0
C*****************************************************************
C
C             SET SIGNAL TO NOISE RATIO (SNR)= 1.0
C
C*****************************************************************
      SNR = 1.0
C
C             CALL DATE
C
      CALL DATE(IAR)
         DA1 = IAR(1)
         DA2 = IAR(2)
         DA3 = IAR(3)
C
C             THIS IS TO HELP FINISH AN INTERVAL.
C
      L = 0
      ICOUNT = 5
C
C        READ DATA FROM CARD READER
C
      READ(4,530) E2,EMAX
      READ(4,650) FUZLL,FUZLH,FUZQL,FUZQH
      READ(4,660) DIF
      READ (4,520) IIL
      DIF1 = 1.0*DIF
      DIF2 = 2.0*DIF
      DIF4 = 4.0*DIF
      DIF5 = 5.0*DIF
      IF(IIL .GE. 50) GO TO 12
         IL = IIL
         IL1 = IIL
      GO TO 14
12    CONTINUE
         IL = 50
         IL1 = 50
         IL3 = 50
14    CONTINUE
      READ(4,530) (XIN(I),YIN(I),I=1,IL)
      WRITE(6,590)
      WRITE(6,610)DA2,DA3,DA1
      WRITE(6,580)(I,XIN(I),YIN(I),I=1,IL)
      WRITE(6,600) E2,EMAX
      WRITE(6,640) FUZLL,FUZLH,FUZQL,FUZQH
      EVAL = E2
      CALL TIME(IT)
      ISEC1 = 3600*IT(1)+60*IT(2)+IT(3)
C************************************************************************
C
C             BUILD XINI AND YINI ARRAYS.
C
C*************************************************************************
      DO 15 I=1,IL
      XINI(I) = XIN(I)
15    YINI(I) = YIN(I)
      XSUM = 0.0
      DELTX = 0.0
      DO 20 I=2,M
         DELTX = ABS(XIN(I)-XIN(I-1))
         XSUM = XSUM+DELTX
20    CONTINUE
      IF(XSUM .GT. DIF4) GO TO 950
      IF(XSUM .GE. DIF2) GO TO 109
      GO TO 107
 40   CONTINUE
      CALL SCALE
C
C             SCALE INPUT DATA TO THE INTERVAL (-1,1). THIS IS TO
C             AVOID ROUND OFF ERRORS.
C
C
C             CALL LINEAR SOLUTION
C
      CALL LIN
C
C             IF E2<DELTA W HAVE FOUND A LINEAR FIT FOR THE
C             GIVEN INTERVAL--GO AND READ MORE DATA.
C
      IF(DNORM .GE. E2) GO TO 55
C
C             SET SOME CONSTANTS AND CALL FUZZY ROUTINE
C
      N = 1
C***********************************************************************
C
C             BUILD FYOPT
C
C*************************************************************************
      IF(M .EQ. 3) GO TO 46
      ITAR = MH - 4
      GO TO 47
 46     CONTINUE
        ITAR = MM - 2
 47     CONTINUE
        MFOPT = MFOPT + 1
        FYOPT(MFOPT,1) = ITAR
        FYOPT(MFOPT,2) = MM
        FYOPT(MFOPT,3) = NY
        FYOPT(MFOPT,4) = FSOLIN(1)
        FYOPT(MFOPT,5) = FSOLIN(2)
        FYOPT(MFOPT,6) = 0.0
        FYOPT(MFOPT,7) = EMAX
        FYOPT(MFOPT,8) = DNORM
        CALL FUZZY
        GO TO 60
 55     CONTINUE
        CALL SCALE
C
C               CALL QUAD SOLUTION
C
        CALL QUAD
C
C               IF E2<DELTA WE HAVE FOUND A QUADRATIC FIT FOR THE
C               GIVEN INTERVAL---GO AND READ MORE DATA.
C
        IF(DNORM .GE. E2) GO TO 62
C
C               SET SOME CONSTANTS AND CALL FUZZY ROUTINE
C
        N = 2
C*******************************************************************
C
C               BUILD Q FYOPT
C
C********************************************************************
        IF(M .EQ. 3) GO TO 56
        ITAR = MM-4
        GO TO 57
 56     CONTINUE
        ITAR = MM-2
 57     CONTINUE
        MFOPT = MFOPT + 1
        FYOPT(MFOPT,1) = ITAR
        FYOPT(MFOPT,2) = MM
        FYOPT(MFOPT,3) = NY
        FYOPT(MFOPT,4) = FSOLIN(1)
        FYOPT(MFOPT,5) = FSOLIN(2)
        FYOPT(MFOPT,6) = FSOLIN(3)
        FYOPT(MFOPT,7) = EMAX
        FYOPT(MFOPT,8) = DNORM
        CALL FUZZY
        GO TO 60
 62     CONTINUE
C
C                SPLIT INTERVAL IF E2> DELTA.
C
        M = IFIX((M/2)+1.0)
        MM = MM-2
        GO TO 40
C
C               REARRANGE DATA WITHIN INTERVAL, THAT IS RENUMBER THE
C               INPUT POINTS INDEXES.
C
C
```

```
112    MM = MM+1
       MM = MM -1
C*********************************************************************
C
C              TEST TO SEE IF Y DATA IS PERPENDICULAR TO X
C              AXIS.
C
C*************************************************************************
       XSUM = 0.0
       DELTX = 0.0
       DO 108 I= 2,M
          DELTX = ABS(XIN(I)-XIN(I-1))
          XSUM = XSUM + DELTX
108    CONTINUE
       IF(XSUM .GE. DIF5) GO TO 950
       IF(XSUM .GE. DIF2) GO TO 109
       IF(XSUM .EQ. DIF1) GO TO 107
       IF(XSUM .NE. 0.0) GO TO 130
107    CONTINUE
C***********************************************************************
C
C              REARRANGE THE INPUT DATA AND COMPUTE X = AY + B
C
C*********************************************************************
       MM = MM-4
       KK = 5
       J = 0
       DO 111 I= 1,KK
          J = J+1
          YIN(J) = XINI(MM)
          XIN(J) = YINI(MM)
111    MM = MM+1
       MM = MM-1
       IF( NY .EQ. 1) GO TO 40
          NY = 1
          ISUML = 0
          ISUMQ = 0
       GO TO 40
109    CONTINUE
       IF( NY .EQ. 0) GO TO 40
          NY = 0
          ISUML = 0
          ISUMQ = 0
       GO TO 40
114    CONTINUE
       I = (IL-5)+1
       MM = IL
       DO 116 J=1,5
       YIN(J) = YINI(I)
       XIN(J) = XINI(I)
       I = I+1
116    CONTINUE
C**********************************************************************
C
C              TEST DATA TO DETERMINE IF Y IS PERPENDICULA TO
C              X AXIS.
C
C***********************************************************************
       XSUM = 0.0
       DELTX = 0.0
       DO 115 J=2,M
          DELTX = ABS(XIN(J) - XIN(J-1))
          XSUM = XSUM + DELTX
```

```
C             HOW MANY POINTS MUST BE ADDED TO FILL THE BUFFER OR
C             ARRAY TO 5 FULL POINTS?
C
C
60      CONTINUE
        IF(M .EQ. 5) GO TO 100
        IF((MM .GT. IL) .AND. (DNORM .LT. E2))GO TO 120
        IF(MM .GE. IL) GO TO 120
        IF(M .EQ. 3) GO TO 80
        JJ = 2
        LL = 4
         GO TO 85
80      CONTINUE
        JJ = 3
        LL = 3
85      CONTINUE
        DO 86 I = 1,LL
           XIN(I) =XIN(JJ)
           YIN(I) =YIN(JJ)
           JJ = JJ+1
86      CONTINUE
        LL = LL+1
           DO 88 I=LL,5
                MM = MM+1
                XIN(I) = XINI(MM)
                YIN(I) = YINI(MM)
88         CONTINUE
        IF(MM .GE. IL) GO TO 90
        M = 5
        GO TO 40
90      CONTINUE
        IX1 = MM-IL
        IF(IX1 .EQ. 0) GO TO 92
        M = 5-IX1
        IF(M .EQ. 0) GO TO 120
        GO TO 40
92      CONTINUE
        MM = IL
        IF(M .EQ. 0) GO TO 120
        GO TO 40
100     CONTINUE
C
C             TEST TO DETERMINE IF WE HAVE FINISHED ALL THE DATA.
C
C
        IF(MM .EQ. IL) GO TO 120
        K1 = MM
        KK = 5
        J = 0
        MM = MM+5
C
C             TEST TO SEE IF WE EXCEEDED THE INITIAL INPUT DATA
C             (IL).
C
        IF(MM .GT. IL) GO TO 114
        MM = MM-5
C
C             READ THE NEXT BLOCK OF 5 DATA POINTS.
C
        DO 112 I =1,KK
        J = J+1
        YIN(J) = YINI(MM)
        XIN(J) = XINI(MM)
```

```
 115   CONTINUE
       IF( XSUM .EQ. DIF4) GO TO 119
       IF(XSUM .GE. DIF2) GO TO 109
       IF(XSUM .EQ. DIF1) GO TO 107
       IF(XSUM .NE. 0.0) GO TO 130
C***********************************************************************
C
C             NO--REARRANGE INPUT DATA.
C
C*************************************************************************
       I = (IL-5)+1
       DO 118 J=1,5
          YIN(J) = XINI(I)
          XIN(J) = YINI(I)
          I = I+1
 118   CONTINUE
       IF(NY .EQ. 1) GO TO 117
          NY = 1
          ISUML = 0
          ISUMQ = 0
 117   CONTINUE
       N = 5
       MM = IL
       GO TO 40
 119   CONTINUE
       IF(NY .EQ. 0) GO TO 117
          NY = 0
          ISUML = 0
          ISUMQ = 0
       GO TO 117
C**********************************************
C
C             FOR XSUM = 3.0.
C
C************************************************
 150   CONTINUE
       IF(NY .EQ. 1) GO TO 152
       MM = MM-1
       I= MM-4
       GO TO 160
 152   CONTINUE
       MM = MM-3
       I = MM-4
       GO TO 140
C***************************************************************
C
C             FINISH COMPUTING THE EQUATION FOR THE
C             NUMBER OF POINTS LEFT OVER.
C
C***************************************************************
 130   CONTINUE
       IF((XSUM .GT. 0.0) .AND.
      X (XSUM .LT. DIF1)) GO TO 960
       IXSUM = IFIX(XSUM)
       MM = (MM-4)+IXSUM
C*****************************************************************
C
C         BACK UP SOME NUMBER OF PONITS IN XIN AND YIN.
C
C*****************************************************************
       I = MM-4
       IF( NY .EQ. 1) GO TO 140
 160   CONTINUE
```

```
      DO 132 J = 1,5
         XIN(J) = XINI(I)
         YIN(J) = YINI(I)
         I = I+1
132   CONTINUE
         GO TO 40
140   CONTINUE
      DO 142 J = 1,5
         XIN(J) = YINI(I)
         YIN(J) = XINI(I)
         I = I+1
142   CONTINUE
         GO TO 40
120   CONTINUE
      ANS = IB
      CALL TIME(IT)
      ISEC2 = 3600*IT(1)+60*IT(2)+IT(3)
      ISEC = ISEC2-ISEC1
         WRITE(6,630) ISEC
630      FORMAT(20X,'SEC =',F10.5)
      WRITE(6,590)
      WRITE(6,620)((FYOPT(I,J),J=1,8),I=1,MFOPT)
      WRITE(6,590)
      CALL PLOT
C**********************************************************
      MFOPT = 0
      ISUML = 0
      ISUMQ = 0
      ICOUNL = 0
      ICOUNQ = 0
      INTL = 0
      INTQ = 0
      ICOUNT = 5
      L = 0
      ISKIP = 0
      N = 0
      M = 5
      MM = 5
      DO 134 I = 1,2
134   C3(I) = 0.0
      DO 136 I=1,50
         DO 138 J=1,8
            FYOPT(I,J) = 0.0
138      CONTINUE
136   CONTINUE
C
C             TEST TO SEE IF ALL INPUT DATA HAS
C             BEEN READ.
C
C**********************************************************
      IL1 = IIL - IL1
      IF(IL1) 122,122,124
124   CONTINUE
      IL2 = IL1-50
      IF(IL2) 126,126,128
128   CONTINUE
      IL = 50
      IL3 = IL3+50
      IL1 = IL3
      GO TO 14
126   CONTINUE
      IL = IL1
      IL1 = IL3 + IL1
```

```
       GO TO 14
  122  CONTINUE
       WRITE(1,570)
       READ(1,510) ANS
          IF(ANS .EQ. IY) GO TO 10
       GO TO 1000
  950  CONTINUE
       WRITE(6,920)
  920  FORMAT(10X'*******INPUT DATA NOT ONE
      X  UNIT APART******')
       GO TO 1000
  960  CONTINUE
       WRITE(6,930)
  930  FORMAT(10X,'********INPUT DATA DIFFERENCE
      X  BETWEEN 0.0 AND 1.0*******')
       GO TO 1000
  620  FORMAT(10X,'MSTAR =',I2,5X,'MFIN =',I2,5X,'NY =',I1,5X,
      X'E1 =',E16.8,5X,'E2 =',E16.8,5X,'E3 =',E16.8,/,20X,
      X'EMAX =',E16.8,5X,'DNORM =',E16.8,/,20X,
      X'*******************************************')
  500  FORMAT(35HREAD DATA FROM CARD READER (Y OR N))
  510  FORMAT(A1)
  520  FORMAT(I3)
  530  FORMAT(F6.2,F6.2)
  550  FORMAT(38HDO YOU WANT A LINEAR SOLUTION (Y OR N))
  560  FORMAT(27HDO YOU WANT A PLOT (Y OR N))
  570  FORMAT(44HDO YOU WANT TO RUN ANOTHER SOLUTION (Y OR N))
  580  FORMAT(10X,I3,5X,F6.2,5X,F6.2)
  590  FORMAT(1H1)
  600  FORMAT(10X,4HE2 =,F6.2,5X,6HEMAX =,F6.2)
  610  FORMAT(62X,7HLSQUARE,31X,I2,1H/,I2,1H/,I2,/,/)
  650  FORMAT(F5.3,F5.3,F5.3,F5.3)
  640  FORMAT(10X,'FUZLL =',F5.3,10X,'FUZLH =',F5.3,5X,
      X'FUZQL =',F5.3,5X,'FUZQH =',F5.3)
  660  FORMAT(F5.2)
 1050  FORMAT('IMPORTANCE OF L2 =')
 1060  FORMAT('IMPORTANCE OF SLOPE =')
 1070  FORMAT('IMPORTANCE OF LMAX =')
 2000  FORMAT(A4)
 3000  FORMAT(10X,'L2 =',F10.5,5X,'SLOPE =',F10.5,5X,
      X'LMAX =',F10.5)
 1000  CONTINUE
       STOP
       END
$TITL                             SUMS/LSQ
C**************************************************************
       SUBROUTINE SUMS
C
C         THIS ROUTINE WILL FORM THE SUMS OF THE VARIOUS TERMS
C         WHICH WILL BE REQUIRED LATER ON.
C
       COMMON
      X   /MAIN     /XIN(50),     YIN(50),     A(5,5),     X(5),
      X   N,      M,      EVAL,     ESQVAL,     Y(50),     DNORM,
      X   XINI(50),    YINI(50)
      X   /SUMS     /YY,      XY,      XSQY,      AY(5,5),
      X   XYY,    XXSQY
      X     /SCALE     /C,      D,      IL
C
       DIMENSION XI(50)
C         FIRST TEST TO DETERMINE IF WE ARE GOING TO USE LINEAR
C          OR QUADRATIC APPROXIMATIONS.
C
```

```
C
C          INITIALIZE SOME VARIABLES
C
          XX = 0
          YY = 0
          SQX = 0
          XY = 0
          XCUBI = 0
          XFOUR = 0
          XSQY = 0
          IF(N .EQ. 2) GO TO 20
          DO 2 I =1,5
             DO 4 J =1,5
   4         A(I,J) = 0.0
   2      CONTINUE
C
C            SUMS FOR LINEAR CASE
C
          DO 10 I=1,M
             XX = XX + XIN(I)
             SQX = SQX + (XIN(I))**2
             YY = YY + YIN(I)
             XY = XY + (XIN(I)*YIN(I))
  10      CONTINUE
          A(1,1) = M
          A(1,2) = XX
          A(2,1) = XX
          A(2,2) = SQX
C
C                       SCALE FOR YOPT SOLUTION
C
          XYY = 0
          XX = 0
          SQX = 0
          DO 15 I=1,M
             XI(I) = ((XIN(I)*D)/2.0)+(C/2.0)
             XX = XX+XI(I)
             SQX = SQX+(XI(I))**2
             XYY = XYY+(XI(I)*YIN(I))
  15      CONTINUE
          AY(1,1) = M
          AY(1,2) = XX
          AY(2,1) = XX
          AY(2,2) = SQX
          GO TO 40
  20      CONTINUE
C
C            SUMS FOR QUADRATIC
C
          DO 30 I=1,M
             XX = XX+XIN(I)
             SQX = SQX+(XIN(I))**2
             XCUBI = XCUBI+(XIN(I))**3
             XFOUR = XFOUR+(XIN(I))**4
             XY = XY+(XIN(I)*YIN(I))
             YY = YY+YIN(I)
             XSQY = XSQY+(((XIN(I))**2)*YIN(I))
  30      CONTINUE
          A(1,1) = M
          A(1,2) = XX
          A(1,3) = SQX
          A(2,1) = XX
          A(2,2) = SQX
```

```
      A(2,3) = XCUBI
      A(3,1) = SQX
      A(3,2) = XCUBI
      A(3,3) = XFOUR
C
C                 SCALE FOR YOPT SOLUTION
C
         XXSQY = 0
         XYY = 0
         XX = 0
         SQX = 0
         XXC = 0
         XXF = 0
C
C                 FORM COEFFICIENTS
C
              DO 35 I =1,M
                   XI(I) = ((XIN(I)*D)/2.0)+(C/2.0)
                        XX= XX+XI(I)
                        SQX = SQX + (XI(I))**2
                        XYY = XYY+(XI(I)*YIN(I))
                        XXC = XXC+(XI(I))**3
                        XXF = XXF+(XI(I))**4
                        XXSQY = XXSQY+(((XI(I))**2)*YIN(I))
35            CONTINUE
         AY(1,1) =M
         AY(1,2) = XX
         AY(1,3) = SQX
         AY(2,1) = XX
         AY(2,2) = SQX
         AY(2,3) = XXC
         AY(3,1) = SQX
         AY(3,2) = XXC
         AY(3,3) = XXF
 40   CONTINUE
      RETURN
      END
$TITL                                 GENINV/LSQ
C*************************************************************
      SUBROUTINE GENINV
      COMMON
     X   /MAIN     /XIN(50),     YIN(50),     A(5,5),     X(5),
     X   N,    M,     EVAL,      ESQVAL,     Y(50),      DNORM,
     X   XINI(50),    YINI(50)
     X   /GENINV     /YG(10,10)
      DIMENSION YZ(10,10),AT(10,10),YY(10,10),
     X YI(10,10),Y2(10,10),Y1(10,10)
C
C        FIRST FORM THE INVERSE AFTER INITIALIZING
C
C
C        FIRST TEST TO DETERMINE IF WE ARE COMPUTINE THE LINEAR
C        SOLUTION OR  QUADRATIC.
      IF( N .EQ. 2) GO TO 10
      K = 2
      GO TO 20
 10   CONTINUE
      K = 3
 20   CONTINUE
C
C        INITIALIZE YZ MATRIX TO ZERO
C
      DO 30 I=1,10
```

```
             DO 40 J=1,10
   40       . YZ(I,J) = 0.0
   30     CONTINUE
             GO TO 700
C
C            COMPUTE THE TRANSPOSE XT
C
          DO 50 I=1,K
             DO 60 J=1,K
   60        AT(J,I) = A(I,J)
   50     CONTINUE
C
C            FORM YZ = (AT)*A
C
          DO 70 II=1,K
             DO 80 I = 1,K
               AA = 0.0
                  DO 90 J=1,K
                     AA= AT(I,J)*A(J,II)
   90             YZ(I,II) = YZ(I,II)+AA
   80        CONTINUE
   70     CONTINUE
  700     CONTINUE
          DO 710 I=1,K
             DO 720 J=1,K
             YZ(I,J) = A(I,J)
  720        CONTINUE
  710     CONTINUE
C
C            ADD UNIT MATRIX TO YZ
C
          DO 100 I=1,K
             DO 110 J=1,K
                  YY(I,J) = 0.0
                  YY(I,I) = 1.0
  110        CONTINUE
  100     CONTINUE
          DO 120 I =1,K
             KK = K
                  DO 130 J=1,K
                    KK=KK+1
  130           YZ(I,KK) = YY(I,J)
  120     CONTINUE
C            MATRIX INVERSION AND GENERALIZE INVERSE
C            ROUTINE.
C
C            FIRST TEST TO DETERMINE NOW MANY PASSES
C            WE MUST MAKE THROUGH THIS SECTION.
          IF(N .EQ. 2) GO TO 200
C
C            THIS NEXT SECTION WILL SOLVE FOR
C            THE LINEAR CASE.
C
          KK = 1
          I = 1
C
C            START FIRST PASS
C
          DO 140 J=1,4
  140     YI(I,J) = YZ(I,J) / YZ(KK,KK)
          I = 2
          DO 150 J=1,4
  150     YI(I,J) = YZ(I,J) - (YZ(I,KK)*YI(KK,J))
```

```
C
C
C         START SECOND PASS
C
       KK = 2
       I = 2
       DO 160 J=1,4
 160   Y2(I,J) = YI(I,J)/YI(KK,KK)
       I = 1
       DO 170 J=1,4
 170   Y2(I,J) = YI(I,J)-(YI(I,KK)*Y2(KK,J))
       GO TO 300
 200   CONTINUE
C
C         THIS SECTION WILL SOLVE FOR THE QUADRATIC
C         CASE AS EVIDENT BY THE THREE PASSES.
C
       KK = 1
       I = 1
C
C         START FIRST PASS
C
       DO 210 J=1,6
 210   YI(I,J) = YZ(I,J)/YZ(KK,KK)
       DO 220 I=2,3
          DO 230 J=1,6
 230      YI(I,J) = YZ(I,J)-(YZ(I,KK)*YI(KK,J))
 220   CONTINUE
C
C
C         START SECOND PASS
C
       KK = 2
       I = 2
       DO 240 J=1,6
 240   Y1(I,J) = YI(I,J)/YI(KK,KK)
       DO 250 I=1,3,2
          DO 260 J=1,6
 260      Y1(I,J) = YI(I,J)-(YI(I,KK)*Y1(KK,J))
 250   CONTINUE
C
C
C         START THIRD PASS
C
       KK = 3
       I = 3
       DO 270 J=1,6
 270   Y2(I,J) = Y1(I,J)/Y1(KK,KK)
       DO 280 I=1,2
          DO 290 J=1,6
 290      Y2(I,J) = Y1(I,J)-(Y1(I,KK)*Y2(KK,J))
 280   CONTINUE
 300   CONTINUE
C
C         COMPUTE THE GENERALIZED INVERSE
C
C
C         FIRST SET THE MATRIX YG = 0
C
       DO 310 I=1,10
          DO 320 J=1,10
 320      YG(I,J) = 0.0
 310   CONTINUE
```

```
C
C        TEST TO DETERMINE WHICH SOLUTION IS DESIRED
C        I.E. LINEAR(N=1) OR QUADRATIC(N=2).
C
      IF(N .EQ. 2) GO TO 330
      LL = 2
      K = 2
      GO TO 340
330   CONTINUE
      LL = 3
        K = 3
340   CONTINUE
C
C        NOW COMPUTE THE INVERSE
C
      DO 350 I=1,K
      L= LL
         DO 360 J=1,K
         L = L+1
         YG(I,J) = Y2(I,L)
360   CONTINUE
350   CONTINUE
      RETURN
      END
$TITL                                SOLIN/LSQ
C*******************************************************************
      SUBROUTINE SOLIN
C
C        COMPUTE THE LINEAR SOLUTION AND ERRORS.
C
      COMMON
     X   /MAIN     /XIN(50),      YIN(50),      A(5,5),      X(5),
     X   N,      M,     EVAL,     ESQVAL,      Y(50),      DNORM,
     X   XINI(50),    YINI(50)
     X   /SUMS     /YY,      XY,      XSQY,      AY(5,5),
     X   XYY,      XXSQY
     X   /GENINV     /YG(10,10)
     X     /SOLIN     /E(5),      ISKIP,    FSOLIN(3)
     X     /FUZZY     /SIG(50),      EMAX,      ICOUNL,      ICOUNQ,
     X   FUZLL,      FUZLH,      F1,      F2,      F3,
     X   FUZQL,      FUZQH
      DIMENSION BB(5)
C
C        FIRST INILIZE X(I) TO ZERO
C
      DO 10 I =1,5
      E(I) = 0.0
10    X(I) = 0.0
C
C        TEST TO DETERMINE IF N = 1 OR 2.
C
      IF(N .EQ. 2) GO TO 15
C
C        THEN BUILD THE B MATRIX
C
      BB(1) = YY
      BB(2) = XY
      L = 2
      GO TO 18
15    CONTINUE
C
C        BUILD THE B MATRIX FOR QUADRATIC.
C
```

```
      BB(1) = YY
      BB(2) = XY
      BB(3) = XSQY
      L = 3
18    CONTINUE
C
C        FIND COEFFICIENTS TO LINEAR EQUATION I.E. K1
C        AND K2.
C
      DO 20 I =1,L
      AA = 0.0
         DO 30 J=1,L
              AA= YG(I,J)*BB(J)
30       E(I) = E(I) + AA
20    CONTINUE
      IF(N .EQ. 2) GO TO 45
22       CONTINUE
C
C                   TEST TO SEE IF WE SHOULD SKIP RESCALE
C                   BECAUSE THIS IS FOR YOPTL.
C
C
C
C              RESCALE INTO ORIGINAL VALUES.
C
      CALL RESCAL
C
C        FORM LINEAR EQUATION ( Y= K(1) +K(2)*XIN(I))
C        AT THE GIVEN X'S
C
C***********************************************************************
C
C        BUILD LINEAR MATRIX OF COEFFICIENTS.
C
C************************************************************************
      FSOLIN(1) = E(1)
      FSOLIN(2) = E(2)
      DO 40 I=1,M
40    Y(I) = E(1) +E(2)*XIN(I)
C
C        NOW FORM THE ERROR TERM E BETWEEN Y(I)
C        AND YIN(I). THAT IS E=SUM(YIN(I)-Y).
C
      GO TO 55
45    CONTINUE
C              RESCALE INTO ORGINIAL VALUES.
C
      CALL RESCAL
C***********************************************************************
C
C        BUILD QUADRATIC MATRIX OF COEFFICIENTS.
C
C************************************************************************
      FSOLIN(1) =.E(1)
      FSOLIN(2) = E(2)
      FSOLIN(3) = E(3)
C        FORM QUADRATIC EQUATION AT THE GIVEN X'S.
C
      DO 48 I=1,M
48    Y(I)=E(1)+(E(2)*XIN(I))+(E(3)*(XIN(I)**2))
55    CONTINUE
      EE = 0.0
      AA = 0.0
      DNORM = 0.0
      DNORM1 = 0.0
      E2D = 0.0
         X4 = 0.0
         X1 = 0.0
      IF(N .EQ. 2) GO TO 54
      GAM = ATAN(E(2))
54    CONTINUE
      DO 50 I=1,M
         AA = YIN(I) - Y(I)
C
C        SIG(I) WILL BE ASSIGNED THE DIFFERENCE BETWEEN
C        YIN AND Y COMPUTED. THIS IS USED IN THE FUZZY
C        ROUTINE.
C
C
      SIG(I) = AA
C
C               COMPUTE THE PERPENDICULAR DISTANCE BETWEEN
C               YIN AND Y.
C
C
      IF(AA .LT. 0.0) GO TO 56
C
C               FOR VALUES OF YIN ABOVE Y.
C
      IF(N .EQ. 2) GO TO 58
      E2D = (AA*SIN(90.0-GAM))**2
      GO TO 64
58    CONTINUE
      AM = (2.0*E(3))*XIN(I)
      GAM = 90.0-(ATAN(AM))
      E2D = (AA*SIN(GAM))**2
      GO TO 64
56    CONTINUE
C
C               FOR VALUES OF YIN BELOW Y.
C
      IF(N .EQ. 2) GO TO 62
      E2D = (AA*SIN(GAM))**2
      GO TO 64
62    CONTINUE
      AM = (2.0*E(3))*XIN(I)
      GAM = ATAN(AM)
      E2D = (AA*SIN(GAM))**2
64    CONTINUE
      DNORM1 = DNORM1 + E2D
50    EE = EE+AA
         DNORM = SQRT(DNORM1)
C
C        COMPUTE SUM E SQUARE ERROR
C
      ESQ = 0.0
      AA = 0.0
      DO 60 I=1,M
         AA = (YIN(I)-Y(I))**2
60    ESQ = ESQ + AA
      ESQVAL = SQRT(ESQ)
C********************::*********************************************************
C               FIND EMAX FROM DELTA-Y.
C***************************************************************
      EMAX = ABS(SIG(1))
      DO 70 I=2,M
```

```
         EMAX = AMAX1(EMAX,ABS(SIG(I)))
 70     CONTINUE
 90     CONTINUE
C
C
C
C                    CALL FUZZY LOGIC ROUTINE
C
        RETURN
        END
STITL                                   RESCAL/LSO
C*********************************************************************
        SUBROUTINE RESCAL
        COMMON
      X   /MAIN     /XIN(50),      YIN(50),     A(5,5),      X(5),
      X   N,      M,      EVAL,      ESQVAL,     Y(50),      DNORM,
      X   XINI(50),     YINI(50)
      X   /SOLIN      /E(5),      ISKIP,     FSOLIN(3)
      X   /SCALE      /C,      D,      IL
        DO 10 I =1,M
 10     XIN(I) = ((XIN(I)*D)/2.0)+(C/2.0)
        IF(N .EQ. 2) GO TO 20
        E(1) = E(1) - (C/D)*E(2)
        E(2) = (2.0*E(2))/D
        GO TO 30
 20     CONTINUE
        E(1) = E(1)-((C/D)*E(2))+(((C/D)**2)*E(3))
        E(2) = ((2.0*E(2))/D)-(4.0*C*E(3))/(D**2)
        E(3) = (4.0*E(3))/(D**2)
 30     CONTINUE
        RETURN
        END
STITL                                   SCALE/LSO
C*********************************************************************
        SUBROUTINE SCALE
C
C               THIS ROUTINE SCALES THE INPUT DATA TO THE INTERVAL
C               (-1,1). THIS IS TO AVOID ROUND OFF ERRORS.
C
        COMMON
      X   /MAIN     /XIN(50),      YIN(50),     A(5,5),      X(5),
      X   N,      M,      EVAL,      ESQVAL,     Y(50),      DNORM,
      X   XINI(50),     YINI(50)
      X   /SCALE      /C,      D,      IL
        AA = XIN(1)
        B = XIN(1)
C
C               TEST FOR MIN AND MAX VALUES.
C
        DO 10 I=1,M
        IF(XIN(I) .LE. AA) GO TO 15
        GO TO 20
 15     AA = XIN(I)
 20     CONTINUE
        IF(B .GE. XIN(I)) GO TO 10
        B = XIN(I)
 10     CONTINUE
        C = B+AA
        D = B-AA
C
C               NOW COMPUTE THE NEW SCALED VALUES.
C
        DO 30 I =1,M
```

```
30    XIN(I) = ((2.0*XIN(I))-C)/D
      RETURN
      END
$TITL                                   QUAD/LSQ
C************************************************************
      SUBROUTINE QUAD
C
C             THIS ROUTINE WILL COMPUTE THE QUADRATIC CURVE FIT
C             FOR THE GIVEN DATA.
C
      COMMON
     X   /MAIN     /XIN(50),      YIN(50),      A(5,5),      X(5),
     X   N,      M,      EVAL,      ESGVAL,      Y(50),      DNORM,
     X   XINI(50),     YINI(50)
      N = 2
      CALL SUMS
      CALL GENINV
      CALL SOLIN
      RETURN
      END
$TITL                                   LIN/LSQ
C************************************************************
      SUBROUTINE LIN
C
C             THIS ROUTINE WILL COMPUTE THE LINEAR CURVE FIT FOR
C             THE GIVEN DATA.
C
      COMMON
     X   /MAIN     /XIN(50),      YIN(50),      A(5,5),      X(5),
     X   N,      M,      EVAL,      ESQVAL,      Y(50),      DNORM,
     X   XINI(50),     YINI(50)
C
C             SET N = 1
C
      N = 1
      CALL SUMS
      CALL GENINV
      CALL SOLIN
      RETURN
      END
$TITL                                   FUZZY/LSQ
C*******************************************************************
      SUBROUTINE FUZZY
C
C             THIS ROUTINE WILL COMPUTE THE FUZZY CHARACTERISTIC
C             FUNCTION FOR THE BREAKPOINT.
C
      COMMON
     X   /MAIN     /XIN(50),      YIN(50),      A(5,5),      X(5),
     X   N,      M,      EVAL,      ESQVAL,      Y(50),      DNORM,
     X   XINI(50),     YINI(50)
     X   /FUZZY     /SIG(50),      EMAX,      ICOUNL,      ICOUNQ,
     X   FUZLL,      FUZLH,      F1,      F2,      F3,
     X   FUZQL,      FUZQH
     X   /SOLIN     /E(5),      ISKIP,      FSOLIN(3)
     X   /YOPT      /XX(5),      ISUML,      ISUMQ,      C3(2),
     X   INTL,      INTQ,      MM,      RHO1,      SNR,      RHO
     X   /OPTSOL    /MSTAR,      MFIN,      E2,      MFOPT,
     X   FYOPT(50,8),      NY
      DIMENSION C1(5),C2(5),G(10),C(3),C4(5),
     XC6(5),D1(5,3),D2(5,3),D3(5,3),D4(5,3)
C
C             THE FOLLOWING IS A LIST OF SYMBLOS USED IN THE
```

```
C             PROGRAM WHICH REPRESENTS THE EQUATION SYMBOLS.
C
C         C(1) = G([X1])
C         C(2) = G([X2])
C         C(3) = G([X3])
C
C
C             C1(I) = SIGMA-Y([DELTAY(I)/L2]!X(I))
C             C2(I) = SIGMA-Y([DELTAY/LMAX]!X(I))
C             C3(I) = SIGMA-Y([S(I)/M]!X(I))
C             C4(I) = SIGMA-Y([L2/DELTAY(I)]!X(I))
      YY = M
      C(1) = F1
      C(2) = F3
      C(3) = F2
C
C                 TEST TO SEE IF L2 IS ZERO.
C
      IF(DNORM .GE. .01) GO TO 5
      DNORM3 = 0.0
5     CONTINUE
C
C        TEST TO SEE IF EMAX IS ZERO.
C
      IF(EMAX .GE. .1) GO TO 8
      EMAX2 = 0.0
8     CONTINUE
C
C             TESAT TO SEE IF SIG(I) IS ZERO
C
      IF(DNORM3 .EQ. 0.0) GO TO 9
      DO 10 I=1,M
         C1(I) = ABS(SIG(I)/DNORM)
         C2(I) = ABS(SIG(I)/EMAX)
10    CONTINUE
      GO TO 13
9     CONTINUE
      DO 11 I=1,M
         C1(I) = 0.0
         C2(I) = 0.0
11       CONTINUE
13    CONTINUE
C
C             TEST ORDER OF POLY.
C
      IF(N .EQ. 2) GO TO 20
         ICOUNL = ISUML + 1
         ICOUNT = ICOUNL
         ICOUNQ = 0
      DO 15 I=1,M
      C6(I) = ((ATAN(E(2)))/90.0)*57.3
15    CONTINUE
      GO TO 30
20    CONTINUE
C
C                 FORM L2 OF THE QUADRATIC SLOPE.
C
         ICOUNQ = ISUMQ + 1
         ICOUNT = ICOUNQ
         ICOUNL = 0
      Q = 0.0
      DO 26 I=1,M
         Q = E(2)+(2.0*E(3)*XIN(I))
      C6(I) = ((ATAN(Q))/90.0)*57.3
         Q = 0.0
26    CONTINUE
30    CONTINUE
C
C
C              ORDER SIGMA-Y FROM LARGEST TO SMALLEST.
C
C
      AA = 0.0
      K = 1
      J = 1
60    CONTINUE
      DO 70 I=1,M
         IF(C1(I) .GT. C1(J)) GO TO 75
         AA = C1(J)
         C1(J) = C1(I)
         C1(I) = AA
75    CONTINUE
         AA = 0.0
         IF(C2(I) .GT. C2(J)) GO TO 80
         AA = C2(J)
         C2(J) = C2(I)
         C2(I) = AA
80    CONTINUE
         AA = 0.0
         IF(C6(I) .GT. C6(J)) GO TO 85
         AA = C6(J)
         C6(J) = C6(I)
         C6(I) = AA
85    CONTINUE
70    CONTINUE
      IF(J .EQ. M) GO TO 95
      J = J+1
      K = J
         C1(J) = C1(K)
         C2(J) = C2(K)
         C6(J) = C6(K)
         GO TO 60
95    CONTINUE
C
C
C              NOW FIND MIN(SIGMA-Y(I),C(I))
C
C
      DO 210 I=1,M
              D1(I,1) = AMIN1(C1(I),C(1))
              D2(I,2) = AMIN1(C2(I),C(2))
              D3(I,3) = AMIN1(C6(I),C(3))
210   CONTINUE
C
C
C              FIND MAX VALUES FROM D1(I),D2(I),D3(I),AND
C        D4(I). THIS IS THE FUZZY BAYES' THEOREM MODEL.
C
C
      E5 = D1(1,1)
      E6 = D2(1,2)
      E7 = D3(1,3)
      DO 220 I = 2,M
         E5 = AMAX1(E5,D1(I,1))
         E6 = AMAX1(E6,D2(I,2))
         E7 = AMAX1(E7,D3(I,3))
```

```
 220   CONTINUE
       E5 = 1.0+E5
       E6 = 1.0+E6
       E7 = 1.0+E7
       E5 = 1.0-E5
       E6 = 1.0-E6
       E7 = 1.0-E7
C****************************************
C
C              FIND FUZZY DECISION BY USING
C              DISJUNCTIVE SUM.
C
C****************************************
       E9 = 1.0-E5
       E10 = 1.0-E6
       E11 = AMIN1(E5,E10)
       E12 = AMIN1(E9,E6)
       E13 = AMAX1(E11,E12)
       E14 = 1.0-E13
       E15 = 1.0-E7
       E16 = AMIN1(E13,E15)
       E17 = AMIN1(E14,E7)
       E18 = AMAX1(E16,E17)
       XX(ICOUNT) = E18
          FF = XX(ICOUNT)
C              TEST TO DETERMINE IF FUZZY VALUE IS
C              LESS THAN .25.
C
       IF(ICOUNT .NE. 2) GO TO 240
       RHO1 = ABS(XX(1)-XX(2))
C
C              NOW COMPUTE RHO USING 1-(MINMAX RULE).
C
       RHO = 1.0-RHO1
       IF(N .EQ. 2) GO TO 250
C
C
C              NOW TEST TO SEE IF WE HAVE A GOOD FIT.
C
C
       IF((RHO .LT. FUZLH) .AND. (RHO .GE. FUZLL)) GO TO 240
C
C         NO--BAD FIT--SET XX(1) = XX(2) AND ICOUNT = 1.
C
       XX(1) = XX(2)
          ISUML = 0
          CALL YOPT
       GO TO 230
 250   CONTINUE
       IF((RHO .LT. FUZQH) .AND. (RHO .GE. FUZQL)) GO TO 240
               XX(1) = XX(2)
          ISUMQ = 0
          CALL YOPT
          GO TO 230
 240   CONTINUE
C
C         WE HAVE A GOOD FIT. COMPUTE YOPT AND COMPUTE
C         A NEW FUZZY CHARACTERISTIC VALUE BASED ON YOPT.
C
C
       CALL YOPT
C
C              TEST WHICH PASS THIS IS I.E. 1 OR 2.
```

```
C
      IF(ICOUNT .NE. 2) GO TO 230
      XX(1) = XX(2)
230   CONTINUE
      RETURN
      END
$TITL                                  YOPT/LSQ
C*******************************************************************
      SUBROUTINE YOPT
C
C             THIS ROUTINE WILL COMPUTE A NEW YOPT ACROSS
C             TWO INTERVALS (IN AND IN+1).
C
      COMMON
     X   /MAIN     /XIN(50),      YIN(50),      A(5,5),      X(5),
     X   N,      M,      EVAL,      ESQVAL,      Y(50),      DNORM,
     X   XINI(50),    YINI(50)
     X   /YOPT     /XX(5),     ISUML,      ISUMQ,      C3(2),
     X   INTL,      INTQ,      MM,      RHO1,      SNR,      RHO
     X   /SUMS     /YY,      XY,      XSQY,      AY(5,5),
     X   XYY,    XXSQY
     X   /SCALE     /C,      D,      IL
     X   /SOLIN     /E(5),     ISKIP,     FSOLIN(3)
     X   /GENINV    /YG(10,10)
     X   /OPTSOL     /MSTAR,     MFIN,     E2,     MFOPT,
     X   FYOPT(50,8),     NY
      DIMENSION AL(20),BL(10),PI1(20),THETA(20),XI(50),
     X   ML(50),MQ(50),ALL(10),BLL(10),PI2(20)
      IF(N .EQ. 2) GO TO 30
      ISUML = ISUML + 1
C
C             SOLVE FOR LINEAR CASE BY FORMING TWO SEPARATE ARRAYS
C             THEN COMPUTE THE YOPT AND THEN SUBSTITUTE YOPT BACK
C             INTO THE INITIAL LINEAR ARRAY.
C
C             TEST TO DETERMINE WHICH PASS THIS IS.
C
C*******************************************************************
C
C             INITIALIZE SOME MULIPYING FACTORS I.E. PI(LINCASE) AND
C             THETA(QUADRATIC CASE).
C
C**********************************************************************
      PI1(1) = .277777779
      PI1(2) = .12820512
      PI1(3) = .07352942
      PI1(4) = .047619049
      PI1(5) = .03333339
      PI2(1) = .33333333
      PI2(2) = .16483533
      PI2(3) = .098039269
      PI2(4) = .064935207
      PI2(5) = .046153914
         IF(ISUML .EQ. 2) GO TO 20
      ML(1) = 5
      INTL = INTL + 1
C
C             FORM INITIAL LINAER ARRAY AND SCALE BY 10.
C
      AL(1) = AY(1,1)/10.0
      AL(2) = AY(1,2)/10.0
      AL(3) = AY(2,1)/10.0
      AL(4) = AY(2,2)/10.0
```

```
         BL(1) = YY/10.0
      BL(2) = XYY/10.0
      ALL(1) = YG(1,1)
      ALL(2) = YG(2,2)
C
C             SET ISUMQ = 0
C
         ISUMQ = 0
      ICOUNT = 0
C***************************************************************
C
C             FIN STARTING VALUE
C
C***************************************************************
      MSTAR = MM-4
      NUMCO = MM
      GO TO 200
20    CONTINUE
C
C             FORM SECOND LINEAR ARRAY AND SCALE BY 10.
C
      AL(5) = AY(1,1)/10.0
      AL(6) = AY(1,2)/10.0
      AL(7) = AY(2,1)/10.0
      AL(8) = AY(2,2)/10.0
         BL(3) = YY/10.0
      BL(4) = XYY/10.0
C
C             SET ISUMQ = 0
C
         ISUMQ = 0
      ICOUNT = ICOUNT + 1
      ALL(3) = YG(1,1)
      ALL(4) = YG(2,2)
      ALL1 = ALL(1)+ALL(3)
      ALL2 = ALL(2)+ALL(4)
      ALL3 = ALL1*PI1(ICOUNT)
      ALL4 = ALL2*PI2(ICOUNT)
C
C             NOW FORM LINEAR YOPT.
C
      A(1,1) = AL(1) + AL(5)
      A(1,2) = AL(2) + AL(6)
      A(2,1) = AL(3) + AL(7)
      A(2,2) = AL(4) + AL(8)
      YY = BL(1) + BL(3)
      XY = BL(2)+BL(4)
C*******************************************************************
C
C             FIND THE FINAL X OR Y VALUE.
C
      MFIN = MM
      NUMCO = NUMCO+4
      NUMCO1 = NUMCO - MM
      IF(NUMCO1) 102,104,106
102   CONTINUE
         WRITE(6,600)
600   FORMAT(20X,'ERROR INTERVAL NUMBER IS NEGATIVE')
      GO TO 200
104   CONTINUE
C***************************************************************
C
C             COMPUTE THE Y = A'X MATRIX THE SHORT WAY.
```

```
C
C***************************************************************
      IF(NY .EQ. 1) GO TO 36
      D1 = XINI(MSTAR)
      D2 = XINI(MSTAR)
      DO 26 I=MSTAR,MFIN
         D1 = AMIN1(D1,XINI(I))
         D2 = AMAX1(D2,XINI(I))
26    CONTINUE
      DMIN = D2-D1
      DMAX = D2+D1
C***************************************************************
C
C        SCALE XINI VALUES AND FORM THE PRODUCT
C        Y = A'X.
C
C***************************************************************
      VA1 = 0.0
      VAL = 0.0
      VA2 = 0.0
      DO 28 I=MSTAR,MFIN
         VA1 = (((2.0*XINI(I))-DMAX)/DMIN)*YINI(I)
         VAL = VAL+VA1
         VA2 = VA2 + YINI(I)
28    CONTINUE
      GO TO 39
36    CONTINUE
      D1 = YINI(MSTAR)
      D2 = YINI(MSTAR)
      DO 37 I=MSTAR,MFIN
         D1 = AMIN1(D1,YINI(I))
         D2 = AMAX1(D2,YINI(I))
37    CONTINUE
      DMIN = D2-D1
      DMAX = D2+D1
      VA1 = 0.0
      VAL = 0.0
      VA2 = 0.0
      DO 38 I=MSTAR,MFIN
         VA1 = (((2.0*YINI(I))-DMAX)/DMIN)*XINI(I)
         VAL = VAL + VA1
         VA2 = VA2 + XINI(I)
38    CONTINUE
39    CONTINUE
C**********************************************
C
C        NOW SOLVE THE MATRIX.
C
C**********************************************
      ALL5 = ALL3*VA2
      ALL6 = ALL4*VAL
C**********************************************
C        RESCALE ALL5 AND ALL6.
C**********************************************
      ALL7 = ALL5-(DMAX/DMIN)*ALL6
      ALL8 = (2.0*ALL6)/DMIN
106   CONTINUE
C***************************************************************
C
C              COMPUTE THE UPPER ERROR BOUND ALLOWED
C              FOR A YOPT SOLUTION.
C
C***************************************************************
```

```
        ECONS = E2
        CONS = RHO1*SNR
        CONS1 = EXP(RHO)
        E2 = EVAL+CONS+CONS1
C
C
C               NOW CALL THE APPROPRIATE ROUTINES.
C
        CALL GENINV
        CALL OPTSOL
        E2 = ECONS
C
C               SET INITIAL LINEAR ARRAY EQUAL TO YOPT.
C
           AL(1) = A(1,1)
           AL(2) = A(1,2)
           AL(3) = A(2,1)
           AL(4) = A(2,2)
           BL(1) = YY
        BL(2) = XY
        INTQ = 0
C
C               SET ISUML = 1 AND SET ISKIP = 0
C
        ISUML = 1
        ISKIP = 0
        GO TO 200
30      CONTINUE
        ISUMQ = ISUMQ + 1
        ISUML = 0
C
C               SOLVE FOR QUADRATIC CASE BY FORMING TWO SEPARATE
C               ARRAYS AND THEN COMPUTE YOTPQ AND THEN SUBSTITUTE
C               YOPTQ BACK INTO THE INITIAL QUADRATIC ARRAY.
C
C
C                    TEST TO DETERMINE WHICH PASS THIS IS.
C
        IF(ISUMQ .EQ. 2) GO TO 40
C
C                    FORM INITIAL QUADRATIC ARRAY.
C
           K = 0
           DO 32 I=1,3
                DO 34 J =1,3
                     K = K+1
                     AL(K) = AY(I,J)/10.0
34              CONTINUE
32         CONTINUE
        BL(1) = YY/10.0
        BL(2) = XYY/10.0
        BL(3) = XXSQY/10.0
C
C                    SET ISUML = 0
C
        ISUML = 0
        MSTAR = MM-4
        GO TO 200
40      CONTINUE
C
C                    FORM SECOND QUADRATIC ARRAY
C
            K=9
```

```
             DO 42 I =1,3
                DO 44 J =1,3
                   K = K + 1
                   AL(K) = AY(I,J)/10.0
44              CONTINUE
42           CONTINUE
          BL(4) = YY/10.0
          BL(5) = XYY/10.0
          BL(6) = XXSQY/10.0
C
C                  SET ISUML = 0
C
          ISUML = 0
C
C                  NOW FORM QUADRATIC YOPT
C
          L = 9
          K = 0
          DO 50 I=1,3
             DO 60 J=1,3
                L = L+1
                K= K+1
                A(I,J) = AL(K) + AL(L)
60           CONTINUE
50        CONTINUE
          YY = BL(1) + BL(4)
          XY = BL(2) + BL(5)
          XSQY = BL(3) + BL(6)
          MFIN = MM
C*******************************************************************
C
C               COMPUTE THE UPPER ERROR BOUND.
C
C********************************************************************
          ECONS = E2
          CONS = (1.0-RHO1)*SNR
          CONS1 = EXP(RHO)
          E2 = EVAL+CONS+CONS1
C
C                  SET A FLAG TO SKIP RESCALE.
C
          ISKIP = 1
C
C                  NOW CALL THE APPROPRIAT ROUTINES
C
          CALL GENINV
          CALL OPTSOL
          E2 = ECONS
C
C                  SET INITIAL QUADRATIC ARRAY EQUAL TO YOPT.
C
          K = 0
          DO 70 I =1,3
             DO 80 J=1,3
                K = K+1
                AL(K) = A(I,J)
80           CONTINUE
70        CONTINUE
          BL(1) = YY
          BL(2) = XY
          BL(3) = XSQY
C
C                  SET ISUMQ = 1 AND SET ISKIP = 0.
```

```
C
       ISKIP = 0
        ISUML = 0
       ISUMQ = 1
       INTL = 0
200    CONTINUE
       RETURN
       END
$TITL                                                           OPTSOL/LSQ
       SUBROUTINE OPTSOL
C*******************************************************************
C
C              COMPUTE THE OPTIMUM SOLUTION AND ERRORS
C              ASSOCIATED WITH THIS SOLUTION.
C
C**********************************************************************
       COMMON
     X   /MAIN    /XIN(50),    YIN(50),    A(5,5),    X(5),
     X   N,    M,    EVAL,    ESQVAL,    Y(50),    DNORM,
     X   XINI(50),    YINI(50)
     X   /SUMS    /YY,    XY,    XSQY,    AY(5,5),
     X   XYY,    XXSQY
     X   /GENINV    /YG(10,10)
     X   /SOLIN    /E(5),    ISKIP,    FSOLIN(3)
     X   /FUZZY    /SIG(50),    EMAX,    ICOUNL,    ICOUNQ,
     X   FUZLL,    FUZLH,    F1,    F2,    F3,
     X   FUZQL,    FUZQH
     X   /OPTSOL    /MSTAR,    MFIN,    E2,    MFOPT,
     X   FYOPT(50,8),    NY
     X   /YOPT    /XX(5),    ISUML,    ISUMQ,    C3(2),
     X   INTL,    INTQ,    MM,    RHO1,    SNR,    RHO
       DIMENSION BB(5),CE(3)
C********************************************************************
C
C              INITIALIZE X(I) TO ZERO
C
C*******************************************************************
       DO 10 I = 1,3
          CE(I) = 0.0
10     CONTINUE
C******************************************************************
C
C              TEST TO DETERMINE IF N = 1 OR 2.
C
C****************************************************************
       IF(N .EQ. 2) GO TO 15
C******************************************************************
C
C              THEN BUILD THE B MATRIX.
C
C*******************************************************************
       BB(1) = YY
       BB(2) = XY
       L= 2
       GO TO 18
15     CONTINUE
C*******************************************************************
C
C              BUIL THE B MATRIX FOR QUADRATIC.
C
C*******************************************************************
       BB(1) = YY
       BB(2) = XY
```

```
      BB(3) = XSOY
      L = 3
18    CONTINUE
C*********************************************************************
C
C               FIND COEFFICIENTS TO LINEAR EQUATION
C
C*********************************************************************
      DO 20 I=1,L
      AA = 0.0
         DO 30 J=1,L
               AA = YG(I,J)*BB(J)
30       CE(I) = CE(I)+AA
20    CONTINUE
      IF(N .EQ. 2) GO TO 45
22    CONTINUE
C**********************************************************************
C
C               FORM LINEAR EQUATION AT THE GIVEN XS
C
C************************************************************************
      IF(NY .EQ. 0) GO TO 32
         DO 34 I = MSTAR,MFIN
34       Y(I) = CE(1) + CE(2)*YINI(I)
      GO TO 36
32    CONTINUE
      DO 40 I=MSTAR,MFIN
40    Y(I) = CE(1)+CE(2)*XINI(I)
36    CONTINUE
C**********************************************************************
C
C        TEST TO SEE IF WE CAN REPLACE THE
C        PAST 2 EQUATIONS BY ONE.
C
C**********************************************************************
      II = MFOPT-1
      IF((FYOPT(II,3) .EQ. 0.0) .AND.
     X (FYOPT(MFOPT,3) .EQ. 0.0)) GO TO 42
      IF((FYOPT(II,3) .EQ. 1.0) .AND.
     X (FYOPT(MFOPT,3) .EQ. 1.0)) GO TO 42
      GO TO 55
42    CONTINUE
      STAR = FLOAT(MSTAR)
      FIN = FLOAT(MFIN)
      IF((FYOPT(II,1) .NE. STAR) .AND.
     X (FYOPT(MFOPT,2) .NE. FIN)) GO TO 55
      MFOPT = II
      GO TO 55
45    CONTINUE
C**************************************************************************
C
C        FORM QUADRATIC COEFFICIENTS.
C
C*************************************************************************
      IF(NY .EQ. 0) GO TO 47
         DO 46  I = MSTAR,MFIN
46       Y(I) = CE(1)+(CE(2)*YINI(I))+(CE(3)*(YINI(I)**2))
      GO TO 49
47    CONTINUE
      DO 48 I=MSTAR,MFIN
48    Y(I) = CE(1)+(CE(2)*XINI(I))+(CE(3)*(XINI(I)**2))
49    CONTINUE
C*********************************************************************
```

```
C
C        TEST TO SEE IF WE CAN REPLACE THE
C        PAST 2 EQUATIONS BY ONE.
C
C*********************************************************************
      II = MFOPT-1
      IF((FYOPT(II,3) .EQ. 0.0) .AND.
     X (FYOPT(MFOPT,3) .EQ. 0.0)) GO TO 52
      IF((FYOPT(II,3) .EQ. 1.0) .AND.
     X (FYOPT(MFOPT,3) .EQ. 1.0)) GO TO 52
      GO TO 55
52    CONTINUE
      STAR = FLOAT(MSTAR)
      FIN = FLOAT(MFIN)
      IF((FYOPT(II,1) .NE. STAR) .AND.
     X (FYOPT(MFOPT,2) .NE. FIN)) GO TO 55
      MFOPT = II
55    CONTINUE
      EE = 0.0
      AA = 0.0
      DNORM = 0.0
      DNORM1 = 0.0
      E2D = 0.0
      X1 = 0.0
      X4 = 0.0
      IF(N .EQ. 2) GO TO 54
      GAM = ATAN(E(2))
54    CONTINUE
      DO 50 I=MSTAR,MFIN
         IF(NY .EQ. 0) GO TO 51
                AA = XINI(I)-Y(I)
      GO TO 53
51    CONTINUE
                AA = YINI(I)-Y(I)
53    CONTINUE
         SIG(I) = AA
         IF(AA .LT. 0.0) GO TO 56
         IF(N .EQ. 2) GO TO 58
                E2D = (AA*SIN(90.0-GAM))**2
                GO TO 64
58       CONTINUE
      IF(NY .EQ. 0) GO TO 57
         AM = (2.0*CE(3))*YINI(I)
      GO TO 61
57    CONTINUE
                AM = (2.0*CE(3))*XINI(I)
61    CONTINUE
                GAM = 90.0-(ATAN(AM))
                E2D = (AA*SIN(GAM))**2
                GO TO 64
56       CONTINUE
                IF(N .EQ. 2) GO TO 62
                E2D = (AA*SIN(GAM))**2
                GO TO 64
62       CONTINUE
      IF(NY .EQ. 0) GO TO 59
         AM = (2.0*CE(3))*YINI(I)
      GO TO 63
59    CONTINUE
                AM = (2.0*CE(3))*XINI(I)
63    CONTINUE
                GAM = ATAN(AM)
                E2D = (AA*SIN(GAM))**2
```

```
 64        CONTINUE
                DNORM1 = DNORM1+E2D
 50     EE = EE+AA
           DNORM = SQRT(DNORM1)
C*********************************************************************
C
C               COMPUTE SUM E SQUARE ERROR.
C
C*********************************************************************
        ESQ = 0.0
        AA = 0.0
           DO 60 I=MSTAR,MFIN
                AA = (YINI(I)-Y(I))**2
 60        ESQ = ESQ+AA
        ESQVAL = SQRT(ESQ)
C******************************************************************
C
C               FIND EMAX.
C
C******************************************************************
        EMAX = ABS(SIG(MSTAR))
        KK = MSTAR+1
        DO 70 I=KK,MFIN
           EMAX = AMAX1(EMAX,ABS(SIG(I)))
 70     CONTINUE
 90     CONTINUE
        IF(DNORM .GT. E2) GO TO 1000
        MFOPT = II
        FYOPT(II,1) = MSTAR
        FYOPT(II,2) = MFIN
        FYOPT(II,3) = NY
        FYOPT(II,4) = CE(1)
        FYOPT(II,5) = CE(2)
        IF(N .EQ. 2) GO TO 1010
        FYOPT(II,6) = 0.0
        GO TO 1020
 1010   CONTINUE
        FYOPT(II,6) = CE(3)
 1020   CONTINUE
        FYOPT(II,7) = EMAX
        FYOPT(II,8) = DNORM
        GO TO 1200
 1000   CONTINUE
        MFOPT = II+1
 1200   CONTINUE
        RETURN
        END
 STITL                                                       PLOT/LSQ
        SUBROUTINE PLOT
C**************************************************************
C
C              THIS ROUTINE WILL COMPUTE THE VALUES OF THE FUNCTION
C              AND PLOT THEM AGANIST THE INITIAL VALUES.
C
C**************************************************************
        COMMON
      X /MAIN    /XIN(50),     YIN(50),     A(5,5),     X(5),
      X N,    M,     EVAL,    ESQVAL,    Y(50),     DNORM,
      X XINI(50),     YINI(50)
      X   /OPTSOL     /MSTAR,     MFIN,     E2,     MFOPT,
      X   FYOPT(50,8),     NY
       DIMENSION AA(50),BX(50),C(50),D(50),E(50),F(50),
      XIAA(50),IBX(50)
      REAL NN,IYY
      DATA NN,IYY,IB/1HN,1HY,1H /
C*****************************************************
C
C           TEST TO DETERMINE WHICH FUNCTION WE NEED
C           TO PLOT.
C
C**********************************************************
      ANS = IB
      KK = 0
      K = 0
      J = 1
      I = 1
      IY = 0
      IX = 0
5     CONTINUE
      II = IFIX(FYOPT(I,3))
      IF(II .EQ. 1) GO TO 20
      IY = IY+1
C**************************************************************
C
C           COMPUTE Y = AX + B
C
C*************************************************************
      MSTAR = IFIX(FYOPT(I,1))
      MFIN = IFIX(FYOPT(I,2))
      DO 10 J=MSTAR,MFIN
         K = K+1
            AA(K) = FYOPT(I,4)+(FYOPT(I,5)*XINI(J))+
     X(FYOPT(I,6)*(XINI(J)**2))
      C(K) = XINI(J)
      D(K) = YINI(J)
10    CONTINUE
      I = I+1
      IF(I .GT. MFOPT) GO TO 100
      GO TO 5
20    CONTINUE
      IX = IX+1
      MSTAR = IFIX(FYOPT(I,1))
      MFIN = IFIX(FYOPT(I,2))
      DO 30 J=MSTAR,MFIN
         KK = KK+1
            BX(KK) = FYOPT(I,4)+(FYOPT(I,5)*YINI(J))+
     X(FYOPT(I,6)*(YINI(J)**2))
      E(KK) = XINI(J)
      F(KK) = YINI(J)
30    CONTINUE
      I = I+1
      IF(I .GT. MFOPT) GO TO 100
      GO TO 5
100   CONTINUE
      IF(IY .EQ. 0) GO TO 110
      WRITE(6,530)
      WRITE(1,550)
      READ(1,560) ANS
      IF(ANS .EQ. IYY) GO TO 130
      WRITE(6,500)(AA(L),D(L),C(L),L=1,K)
      GO TO 110
130   CONTINUE
      DO 140 L=1,K
         IAA(L) = IFIX(AA(L))
         AAB = FLOAT(IAA(L))
         AAA = ABS(AA(L)-AAB)
```

```
            IF(AAA .LT. .50) GO TO 140
            IAA(L) = IAA(L)+1
  140    CONTINUE
         WRITE(6,570)(IAA(L),D(L),C(L),L=1,K)
  110    CONTINUE
         IF(IX .EQ. 0) GO TO 120
         WRITE(6,510)
         WRITE(6,540)
         WRITE(6,510)
         IF(ANS .EQ. IYY) GO TO 150
         WRITE(6,520)(BX(L),E(L),F(L),L=1,KK)
         GO TO 120
  150    CONTINUE
         DO 160 L=1,KK
            IBX(L) = IFIX(BX(L))
            BAA = FLOAT(IBX(L))
            BBA = ABS(BX(L)-BAA)
            IF(BBA .LT. .50) GO TO 160
            IBX(L) = IBX(L)+1
  160    CONTINUE
         WRITE(6,580)(IBX(L),E(L),F(L),L=1,KK)
  500    FORMAT(10X,'Y =',F10.5,5X,'YIN =',F10.5,5X,
        X    'XIN =',F10.5)
  510    FORMAT(10X,'*******************************************')
  520    FORMAT(10X,'X =',F10.5,5X,'XIN =',F10.5,5X,
        X    'YIN =',F10.5)
  530    FORMAT(33X,'Y = AX + B')
  540    FORMAT(33X,'X = CY + D')
  550    FORMAT('DO YOU WANT INTEGER DATA (Y OR N)?')
  560    FORMAT(A1)
  570    FORMAT(10X,'Y =',F10.5,5X,'YIN =',F10.5,5X,
        X'XIN =',F10.5)
  580    FORMAT(10X,'X =',F10.5,5X,'XIN =',F10.5,5X,
        X'YIN =',F10.5)
  120    CONTINUE
         RETURN
         END
 $BEND
```

BIBLIOGRAPHY

1. Ackland, Band N. Weste, "The Edge Flag Algorithm - A Fill Method for Raster Scan Displays", IEEE Transactions on Computers, Vol. C-30, No. 2, Jan. 1980, pp. 4-48.

2. Agrawala, A.K., Machine Recognition of Patterns, IEEE Press, New York, 1977.

3. Aklberg, Nilson, and Walsh, The Theory of Splines and Their Applications, Academic Press, New York and London, 1967.

4. Albano, A.A., "Representation of Digitized Contours in Terms of Conic Arcs and Straight-Line Segments", Computer Graphics and Image Processing, Vol. 3, 1974, pp. 23-33.

5. Andrews, H., Introduction to Mathematical Techniques in Pattern Recognition, Wiley-Interscience, New York, 1972.

6. Attneave, Fred, "Some Informational Aspects of Visual Perception", Psychological Review, Vol. 61, No. 3, 1954, pp. 183-193.

7. Ben-Israel, A. and T. Greville, Generalized Inverses: Theory and Applications, Wiley-Interscience Series, 1974.

8. Bezdek, J.C. and J.C. Dunn, "Optimal Fuzzy Partitions: A Heuristic for Estimating the Parameters in a Mixture of Normal Distributions", IEEE

Transactions on Computers, Vol. C-24; No. 8, August 1975, pp. 835-838.

9. Boullion, T. L. and P. L. Odell, Generalized Inverse Matrices, Wiley, 1971.

10. Chang, R. and T. Pavlidis, "Application of Fuzzy Sets in Curve Fitting", Fuzzy Sets in Curve Fitting", Fuzzy Sets and Systems, Vol. 2, 1979, pp 67-74.

11. Cook, S. A., "The Complexity of Theorem - Proving Procedures", Proc. Third ACM Symp. on Theory of Computing 1971, pp. 151-158.

12. Dahlquist, G. and A. Bjorck, Numerical Methods, Prentice-Hall, Englewood Cliffs, N.J., 1974.

13. Davis, P.J., Interpolation and Approximation, N.Y. Blaisdell, 1963.

14. De Boor, C., "Good Approximation by Splines with Variable Knots", Int. Series Numer. Math., Vol. 21, Birkhauser Verlag, Basel 1973, pp. 57-72.

15. De Mori, R. and P. Laface, " Use of Fuzzy Algorithms for Phonetic and Phonemic Labeling of Continuous Speech", IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 2, March 1980, pp. 136-148.

16. Duda, R. O. and P. E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

17. Fiduccia, C. M., Complexity of Computer Computations, Plenum-Press, Inc.,

New York, 1972.

18. Franklin, J. N., Matrix Theory, Prentice-Hall, Inc., New York, 1968.

19. Fu, K. S., Syntactic Methods in Pattern Recognition, Academic Press, New York, 1974.

20. Fukunaga, K., Introduction to Statistical Pattern Recognition, Academic Press, New York and London, 1972.

21. Gebb, A., Applied Optimal Estimation. M.I.T. Press, Cambridge, 1974.

22. Goguen, J. "L-Fuzzy Sets", J. Math. Appl., No. 18, 1967, pp. 145-174.

23. Halmos, P. R., Finite-Dimensional Vector Spaces, Van Nostrand, Princeton, 1958.

24. Halmos, P. R., Measure Theory, Van Nostrand, Princeton, 1950.

25. Jacobson, N., Basic Algebra I, W. H. Freeman and Co., San Francisco, 1974.

26. Kandel, A. and W, Byatt, "Fuzzy Sets, Fuzzy Algebra, and Fuzzy Statistics", Proceedings of the IEEE, Vol. 66, No. 23, Dec. 1978, pp. 1624-1639.

27. Kandel, A. and J. Francioni, "On the Properties and Applications of Fuzzy - Valued Switching Functions", IEEE Transactions on Computers, Vol. C-29, No. 11, Nov. 1980., pp. 986-994.

28. Kaufmann, A., Introduction to the Theory of Fuzzy Subsets, Vol. I, Academic Press, 1975.

29. Kells, Lyman, Analytic Geometry and Calculus, Prentice Hall, Inc., 1950.

30. Koczy, L., "Interactive σ -Algebras and Fuzzy Objects of Type N", Journal of Cybernetics, 1978, pp. 273-290.

31. Lane. J. M. and R. F. Riesenfeld, "A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2., No. 1, Jan. 1980, pp. 35-46.

32. McCaughey, D. and H. Andrews, "Image Approximation by Variable Knot Bicubic Splines", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-3, No. 3, May 1981, pp. 299-310.

33. McClure, D. E., "Nonlinear Segmented Function Approximation and Analysis of Line Patterns", Quart. Applied Math., Vol. 33, April 1975, pp. 1-37.

34. McClure, D. E., "Computation of Approximately Optimal Compressed Representations of Discretized Plane Curves", NSF grant number MCS 76-04002 and ONR grant number N0014-75-C-0427.

35. Mizumoto, M., "Note On the Arithmetic Rule By Zadeh for Fuzzy Conditional Inference", Cybernetics and Systems An International Journal, Vol. 12, No. 3, July-Sept. 1981.

36. Negoia, C. V. and D. A. Ralescu, Applications of Fuzzy Sets to Systems Analysis, Holsted Press Book, Wiley and Sons, New York and Toronto, 1975.

37. Pal, S. and R. King, "Image Enhancement Using Smoothing with Fuzzy Sets", IEEE Transations on Systems, Man, and Cybernetics, Vol. SMC-11, No. 7, July 1981, pp. 494-501.

38. Papoulis, A., Probability, Random Variables, and Stochastic Processes, McGraw Hall, 1965.

39. Pavlidis, T., Structural Pattern Recognition, Springer Series in Electrophysics 1, Springer - Verlag, Berlin, Heidelberg, New York, 1977.

40. Pavlidis, T., "Algorithms for Shape Analysis of Contours and Waveforms", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No.4, July 1980, pp. 301-312.

41. Pavlidis, T. and F. Ali, "A Hierarchical Syntactic Shape Analyzer", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 1, Jan. 1979, pp. 2-9.

42. Pavlidis, T. and S. Horowitz, "Segmentation of Plane Curves", IEEE Transactions on Computers, Vol. C-23, No. 8, August 1974, pp. 860-870.

43. Perkins, W. A., "A Model-Based Vision System for Industrial parts", IEEE Transactions on Computers, Vol. C-27, No. 2, Feb. 1978, pp. 126-143.

44. Ralston, A., A First Course in Numerical Analysis, McGraw-Hill Book Co., 1965.

45. Riesenfeld, R., "Applications of B-Spline Approximation to Geometric Problems of Computer - Aided Design", University of Utah, UTEC-CSC-73-126, March 1973.

46. Sammon, Jr., J., "Interactive Patten Analysis and Classification", IEEE Transactions on Computers, Vol. C-19, No. 7, July 1970, pp. 594-616.

47. Sokolnikoff, I.S. and R.M. Redheffer, Mathematics of Physics and Modern Engineering, McGraw-Hill Book Co., New York, 1958.

48. Special Symposium of Fuzzy Set Theory and Applications held during 1977 IEEE Conference on Decision and Control, New Orleans, L.A.

49. Steiglitz, K., "Computational Complexity Theory and Communication Network Problems", NATO Advanced Study Institute: "New Concepts in Multi-User Communications", Norwick, UK, 4-16 August 1980.

50. Swerling, P., "Modern State Estimation Methods from the Viewpoint of the Method of Least Squares", IEEE Transactions on Automatic Control, Vol. AC-16, No. 6, Dec. 1971, pp. 707-719.

51. Tamura, S., S. Higuchi, and K. Tanaka, " Pattern Classification Based on Fuzzy Relations", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-1, No. 1, Jan 1971, pp. 61-66.

52. Tamura, S. and K. Tanaka, "Learning of Fuzzy Formal Language", IEEE Transactions on Systems, Man, and Gybernetics, Vol. SNC-3, No. 1, Jan 1973., pp 98-102.

53. Terano, T. and ,. Sugeno, "Conditional Fuzzy Measures and Their Application", Fuzzy Sets and Their Applications to Cognitive and Decision Processes, Academic Press, 1975.

54. Tou, J. and R. Gonzalez, Pattern Recognition Principles, Addison-Wesley Publishing Co., Reading, Mass. 1974.

55. Watson, S. R., J. J. Weiss and M. L. Donnell, "Fuzzy Decision Analysis", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-9, No. 1, Jan 1979, pp. 1-9.

56. Winograd, S., "A New Algorithm for Inner Product", IEEE Transactions on Computers, July 1968, pp. 693-694.

57. Wong, A. R. C. and T. S. Liu, "A Decision-Directed Clustering Algorithm for Discrete Data", IEEE Transactions on Computers, Vol C-26, No. 1, Jan. 1977, pp. 75-82.

58. Zadeh,L. A., "Fuzzy Sets", Inf. and Control, Vol. 8, 1965, pp. 338-353.

59. Zadeh, L. A., "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-3, No. 1, Jan. 1973, pp. 28-44.

60. Zadeh, L., "Possibility Theory and Its Application To Information Analysis", Theoric De L' Information, No. 276, pp. 173-182.

61. Zadeh, L., "Similarity Relations and Fuzzy Orderings", Information Sciences, No. 3, 1971, pp. 177-200.