Moment-preserving piecewise approximation for 1-D and 2-D signals

Soha M. A. A. Seif

2004 / 11

The American University in Cairo
School of Sciences and Engineering

Moment-Preserving Piecewise Approximation For 1-D and 2-D Signals

A Thesis Submitted to
The Computer Science Department

In partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

By

Soha Mohamed Aly Abou Seif
B.Sc. Computer Science

Under the supervision of
Prof. Dr. Amr Goneid
May 2004

The American University in Cairo 2004/11

Moment-Preserving Piecewise Approximation for 1-D and 2-D Signals

A thesis submitted by
Soha Mohamed Aly Abouseif

To the department of Computer Science
May 2004

In partial fulfillment of the requirements for
The degree of **Master of Science**

Has been approved by

Dr Amr Goneid
Thesis Committee Chair/ Advisor [redacted]
Affliation AUC

Dr. [redacted]
Thesis Committee Reader/ examiner
Affliation Dean / College of Computers and Informatics / Zagazig Univ

Dr.
Thesis Committee Reader/ examiner
Affliation

Dr.
Thesis Committee Reader/ examiner [redacted]
Affliation AUC

[redacted] 27/5/04
Department Chair/ Date
Program Director

[redacted] May 31, 2004
Dean Date

Acknowledgement

First, I would like to direct a very special thanks to my dear supervisor Professor Amr Goneid who has always been a source of help. He encouraged me with his continuous confidence in my capabilities.

I would also like to thank Dr. Ismail Amr Ismail, Dr. Sherif El Kassas and Dr. Ashraf Abd El Bar for giving my thesis the time for evaluation and revision, in addition to their positive comments.

A special thanks and deep appreciation is for Professor B. John Oommen from Carleton University who however busy, was instantly answering my questions and requests for materials.

Also, I want to thank my dear professors, who participated in developing my capabilities till I became able to carry out with this work.

I would also love to thank the ACS family who were always around me during my pressure time and encouraged me always with their spirit.

Among the people I should thank, my friends and colleagues who were a great help with their spirit and cooperation.

Also, the computer science family with every single aid that they provided, I owe them a 'thank you'.

Last, but not least, I want to thank my beloved parents, fiancée Karim and family members who pushed me to meet deadlines for this thesis and strengthened my belief in thy self until I become able to make it.

Abstract

Approximations of 1-D and 2-D signals are important for noise reduction and signal space compression. Current techniques address the approximation process either in the signal space or in its transform space but not in both. A moment-preserving constraint can couple both spaces for better evaluation of approximations at nodal signal points.

In this thesis, we start by examining the effect of using the moment-preserving piecewise approximation technique on 1-D signals and we compare it against the Interpolation piecewise technique. On Linear, Quadratic and Cubic Spline polynomial method, the moment-preserving constraint proves to be a better approximation technique.

The second experiment examines the moment-preserving constraint effect on the approximation of 2-D closed boundaries and the results again are promising on the Linear, Quadratic and Cubic Spline polynomial level when compared to the Interpolation technique.

Thirdly, we examine the moment-preserving constraint on the Digital images approximation and it is successful as far as comparison with the Discrete Cosine Transform technique for image compression is concerned.

Finally, we experiment the effect of the moment-preserving piecewise approximation technique on 2-D Surfaces. The moment-preserving linear piecewise approximation technique excels over Bilinear Interpolation with much better result.

Index Terms – Signal Processing, Image Processing, Pattern Analysis, Moments, Piecewise Approximation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1. Introduction

1.1. Introduction

Signal Processing is concerned with the sequence of operations carried out on a signal to simplify its analysis and storage. This makes approximation techniques an important aspect of signal processing. For example, an approximation for the analog signal represents it with a digital signal instead of the original analog one. Electronic voice mail messages and photographic X-ray images are often digitized to reduce its storage and transmission costs. Approximation and encoding are small segments of the field of signal processing which also includes scaling, filtering, encoding, structural pattern recognition, feature extraction, scene analysis, motion detection and image understanding. [17] This makes approximation an interesting field for focus through this thesis.

The process of approximating digitized 1-D signals and images has always been a regular practice when there was a need for noise reduction or signal space lossy compression. A linear approximation technique can be used to remove undesirable noise from the actual signal as in [13], [15], [21], [8] and [9]. Unfiltered noisy signals can lead to misinterpretation of data and this is illustrated through a case in [3]. There, a boundary detection algorithm was applied to a radiograph of a chest in which the accurate detection of the rib was almost impossible due to unfiltered noise.

In the 1-D signals case, the noisy signals have been considered as functions f(x) sampled at distinct points $\{x_i , i = 0 ,1 , .. n\}$. The purpose of approximation in such case is to find an approximation function g(x) defined as a set of distinct nodal points $\{z_j , j = 0 ,1 ,...., m\}$ such that $m<n$, subject to a certain error minimization criteria. These nodal points are then joined using one of the interpolation techniques giving an approximation of the original signal. This methodology is also applied to 2-D signals and images when approximation is needed.

The approximation problem is relevant to many applications in signal processing, pattern analysis and image processing. Hence, a considerable amount of research has been advanced in this area.

1.2. Existing Approximation Approaches

Existing well-known approaches derive the approximations either through constraints in the signal domain or in the transform domain. Thai Nguyen presents a full survey of existing approximation techniques, with experiments illustrating their performance in his master thesis. [12]

1.2.1. Approximation in Signal Domain:

Some distinct nodal points are chosen on the original signal. Regenerating the signal, we take those chosen nodal points and apply one of the interpolation methodologies. The resulting signal is an approximation of the original signal. In the signal domain, examples of approximation approaches are Interpolation (Piecewise Approximation technique), [4], [18], [16] and others.

1.2.2. Approximation in Transform Domain:

The original signal is transferred from signal domain to the frequency domain. In the frequency domain, we ignore the low amplitude frequencies and track only the high amplitude frequencies to preserve the main features of the signal. Converting these tracked frequencies from the frequency domain to the signal domain returns an approximation of the original signal. In the signal domain, examples of approximation approaches are Fourier Transform, Discrete Cosine Transform and others.

1.3. Moment-Preserving Piecewise Approximation

This method was first introduced by Professor Oommen and his student Nguyen in 1994 and it was published in an IEEE conference in 1997 by the same authors.[12] They introduced the method and applied it only on the piecewise linear approximations case. Their main motivation was that this method is, unlike the already existing methods that derives the approximation in either the signal domain or the transform domain, utilizes the approximation in both domains. It derives the approximation in the Signal Domain while preserving a finite number of geometric moments that are related to its Fourier Domain. In the present work, we extend Thai's work by using high order polynomials on 1-D and 2-D signals.

1.4. Thesis Objective

The thesis contribution, presented in this work, the extension of the moment-preserving method for 1-D Signals, 2-D Closed Boundaries, Digitized Images and 2-D Surfaces through higher order polynomials. Specifically, we aim to derive the theory of the moment preserving approach for linear, quadratic and cubic spline polynomials for 1-D Signals and 2-D Closed Boundaries, the linear and quadratic polynomials for Digitized Images and the linear polynomial for 2-D Surfaces will be presented. The computational results obtained from the derived theory are to be compared with the usual interpolation approximation methods.

1.5. Thesis Outline

The thesis is divided into eight chapters. Chapter two demonstrates the background of approximation as a research topic. In Chapter three, we develop the theory that is needed for applying the moment-preserving approximation method. We also introduce the mathematical formulation for Piecewise Linear, Quadratic and Cubic Spline approximations that are applied on 1-D Signals and 2-D Closed Boundaries. Chapter four demonstrates the effect of Moment-Preserving Piecewise Approximation technique compared to Interpolation on 1-D Signals.

Chapter five is similar to chapter four but for 2-D Closed Boundaries. Chapter six handles the experimentation of Moment-Preserving technique on Digitized Images and the seventh chapter experiments the method on 2-D Surfaces. The last/eighth chapter concludes the thesis and briefs the expected future work.

Part of the present work has been published as a full paper [1] in the proceeding of the international Conference CCCT'03.

handles the experimentation of Moment-Preserving technique on Digitized Images and the seventh chapter experiments the method on 2-D Surfaces. The last/eighth chapter concludes the thesis and briefs the expected future work.

Part of the present work has been published as a full paper [1] in the proceeding of the international Conference CCCT'03.

Chapter 2

Approximation Methods for 1-D and 2-D Signals

2. Approximation Methods for 1-D and 2-D Signals

2.1. Introduction

There are various methods that are used for 1-D and 2-D signal approximation. They utilize the approximation technique in either the Signal Domain or in the Transform Domain. In the next subsections, we are introducing examples of existing approximation technique, some constraints approximation in the Signal Domain and others constraint approximation in the Transform Domain. The effect of the Moment-Preserving constraint is demonstrated in the coming sections. We compare the Moment-Preserving piecewise approximation technique with Interpolation and DCT.

2.2. Signal Domain Approximation Techniques

Given a 1-D signal f(x) in its signal domain (x), the approximation techniques introduced in the following subsections calculate its approximation, g(x), by directly using the given values of x and f(x). This is why they are considered to be approximation methods in the actual signal domain.

2.2.1. Subsampling Approximation

In many signal-processing applications, it is necessary to convert an analog signal to a discrete sequence of numbers. The conversion process is commonly done as explained in [12] by sampling the continous analog signal, f(x) at regular intervals Δx to produce a sequence $S = \{f(x_0), f(x_1), f(x_2), \ldots, f(x_{n-1})\}$ where $x_i = x_0 + i\Delta x$ and x_0 is some given starting value. There are special cases where the sampling rate is not regular and in such a case, both the x and the value of the signal at that particular x have to be specified. The work in [16] is an example of a proposed subsampling scheme. This type of approximation is easy to apply and relatively inexpensive due to its low computational complexity. However, the lack of computational complexity introduces aliasing problem, which causes the signals to lose their high frequencies and the signals to be very inaccurate.

2.2.2. Least Square Polynomial Approximation

Least Square Polynomial is a well-known approximation technique that is based on Legendre's principle of least squares which states that "when the available data in the domain D are either exact of equal reliability, then the 'best approximation' over D is that one for which the aggregate (sum or integral) of the squared error in D is least" [7] This technique is not practical when applied on high polynomial orders due to the computational cost and the many local minima, maxima and inflection points which inherently exists in polynomials.

2.2.3. Piecewise Approximation

By piecewise approximation, we mean dividing our signal into segments and approximating each and every segment separately. Then the resulting segments are recombined to come up with the whole approximated signal. This is a common method that helps when we have a highly complicated signal, and it is better to handle one segment at a time to reduce the complexity of the operation. Usually the point at which the resulting segments are joined is called 'knot' points. As mentioned previously, Linear Piecewise approximation is a common practice for noise reduction as in in [13], [15], [21], [8] and [9]. They assume that each and every pair of points over the signal are joined by a straight line. Among the most famous Piecewise Approximation technique currently in use is Interpolation.

2.2.3.1. Interpolation

Provided that approximation takes place either on the whole signal or piecewise, Interpolation is the most famous technique used in piecewise approximation (which is the approximation style we are interested in).

Interpolation is the process used to estimate values that lie between known data points. Conceptually, the interpolation process has 2 stages:

1. Fit an interpolating function to the data points provided.
2. Evaluate the interpolating function at the target point x.

Interpolation operates on various polynomial orders. The higher the polynomial order, the more the number of points needed for interpolation; i.e. Linear interpolation needs 2 points while Quadratic interpolation needs 3 points. However, this does not necessarily mean higher accuracy.

The interpolation orders we are interested in through this thesis are Linear, Quadratic, Cubic Spline and Bilinear interpolation.

Interpolation utilizes the approximation process in the Signal Domain.

2.2.3. Polynomial Spline Approximation

The difference between this approximation technique and the piecewise approximation technique is that in piecewise approximation, the resulting signal is not necessarily smooth due to the transitions from one segment to another at the knot points. The polynomial spline approximation smoothens the segment at the knot points hence resulting in a smooth segment. [8] is an example of an application for spline approximation.

2.3. Transform Domain Approximation Techniques

The approximation techniques introduced in the following subsections calculates the approximation for f by transforming it into another function F, approximating F and then an inverse transformation is applied to obtain the approximation of f which we call g. The approximation technique works on the transformed function and in the domain of the transformed function. That is why it is called Transform domain approximation.

2.3.1. Karhunen-Loeve Transform

Karhunen-Loeve Transform (KLT) is based on statistical properties of vector representation. In [21], it is explained as follows. Consider a population of random vectors of the form

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \tag{2.1}$$

The mean vector of the population is defined as

$$m_x = E\{x\} \tag{2.2}$$

where E{arg} is the expected value of the argument, and the subscript of m is associated with the population of x.

The covariance matrix of the vector population is defined as

$$C_x = E\{(x-m_x)(x-m_x)^T\} \tag{2.3}$$

where T is the vector transposition. Because x is n dimensional, C_x and $(x-m_x)(x-m_x)^T$ are matrices of order n X n. Element c_{ii} of C_x is the variance of x_i, the ith component of the x in the population, and c_{ij} of C_x is the covariance between elements x_i and x_j of these vectors. The matrix C_x is real and symmetric. If elements x_i and x_j are uncorrelated, their covariance is zero and therefore, c_{ij} = c_{ji}=0.

For M vector samples from a random population, the mean vector and covariance matrix can be approximated from the samples by

$$m_x = \frac{1}{M}\sum_{k=1}^{M} x_k \tag{2.4}$$

and

$$C_x = \frac{1}{M}\sum_{k=1}^{M} x_k x_k^T - m_x m_x^T \tag{2.5}$$

In [28] a summary of the KLT is presented. It concludes that KLT is an optimal transform because it completely decorrelates the signal in the transform domain, minimizes the MSE in bandwidth reduction or data compression, contains the most variance in the fewest number of transform coefficients and minimizes the total representation entropy of the sequence. This method is mainly used as a measuring tool against sub optimal transforms and it is not often applied because it is dependent on the input data.

2.3.2. Discrete Fourier Transform (DFT)

Fourier's theorem states that for any function f(x) defined in the interval $-\infty < x < +\infty$, it is possible to express it as a summation of a series of sine and cosine terms of increasing frequency. The Fourier transform of the function f(x), F(w), describes the amount of each frequency term that must be added together to make f(x). It is often described as [21],[20] as,

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N} \qquad (2.6)$$

Eq. 2.6 has a very high complexity because of a lot of multiplications and additions involved. The complexity reaches N^2. Hence, a proper decomposition of Eq. 2,6 can make the number of addition and multiplication operations proportional to N lg N. The decomposition procedure is called *Fast Fourier Transform (FFT)* algorithm. Eq. 2.6 is written in the form

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)W_N^{ux} \qquad (2.7)$$

where

$$W_N = e^{-j2\pi / N} \tag{2.8}$$

2.3.2. Discrete Cosine Transform (DCT)

Discrete Cosine Transform (DCT) is the most famous technique used for Images Lossy Compression techniques including JPEG and MPEG. Although it is not as optimal as Karahunen-Loeve transform, it is more efficient in an energy-packing sense than most of the other transform techniques such as Fourier transform and others. The main advantage of DCT is its less complexity that makes it fast in operation. It is defined in [21] as

$$C(u) = \alpha(u)\sum_{x=0}^{N-1} f(x)\cos\left[\frac{(2x+1)u\pi}{2N}\right] \tag{2.9}$$

In DCT, an image is divided into 8X8 blocks. Most of the block information is scattered near its corner, so what happens is that it starts from the left upper corner moving in a zigzag line along the cells, it saves only the first 4 cells from the zigzag path and chops the rest of the 60 cells. Hence, it reduced the image size by 1/16 of the original with almost same information.

Discrete Cosine Transform is a Transform Domain approximation technique.

A summary of some of the approximation techniques has been given by [12] among these is the Moment-Preserving method which is the focus of the present work. The summary is given in the following table:

Table 2.1.: Comparison of different approximation techniques

Approximation Method	Type of original function	type of knot values	Domain of operation
Linear Split [19]	Continous/Discrete	Non-Uniform	Signal
Split-and-Merge [14]	Continous/Discrete	Non-Uniform	Signal
Nearest-Neighbor [24]	Discrete	Uniform	Signal
Hologram-Like[17]	Discrete	Uniform	Signal
DFT	Discrete	N/A	Transform
DCT	Discrete	N/A	Transform
Moment-Preserving	Continous/Discrete	Uniform	Signal/Transform

All of the above mentioned approximation techniques are 1-D approximation techniques. Approximation in 2-D for planar curves and digitized images, which is still an important aspect of pattern recognition, image processing and graphics applications, has received a lot of attention by many authors such as [21], [9], [18] and many others. The results of the researches noted that the 2-D approximation techniques are in a way based on the 1-D techniques presented above. The same approach is followed through this thesis when the approximation for 2-D Surfaces and digitized images approximation are concerned.

2.4. Signal/Transform Domain Approximation Techniques

The thought of combining both domains for better approximation results has been addressed by few authors [15] [25]. Both addressed the problem of noise removal from over images. The first required that the image be band limited and the band limits be known. However, this constraint is not required in [25]. The removal of data in the second example is based on Projection onto Convex Sets (POCS) algorithms that use frequency and spatial domain information

In the present work, we choose the ***Interpolation*** method to be our benchmark. The reason for that is that it is the most famous technique for Piecewise Approximation. In addition, it is a signal domain technique and our proposed Moment-Preserving technique operates in the domain of the original signal though it utilizes both domains. As for digitized images, our benchmark is on ***DCT***, being the most famous technique for image compression.

2.4. Moment-Preserving Piecewise

The properties of geometric moments have been addressed by many researchers including [27], [29] and [5] to perform pattern recognition and classification. Because of the close relationship between the geometric moments and Fourier transform, they are considered to be operating in the transform domain.

Moments, by their geometric nature, go deeper into the function extracting its features rather than just considering its superficial surface features. They are considered to be the pillars upon which a function is built.

Moment-Preserving Piecewise approximation is a method introduced in 1994 by Professor John Oommen and his student Nguyen [13]. It has been experimented only on the Piecewise Linear case. This method relies on the idea of Moments.

Unlike the already existing approaches, the method utilizes the approximation of signals in both the Signal Domain and the Transform Domain. A paper about it was published in an IEEE Conference on 1997 [12].

Chapter 3

Moment Preserving Theory

3. Moment-Preserving Theory

3.1. Relationship between Fourier Transform and Moments

In signal theory, it is well known that the characteristic function of a random signal is the Fourier transform of its density function p(x)(with a reversal in sign), i.e.

$$\phi(jv) = E_x[\exp(jvx)] =$$

$$= \int \exp(jvx)\, p(x)\, dx = \tau\, \{p(x)\} \qquad (3.1)$$

If S_k represents the kth geometric moment of a function p(x), then

$$S_k = E_x[x^k] = \int x^k\ p(x)\, dx = (-j)^k\ d^k\phi(jv)/dv^k\,]_{v=0} \qquad (3.2)$$

Suppose that the characteristic function has a Taylor-series expansion, then

$$\phi(jv) = \Sigma_k\, d^k\phi(jv)/dv^k\,]_{v=0}\ v^k/\, k! = \Sigma_k\, S_k\, (jv)^k\, /\, k! \qquad (3.3)$$

Therefore, if the characteristic function has a Taylor-series expansion valid in some region about the origin, it is uniquely determined in this interval by the geometric moments. If the moments do uniquely determine the characteristic function (and hence the Fourier transform of the density function) then they also uniquely determine the density function. The consequence of this uniqueness is that a moment-preserving approximation to the function p(x) in the x-domain will also serve as an approximation constraint in the ν-domain.

3.2. *General Theory Proposed for Moment-Preserving Approximation*

Consider the kth moment of the variable x over the finite interval (i,j) of the function f(x) to be $S_k(i,j) = E_x[x^k]_{i,j}$. Let α be a scale reduction factor so that x = αy and hence we define a scaled moment as

$$\sigma_k(i,j) = \alpha^{-(k+1)}\, S_k(i,j) = \int_{i,j} y^k\ f(\alpha y)\, dy = E_{\alpha y}[y^k]_{i,j} \qquad (3.4)$$

With the function f(x) specified by a finite set of discrete points $\{x_i\ , i = 0\ ,1\ , ..\ n\}$, the scaled moment σ_k is the sum over all (n) segments (i,i+1) covering the above domain:

$$\sigma_k\ = \Sigma_i\, \sigma_k(i,i+1)\quad ,\ i = 0,1,\ldots,n\text{-}1 \qquad (3.5)$$

On the other hand, if we seek an approximating function g(x) defined at a set of distinct nodal points $\{z_j\ , j = 0\ ,1\ ,\ldots, m\}$, then over the interval between two nodal points (p,q) we obtain scaled moments $\mu_k(p,q)$ whose sum over the nodal intervals gives the scaled moments μ_k. For the moment preserving approximation, we require that

$$\sigma_k = \mu_k \qquad \text{for } k = 0,1, .., m \tag{3.6}$$

The above moment-preserving constraint leads to a system of m+1 equations

$$\sigma = \mathbf{E} \, . \, \mathbf{G} \tag{3.7}$$

where **E** is an m+1 by m+1 square matrix of coefficients depending on the approximating polynomial, and **G** is a column vector representing the approximations $g(z_j)$ to the function $f(x)$ at the nodal points $\{z_j \; , j = 0 \, ,1 \, ,...., m\}$.

3.3. 1-D Piecewise Approximation Mathematical Formulas Derivations

3.3.1. Piecewise Linear Approximation

Assuming that between the nodal points z_p and z_q the function is piecewise linear, we use Lagrange's classical formula.

$$g_{pq}(\alpha y) = (y_q - y)/(y_q - y_p)g(\alpha y_p) + (y - y_p)/(y_q - y_p)g(\alpha y_q)\,z$$
$$= (y_q - y)/(y_q - y_p)g(\alpha y_p) - (y_p - y)/(y_q - y_p)g(\alpha y_q)$$
$$= 1/(y_q - y_p)[(y_q - y)g(\alpha y_p) - (y_p - y)g(\alpha y_q)\ ] \qquad (3.8)$$

If we assume $1/(y_q - y_p)\,\&\,g(\alpha y_p)\,\&\,g(\alpha y_q)$ are constant values

and $\mu_k = \alpha^{-(k+1)}S_k(p,q) = \int_q^p y^k g_{pq}(\alpha y)dy = E[y_k]_q^p$

$$\therefore \mu_k(p,q) = g(\alpha y_p)/(y_q - y_p)\int_q^p y^k(y_q - y).dy - g(\alpha y_q)/(y_q - y_p)\int_q^p y^k(y_p - y).dy$$

Substitute y with t in the integral

$$= g(\alpha y_p)/(y_q - y_p)\int_q^p (t^k t_q - t^{k+1}).dt - g(\alpha y_q)/(y_q - y_p)\int_q^p (t^k t_p - t^{k+1}).dt$$

$$= g(\alpha y_p)/(y_q - y_p)\left[t_q\int_q^p t^k.dt - \int_q^p t^{k+1}.dt\right] - g(\alpha y_q)/(y_q - y_p)\left[t_p\int_q^p t^k.dt - \int_q^p t^{k+1}.dt\right]$$

Let $D_{pq}(t,n) = \int_q^p t^n.dt = (t_q^{n+1} - t_p^{n+1})/(n+1)$

$$\therefore \ = g(\alpha y_p)/(t_q - t_p)(t_q D(t,k) - D(t,k+1)) - g(\alpha y_q)/(t_q - t_p)(t_p D(t,k) - D(t,k+1))$$

$\because D_{pq}(t,0) = t_q - t_p$

$$= g(\alpha y_p)/D_{pq}(t,0)(t_q D(t,k) - D(t,k+1)) - g(\alpha y_q)/D_{pq}(t,0)(t_p D(t,k) - D(t,k+1))$$

Suppose $Bq(b,k,t) = \{t_b\ Dpq(t,k) - Dpq(t,k+1)\}/Dpq(t,0)$

$$\therefore \mu_k(p,q) = g(\alpha y_p)B_q(q,k,y) - g(\alpha y_q)B_q(p,k,y) \qquad (3.9)$$

For nodal points $\{z_j\ , j = 0, 1, \ldots, m\}$, then the scaled moments for the m segments will be:

$$\mu_k(0,1) \quad = g(\alpha y_0)\,B_1(1,k,y) - g(\alpha y_1)\,B_1(0,k,y)$$

$$\mu_k(1,2) \quad = g(\alpha y_1)\,B_2(2,k,y) - g(\alpha y_2)\,B_2(1,k,y)$$

……………………………………………………

$$\mu_k(m-1,m) = g(\alpha y_{m-1})\,B_m(m,k,y) - g(\alpha y_m)\,B_m(m-1,k,y) \qquad (3.10)$$

Let us define a function

$$C_j(k,t) = \begin{cases} B_1(1,k,t) & \text{for } j = 0 \\ B_{j+1}(j+1,k,t) - B_j(j-1,k,t) & \text{for } j = 1,\ldots,m-1 \\ -B_m(m-1,k,t) & \text{for } j = m \end{cases} \qquad (3.11)$$

Hence, the total scaled moment can be expressed as a vector μ with elements

$$\mu_k = \Sigma_j \, C_j(k,y) \, g(\alpha y_j) \quad , \; j\,,\, k = 0,1,\ldots,m \qquad (3.12)$$

Notice that in the above equation, the values of y_j represent the scaled coordinates of the nodal points. When the scaled coordinates of the actual function points are used, then we obtain the actual scaled moments vector **σ**. Accordingly, moment preservation *($\mu = \sigma$)* leads to the system

$$\boldsymbol{G = E^{1} \, . \, \sigma} \qquad (3.13)$$

where the elements of the square matrix **E** are given by

$$e(k,j) = C_j(k,y) \qquad (3.14)$$

3.3.2. Piecewise Quadratic Approximation

Here, we use equally spaced nodal points with an internal point z_r between the points z_p and z_q. With $\Delta = (z_r - z_p) = (z_q - z_r)$, Lagrange's formula can be written in the form: ,

$$2\Delta^2 g_{pq}(x) = (x - z_r)(x - z_q)\, g(z_p) - 2\,(x - z_p)(x - z_q)\, g(z_r) + \\ + (x - z_p)(x - z_r)\, g(z_q) \tag{3.15}$$

With $\mu_k(p,q) = \int_{p,q} y^k\, g_{pq}(\alpha y)\; dy$, we repeat steps similar to the ones illustrated in the piecewise linear approximation while adjusting the differences,

$$\mu_k(p,q) = g(\alpha y_p)\, B_{p,q}(r,q,k,y) - 2\, g(\alpha y_r)\, B_{p,q}(p,q,k,y) + \\ + g(\alpha y_q)\, B_{p,q}(p,r,k,y) \tag{3.16}$$

where $B_{p,q}(i,j,k,t) = [2/D^2_{pq}(t,0)]\,\{ D_{pq}(t,k+2) -$

$$- (t_i + t_j)\, D_{pq}(t,k+1) + t_i\, t_j\, D_{pq}(t,k)\, \} \tag{3.17}$$

Similar to the method used for the piecewise linear approximation, we may write

$$\mu_k = \Sigma_j\, C_j(k,y)\, g(\alpha y_j) \quad ,\; j\,,\, k = 0,1,\ldots,m \text{ , m even} \tag{3.18}$$

where

$$C_j(k,y) = \begin{cases} B_{0,2}(1,2,k,y) & \text{for j} = 0, \\ 2\, B_{j-1,j+1}(j-1,j+1,k,y) & \text{for j odd,} \\ B_{j-2,j}(j-2,j-1,k,y) + B_{j,j+2}(j+1,j+2,k,y) & \text{for j even,} \\ B_{m-2,m}(m-2,m-1,k,y) & \text{for j} = \text{m} \end{cases} \tag{3.19}$$

As before, the elements of the matrix **E** are $e(k,j) = C_j(k,y)$

3.3.3. *Piecewise Cubic Spline Approximation*

Lagrange's formula for the piecewise linear interpolation in the interval between z_p and z_q may be written in the form:

$$g_{pq}(z) = a\, g(z_p) + b\, g(z_q) \tag{3.20}$$

with $a = (z_q - z)/(z_q - z_p)$ and $b = (z - z_p)/(z_q - z_p) = 1 - a$

In a cubic spline approximation, we add a cubic polynomial whose second derivative varies linearly over the (p,q) interval and with zero values at z_p and z_q leading to

$$g_{pq}(z) = a\, g(z_p) + b\, g(z_q) + c\, g''(z_p) + d\, g''(z_q) \tag{3.21}$$

where

$c = (1/6)\,(a^3\text{-}a)(z_q - z_p)^2$ and $d = (1/6)\,(b^3\text{-}b)(z_q - z_p)^2$

Hence, the interpolating polynomial can be expressed as:

$g_{pq}(\alpha y) = [(y_q - y)/\Delta_y]\, g(\alpha y_p) + [(y - y_p)/\Delta_y]\, g(\alpha y_q) +$

$$+ P(\alpha y) + Q(\alpha y) \tag{3.22}$$

where $\Delta_y = y_q - y_p$,

$$P(\alpha y) = [(y_q - y)^3 - \Delta^2_y (y_q - y)] (\alpha^2/6\Delta_y)\ \varphi_p \tag{3.23}$$

$$Q(\alpha y) = [(y - y_p)^3 - \Delta^2_y (y - y_p)] (\alpha^2/6\Delta_y)\ \varphi_q \tag{3.24}$$

φ_p and φ_q are the second derivatives at the two nodal points.

Let $U_k(p,q) = \int_{p,q} y^k\ [P(\alpha y) + Q(\alpha y)]\ dy$, so that

$$\mu_k(p,q) - U_k(p,q) = g(\alpha y_p)\ B_q(q,k,y) - g(\alpha y_q)\ B_q(p,k,y) \tag{3.25}$$

where $B_q(b,k,t)$ is as defined for the linear case. Therefore, the problem is similar to the linear case except for the term $U_k(p,q)$. Evaluation of this term gives

$$\begin{aligned} U_k(p,q) = {} & D_{pq}(y,k)\ [\gamma_p\ \beta_0(q) - \gamma_q\ \beta_0(p)] + \\ & + D_{pq}(y,k+1)\ [\gamma_p\ \beta_1(q) - \gamma_q\ \beta_1(p)] + \\ & + D_{pq}(y,k+2)\ [\gamma_p\ \beta_2(q) - \gamma_q\ \beta_2(p)] - \\ & - D_{pq}(y,k+3)\ [\gamma_p - \gamma_q] \end{aligned} \tag{3.26}$$

where $\gamma_p = (\alpha^2/6\Delta_y)\ \varphi_p$, $\beta_0(i) = y^3_i - \Delta^2_y\, y_i$, $\beta_1(i) = \Delta^2_y - 3\, y^2_i$,

$\beta_2(i) = 3\, y_i$

It follows that

$$\mu_k - U_k = \Sigma_j C_j(k,y)\, g(\alpha y_j) \qquad , j , k = 0,1,\ldots,m \qquad (3.27)$$

where $U_k = \Sigma_j U_k(j-1,j) \qquad ,j = 1,2,\ldots,m$

Accordingly, moment preservation $(\mu = \sigma)$ leads to the system

$$G = E^{-1} . (\sigma - U_\mu + U_\sigma) \qquad (3.28)$$

In the above equation, **E** and σ are respectively the coefficient matrix and moments vector for the piecewise linear approximation, while the vectors U_μ and U_σ are computed using the nodal points and the actual function points, respectively.

Chapter 4

Moment-Preserving Results for 1-D Signal

4. Moment-Preserving Results for 1-D Signal

4.1. Introduction

The moment-preserving method has been applied to obtain piecewise approximations for various 1-D signals f(x). For a given approximation, nodal points $\{z_j, j = 0,1,\ldots, m\}$ were chosen to be evenly spaced across the x-space. The vector of approximants **G** at those points was computed using the linear, quadratic or cubic spline methods outlined above and an approximation g(x) to the function is obtained by the respective interpolation method. For comparison with usual interpolation techniques, an approximation h(x) was also obtained using the function values $f(z_j)$. We have used the mean-squared error (MSE) as a measure of the error norm between f(x) and each of the approximations g(x) and h(x).

As an example, we show here the results for the function

$$f(x) = 2\sin(0.2\,x) + 5\cos(0.3\,x) + w*r \tag{4.1}$$

where r is a uniformly distributed random noise {0,1} and w is an amplitude factor. For the above example, we have used an x-domain covering 10 blocks with 161 function points and 5 nodal points in each block (i.e. one nodal point every 40 function points). For more accuracy and to reduce the need for reconditioning the matrices in the inversion process, we have used a scale factor $\alpha = x_{n-1}$.

4.2. Piecewise Linear Approximation

Fig.4.1. Demonstrates the results of applying the traditional Linear Interpolation method and the suggested Moment-Preserving method.

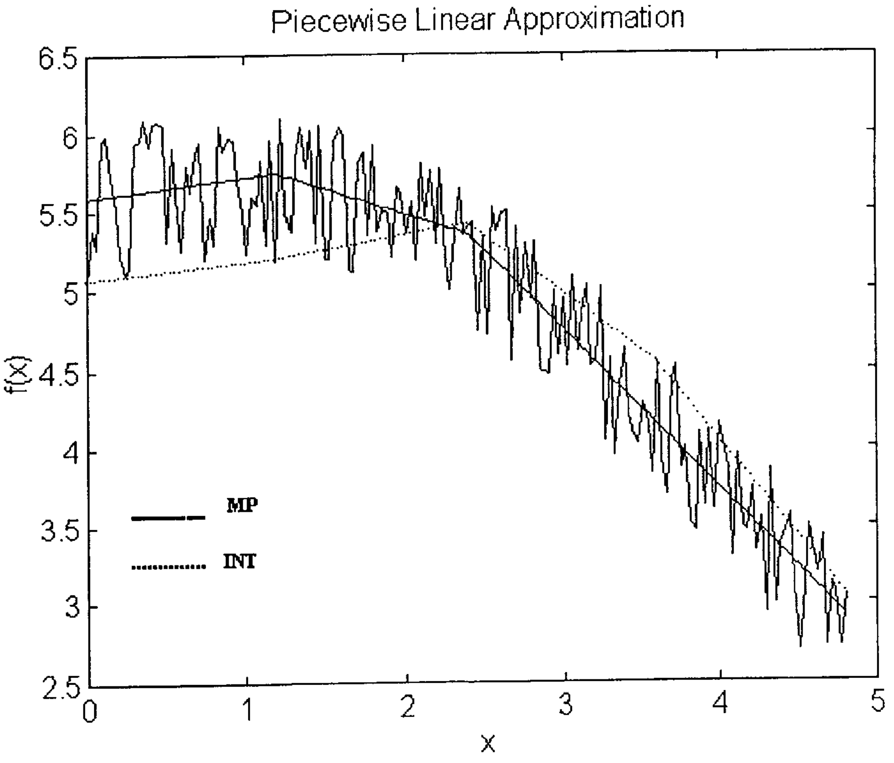


Fig.4.1.Piecewise Linear Approximation for 1-D Signal.

MP: Moment-Preserving. Int:Interpolation

The figure shows the original function with noise, the linear Interpolation method is represented using the dotted line and the Moment-Preserving method is the black line.

It is obvious that the Moment-Preserving approximation is closer to the original function while the linear Interpolation method is very inaccurate compared to the original function.

This is because Linear Interpolation considers the points on the surface of the function and the Moment-Preserving method considers the weight of the function in terms of its Moment.

In such graph, the noise level is of high amplitude over the function and trying to take the nodal points on the surface of the function causes the result to be very inaccurate. The Moment-Preserving method takes the nodal points and does not just join.

4.3. Piecewise Quadratic Approximation

Fig.4.2. demonstrates the results of applying the traditional Quadratic Interpolation method and the suggested Moment-Preserving method.

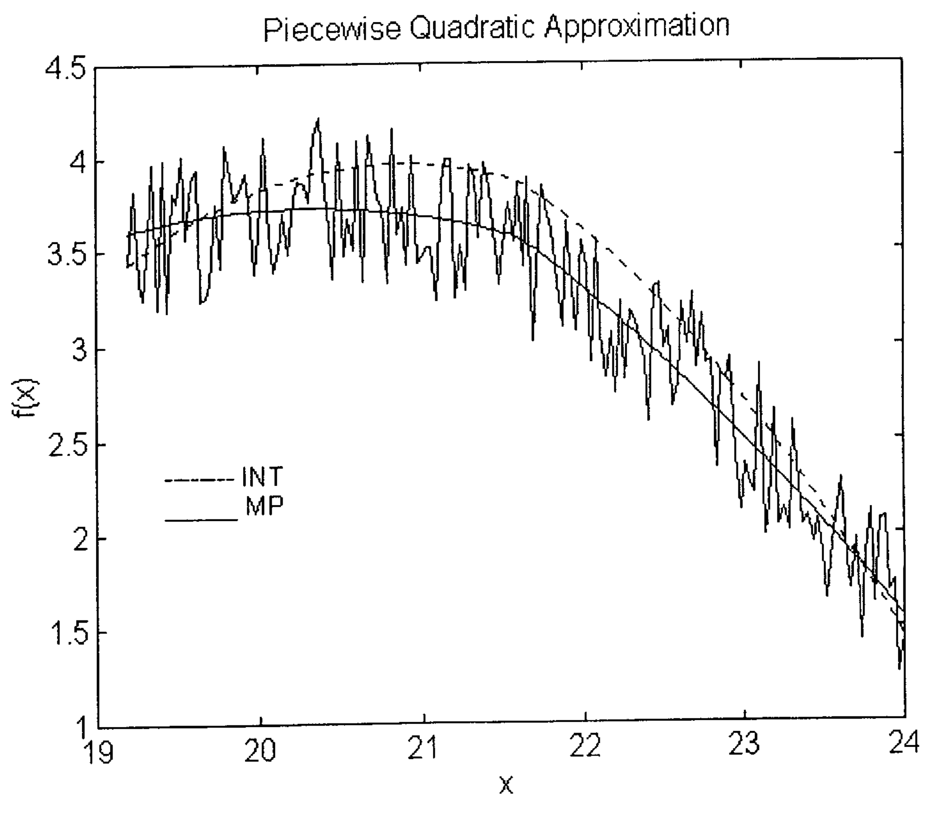


Fig 4.2.Piecewise Quadratic Approximation for 1-D Signal

MP: Moment-Preserving. Int:Interpolation

The figure shows the original function with noise, the Quadratic Interpolation method is represented using the dotted line and the Moment-Preserving method is the black line.

The Moment-Preserving approximation is closer to the original function while the Quadratic Interpolation method is results in a distorted signal compared to the original function.

4.3.1. Zoom In Piecewise Quadratic Approximation

The Piecewise Quadratic Approximation is the experimented for the first time in the present work. The paper that introduced the Moment-Preserving technique in 1997 focused only on Linear Approximation. Here, we show that the method is successful also for higher order (Quadratic) polynomial.

Fig.4.3. shows the effect of various noise levels on Quadratic Interpolation and Moment-Preserving Piecewise approximation method.

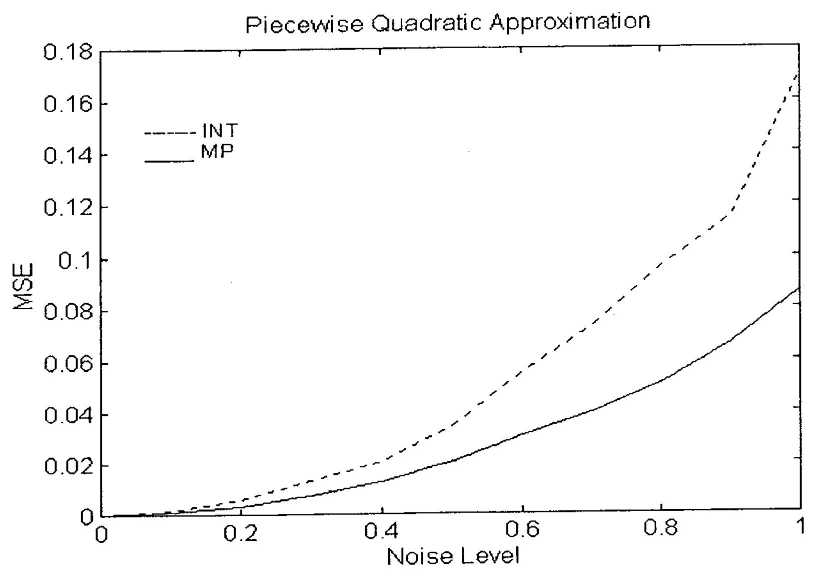


Fig.4.3. Mean Square Error versus Noise Level for Piecewise Quadratic Approximation for 1-D Signal.

MP: Moment-Preserving. Int:Interpolation

4.4. Piecewise Cubic Spline Approximation

Fig.4.4. demonstrates the results of applying the traditional Cubic Spline Interpolation method and the proposed Moment-Preserving method.

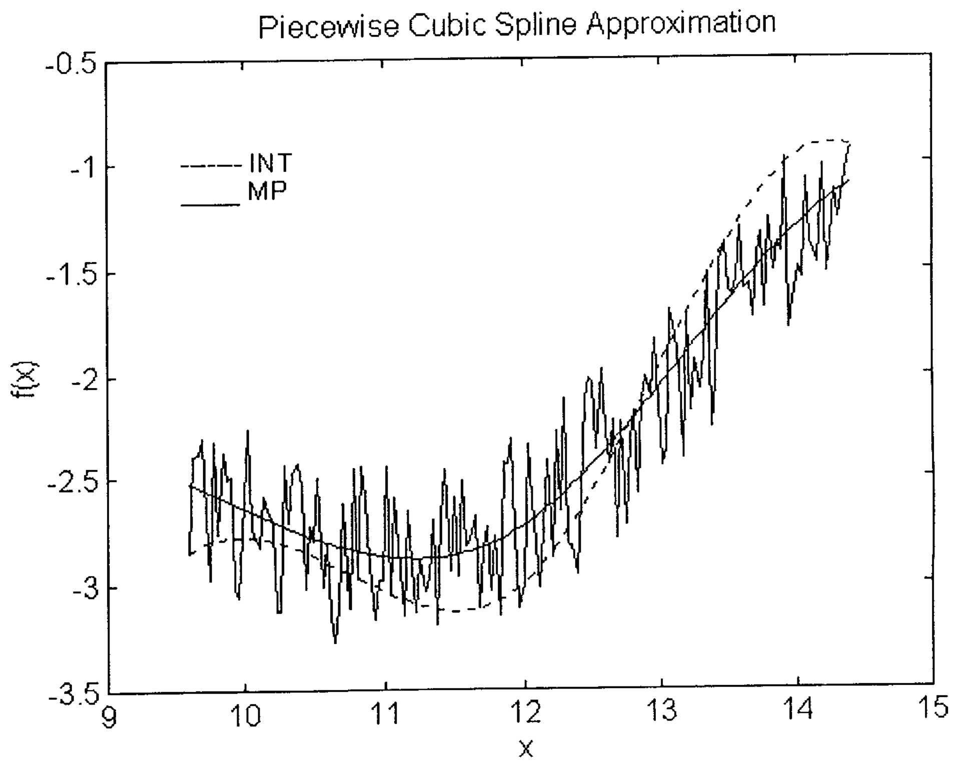


Fig.4.4. Piecewise Cubic Spline Approximation for 1-D Signal.

MP: Moment-Preserving. Int:Interpolation

The figure shows the original function with noise, the Quadratic Interpolation method is represented using the dotted line and the Moment-Preserving method is the black line.

Again, it is clear that the Moment-Preserving approximation is closer to the original function while the Quadratic Interpolation method is not as close to the original function.

Linear Piecewise Approximation and Cubic Piecewise Approximation are very similar in the concept. The difference between them appears at the knot points. The Linear Piecewise Approximated curve is sharp at the nodal points while the application of Cubic Spline makes the curve smoother and this makes the curve look better.

4.5. Comparison between the 3 Approaches

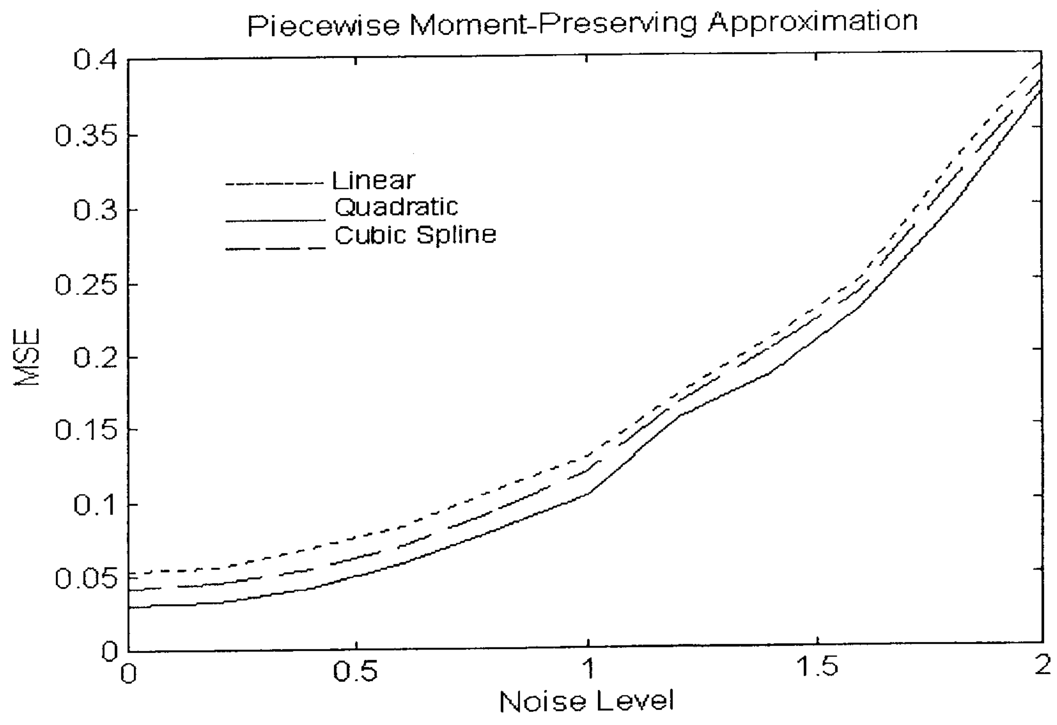


Fig.4.5. Comparison between 3 approaches for 1-D Signal

Linear: MP Piecewise Linear Approximation

Quadratic: MP Piecewise Quadratic Approximation

Cubic Spline: MP Piecewise Cubic Spline Approximation

Fig4.5. illustrates a comparison between Moment-Preserving Piecewise Linear, Quadratic and Cubic Spline Approximation.

As shown in the graph, that the Quadratic method gives the least percentage of error followed by Cubic Spline. The least accurate results happen to be when applying the Piecewise Linear Moment-Preserving approximation.

This does not denounce the effectiveness of Linear Moment-Preserving approximation when compared to the Linear Interpolation traditional method. However, it is the least efficient when compared to Moment-Preserving higher order polynomial approximations.

Chapter 5

Moment-Preserving Results for 2-D Closed Boundary

5. Moment-Preserving Results for 2- D Closed Boundary

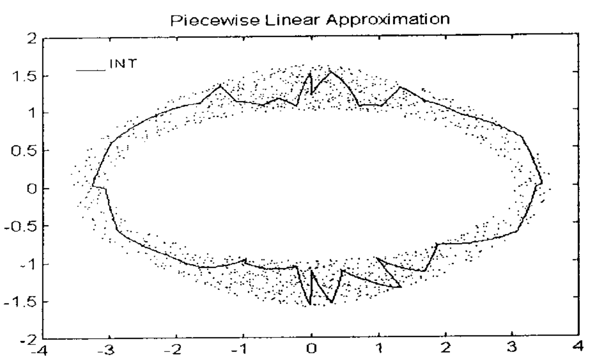
5.1. Introduction

Approximation of 2-D Closed Boundaries works as a method for the recognition and classification of deformable shapes. In our present demonstration, the deformation comes in the form of noise. The removal of noise extracts the actual shape of the boundary; hence, the comparison between objects, the searching of an object in a database or the classification of the approximated object into a globally consistent interpretation can take place.

The present method, Moment-Preserving piecewise approximation, has always been tested on noisy closed binary boundaries by obtaining their r(θ) signatures and computing the approximations for the resulting 1-D signatures. After approximating the 1-D signatures, treating the signature as a signal, the boundary is brought back from the approximated signature.

The θ-space, normalized to $\{0,2\pi\}$, is divided into 8 sectors. In each sector, one nodal θ point is selected every 40 points.

5.2. Piecewise Linear Approximation



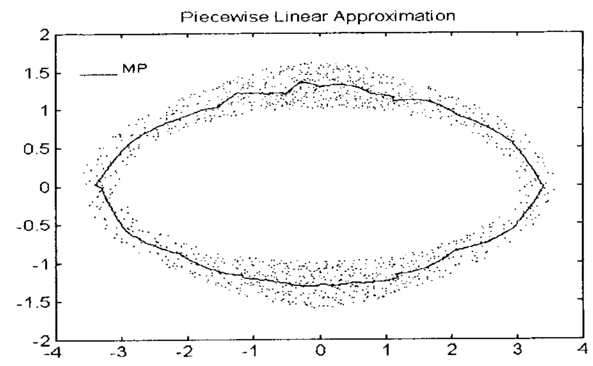


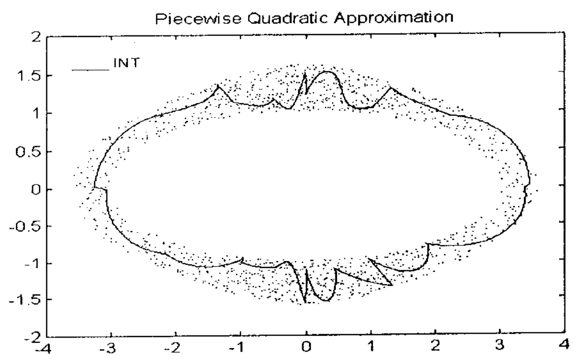
Fig.5.1.a Int. Piecewise Linear Approx. for 2-D Closed Boundaries

Fig.5.1.b MP Piecewise Linear Approx for 2-D Closed Boundaries

Fig.5.1.a. shows a sample of the results obtained from a noisy closed boundary when applying the Linear interpolation. Fig5.1.b. shows the boundary resulting from the approximation using the moment-preserving method.

The problem with using the linear piecewise approximations without moment preservation is very clear in the presence of significant noise levels. Values of the function at the nodal points could be the extreme points of noise amplitude leading to severe fluctuations.

5.3. Piecewise Quadratic Approximation



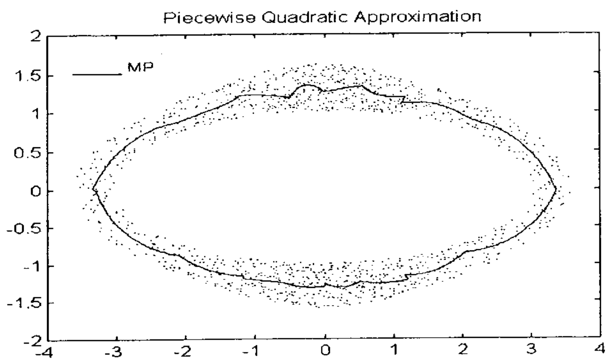


Fig.5.2.a Int. Piecewise Quadratic Approx. for 2-D Closed Boundaries

Fig.5.2.b MP Piecewise Quadratic Approx for 2-D Closed Boundaries

Fig.5.2.a. shows a sample of the results obtained from a noisy closed boundary using Quadratic interpolation. Fig.5.2.b. shows the boundary resulting from the approximation using the moment-preserving method.

The next sub-section takes a closer look at the quadratic piecewise approximation

5.3.1. Zoom In Piecewise Approximation

A typical dependence of the MSE on the noise level in a boundary is shown in Fig.5.3. It can be seen that imposing the moment-preserving criteria leads to a significant reduction in the MSE resulting from a high level of noise on the boundary.

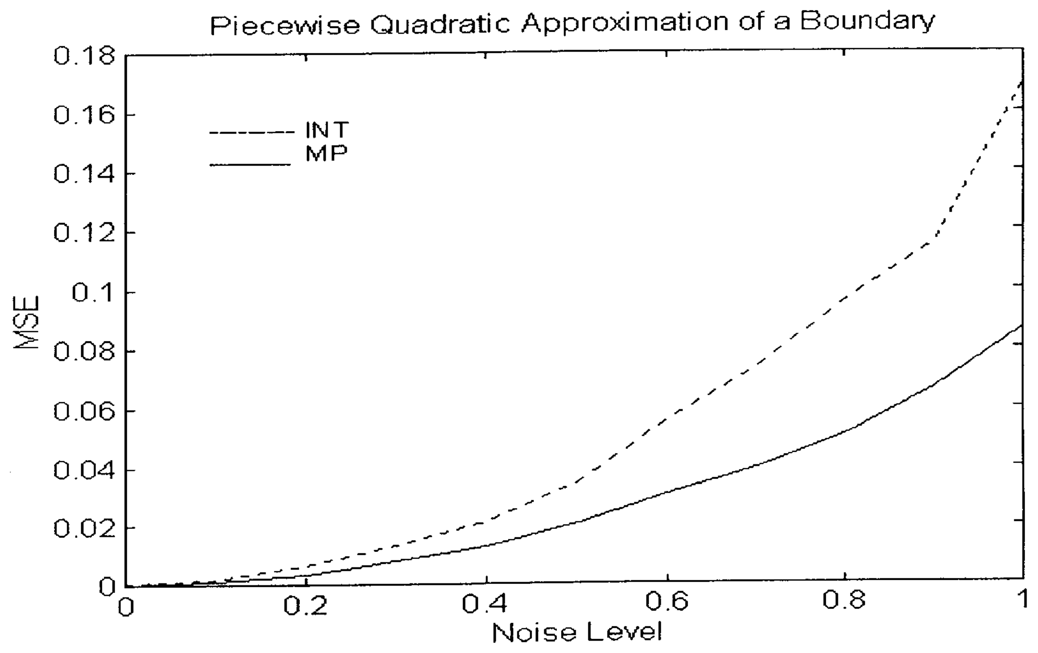
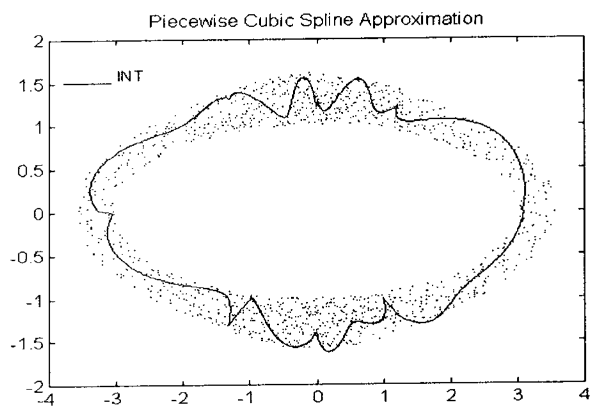


Fig5.3. Mean Square Error versus Noise Level for Piecewise Quadratic Approximation for 2-D Closed Boundary

Int: Interpolation; MP: Moment-Preserving

5.4. Piecewise Cubic Spline Approximation



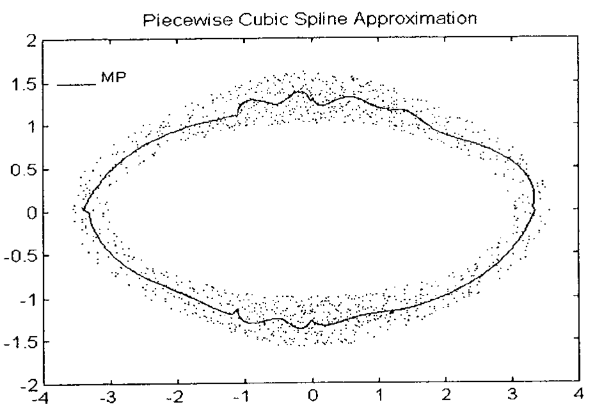


Fig.5.4.a Int. PiecewiseCubic-Spline Approx. for 2-D Closed Boundaries

Fig.5.4.b MP Piecewise Cubic-Spline Approx for 2-D Closed Boundaries

Fig.5.4.a. shows a sample of the results obtained for a noisy closed boundary using cubic spline interpolation. Fig.5.4.b. shows the boundary resulting from the approximation using the moment-preserving method.

5.5. Comparison between the 3 Approaches

It should be noted that the degree of accuracy of moment-preserving approximation for noisy boundaries does not depend strongly on the degree of the approximating polynomial. The figures of the results show clearly that the MSE for the same noise level will not differ significantly between linear, quadratic and cubic spline approximations. Therefore, a low order polynomial with moment preservation can serve to define the skeleton of a noisy shape to a degree of accuracy significantly higher than the usual piecewise interpolation techniques.

Chapter 6

Moment-Preserving Results for Digitized Images

6. Moment-Preserving Results for Digitized Images

6.1. Introduction

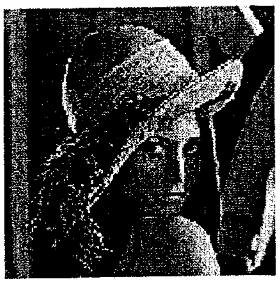
An image, considered as a matrix of pixel values, can be approximated to achieve lossy compression. The most efficient technique used so far is the Discrete Cosine Transform (DCT), which operates in the frequency domain rather than the spatial domain of the image. The usual method for applying the DCT is to process image blocks of size 8 x 8 pixels using a 2-D DCT. A zigzag mapping is used to preserve DCT coefficients of maximum variance while setting the rest of the 64 coefficients to zero. Beside its low computational complexity, the advantages of using the DCT is that it packs the information in the maximum variance coefficients and that it minimizes the boundary discontinuities between the blocks.

In order to apply moment-preserving techniques to a digitized image, we have to use a piecewise approximation in the spatial domain. Block processing can be used after mapping the block pixels into a 1-D vector from which nodal points can be selected for the approximation process using the methods developed in the present work.

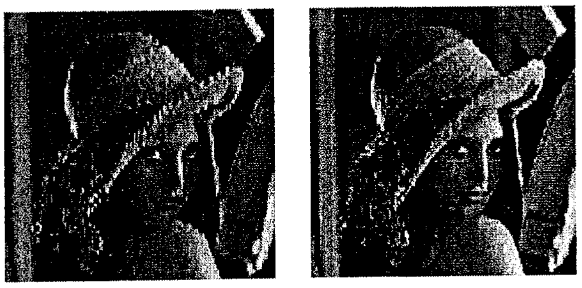
We have experimented with different sub-image geometries and have selected column processing as the block processing method. In this case, a block is chosen to be a sub-column with a number of pixels depending on the degree of variance in the column. As an example, we have used a block size of 17 pixels with 5 nodal points (one nodal pixel every 4 pixels). The image whose results are show here is a 256 grey level image of size 256 x 256 pixels.

6.2. Linear & Quadratic Piecewise Approximation

Fig. 6.1.a. shows the original image. Using usual linear interpolation and moment-preserving linear approximation, the resulting images are shown in Fig. 6.1.b. and Fig. 6.1.c. respectively. Table 6.1. gives the MSE for these two cases. It can be seen that the use of the moment-preserving approximation significantly enhances the quality of the image relative to the usual interpolation method. Similar conclusions can be derived from the use of higher approximating polynomials, as shown in Figs. 6.2.a. and 6.2.b. and Table 6.1. for the case of quadratic approximation.

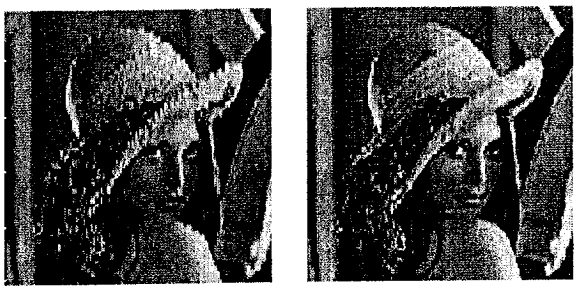


(a) Original Picture



(b) Linear Int, (c) Linear MP.

Fig.6.1.



(a) Quadratic Int. (b) Quadratic MP.

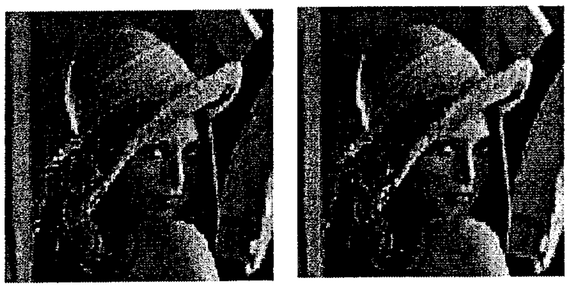
Fig.6.2.

Table 6.1. MSE for Image Linear & Quadratic Approximations (x 10^{-4})

Method	Interpolation	Moment-Preserving
Linear	33	24
Quadratic	37	23

In order to compare the present spatial approximation method with frequency domain processing, we have used the DCT with a block size of 8 x 8 pixels and preserving the highest variance K coefficients of the zigzag mapped 64 coefficients for each block. We have also processed the blocks after adding N nodal coefficients from the remaining (64 – K). These nodal points have been samples every 4 coefficients and their values have been determined by a moment-preserving quadratic method. With K = 12, and N = 3, the packing ratio is approximately comparable to the case of spatial processing shown above.

Fig.6.3. shows the reconstructed images using the above parameters, and Table 6.2. gives the MSE for different values of K and N.



(a) DCT (b) DCT with MP

Fig. 6.3.

Table 6.2. MSE of DCT Processing for Different Packing Parameters

K	8	8	12	12	16	16
N	0	3	0	3	0	3

It is to be noted that the MSE of 0.0023 obtained for quadratic spatial processing (see Table 6.1.) is quite close to the value of 0.0022 obtained from the DCT with K = 12 and N = 3 moment preserving quadratic nodal points. This indicates that moment-preserving approximations of images in the spatial domain can compete with transform methods such as the DCT as far as packing efficiency is concerned. However, we must recognize the advantage of low computational complexity offered by the DCT relative to moment-preserving spatial processing. This is in view of the latter methods being dependent on matrix inversion computations that increase their computational cost for image approximations.

Chapter 7

Moment-Preserving Results for 2-D Surfaces

7. Moment-Preserving Results for 2-D Surfaces

7.1. Introduction

Through this chapter, we demonstrate the effect of applying the moment-preserving constraint on approximation of 2-D Surface. We compare the result with the Bilinear Interpolation technique.

In Bilinear Interpolation, we need four points on the corner of a square like shape to interpolate. So, a mesh of knot points over the 2-D surface is chosen and bilinear interpolation is applied.

The moment-preserving constraint treats the 2-D Surface as a set of lines in row and a set of lines in column. Moment-Preserving technique deals with each row as a signal and approximates each and every row of the 2-D Surface "i.e. represented by 2-D Matrix". The same procedure is then repeated for every column. The result of rows approximation and the result of columns approximation are then averaged together to give us the resulting approximation matrix.

We experiment the method on 3 Surfaces as shown below. The surfaces as:

1. $f(x,y) = \sin\frac{\sqrt{x^2+y^2}+eps}{\sqrt{x^2+y^2}+eps}$

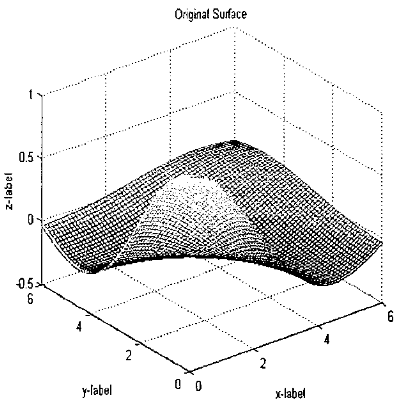
2. $f(x,y) = pi * \frac{(\sqrt{x^2+y^2}+eps)^2}{\sqrt{x^2+y^2}+eps}$

3. $f(x,y) = peaks(row(x))$

In the case of the above-mentioned 3 functions, we take 5 nodal points on each row and 5 nodal points for each column. Hence, for a matrix of nXn, we have a mesh of 25 nodal points.

7.2. 2-D Surface

Shown below the results of applying the moment-preserving constraint on 2-D surfaces. The figures illustrate the results of using the above functions respectively. Figures 7.1.a., 7.2.a.and 7.3.a. represent the 3 pure surfaces that we will demonstrate on. Figures 7.1.b., 7.2.b.and 7.3.b. are the surfaces after random noise has been added to them. Figures 7.1.c., 7.2.c.and 7.3.c. show the effect of moment-preserving constraint when applied on the noisy surface for approximation in purpose of extracting the pure original surface while Figures 7.1.d., 7.2.d.and 7.3.d. are the result of applying bilinear interpolation to the same noisy surfaces.



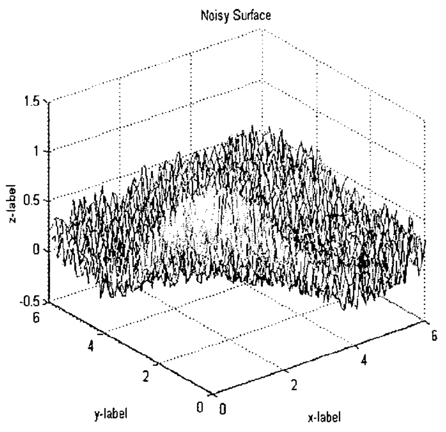
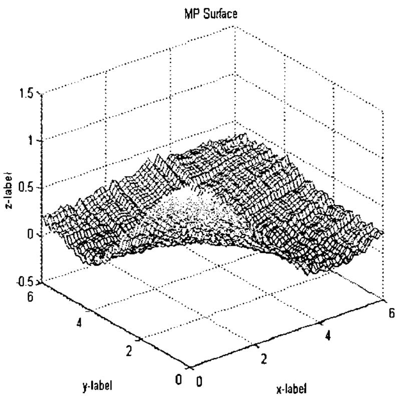


Fig. 7.1.a. Original Surface Fig.7.1.b. Noisy Surface



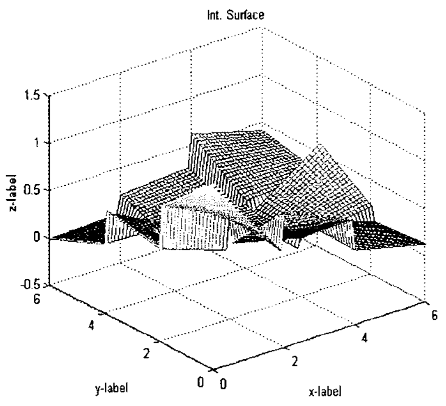
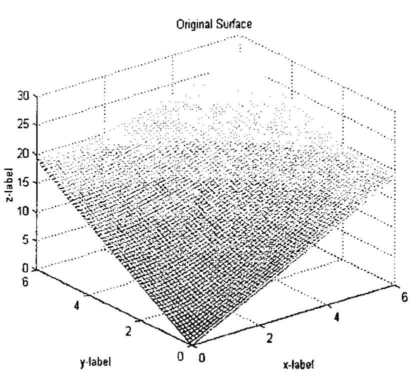


Fig. 7.1.c. MP Surface Fig.7.1.d. Int.Surface



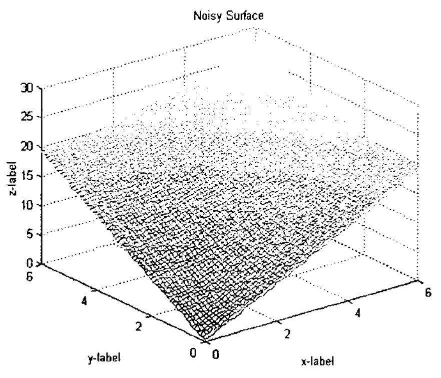
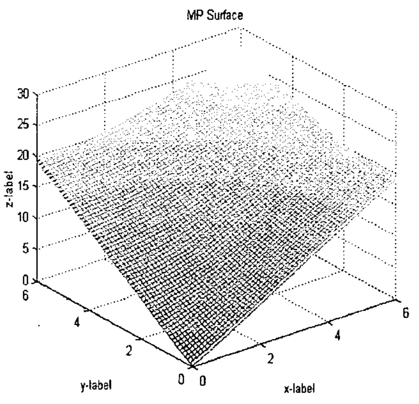


Fig.7.2.a.Original Surface Fig.7.2.b. Noisy Surface



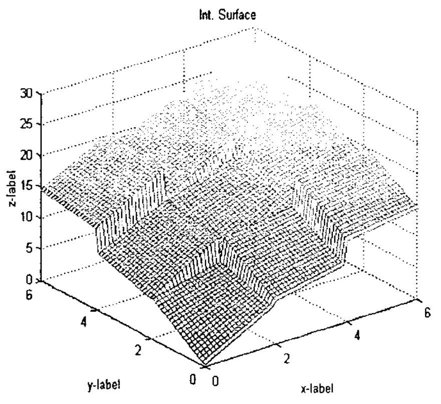
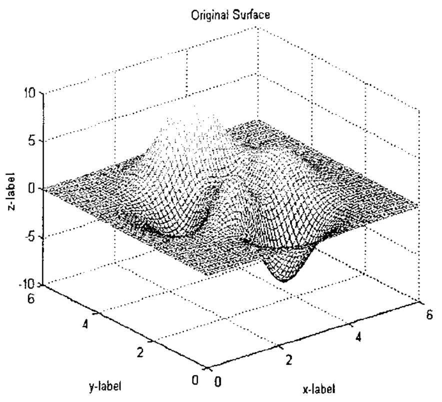


Fig.7.2.c.MP Surface Fig.7.2.d. Int. Surface.



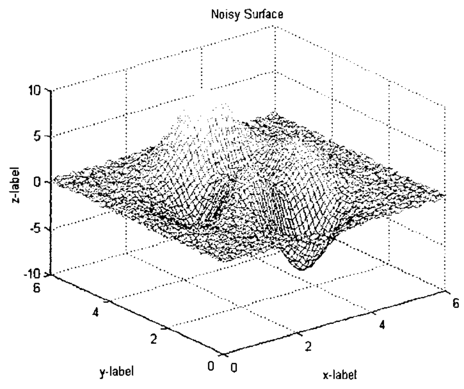
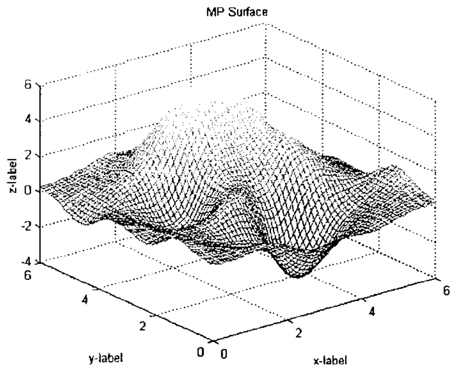


Fig.7.3.a.Original Surface Fig.7.3.b. Noisy Surface



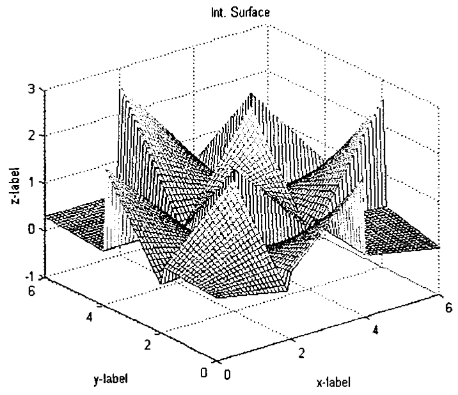


Fig.7.3.c.MP Surface Fig.7.3.d. Int. Surface

The above 3 illustrative examples show the effect of applying linear piecewise moment-preserving approximation relative to bilinear interpolation. By the naked eye, it is clear that moment-preserving technique demonstrates a much better result than bilinear interpolation. Fig. 7.4. reinforces this idea by demonstrating the MSE relative to noise level for both Bilinear Interpolation and Linear Piecewise Approximation. The Bilinear interpolation appears as a dotted line while the Moment-Preserving appears as the black line.

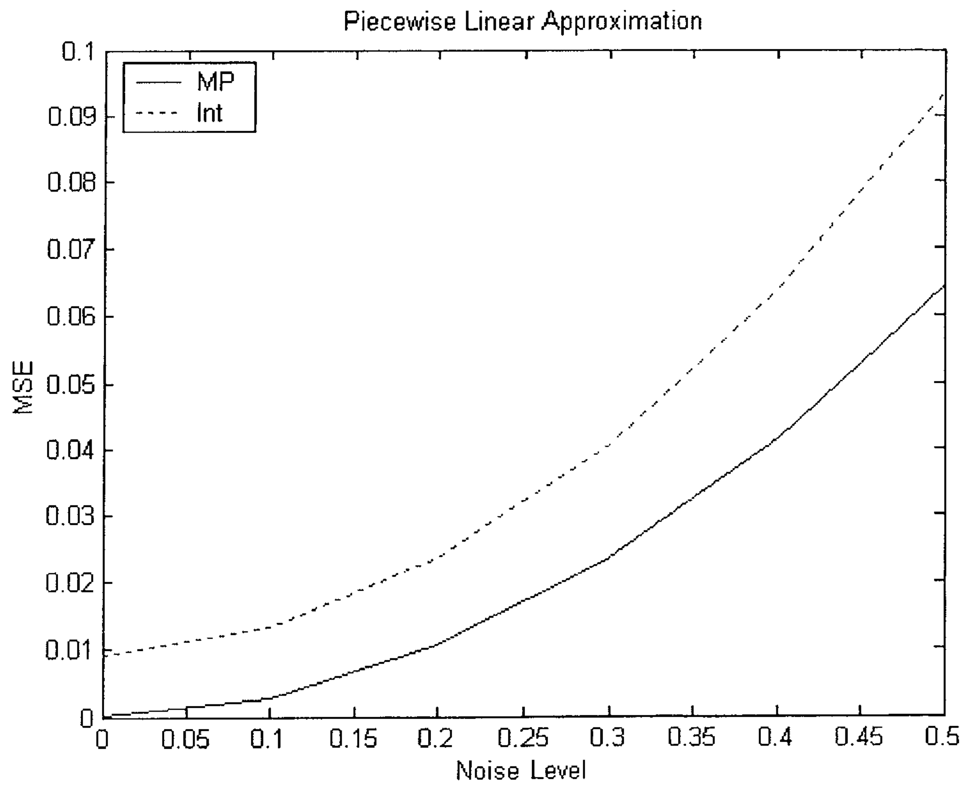


Fig.7.4.Mean Square Error versus Noise Level for Bilinear Interpolation and Linear MP Piecewise Approximation for 2-D Surface.

Int: Interpolation. MP: Moment-Preserving

Chapter 8

Conclusion and Future Work

8. Conclusion and Future Work

8.1. Summary and Conclusion

In the present work, experimentation of the potential of using the moment-preserving constraint as a technique for signals and images approximation is demonstrated. Its ability to combine the Signal Domain and the Transform Domain makes it an attractive technique for experimentation.

Through this thesis, deriving the mathematical formulas as well as experimenting them on the Moment-Preserving Piecewise approximation technique was illustrated in:

- 1-D Signals using the Linear, Quadratic and Cubic Spline polynomial orders,
- 2-D Closed Boundaries using the Linear, Quadratic and Cubic Spline polynomial orders,
- Digitized Images using the Linear and Quadratic polynomial orders and
- 2-D Surfaces using the Linear polynomial order.

In the case of 1-D Signals, the Moment-Preserving Piecewise Approximation performance is much better than Interpolation on the 3 experimented polynomial orders. On the other hand, the higher the Moment-Preserving polynomial order, the better was the accuracy of the resulting signal. The Moment-Preserving

Piecewise Quadratic approximation is the best performer among the 3 demonstrated polynomial orders.

In the case of 2-D Closed, the Moment-Preserving Piecewise Approximation performance is much better than Interpolation on the 3 experimented polynomial orders. However, unlike the 1-D signals, the 3 illustrated polynomial orders gave almost the same result when applied in Moment-Preserving Piecewise approximation

As for Digitized Images, Moment-Preserving Piecewise approximation gave results as close in accuracy as that achieved by the Discrete Cosine Transform. Since Discrete Cosine Transform is the most famous technique for Image compression, this means that the effect of Moment-Preserving Piecewise technique is promising. However, Moment-Preserving Piecewise techniques lag behind Discrete Cosine Transform as far as time is concerned.

An interesting observation is that the behavior of Moment-Preserving compared with Interpolation on various polynomial levels is not the same. This is clear from the difference between the efficiency of Quadratic Moment-Preserving piecewise approximation over the Linear Moment-Preserving piecewise approximation, which is a reversed result in the case of Interpolation.

The Linear Moment-Preserving piecewise approximation experiment on 2-D Surfaces added more credit to the method since again it proves its successfulness over the Bilinear Interpolation for various surfaces.

Moment-Preserving piecewise approximation technique, tested in various experiments, proves to be an excellent method for approximation serving the purpose of noise-reduction and signal-space compression. However, this higher accuracy in approximation has its price, which is paid in terms of the time consumed to perform the approximation process. Aside from that, Moment-Preserving piecewise approximation is a great performer for approximation.

Performance is an issue of concern that is observed during generating the experimental results. The aim of the mathematical derivation is to find the approximation function G. This requires dealing with a system of m+1 equations (i.e. m is the number of chosen knot points), which requires matrix operations to solve it. The operations of matrices lengthens the running time because its complexity is almost O((m+1)^2) which highly increases as m increases in value. However, in this type of problem i.e. approximation of signals, usually a compromise between Time and Quality is a question. In our case, the addresses Quality more than Time; the fact that makes it beneficial to safety critical cases like the medical case we referred to earlier.

8.2. Future Work

My original inclination when I started working on this thesis was to extend the Moment-Preserving constraint to 3-dimensional space. In 3-D space, Moments are expected to be of great help. This is because of the synchronization process it is expected to provide when viewing a 3-D object from various angles. Hence, it is promising when approximating 3-D objects.

The achieved results and the future work could be applied on medical images. We could not put that in the thesis time plan due to the huge time consumption in the mathematical derivations, which, together with the implementation of the theories for demonstrating the strength of the Moment-Preserving method, were very challenging. Though it is time consuming compared to other present techniques. Medical images could be highly enhanced using our method.

In medicine, diagnosis of diseases is a serious issue. Without an accurate approximation technique, wrong diagnosis could happen which affects the life of human being. So, we expect consumption of time not to be an obstacle in similar critical fields.

References

References

[1] Goneid, A. and Abou Seif,, S. "Moment-Preserving Piecewise Approximation for 1-D and 2-D Signals", CCCT'04, vol 4.

[2] Ahmed, H., Natarajan, T., and Rao, K. R., *Discrete Cosine Transform*, Discrete Transforms and Their Applications, Van Nostrand Reinhold Company Inc., pp 9-12, 1985.

[3] Ballard, D. H., and Brown, C. M., Computer Vision, Prentice-Hall, Inc., 1982.

[4] Bellman, R., *On the approximation of curves by line segments using dynamic programming*, Comm. ACM, Vol. 4, No. 6, pg. 284, June 1961.

[5] Chone-Huah Lo, "3-D Moment Forms: Their Construction and Application to Object Identification and Positioning.", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol 11, No 10, October 1989.

[6] de Boor, C., A Practical Guide to Splines, Springer-Verlag, New York, 1978.

[7] Hanselman, D. and Littlefield, B. , "The Student Edition of MatLab", The MathWorks, Inc., 1997

[8] Kuijt F.and Van Damme, R.M.J. , "A linear Approach to shape preserving spline approximation", University of Twente, July 1998

[9] Hildebrand, F. B., *Introduction to Numerical Analysis*, 2nd Edition, Dover Publications, 1987.

[10] Anil, K. ,"Fundamentals of Digital Image Processing", Prentice Hall, Inc. 1989

[11] Montanari, U., *A Note on Minimal Length Polygonal Approximation to a Digitized Contour*, Communications of the A.C.M., Vol. 13, No. 1, pp.41-47, January 1970.

[12] Nguyen, T.B. and Oommen, B.J. ,"Moment-Preserving Piecewise Linear Approximations of Signals and Images", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no.1, pp. 84-91, 1997

[13] Nguyen, T.B. , *Moment-Preserving Piecewise Linear Approximations of Signals and Images*, master's thesis, Carleton Univ., Ottawa, Ontario, Canada, 1994.

[14] Pavlidis, T., *Waveform Segmentation Through Functional Approximation*, I.E.E.E. Transactions on Computers, Vol. C-22, No. 7, pp. 689-697, July 1973.

[15] Papoulis, A. "A New Algorithm in Spectral Analysis and Band-Limited Extrapolation", IEEE Trans. Cir. & Sys.,22,No.9, 735-742, 1975

[16] Pavlidis, T., and Horowitz, S. L. *Segmentation of Plane Curves*, I.E.E.E. Transactions on Computers, Vol. C-23, No. 8, pp. 860-870, August 1974.

[17] Prusinkiewicz, P., and Christopher, M., *Hologram-like transmission of pictures*, The Visual Computer, Springer Verlag, pp. 185-193, 1985.

[18] Pratt, W. K., *Digital Image Processing*, Wiley-Interscience, New York, 1978.

[19] Ramer, U., *An Iterative Procedure for the Polygonal Approximation of Plane Curves*, Computer Graphics and Image Processing, Vol. 1, pp. 244-256, 1972.

[20] Rosin, P. L., and West, G. A. W., *Techniques for Segmenting Image Curves into Meaningful Descriptions*, Pattern Recognition, Vol. 24, No. 7, pp. 643-652, 1991.

[21] Gonzalez, R. C. and Woods, R. E. , Digital Image Processing. Addison-Wesley Publishing Co., Inc., 1993

[22] Sklansky, J., *Recognition of Convex Blobs*, Pattern Recognition, Pergamon Press, Vol.2, pp. 3-10, 1970.

[23] Stone, H., *Approximation of curves by line segments*, Mathematics of Computation, Vol. 15, pp 40-47, 1961.

[24] Tanimoto, S. L., *Image Transmission with Gross Information First*, Computer Graphics and Image Processing, Vol. 9, pp. 72-76, 1979.

[25] Totsuka, T. and Hirani, A. N., "Combining Frequencey and Spatial Domain Information for fast Interactive Image Noise Removal", Sony Corporation Research Center.

[26] Press, W, H. , Teukolsky, S. A. , Veltering, W. T. and Falnnercy, B. P. , "Numerical Recipes in C." pp.105-128, 1995

[27] Hu, M., Visual Pattern Recognition by Moment Invariants, I.E.E.E. Transactions on Information Theory, Vol. 2, pp. 179-187, 1962.

[28] Rao, K. R., and Yip, P., *Discrete Cosine Transform - Algorithms, Advantages, Applications*, Academic Press, 1990.

[29] Pavlidis, T., *A Review of Algorithms for Shape Analysis*, Computer Graphics and Image Processing, Vol. 7, pp. 243-258, 1978.

Appendix A.

Published Paper

Appendix A: Published Paper

Moment-Preserving Piecewise Approximations for 1-D and 2-D Signals

Amr Goneid and **Soha AbuSeif**
Computer Science Department, The American University in Cairo

ABSTRACT

Approximations of 1-D and 2-D signals are important for noise reduction and signal space compression. Current techniques address the approximation process either in the signal space or in its transform space but not in both. A moment-preserving constraint can couple both spaces for better evaluation of approximations at nodal signal points. We present results for applying moment-preserving piecewise approximations of 1-D and 2-D signals using Linear, Quadratic and Cubic Spline polynomials. Results demonstrate higher accuracy of the method compared to approximations obtained without moment preservation.

Keywords: Signal Processing, Image Processing, Pattern Analysis, Piecewise Approximations

1. INTRODUCTION

The process of approximating digitised 1-D signals and images has always been a regular practice when there was the need for reducing noise or for the purpose of lossy compression of the signals. In the 1-D case, the noisy signal is considered to be a function f(x) sampled at a set of distinct points $\{x_i , i = 0 ,1 , .. n\}$. The objective of an approximation method is to find an approximating function g(x) defined at a set of distinct nodal points $\{z_j , j = 0 ,1 ,...., m\}$, $m < n$, subject to a certain error minimization criterion. The approximated values can be joined using some interpolation technique. Obviously, this methodology also applies to 2-D signals or images.
Because the approximation problem is relevant to many applications in signal processing, pattern analysis and image processing, a considerable amount of research has been advanced in this area. Existing well known approaches derive the approximations either through constraints in the signal domain or in its transform domain but not in both. In the signal domain, several simple sub-sampling techniques have been commonly used, e.g.[1] beside the more complex least squares polynomial methodology. Other approaches utilize piecewise linear approximations, e.g. [2] as well as the smoother polynomial splines, e.g. [3]. In the transform domain, common approaches are the FFT, DCT, and KLT transforms [e.g. 4,5,6]. More recently, a method developed in [7] approaches the problem by deriving the approximation in the signal domain while preserving a finite number of geometric moments that are related to its Fourier domain. The method has been applied to piecewise linear approximations. In the present work, we extend the moment-preserving method for 1-D and 2-D signals to higher order polynomials. Specifically, we present the derivation of the moment preserving method for linear, quadratic and cubic spline polynomials. We also present results of applying this technique to 1-D signals, closed binary boundaries and images.

2. THEORY

2.1 General

In signal theory, it is well known that the characteristic function of a random signal is the Fourier transform of its density function p(x), i.e.

$$\phi(jv) = E_x[\exp(jvx)] = \\ = \int \exp(jvx)\, p(x)\, dx = \tau\,\{p(x)\} \qquad (1)$$

If S_k represents the kth geometric moment of a function p(x), then

$$S_k = E_x[x^k] = \int x^k\, p(x)\, dx = (-j)^k\, d^k\phi(jv)/dv^k\,]_{v=0} \qquad (2)$$

Suppose that the characteristic function has a Taylor-series expansion, then

$$\phi(jv) = \Sigma_k\, d^k\phi(jv)/dv^k\,]_{v=0}\, v^k/k! = \Sigma_k\, S_k\,(jv)^k/k! \qquad (3)$$

Therefore, if the characteristic function has a Taylor-series expansion valid in some region about the origin, it is uniquely determined in this interval by the geometric moments. If the moments do uniquely determine the characteristic function (and hence the Fourier transform of the density function) then they also uniquely determine the density function. The consequence of this uniqueness is that a moment-preserving approximation to the function p(x) in the x-domain will also serve as an approximation constraint in the v-domain.

2.2 Moment-Preserving Approximation

Consider the kth moment of the variable x over the finite interval (i,j) of the function f(x) to be $S_k(i,j) = E_x[x^k]_{i,j}$. Let α be a scale reduction factor so that $x = \alpha y$ and hence we define a scaled moment as

$$\sigma_k(i,j) = \alpha^{-(k+1)}\, S_k(i,j) = \int_{i,j} y^k\, f(\alpha y)\, dy = E_{\alpha y}[y^k]_{i,j} \qquad (4)$$

With the function f(x) specified by a finite set of discrete points $\{x_i , i = 0 ,1 , .. n\}$, the scaled moment σ_k is the sum over all (n) segments (i,i+1) covering the above domain:

$$\sigma_k = \Sigma_i\, \sigma_k(i,i+1) \quad , \ i = 0,1,\ldots,n-1 \qquad (5)$$

On the other hand, if we seek an approximating function g(x) defined at a set of distinct nodal points $\{z_j , j = 0 ,1 ,...., m\}$, then over the interval between two nodal points (p,q) we obtain scaled moments $\mu_k(p,q)$ whose sum over the nodal intervals gives the scaled moments μ_k. For the moment preserving approximation, we require that

$$\sigma_k = \mu_k \qquad \text{for } k = 0,1,..,m \qquad (6)$$

The above moment-preserving constraint leads to a system of m+1 equations

$$\boldsymbol{\sigma} = \mathbf{E} \cdot \mathbf{G} \qquad (7)$$

where **E** is an m+1 by m+1 square matrix of coefficients depending on the approximating polynomial, and **G** is a column vector representing the approximations $g(z_j)$ to the function f(x) at the nodal points $\{z_j \;, j = 0, 1, \ldots, m\}$.

3. PIECEWISE LINEAR APPROXIMATION

Assuming that between the nodal points z_p and z_q the function is piecewise linear, we use Lagrange's classical formula

$$g_{pq}(\alpha y) = (y_q - y)/(y_q - y_p)\, g(\alpha y_p) + + (y - y_p)/(y_q - y_p)\, g(\alpha y_q) \qquad (8)$$

to compute the kth scaled moment over that region:

$$\mu_k(p,q) = \alpha^{-(k+1)} S_k(p,q) = = \int_{p,q} y^k g_{pq}(\alpha y)\, dy = E_{\alpha y}[y^k]_{p,q}$$

With $D_{pq}(t,n) = \int_{p,q} t^n\, dt = (t_q^{n+1} - t_p^{n+1})/(n+1)$ then evaluation of the integral gives

$$\mu_k(p,q) = g(\alpha y_p)\, B_q(q,k,y) - g(\alpha y_q)\, B_q(p,k,y) \qquad (9)$$

where $B_q(b,k,t) = \{t_b\, D_{pq}(t,k) - D_{pq}(t,k+1)\} / D_{pq}(t,0)$

For nodal points $\{z_j \;, j = 0, 1, \ldots, m\}$, then the scaled moments for the m segments will be:

$\mu_k(0,1) = g(\alpha y_0)\, B_1(1,k,y) - g(\alpha y_1)\, B_1(0,k,y)$
$\mu_k(1,2) = g(\alpha y_1)\, B_2(2,k,y) - g(\alpha y_2)\, B_2(1,k,y)$
....
$\mu_k(m-1,m) = g(\alpha y_{m-1})\, B_m(m,k,y) - g(\alpha y_m)\, B_m(m-1,k,y)$

Let us define a function

$$C_j(k,t) = \begin{cases} B_1(1,k,t) & \text{for } j = 0 \\ B_{j+1}(j+1,k,t) - B_j(j-1,k,t) & \text{for } j = 1,\ldots,m-1 \\ -\, B_m(m-1,k,t) & \text{for } j = m \end{cases} \qquad (10)$$

Hence, the total scaled moment can be expressed as a vector $\boldsymbol{\mu}$ with elements

$$\mu_k = \Sigma_j\, C_j(k,y)\, g(\alpha y_j) \qquad , j, k = 0,1,\ldots,m \qquad (11)$$

Notice that in the above equation, the values of y_j represent the scaled coordinates of the nodal points. When the scaled coordinates of the actual function points are used, then we obtain the actual scaled moments vector $\boldsymbol{\sigma}$. Accordingly, moment preservation ($\boldsymbol{\mu} = \boldsymbol{\sigma}$) leads to the system

$$\mathbf{G} = \mathbf{E}^{-1} \cdot \boldsymbol{\sigma} \qquad (12)$$

where the elements of the square matrix **E** are given by $e(k,j) = C_j(k,y)$

4. PIECEWISE QUADRATIC APPROXIMATION

Here, we use equally spaced nodal points with an internal point z_r between the points z_p and z_q. With $\Delta = (z_r - z_p) = (z_q - z_r)$, Lagrange's formula can be written in the form:

$$2\Delta^2 g_{pq}(x) = (x - z_r)(x - z_q)\, g(z_p) - 2(x - z_p)(x - z_q)\, g(z_r) + + (x - z_p)(x - z_r)\, g(z_q) \qquad (13)$$

With $\mu_k(p,q) = \int_{p,q} y^k g_{pq}(\alpha y)\, dy$ one obtains

$$\mu_k(p,q) = g(\alpha y_p)\, B_{p,q}(r,q,k,y) - 2\, g(\alpha y_r)\, B_{p,q}(p,q,k,y) + + g(\alpha y_q)\, B_{p,q}(p,r,k,y) \qquad (14)$$

where $B_{p,q}(i,j,k,t) = [2/ D^2_{pq}(t,0)]\, \{ D_{pq}(t,k+2) - - (t_i + t_j)\, D_{pq}(t,k+1) + t_i\, t_j\, D_{pq}(t,k) \}$

Similar to the method used for the piecewise linear approximation, we may write

$\mu_k = \Sigma_j\, C_j(k,y)\, g(\alpha y_j) \qquad , j, k = 0,1,\ldots,m$, m even

$$\text{where} \quad C_j(k,y) = \begin{cases} B_{0,2}(1,2,k,y) & \text{for } j = 0, \\ -2\, B_{j-1,j+1}(j-1,j+1,k,y) & \text{for } j \text{ odd}, \\ B_{j-2,j}(j-2,j-1,k,y) + B_{j,j+2}(j+1,j+2,k,y) & \text{for } j \text{ even}, \\ B_{m-2,m}(m-2,m-1,k,y) & \text{for } j = m \end{cases} \qquad (15)$$

As before, the elements of the matrix **E** are $e(k,j) = C_j(k,y)$

5. PIECEWISE CUBIC SPLINE APPROXIMATION

Lagrange's formula for the piecewise linear interpolation in the interval between z_p and z_q may be written in the form:

$$g_{pq}(z) = a\, g(z_p) + b\, g(z_q)$$

with $a = (z_q - z)/(z_q - z_p)$ and $b = (z - z_p)/(z_q - z_p) = 1 - a$
In a cubic spline approximation, we add a cubic polynomial whose second derivative varies linearly over the (p,q) interval and with zero values at z_p and z_q leading to

$$g_{pq}(z) = a\, g(z_p) + b\, g(z_q) + c\, g''(z_p) + d\, g''(z_q)$$

where
$c = (1/6)(a^3 - a)(z_q - z_p)^2$ and $d = (1/6)(b^3 - b)(z_q - z_p)^2$

Hence, the interpolating polynomial can be expressed as:

$$g_{pq}(\alpha y) = [(y_q - y)/\Delta_y]\, g(\alpha y_p) + [(y - y_p)/\Delta_y]\, g(\alpha y_q) + + P(\alpha y) + Q(\alpha y) \qquad (16)$$

where $\Delta_y = y_q - y_p$,
$P(\alpha y) = [(y_q - y)^3 - \Delta^2_y\, (y_q - y)]\, (\alpha^2/6\Delta_y)\, \varphi_p$

$Q(\alpha y) = [(y - y_p)^3 - \Delta^2_y (y - y_p)] (\alpha^2/6\Delta_y) \varphi_q$
φ_p and φ_q are the second derivatives at the two nodal points.

Let $U_k(p,q) = \int_{p,q} y^k [P(\alpha y) + Q(\alpha y)]\ dy$, so that

$$\mu_k(p,q) - U_k(p,q) = g(\alpha y_p) B_q(q,k,y) - g(\alpha y_q) B_q(p,k,y)$$

where $B_q(b,k,t)$ is as defined for the linear case. Therefore, the problem is similar to the linear case except for the term $U_k(p,q)$. Evaluation of this term gives

$$\begin{aligned} U_k(p,q) = \ & D_{pq}(y,k) [\gamma_p \beta_0(q) - \gamma_q \beta_0(p)] + \\ & + D_{pq}(y,k+1) [\gamma_p \beta_1(q) - \gamma_q \beta_1(p)] + \\ & + D_{pq}(y,k+2) [\gamma_p \beta_2(q) - \gamma_q \beta_2(p)] - \\ & - D_{pq}(y,k+3) [\gamma_p - \gamma_q] \end{aligned} \qquad (17)$$

where $\gamma_p = (\alpha^2/6\Delta_y) \varphi_p$, $\beta_0(i) = y^3_i - \Delta^2_y y_i$, $\beta_1(i) = \Delta^2_y - 3 y^2_i$, $\beta_2(i) = 3 y_i$

It follows that

$$\mu_k - U_k = \Sigma_j C_j(k,y) g(\alpha y_j) \quad , j , k = 0,1,\ldots,m \qquad (18)$$

where $U_k = \Sigma_j U_k(j-1,j) \quad , j = 1,2,\ldots,m$

Accordingly, moment preservation ($\mu = \sigma$) leads to the system

$$\mathbf{G} = \mathbf{E}^{-1} . (\sigma - \mathbf{U}_\mu + \mathbf{U}_\sigma) \qquad (19)$$

In the above equation, **E** and σ are respectively the coefficient matrix and moments vector for the piecewise linear approximation, while the vectors $\mathbf{U}_\mu$ and $\mathbf{U}_\sigma$ are computed using the nodal points and the actual function points, respectively.

6. EXPERIMENTAL RESULTS

6.1 1-D Signal Approximation

The above moment-preserving method has been applied to obtain piecewise approximations for various 1-D signals f(x). For a given approximation, nodal points $\{z_j, j = 0,1,\ldots, m\}$were chosen to be evenly spaced across the x-space. The vector of approximants **G** at those points was computed using the linear, quadratic or cubic spline methods outlined above and an approximation g(x) to the function is obtained by the respective interpolation method. For comparison with usual interpolation techniques, an approximation h(x) was also obtained using the function values $f(z_j)$. We have used the mean-squared error (MSE) as a measure of the error norm between f(x) and each of the approximations g(x) and h(x).
As an example, we show here the results for the function

$$f(x) = 2\sin(0.2\,x) + 5\cos(0.3\,x) + w\,r \qquad (20)$$

where r is a uniformly distributed random noise {0,1} and w is an amplitude factor. For the above example, we have used an x-domain covering 10 blocks with 161 function points and 5 nodal points in each block (i.e. one nodal point every 40 function points). For more accuracy and to reduce the need for reconditioning the matrices in the inversion process, we have used a scale factor $\alpha = x_{n-1}$.
As an example, Fig.(1) compares the obtained piecewise approximations without and with moment-preserving constraint

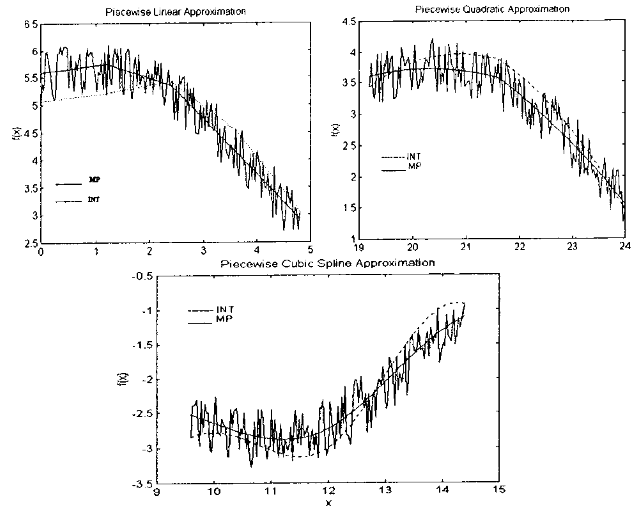


Fig.(1) Piecewise linear, quadratic and cubic spline approximations.

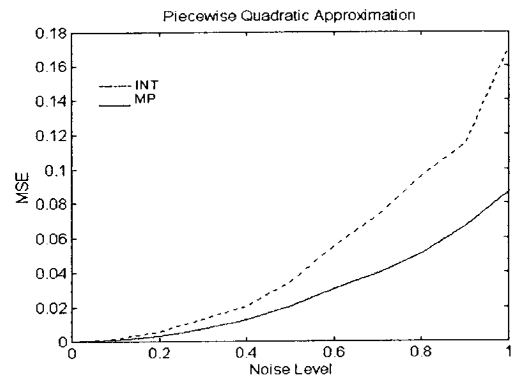


Fig.(2) MSE at different noise levels (Quadratic)

for the linear, quadratic and cubic spline approximations. Fig.(2) shows the MSE as a function of noise level for the quadratic approximation. The solid (MP) and dotted (INT) curves relate to moment-preserving and usual interpolation MSE, respectively. The above figure shows the expected result that the MSE increases by increase of the noise level and also clearly illustrates how the MSE is significantly decreased, even

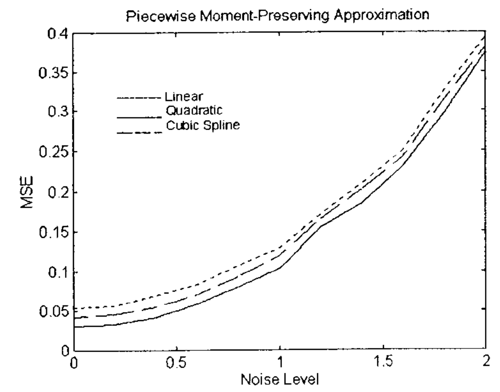


Fig.(3) MSE for the different moment-preserving approximations.

at high noise level, by imposing moment preservation. Similar results are obtained for linear and cubic spline approximations.
In Fig.(3), we show the results for the MSE obtained for the different moment-preserving approximations at different noise levels. The shown results indicate that the quadratic approximation is better than the other two. Although cubic splines produce smoother approximations, higher accuracy is obtained with the quadratic approximation.

6.2 2-D Closed Boundaries

We have also applied the present methodology to noisy closed binary boundaries by obtaining their r(θ) signatures and computing the approximations for the resulting 1-D signatures. The θ-space, normalized to {0,2π}, is divided into 8 sectors. In each sector, one nodal θ point is selected every 40 points.

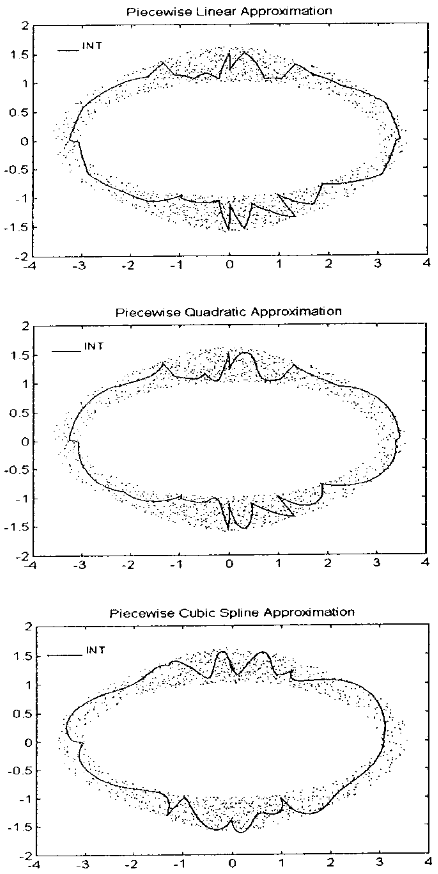


Fig.(4) Interpolation approximation for a boundary

A sample of the results obtained for a noisy closed boundary using usual interpolation is shown in Fig.(4) for linear, quadratic and cubic spline approximations. The problem with using such piecewise approximations without moment preservation is quite clear in the presence of significant noise levels. Values of the function at the nodal points could well be extremum points of noise amplitude leading to fluctuations between these points.
The boundaries shown in Fig.(4) have been approximated with moment preservation with the results shown in Fig.(5). It can be clearly seen that moment-preserving approximations are superior to the usual interpolation methods for boundary representation.

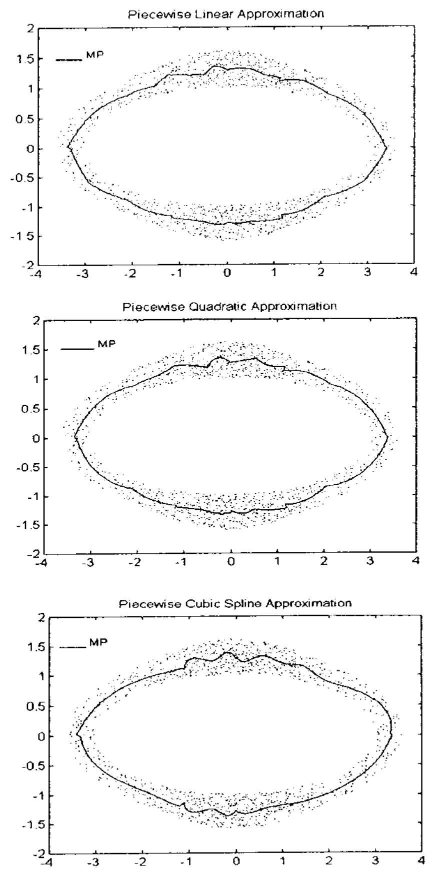


Fig.(5) Moment-preserving approximation for a boundary

A typical dependence of the MSE on the noise level in a boundary is shown in Fig.(6). It can be seen that imposing the moment-preserving criteria leads to a significant reduction in the MSE resulting from a high level of noise on the boundary.

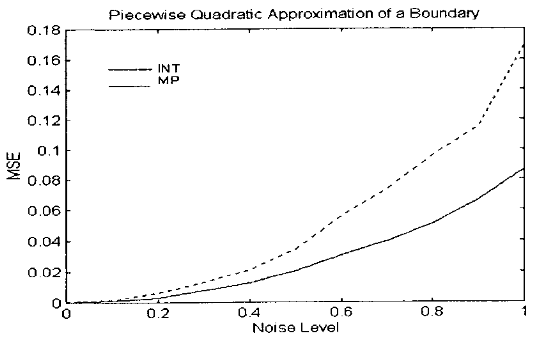


Fig.(6) Typical MSE graph for boundary approximation.

It should be noted that the degree of accuracy of moment-preserving approximation for noisy boundaries does not depend strongly on the degree of the approximating polynomial. The results we obtained show that the MSE for the same noise level will not differ significantly between linear, quadratic and cubic spline approximations. Therefore, a low order polynomial with moment preservation can serve to define the skeleton of a noisy shape to a degree of accuracy significantly higher than the usual piecewise interpolation techniques. Fig.(7) shows some examples of skeleton detection using quadratic moment-preserving piecewise approximations.

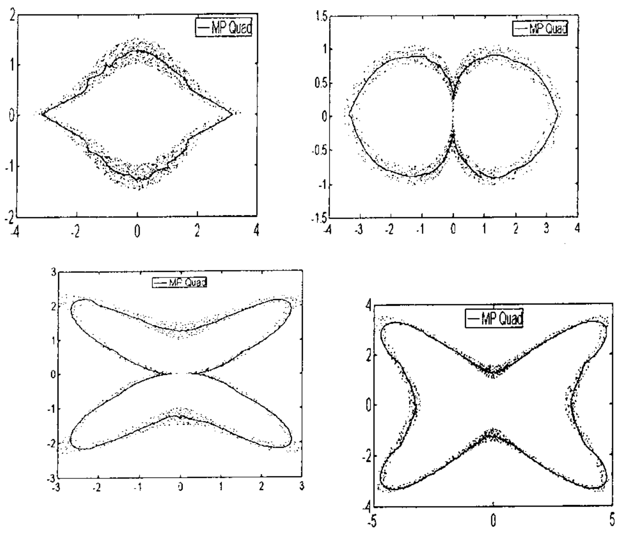


Fig.(7) Examples of skeleton detection from noisy boundaries using quadratic moment-preserving approximations.

6.3 Approximation of Digitized Images

An image, considered as a matrix of pixel values, can be approximated to achieve lossy compression. The most efficient technique used so far is the Discrete Cosine Transform (DCT) which operates in the frequency domain rather than the spatial domain of the image. The usual method for applying the DCT is to process image blocks of size 8 x 8 pixels using a 2-D DCT. A zigzag mapping is used to preserve DCT coefficients of maximum variance while setting the rest of the 64 coefficients to zero. Beside its low computational complexity, the advantages of using the DCT is that it packs the information in the maximum variance coefficients and that it minimizes the boundary discontinuities between the blocks.

In order to apply moment-preserving techniques to a digitised image, we have to use a piecewise approximation in the spatial domain. Block processing can be used after mapping the block pixels into a 1-D vector from which nodal points can be selected for the approximation process using the methods developed in the present work.

We have experimented with different sub-image geometries and have selected column processing as the block processing method. In this case, a block is chosen to be a sub-column with a number of pixels depending on the degree of variance in the column. As an example, we have used a block size of 17 pixels with 5 nodal points (one nodal pixel every 4 pixels). The image whose results are show here is a 256 grey level image of size 256 x 256 pixels.

Fig. (8a) shows the original image. Using usual linear interpolation and moment-preserving linear approximation, the resulting images are shown in Fig. (8b) and Fig. (8c), respectively. Table (1) gives the MSE for these two cases. It can be seen that the use of the moment-preserving approximation significantly enhances the quality of the image relative to the usual interpolation method. Similar conclusions can be derived from the use of higher approximating polynomials, as shown in Figs. (9a) and (9b) and Table (1) for the case of quadratic approximation.

(a) Original Image

(b) Interpolation (c) Moment-Preserving

Fig. (8) Image linear approximation

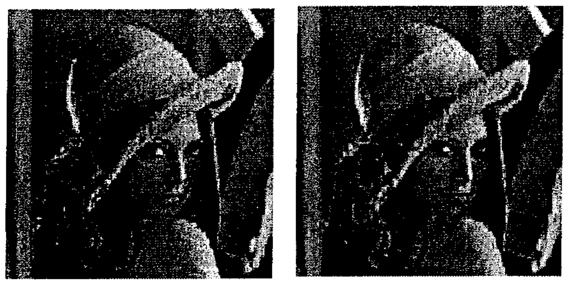
(b) Interpolation (c) Moment-Preserving

Fig. (9) Image quadratic approximation

Table (1) MSE for Image Linear & Quadratic Approximations (x 10^{-4})

Method	Interpolation	Moment-Preserving
Linear	33	24
Quadratic	37	23

In order to compare the present spatial approximation method with frequency domain processing, we have used the DCT with a block size of 8 x 8 pixels and preserving the highest variance K coefficients of the zigzag mapped 64 coefficients for each block. We have also processed the blocks after adding N nodal coefficients from the remaining (64 – K). These nodal points have been samples every 4 coefficients and their values have been determined by a moment-preserving quadratic method. With K = 12, and N = 3, the packing ratio is approximately comparable to the case of spatial processing shown above.
Fig.(10) shows the reconstructed images using the above parameters, and Table (2) gives the MSE for different values of K and N.



(a) DCT, K=12, N = 0 (b) DCT, K=12, N=3

Fig.(10) DCT Processing without and with N moment-preserving quadratic nodal points.

Table (2) MSE of DCT Processing for Different Packing Parameters

K	8	8	12	12	16	16
N	0	3	0	3	0	3
MSE (x 10^{-4})	30	26	25	22	19	19

It is to be noted that the MSE of 0.0023 obtained for quadratic spatial processing (see Table (1)) is quite close to the value of 0.0022 obtained from the DCT with K = 12 and N = 3 moment preserving quadratic nodal points. This indicates that moment-preserving approximations of images in the spatial domain can compete with transform methods such as the DCT as far as packing efficiency is concerned. However, we must recognize the advantage of low computational complexity offered by the DCT relative to moment-preserving spatial processing. This is in view of the latter methods being dependent on matrix inversion computations that increase their computational cost for image approximations.

7. CONCLUSION

Moment-preserving piecewise approximations have been derived for linear, quadratic and cubic spline polynomials. The application of such approximations to noisy 1-D signals and to 2-D boundaries has proven to be superior to the use of ordinary interpolation methods, specially for high level noise content. In case of 1-D signals, the quadratic moment-preserving approximation is more accurate than the other two polynomial approximations. For 2-D boundaries, the accuracy does not depend significantly on the degree of the polynomial used. High packing ratios (e.g. 1/40) can be achieved with high reconstruction accuracy.
For image approximation, spatial moment-preserving methods can compete with the efficient frequency domain DCT processing at comparable packing ratios. However, DCT methods have the advantage of lower computational complexity.

REFERENCES

[1] S.L. Tanimoto, "Image Transmission with Gross Information First", *Computer Graphics and Image Processing*, vol.9, pp. 72-76, 1979
[2] H. Stone, "Approximation of Curves by Line Segments", *Mathematics of Computation*, vol. 15, pp. 40-47, 1961
[3] C. de Boor, *A Practical Guide to Splines*. New York: Springer Verlag,1978
[4] P.L. Rosin and G.A.W. West, "Techniques for Segmenting Image Curves into Meaningful Descriptions", *Pattern Recognition*, vol. 24, no.7, pp. 643-652, 1991
[5] K.R. Rao and P. Yip, *Discrete Cosine Transform-Algorithms, Advantages, Applications*. Academic Press, Inc., 1990
[6] H. Ahmed et al., *Discrete Cosine Transform, Discrete Transforms and their Applications*. Van Nostrand Reinhold Co., Inc., pp. 9-12, 1985
[7] T.B. Nguyen and B.J. Oommen, "Moment-Preserving Piecewise Linear Approximations of Signals and Images", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no.1, pp. 84-91, 1997

Appendix B.

Matlab Code

Appendix B: The Matlab Code

Figures A, B, C, D, E, F, G, H and I are block diagrams that illustrate how the M files are connected to each other in the Matlab. The file name is every diagram is listed under the name of the functionality it performs. The names of the functions are related to the mathematical derivations shown earlier. So for elaboration on the function of the modules within the derivation, link the name of the functionality to the names of the variables in the mathematical derivation.

- Linear MP Piecewise Approximation for 1-D Signal block diagram is shown in Figure A.

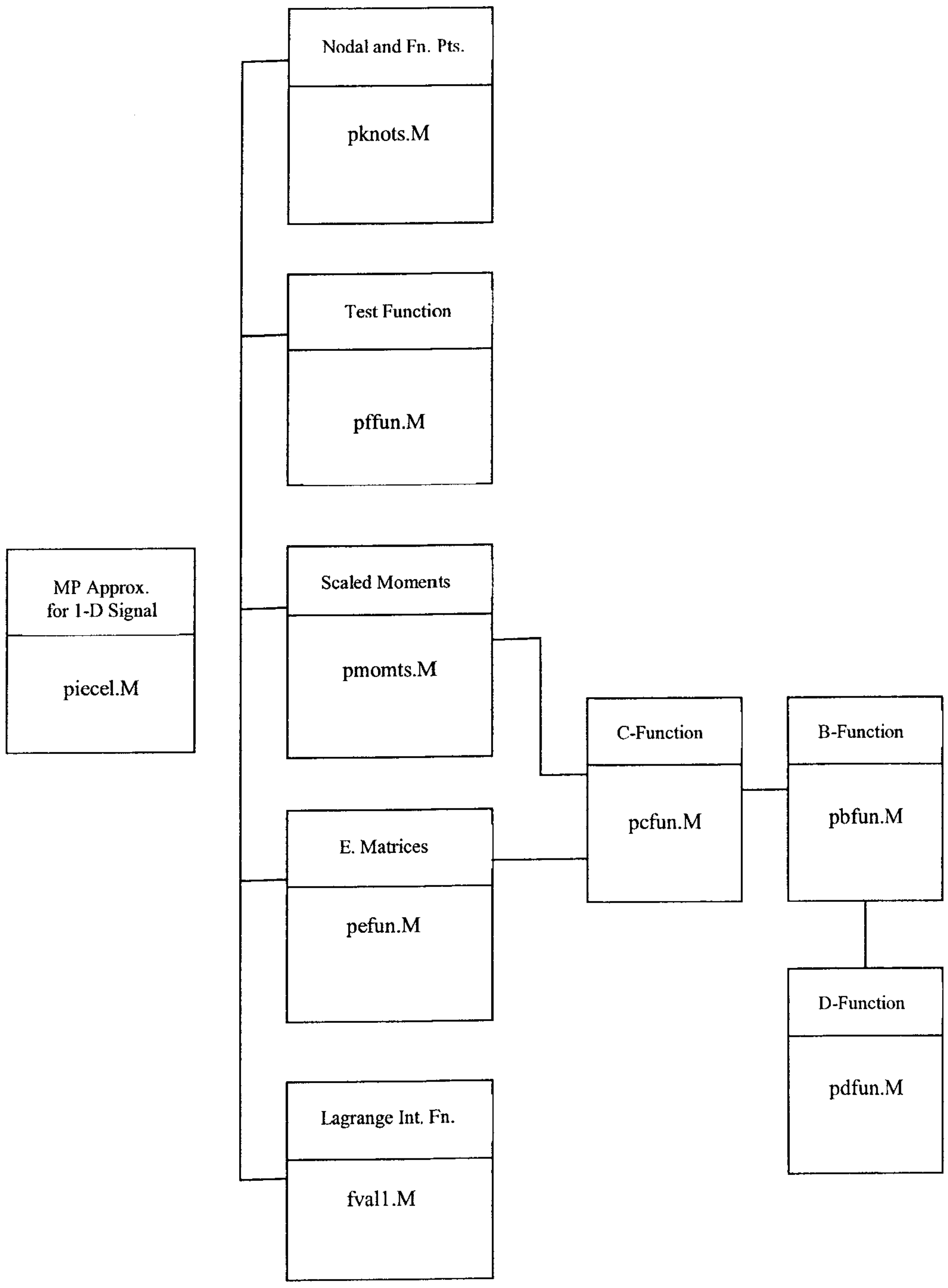


Fig.A.Matlab Code Block Diagram for Linear 1-D Signal

General

- **Test Function**

```
function F = pffun(X,w);

%PFFUN receives a row vector X of n+1 coordinates (x0 .. xn)

% returns a column vector F of n+1 values where F = f(x) + w*r

% r is a vector of random noise (0-1.0). w is the noise weight.


[L,m] = size(X');

r = rand(L,m);

F = 2*sin(0.2*X') + 5*cos(0.3*X') + w*r;
```

- Nodal and Function Points

```
function [n,X,XK] = pknots(d,x0,R,nk);

%PKNOTS produces a row vector X of n+1 coordinates (x0 .. xn)

% in a block,and a row vector XK of nk knot points

% d = x-resolution, R = knot point sampling rate (must be integer)

% s = no. of segments in block = n/R = nk-1


n = R*(nk-1);

for i = 1:n+1

      X(i) = x0 + (i-1)*d;

end

for i = 1:nk

      XK(i) = x0 +(i-1)*R*d;

end
```

Linear Approximation

- **D-Function**

```
function d = pdfun(Z,i,j,k)
%PDFUN receives vector of z-coordinates (Z = z0,z1,..,zn), point
% indices i,j (j>i) and a moment k,
% and returns [z(j)^(k+1) - z(i)^(k+1)]/(k+1)

d = (Z(j+1)^(k+1) - Z(i+1)^(k+1))/(k+1);
```

- **B-Function**

```
function b = pbfun(X,j,a,k)
%PBFUN receives vector of x-coordinates (X), point indices j,a
% and a moment k, and returns  Bj(a,k)

d0 = pdfun(X,j-1,j,0);
d1 = pdfun(X,j-1,j,k);
d2 = pdfun(X,j-1,j,k+1);
b = (X(a+1)*d1-d2)/d0;
```

- C-Function

```
function c = pcfun(X,i,n,k)
%PCFUN receives vector of x-coordinates (X), point index i,
% the total number of segments (n), and moment k,  and returns Ci(k)

if (i == 0) c = pbfun(X,1,1,k);
      elseif (i == n) c = -pbfun(X,n,n-1,k);
      else c = pbfun(X,i+1,i+1,k) - pbfun(X,i,i-1,k);
end
```

- E Matrices

```
function E = pefun(X,n)
%PEFUN receives vector of scaled x-coordinates (X),
% and the total number of segments(n),
%and returns scaled matrix E (n+1)x(n+1) where E(k,i) = Ci(k)

for kk = 1:n+1
      k = kk-1;
      for ii = 1:n+1
            i = ii-1;
            E(kk,ii) = pcfun(X,i,n,k);
      end
end
```

- **Scaled Moments**

```
function [SIG,M] = pmomts(X,F,c,nk)
%PMOMTS receives row vector of point coordinates (X) of size n+1, the
% no. of knot points (nk) and a scale factor (c) and function column vector F
% and computes the corresponding scaled coordinate vector (Y)
% returns a column vector SIG of the first nk scaled moments.
% Ordinary moments are given by M(k) = SIG(k) * c^k



Y = X/c;
[L,m] = size(Y);
n = m-1;



for kk = 1:nk
       k = kk-1;
       SIG(kk) = 0;
       for ii = 1:n+1
              i = ii-1;
              SIG(kk) = SIG(kk) + pcfun(Y,i,n,k)*F(ii) ;
       end
end
SIG = SIG';
for k = 1:nk
       M(k) = SIG(k) * c^k;
end
M = M';
```

▪ Lagrange Interpolation Function

```
function GF = fval1(X,XK,R,G)
%FVAL1 evaluates the function at X using 1st order piecewise
% approximation and Lagrange's Formula. XK is the vector of
% knot points of size nk and G is the vector of values at these points.
% R is the sampling rate.

[k,m] = size(X);
nn = m-1;
[k,nk] = size(XK);
e = 0.001;
for i = 1:nn
       s = fix(i/(R+e))+1;
       x1 = XK(s);
       x2 = XK(s+1);
       a = (X(i)-x2)/(x1-x2);
       b = (X(i)-x1)/(x2-x1);
       GF(i) = a*G(s) + b*G(s+1);
end
GF(nn+1) = G(nk);
```

▪ **Piecewise Moment Preserving Approximation**

```
%Piecel Moment Preserving Approximation of a function
% 1st order piecewise approximation of a function F = f(x)+ w*random noise
% over nb blocks using nk knot points (XK) per block sampled at rate R
% x0 = starting x, d = x-resolution, s = no. of segments per block
% X = x0 ...xn, i.e. n+1 points, c is a scale factor
% SIG is the vector of scaled moments, G1 & G2 are the fuction values at
% the knot points using usual interpolation and moment method, respectively.
% GI & GF are the corresponding interpolated function values using
% Lagrange's formula
% msei, msef are the mean square errors. ei,ef are their means over the blocks
% and are plotted against noise magnitude.

clear;
clc;
clf;

nb = input('Enter no. of blocks: ');
nb = 10;
R  = input('Enter Knot Sampling Rate: ');
R = 40;
nk = input('Enter no. of Knot points: ');
nk = 5; d = 0.1; s = nk - 1;
kw = 10;
rand('seed',0);
for k = 0:kw
x0 = 0; w = 0.1*k; ww(k+1) = w;
for j = 1:nb
```

```
        [n,X,XK] = pknots(d,x0,R,nk);
        c = X(n+1);
        YK = XK/c;
        F = pffun(X,w);
        F0 = pffun(X,0);
        [SIG,M] = pmomts(X,F,c,nk);
        E = pefun(YK,s);
        G2 = pinv(E)*SIG;
        GF = fval1(X,XK,R,G2);
        G1 = interp1(X,F',XK);
        GI = fval1(X,XK,R,G1);
        msef(j) = mean((F0' - GF).^2);
        msei(j) = mean((F0' - GI).^2);
        x0 = X(n+1);
end
ef(k+1) = mean(msef);
ei(k+1) = mean(msei);
end
plot(ww,ef,ww,ei);
disp(ef);
disp(ei);
```

- Quadratic MP Piecewise Approximation for 1-D Signal block diagram is shown in Figure B.

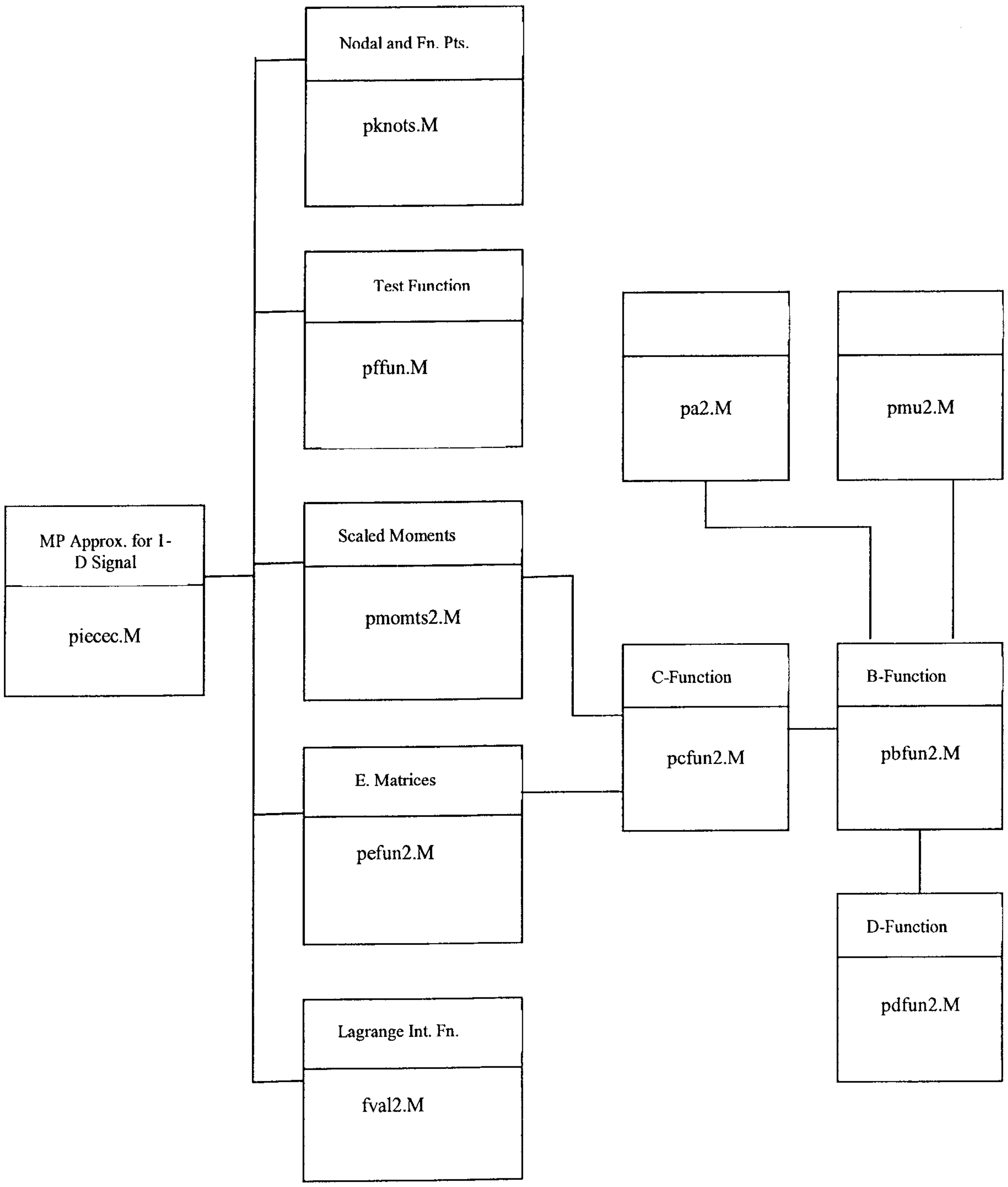


Fig.B. Matlab Code Block Diagram for Quadratic 1-D Signal

Quadratic Approximation

- **D-Function**

```
function d = pdfun2(Z,p,q,k)

%PDFUN2 receives vector of z-coordinates (Z = z0,z1,..,zn), point

% indices p,q (q>p) and a moment k,

% and returns [z(q)^(k+1) - z(p)^(k+1)]/(k+1)


d = (Z(q+1)^(k+1) - Z(p+1)^(k+1))/(k+1);
```

- **B-Function**

```
function b = pbfun2(Z,i,j,p,q,k)

%PBFUN2 receives vector of z-coordinates (Z), point indices i,j,p,q

% and a moment k, and returns scaled Bij(p,q,k)


d0 = pdfun2(Z,i,j,0);

d = pdfun2(Z,i,j,k);

d1 = pdfun2(Z,i,j,k+1);

d2 = pdfun2(Z,i,j,k+2);

a = pa2(Z,p,q);

mu = pmu2(Z,p,q);

b = 2 * (d2 - a*d1 + mu*d) / (d0 * d0);
```

The function pbfun2 uses the following two functions:

```
function a = pa2(Z,p,q)
```

```
%PA2 receives vector Z = z0,z1,..,zn, returns zp + zq


a = Z(p+1) + Z(q+1);


function mu = pmu2(Z,p,q)

%PMU2 receives vector Z = z0,z1,..,zn, returns zp * zq


mu = Z(p+1) * Z(q+1);
```

- C-Function

```
function c = pcfun2(Z,i,n,k)

%PCFUN2 receives vector of z-coordinates Z = z0,z1,..,zn,

% point index i, the value of n and moment k and returns Ci(k,Z)


if (i == 0) c = pbfun2(Z,0,2,1,2,k);

       elseif (i == n) c = pbfun2(Z,n-2,n,n-2,n-1,k);

       elseif (rem(i,2) == 1) c = -2 * pbfun2(Z,i-1,i+1,i-1,i+1,k);

       else c = pbfun2(Z,i-2,i,i-2,i-1,k) + pbfun2(Z,i,i+2,i+1,i+2,k);

end
```

- **E Matrices**

```
function E = pefun2(Z,n)
%PEFUN2 receives vector of scaled z-coordinates (Z = z0,z1,..,zn),
% and the total number of segments(n),
%and returns scaled matrix E (n+1)x(n+1) where E(k,i) = Ci(k,Z)


for kk = 1:n+1
       k = kk-1;
       for ii = 1:n+1
              i = ii-1;
              E(kk,ii) = pcfun2(Z,i,n,k);
       end
end
```

- **Scaled Moments**

```
function [SIG,M] = pmomts2(X,F,c,nk)
%PMOMTS2 receives row vector of point coordinates (X) of size n+1, the
% no. of knot points (nk) and a scale factor (c) and function column vector F
% computes the corresponding scaled coordinate vector (Y) and
% returns a column vector SIG of the first nk scaled moments.
% Ordinary moments are given by M(k) = SIG(k) * c^k



Y = X/c;
[L,m] = size(Y);
n = m-1;
```

```
for kk = 1:nk
        k = kk-1;
        SIG(kk) = 0;
        for ii = 1:n+1
                i = ii-1;
                SIG(kk) = SIG(kk) + pcfun2(Y,i,n,k)*F(ii) ;
        end
end
SIG = SIG';
for k = 1:nk
        M(k) = SIG(k) * c^k;
end
M = M';
```

▪ Lagrange Interpolation Function

```
function GF = fval2(X,XK,R,G)

%FVAL2 evaluates the function at X using 2nd order piecewise

% approximation and Lagrange's Formula. XK is the vector of

% knot points of size nk and G is the vector of values at these points.

% R is the sampling rate.


[k,m] = size(X);

nn = m-1;

[k,nk] = size(XK);

e = 0.001;

for i = 1:nn

       j = fix(i/(2*R+e))+1;

       s = 2*j-1;

       x1 = XK(s);

       x2 = XK(s+1);

       x3 = XK(s+2);

       a = (X(i)-x2)*(X(i)-x3)/((x1-x2)*(x1-x3));

       b = (X(i)-x1)*(X(i)-x3)/((x2-x1)*(x2-x3));

       c = (X(i)-x1)*(X(i)-x2)/((x3-x1)*(x3-x2));

       GF(i) = a*G(s) + b*G(s+1) + c*G(s+2);

end

GF(nn+1) = G(nk);
```

- ## Piecewise Moment Preserving Approximation

```
%Pieceq Moment Preserving Approximation of a function
% 2nd order piecewise approximation of a function F = f(x)+ w*random noise
% over nb blocks using nk knot points (XK) per block sampled at rate R
% x0 = starting x, d = x-resolution, s = no. of segments per block
% X = x0 ...xn, i.e. n+1 points, c is a scale factor
% SIG is the vector of scaled moments, G1 & G2 are the fuction values at
% the knot points using usual interpolation and moment method, respectively.
% GI & GS are the corresponding interpolated function values using
% Lagrange's formula
% msei, mses are the mean square errors. ei,es are their means over the blocks
% and are plotted against noise magnitude.

clear;
clc;
clf;

nb = input('Enter no. of blocks: ');
nb = 10;
R  = input('Enter Knot Sampling Rate: ');
R = 40;
nk = input('Enter no. of Knot points: ');
nk = 5;
d = 0.1;
s = nk - 1;
kw = 10;
rand('seed',0);
for k = 0:kw
x0 = 0; w = 0.1*k; ww(k+1) = w;
```

```
for j = 1:nb

        [n,X,XK] = pknots(d,x0,R,nk);

        c = X(n+1);

        YK = XK/c;

        F = pffun(X,w);

        F0 = pffun(X,0);

        [SIG,M] = pmomts2(X,F,c,nk);

        E = pefun2(YK,s);

        G2 = pinv(E)*SIG;

        GS = fval2(X,XK,R,G2);

        G1 = interp1(X,F',XK);

        GI = fval2(X,XK,R,G1);

        mses(j) = mean((F0' - GS).^2);

        msei(j) = mean((F0' - GI).^2);

        x0 = X(n+1);

end

es(k+1) = mean(mses);

ei(k+1) = mean(msei);

end

plot(ww,es,ww,ei);

disp(es);

disp(ei);
```

- Cubic Spline MP Piecewise Approximation for 1-D Signal block diagram is shown in Figure C.

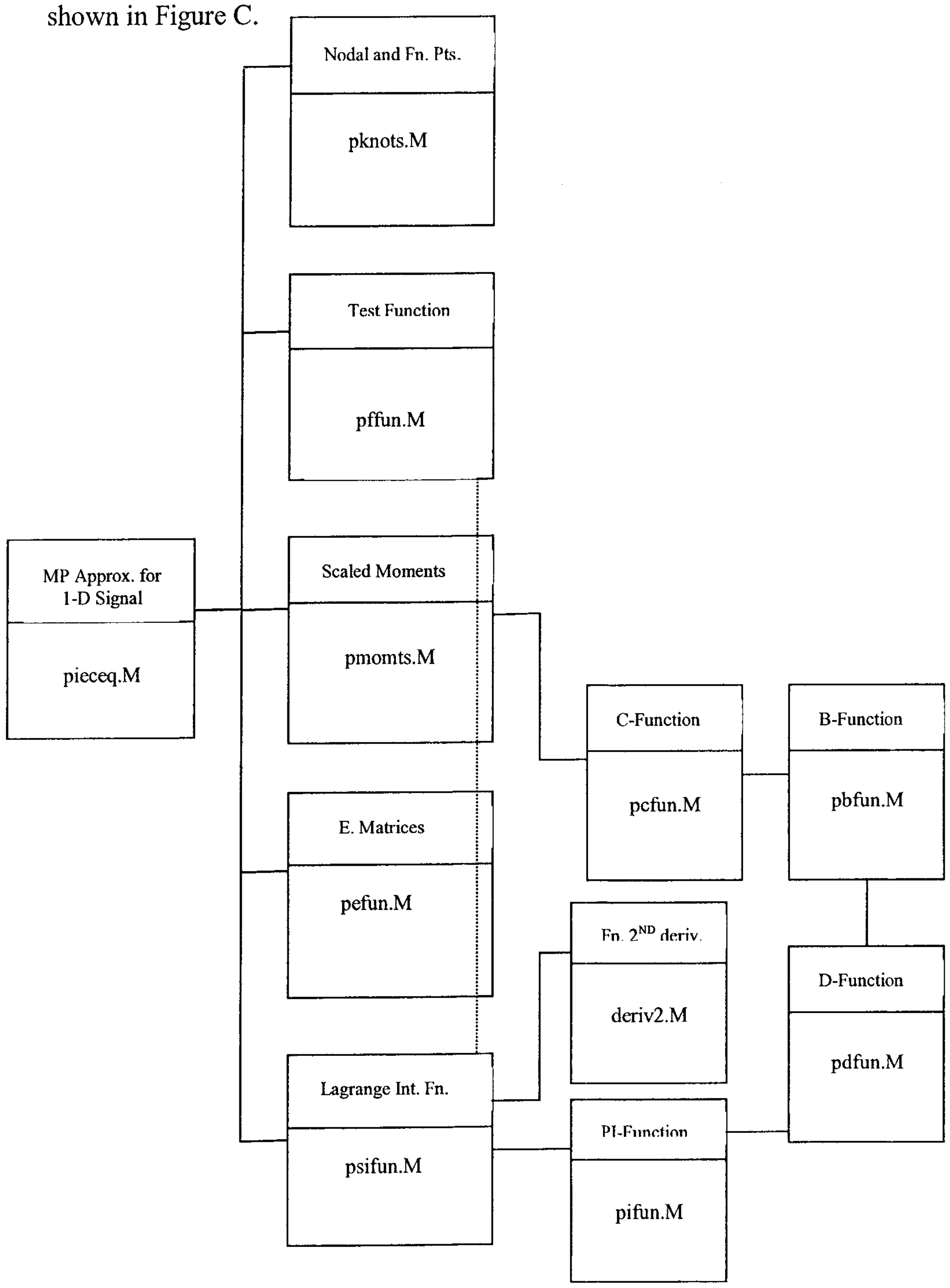


Fig.C. Matlab Code Block Diagram for Cubic Spline 1-D Signal

▪ Second Derivative of a Function

```
function y2 = deriv2(x,y)
%DERIV2
% Given vector x[1..n],and function values f(x)= y[1..n]
% returns 2nd derivative f''(x) = y2[1..n] for cubic spline
% f'(x[1]) = f'(x[n]) = 0 , i.e., natural spline

[L,n] = size(x);
u = zeros(1,n-1);
y2 = zeros(1,n);
%yp1 = (y(2)-y(1))/(x(2)-x(1));
yp1 = 0;
if (yp1 > 0.0)
       y2(1) = -0.5;
       u(1) = (3.0/(x(2)-x(1)))*((y(2)-y(1))/(x(2)-x(1))-yp1);
end
for i = 2:n-1
       sig = (x(i)-x(i-1))/(x(i+1)-x(i-1));
       p = sig*y2(i-1)+2.0;
       y2(i) = (sig-1.0)/p;
       u(i) = (y(i+1)-y(i))/(x(i+1)-x(i)) - (y(i)-y(i-1))/(x(i)-x(i-1));
       u(i) = (6.0 * u(i)/(x(i+1)-x(i-1)) - sig*u(i-1))/p;
end
qn = 0;
un = 0;
%ypn = (y(n)-y(n-1))/(x(n)-x(n-1));
```

```
ypn = 0;

if (ypn > 0.0)

       qn = 0.5;

       un = (3.0/(x(n)-x(n-1)))*(ypn-(y(n)-y(n-1))/(x(n)-x(n-1)));

end

y2(n) = (un - qn*u(n-1))/(qn*y2(n-1)+1.0);

for k = n-1:-1:1

       y2(k) = y2(k) * y2(k+1) + u(k);

end
```

- **Pi - Function**

```
function z = pifun(y,c,p,i,j,k)

%PIFUN

d0 = pdfun(y,i,j,0);

for l = 0:3

       d(l+1) = pdfun(y,i,j,k+1);

end

yi = y(i+1);

yj = y(j+1);

gi = c*c*p(i+1)/d0;

gj = c*c*p(j+1)/d0;

a0i = yi*(yi*yi-d0*d0);

a1i = d0*d0-3*yi*yi;

a2i = 3*yi;

a0j = yj*(yj*yj-d0*d0);

a1j = d0*d0-3*yj*yj;

a2j = 3*yj;

z = d(1)*(gi*a0j-gj*a0i)+d(2)*(gi*a1j-gj*a1i)+d(3)*(gi*a2j-gj*a2i)-d(4)*(gi-
gj);
```

- **Psi - Function**

```
function s = psifun(x,c,nk)

%PSIFUN

%


[m,n] = size(x);

F =pffun(x,0);

p = deriv2(x,F)/6.0;
```

```
y = x/c;

for kk = 1:nk

        k = kk-1;

        sum = 0.0;

        for j = 1:n-1

                i = j-1;

                sum = sum + pifun(y,c,p,i,j,k);

        end

        s(kk) = sum;

end

s = s';
```

- **Piecewise Moment Preserving Approximation**

```
%Piecec Moment Preserving Approximation of a function

% Cubic Spline piecewise approximation of a function F = f(x)+ w*random noise

% over nb blocks using nk knot points (XK) per block sampled at rate R

% x0 = starting x, d = x-resolution, s = no. of segments per block

% X = x0 ...xn, i.e. n+1 points, c is a scale factor

% SIG is the vector of scaled moments, G1 & G2 are the fuction values at

% the knot points using usual interpolation and moment method, respectively.

% GI & GC are the corresponding interpolated function values using

% Lagrange's formula

% msei, msec are the mean square errors. ei,ec are their means over the blocks

% and are plotted against noise magnitude.


clear;

clc;

clf;


nb = input('Enter no. of blocks: ');

nb = 10;

R  = input('Enter Knot Sampling Rate: ');

R = 40;

nk = input('Enter no. of Knot points: ');

nk = 5;

d = 0.1;

s = nk - 1;

kw = 10;

rand('seed',0);

for k = 0:kw

x0 = 0; w = 0.1*k; ww(k+1) = w;

for j = 1:nb
```

```
        [n,X,XK] = pknots(d,x0,R,nk);

        c = X(n+1);

        YK = XK/c;

        F = pffun(X,w);      F0 = pffun(X,0);

        [SIG,M] = pmomts(X,F,c,nk);

        E = pefun(YK,s);

        pk = psifun(XK,c,nk);      ps = psifun(X,c,nk);

        G2 = pinv(E)*(SIG+ps-pk); GC = spline(XK,G2,X);

        G1 = interp1(X,F',XK);           GI = spline(XK,G1,X);

        msec(j) = mean((F0' - GC).^2);

        msei(j) = mean((F0' - GI).^2);

        x0 = X(n+1);

end

ec(k+1) = mean(msec);

ei(k+1) = mean(msei);

end

plot(ww,ec,ww,ei);

disp(ec);

disp(ei);
```

Comparison Between the Three Approximations

▪ Usual Interpolation

```
%Piecei Approximation of a function through usual interpolation
% 1st & 2nd order and cubic spline piecewise approximation of
% a function F = f(x)+ w*random noise over nb blocks using nk knot
% points (XK) per block sampled at rate R. x0 = starting x, d = x-resolution,
% X = x0 ...xn, i.e. n+1 points, G1 is the fuction values at the knot points.
% GF & GS and GC are the interpolated function values. msef, mses,msec are the
% mean square errors. ef,es, ec are their means over the blocks
% and are plotted against noise magnitude.
clear;
clc;
clf;

nb = input('Enter no. of blocks: ');
nb = 10;
R  = input('Enter Knot Sampling Rate: ');
R = 40;
nk = input('Enter no. of Knot points: ');
nk = 5;
d = 0.1;
s = nk - 1;
kw = 10;
rand('seed',0);
for k = 0:kw
x0 = 0; w = 0.1*k; ww(k+1) = w;
for j = 1:nb
       [n,X,XK] = pknots(d,x0,R,nk);
```

```
        F = pffun(X,w);

        F0 = pffun(X,0);

        G1 = interp1(X,F',XK);

        GF = fval1(X,XK,R,G1);

        GS = fval2(X,XK,R,G1);

        GC = spline(XK,G1,X);

        msef(j) = mean((F0' - GF).^2);

        mses(j) = mean((F0' - GS).^2);

        msec(j) = mean((F0' - GC).^2);


%       plot(X,F,XK,G1,'o',X,GF,X,GS,X,GC);

        x0 = X(n+1);

%       pause;

end


ef(k+1) = mean(msef);

es(k+1) = mean(mses);

ec(k+1) = mean(msec);

end

plot(ww,ef,ww,es,ww,ec);

disp(ef);

disp(es);

disp(ec);
```

▪ Moment-Preserving Approximation

```
%Piece Moment Preserving Approximation of a function
% 1st, 2nd order and Cubic Spline piecewise approximation of a function
% F = f(x)+ w*random noise over nb blocks using nk knot points (XK) per block
% sampled at rate R. x0 = starting x, d = x-resolution, s = no. of segments per
block
 % X = x0 ...xn, i.e. n+1 points, c is a scale factor
% SIG is the vector of scaled moments, G1,G2,G3 are the fuction values at
% the knot points using moment method. GF, GS and GC are the interpolated
% function values using Lagrange's formula and spline interpolation
% msef, mses and msec are the mean square errors. ef,es and ec are their means
% over the blocks and are plotted against noise magnitude.


clear;
clc;
clf;


nb = input('Enter no. of blocks: ');
nb = 10;
R  = input('Enter Knot Sampling Rate: ');
R = 40;
nk = input('Enter no. of Knot points: ');
nk = 5;
d = 0.1; s = nk - 1; kw = 10; rand('seed',0);
for k = 10:kw
x0 = 0; w = 0.1*k;  ww(k+1) = w;
for j = 1:nb
       [n,X,XK] = pknots(d,x0,R,nk);
       c = X(n+1);  YK = XK/c;
       F = pffun(X,w);      F0 = pffun(X,0);
```

```
        [SIG,M] = pmomts(X,F,c,nk);
        E = pefun(YK,s);
        G1 = pinv(E)*SIG;    GF = fval1(X,XK,R,G1);
        pk = psifun(XK,c,nk);       ps = psifun(X,c,nk);
        G3 = pinv(E)*(SIG+ps-pk);  GC = spline(XK,G3,X);
        [SIG,M] = pmomts2(X,F,c,nk);
        E = pefun2(YK,s);
        G2 = pinv(E)*SIG;    GS = fval2(X,XK,R,G2);
        msef(j) = mean((F0' - GF).^2);
        mses(j) = mean((F0' - GS).^2);
        msec(j) = mean((F0' - GC).^2);
        plot(X,F,'+',X,GF,X,GS,'-',X,GC,'.');
        x0 = X(n+1);
        pause;
end
ef(k+1) = mean(msef);  es(k+1) = mean(mses);  ec(k+1) = mean(msec);
end
plot(ww,ef,ww,es,ww,ec);
disp(ef); disp(es); disp(ec);
```

- Linear MP Piecewise Approximation for 2-D Closed Boundary block diagram is shown in Figure D.

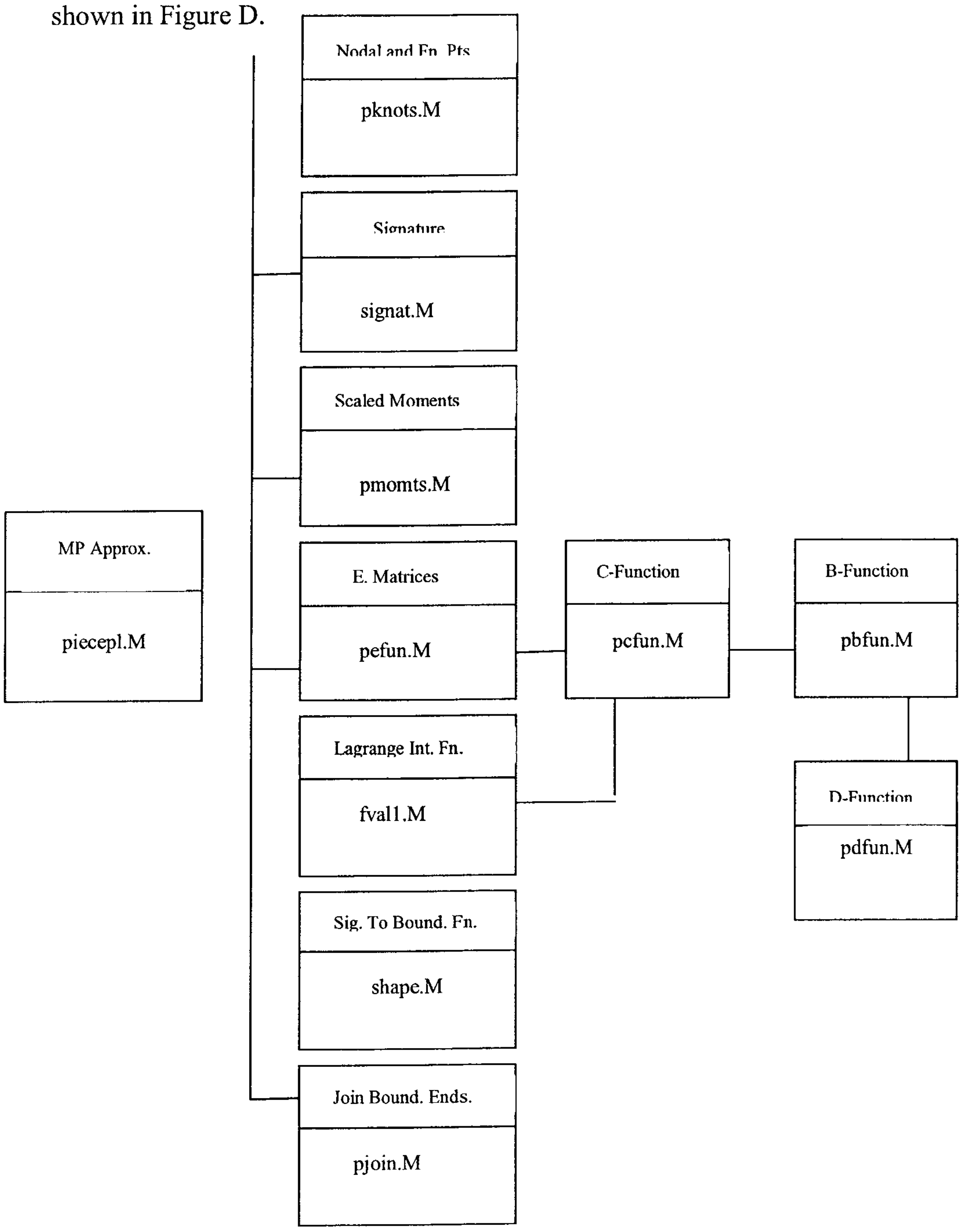


Fig.D. Matlab Code Block Diagram for Linear 2-D Closed Boundary

▪ Signature of a 2-D Function

```
function F = signat(X,xf,a,b,c,w);
%SIGNAT receives a row vector X of n+1 coordinates (x0 .. xn)
% normalizes X {0,xf} to T {2*pi,0} (clockwise)
% returns a column vector F of n+1 values where F = D(T)+ w*r
% representing the signature of a parametric function.
% r is a vector of random noise (0-1.0). w is the noise weight.
% a,b,c are the parameters of the boundary

[L,m] = size(X);
r = rand(L,m); T = 2*pi*(1 - X/xf);
for i = 1:m
 F(i) = a*b*abs(sin(T(i)))/sqrt(b^2 * abs(cos(T(i)))^c + a^2 *
abs(sin(T(i)))^c);
end
F = (F + w*r)';
```

▪ Boundary from Signature

```
function [X,Y] = shape(Z,zf,F);
%SHAPE receives a vector Z of n+1 coordinates (z0 .. zn)and a vector F
% of n+1 values where F = D(T). Normalizes Z {0,zf} to T {2*pi,0} (clockwise) ,
returns the planar transformation (X,Y)

[L,m] = size(Z); T = 2*pi*(1 - Z/zf);
for i = 1:m
      X(i) = F(i) * cos(T(i));
      Y(i) = F(i) * sin(T(i));
end
```

- **Join Ends of a Boundary**

```
function G = pjoin(G)
%PJOIN
% Averages and equates last value in a row with
% first value in the next row

[nb,nk] = size(G);
for j = 2:nb
      v = 0.5*(G(j-1,nk) + G(j,1));
      G(j-1,nk) = v;
      G(j,1) = v;
end
v = 0.5*(G(1,1) + G(nb,nk));
G(1,1) = v; G(nb,nk) = v;
```

```
%PIECPL Moment Preserving Approximation of a closed boundary
% 1st order piecewise approximation of a noisy boundary
% over nb blocks using nk knot points (XK) per block sampled at rate R
% x0 = starting x, d = x-resolution, s = no. of segments per block
% X = x0 ...xn, i.e. n+1 points, c is a scale factor
% SIG is the vector of scaled moments, G1 & G2 are the fuction values at
% the knot points using usual interpolation and moment method, respectively.
% GI & GF are the corresponding interpolated function values using
% Lagrange's formula. msei, msef are the mean square errors. ei,ef are their
% means over the blocks and are plotted against noise magnitude.

clear; clc; clf;

nb = input('Enter no. of blocks: ');
nb = 8;
R  = input('Enter Knot Sampling Rate: ');
R = 40;
nk = input('Enter no. of Knot points: ');
nk = 5;  nn = R*(nk-1)+1; ux = zeros(nb,nn);  uy = ux;
d = 1; s = nk - 1; kw = 7; rand('seed',1943);  xf = nb*R*(nk-1);
p1 = 3;  p2 = 1; p3 = 8;
for k = kw:kw
x0 = 0; w = 0.1*k; ww(k+1) = w;
for j = 1:nb
      [n,X,XK] = pknots(d,x0,R,nk);
      c = X(n+1);  YK = XK/c;
```

```
        F = signat(X,xf,p1,p2,p3,w); F0 = signat(X,xf,p1,p2,p3,0);
        [SIG,M] = pmomts(X,F,c,nk); E = pefun(YK,s);
        G2 = pinv(E)*SIG; GF = fval1(X,XK,R,G2);
        G1 = interp1(X,F',XK);      GI = fval1(X,XK,R,G1);
        msef(j) = mean((F' - GF).^2); msei(j) = mean((F' - GI).^2);
        [x1,y1] = shape(X,xf,F); [x2,y2] = shape(X,xf,GF);
        [x3,y3] = shape(X,xf,GI);
        ux(j,:) = x2;  uy(j,:) = y2;
        plot(x1,y1,'w.'); title('Piecewise Linear Approximation');
        hold on;
        x0 = X(n+1);
%       pause;
end
ef(k+1) = mean(msef); ei(k+1) = mean(msei);
ux = pjoin(ux);  uy = pjoin(uy);
for j = 1:nb
        plot(ux(j,:),uy(j,:),'w');
end
pause;
clf;
end
plot(ww,ef,'w',ww,ei,'w:'); xlabel('Noise Level'); ylabel('MSE');
title('Piecewise Linear Approximation of a Boundary');
legend('w-','MP','w:','INT');
disp(ef); disp(ei);
```

- Quadratic MP Piecewise Approximation for 2-D Closed Boundary block diagram is shown in Figure E.

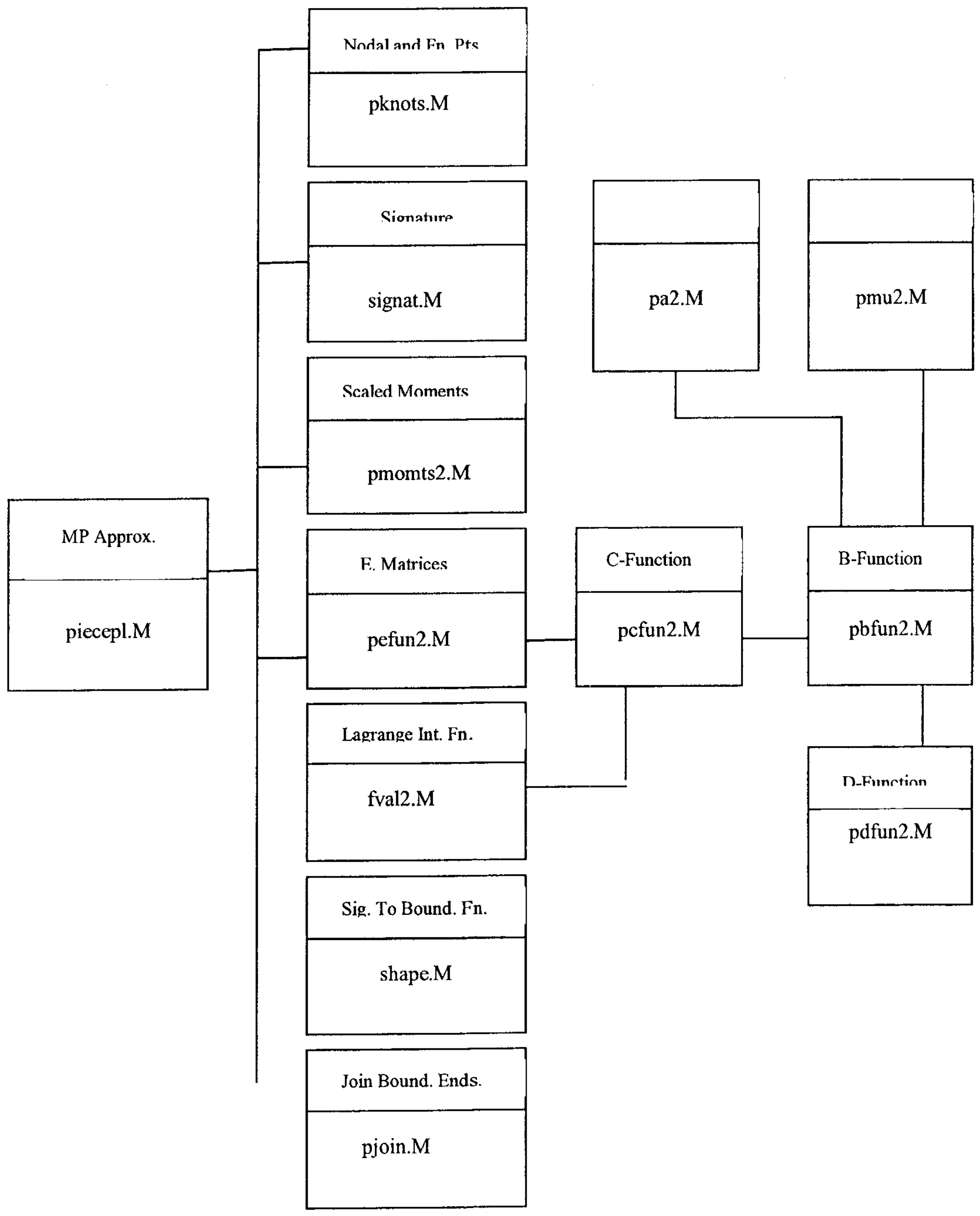


Fig.E. Matlab Code Block Diagram for Quadratic2-D Closed Boundaries

Quadratic Approximation of Noisy Boundary

```
%PIECEPQ Moment Preserving Approximation of a closed boundary
% 2nd order piecewise approximation of a noisy boundary
% over nb blocks using nk knot points (XK) per block sampled at rate R
% x0 = starting x, d = x-resolution, s = no. of segments per block
% X = x0 ...xn, i.e. n+1 points, c is a scale factor
% SIG is the vector of scaled moments, G1 & G2 are the fuction values at
% the knot points using usual interpolation and moment method, respectively.
% GI & GF are the corresponding interpolated function values using
% Lagrange's formula. msei, msef are the mean square errors. ei,ef are their
% means over the blocks and are plotted against noise magnitude.

clear; clc; clf;

nb = input('Enter no. of blocks: ');
nb = 8;
R  = input('Enter Knot Sampling Rate: ');
R = 40;
nk = input('Enter no. of Knot points: ');
nk = 5;
nn = R*(nk-1)+1; ux = zeros(nb,nn);  uy = ux;
d = 1; s = nk - 1; kw = 5;
rand('seed',1943); xf = nb*R*(nk-1);
p1 = 3;  p2 = 1; p3 = 8;
for k = kw:kw
x0 = 0;
w = 0.1*k;
ww(k+1) = w;
```

```
 for j = 1:nb
        [n,X,XK] = pknots(d,x0,R,nk);
        c = X(n+1);  YK = XK/c;
        F = signat(X,xf,p1,p2,p3,w); F0 = signat(X,xf,p1,p2,p3,0);
        [SIG,M] = pmomts2(X,F,c,nk); E = pefun2(YK,s);
        G2 = pinv(E)*SIG; GF = fval2(X,XK,R,G2);
        G1 = interp1(X,F',XK); GI = fval2(X,XK,R,G1);
        msef(j) = mean((F' - GF).^2); msei(j) = mean((F' - GI).^2);
        [x1,y1] = shape(X,xf,F);[x2,y2] = shape(X,xf,GF);
        [x3,y3] = shape(X,xf,GI);
        ux(j,:) = x2;  uy(j,:) = y2;
        plot(x1,y1,'w.'); title('Piecewise Quadratic Approximation');
        legend('w-','MP Quad');
        hold on;
        x0 = X(n+1);
%       pause;
end
ef(k+1) = mean(msef); ei(k+1) = mean(msei);
ux = pjoin(ux);  uy = pjoin(uy);
for j = 1:nb
        plot(ux(j,:),uy(j,:),'w');
end
pause;
clf;
end
plot(ww,ef,'w',ww,ei,'w:'); xlabel('Noise Level'); ylabel('MSE');
title('Piecewise Quadratic Approximation of a Boundary');
legend('w-','MP','w:','INT');
disp(ef);
disp(ei);
```

- Cubic Spline MP Piecewise Approximation for 2-D Closed Boundary block diagram is shown in Figure F.

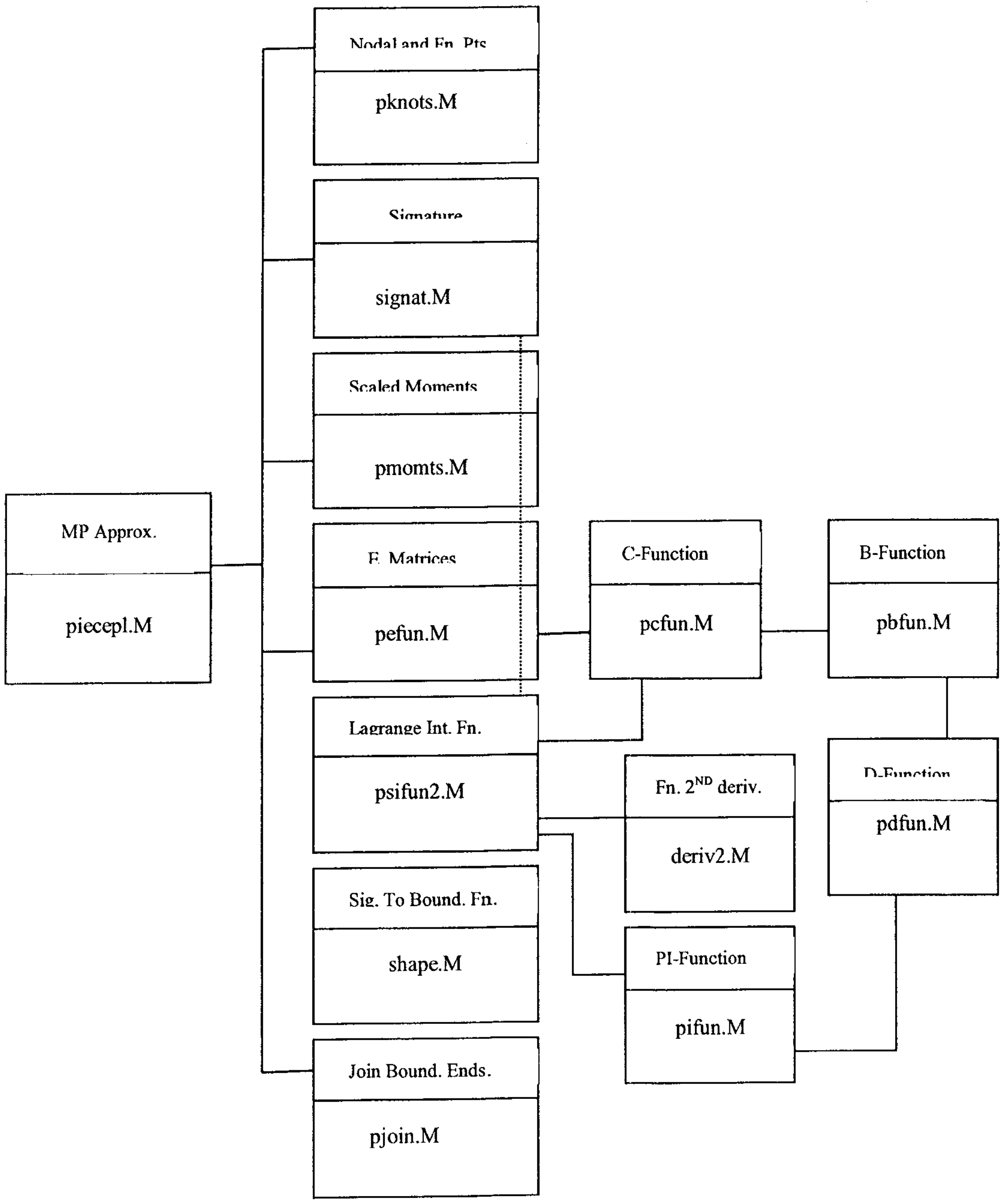


Fig.F. Matlab Code Block Diagram for Cubic Spline 2-D Closed Boundary

```
%PIECEPC Moment Preserving Approximation of a closed boundary
% Cubic Spline piecewise approximation of a noisy boundary
% over nb blocks using nk knot points (XK) per block sampled at rate R
% x0 = starting x, d = x-resolution, s = no. of segments per block
% X = x0 ...xn, i.e. n+1 points, c is a scale factor
% SIG is the vector of scaled moments, G1 & G2 are the fuction values at
% the knot points using usual interpolation and moment method, respectively.
% GI & GF are the corresponding interpolated function values using
% Lagrange's formula. msei, msef are the mean square errors. ei,ef are
% their means over the blocks and are plotted against noise magnitude.

clear;
clc;
clf;

nb = input('Enter no. of blocks: ');
nb = 8;
R  = input('Enter Knot Sampling Rate: ');
R = 40;
nk = input('Enter no. of Knot points: ');
nk = 5;
nn = R*(nk-1)+1; ux = zeros(nb,nn);  uy = ux;
d = 1; s = nk - 1; kw = 10; rand('seed',1943); xf = nb*R*(nk-1);
p1 = 3;  p2 = 1; p3 = 2;
for k = 0:kw
x0 = 0; w = 0.2*k; ww(k+1) = w;
for j = 1:nb
```

```
        [n,X,XK] = pknots(d,x0,R,nk);
        c = X(n+1);   YK = XK/c;    F = signat(X,xf,p1,p2,p3,w);
        F0 = signat(X,xf,p1,p2,p3,0); FK = signat(XK,xf,p1,p2,p3,0);
        [SIG,M] = pmomts(X,F,c,nk); E = pefun(YK,s);
        pk = psifun2(FK,XK,c,nk);  ps = psifun2(F0,X,c,nk);
        G2 = pinv(E)*(SIG+ps-pk);  GF = spline(XK,G2,X);
        G1 = interp1(X,F',XK);      GI = spline(XK,G1,X);
        msef(j) = mean((F' - GF).^2); msei(j) = mean((F' - GI).^2);
        [x1,y1] = shape(X,xf,F); [x2,y2] = shape(X,xf,GF);
        [x3,y3] = shape(X,xf,GI);  ux(j,:) = x2;  uy(j,:) = y2;
        plot(x1,y1,'w.'); title('Piecewise Cubic Spline Approximation');
        hold on;
        x0 = X(n+1);
%       pause;
end
ef(k+1) = mean(msef); ei(k+1) = mean(msei); ux = pjoin(ux);  uy = pjoin(uy);
for j = 1:nb
        plot(ux(j,:),uy(j,:),'w');
end
pause;
clf;
end
plot(ww,ef,'w',ww,ei,'w:'); xlabel('Noise Level'); ylabel('MSE');
title('Piecewise Cubic Spline Approximation of a Boundary');
legend('w-','MP','w:','INT');
disp(ef); disp(ei);
```

- Linear & Quadratic MP Piecewise Approximation for Digitized Images block diagram is shown in Figure G.

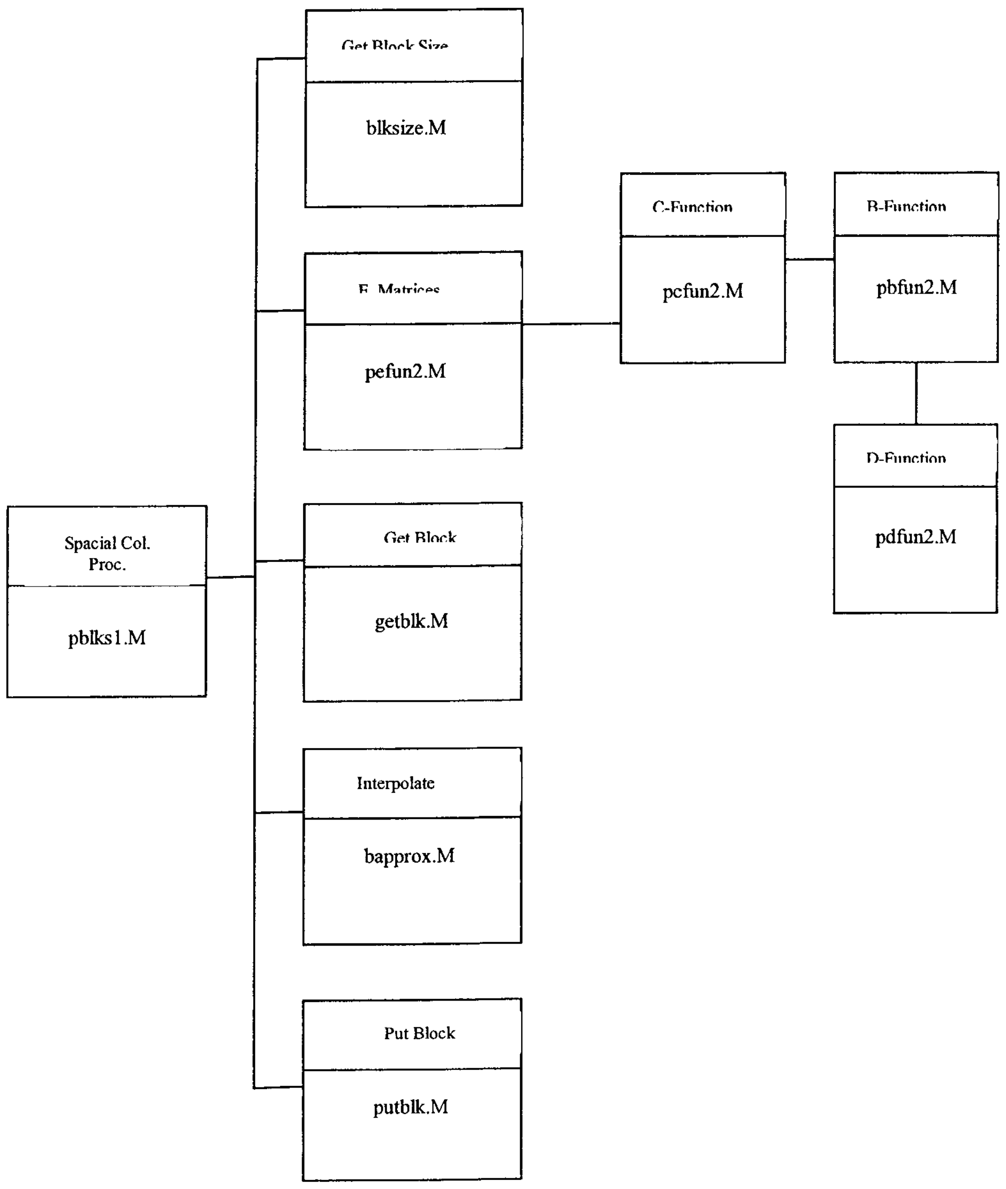


Fig. G. Matlab Code Block Diagram for Digitized Images

Linear and Quadratic Digitized Image Approximation

- Block Size

```
function [np,nb,nt] = blksize(m,nk,R,v);

%BLKSIZE

% m = no. of pixels in an image row,

% nk = no.of nodal points sampled at R pixels

% v = overlap pixels between successive blocks

% returns np = no. of points in block, nb = no. of blocks

% nt = total no. of pixels needed in row


np = (nk-1)*R+1;

nb = ceil((m-v)/(np-v));

nt = nb*(np-v)+v;
```

- Get Block

```
function B = getblk(X,np,v,j)

%GETBLK

% Get a block (j) from an image row X.

% np = no.of points in block. v is no. of overlap

% pixels between current and previous blocks


s = (j-1)*(np-v)+1;

e = s+np-1;

B = zeros(1,np);

B = X(s:e);
```

- Put Block

```
function X = putblk(X,B,np,v,j)

%PUTBLK

% Given an image row in X, overwrite block (j) with

% contents of B. np = no.of points in block.

% v is no. of overlap pixels between current and previous blocks


s = (j-1)*(np-v)+1;

e = s+np-1;

X(s:e) = B;
```

- Interpolation Block Processing

```
function y = bapprox(x,f,xk,nk,c,R,u,p,q)
%BAPPROX

% Interpolation of pixels in a block (f) of np points

% using nodal points sampled at R pixels.

% x = 1:np, f = block, xk = 1:R:nk, nk = no. of nodal points

% c = x(np), u = pinv(E)

% p = 0 usual interpolation, p = 1 Moment-Preserving

% q = 1 Linear, q = 2 Quadratic


if (p > 0)

       if (q == 1)

              [SIG,M] = pmomts(x,f',c,nk);

       else

              [SIG,M] = pmomts2(x,f',c,nk);

       end

       G1 = u*SIG;

end

if (p == 0) G1 = f(xk); end
if (q == 1)
      y = fval1(x,xk,R,G1);
else
      y = fval2(x,xk,R,G1);
end
```

- Spatial Column Processing

```
%pblks1
% Spatial block processing, of an image, row/column by row/column
% A row is divided into nb blocks, with np points/block
% and v overlap points. Total row length is zero padded to be
% exactly nb(np-v)+v points. nk = number of nodal points,
% sampled at R points
% Linear or quadratic interpolation or moment preserving.
% MSE =Mean Square Error. PSNR = Peak Signal-to-Noise Ratio
clear;
clc;
clf;

fn = input('Enter Image File Name: ','s');
[a,map] = bmpread(fn);
X = ind2gray(a,map);
subplot(2,1,1); imshow(X,256);
pause; X = X';
[N,M] = size(X); Y = zeros(N,M);
R = 4;
v = 0;
nk = 5;
[np,nb,nt] = blksize(M,nk,R,v);
row = zeros(1,nt);
```

```
x = 1:np;

xk = 1:R:np; s = nk-1;

c = x(np); YK = xk/c;

El = pefun(YK,s);    lv = pinv(El);

Eq = pefun2(YK,s);  qv = pinv(Eq);

for i = 1:N

       row(1:M) = X(i,1:M);

       for j = 1:nb

              B = getblk(row,np,v,j);

              C = bapprox(x,B,xk,nk,c,R,qv,1,2);

              row = putblk(row,C,np,v,j);

       end

       Y(i,1:M) = row(1:M);

end

X = X'; Y = Y';



subplot(2,1,2); imshow(Y,256);

MSE = sum(sum((X - Y).^2))/(N*M);

PSNR = 10 * log10(255*255/(MSE+eps));

disp(MSE); disp(PSNR);
```

- DCT Block Processing
    - 2-D to 1-D Zigzag Mapping

```
function F = zigzag(B)
%ZIGZAG
% 2-D to 1-D zigzag mapping of an 8x8 block

R = [ 1  2  6  7 15 16 28 29;
      3  5  8 14 17 27 30 43;
      4  9 13 18 26 31 42 44;
     10 12 19 25 32 41 45 54;
     11 20 24 33 40 46 53 55;
     21 23 34 39 47 52 56 61;
     22 35 38 48 51 57 60 62;
     36 37 49 50 58 59 63 64 ];
for i = 1:8
        for j = 1:8
                k = R(i,j);
                F(k) = B(i,j);
        end
end
```

    - 1-D to 2-D Inverse Zigzag Mapping

```
function B = izigzag(F)
%IZIGZAG
% 1-D to 2-D zigzag mapping of an 8x8 block

R = [ 1  2  6  7 15 16 28 29;
      3  5  8 14 17 27 30 43;
      4  9 13 18 26 31 42 44;
     10 12 19 25 32 41 45 54;
     11 20 24 33 40 46 53 55;
     21 23 34 39 47 52 56 61;
     22 35 38 48 51 57 60 62;
     36 37 49 50 58 59 63 64 ];
for i = 1:8
        for j = 1:8
                k = R(i,j);
                B(i,j) = F(k);
        end
end
```

o Block Interpolation

```
function f = avblki(f,k,R)
%AVBLKi
% f is a 1-D vector of length 64 in a block
% elements (1 .. k) are unchanged
% R = nodal point sampling interval, np = no. of points
% (from k+1 to k+np) that will be interpolated
% points from k+1+np to 64 are set to zero
% returns a linear or quadratic interpolation approximation

np = 2*R+1;
y = zeros(1,np);
x = 1:np;
xk = 1:R:np;
y = f(k+1:k+np);
G1 = y(xk);
%y = fval1(x,xk,R,G1);
y = fval2(x,xk,R,G1);
f(k+1:k+np) = y;
f(k+np+1:64) = zeros(1,64-k-np);
```

o Block Moment-Preserving Approximation

```
function f = avblkmp(f,k,R)
%AVBLKMP
% f is a 1-D vector of length 64 in a block
% elements (1 .. k) are unchanged
% R = nodal point sampling interval, np = no. of points
% (from k+1 to k+np) that will be interpolated
% points from k+1+np to 64 are set to zero
% returns a linear or quadratic Moment-Preserving approximation


np = 2*R+1;
y = zeros(1,np);
x = 1:np;
xk = 1:R:np;
nk = length(xk); s = nk-1;
y = f(k+1:k+np);
c = x(np);
YK = xk/c;
%[SIG,M] = pmomts(x,y',c,nk);
%E = pefun(YK,s);
[SIG,M] = pmomts2(x,y',c,nk);
E = pefun2(YK,s);
G1 = pinv(E)*SIG;
%y = fval1(x,xk,R,G1);
y = fval2(x,xk,R,G1);
f(k+1:k+np) = y;
f(k+np+1:64) = zeros(1,64-k-np);
```

o DCT Coefficients Interpolation

```
function y = dctmi(x,k,R)
%DCTMI receives an 8x8 block, computes its DCT, retains
% first k zigzag coefficients,then returns the inverse DCT
% For R = 0, Coef. K+1 to 64 are set to zero. For R > 0 ,
% they are approximated by Linear or quadratic interpolation


d = dct2(x);
f = zigzag(d);
if (R > 0) f = avblki(f,k,R);
      else f(k+1:64) = zeros(1,64-k);
end
d = izigzag(f);
y = idct2(d);
```

o DCT Coefficients Moment-Preserving

```
function y = dctmmp(x,k,R)
%DCTMMP receives an 8x8 block, computes its DCT, retains
% first k zigzag coefficients,then returns the inverse DCT
% For R = 0, Coef. K+1 to 64 are set to zero. For R > 0 ,
% they are approximated by linear or quadratic Moment-preserving

d = dct2(x);
f = zigzag(d);
if (R > 0) f = avblkmp(f,k,R);
      else f(k+1:64) = zeros(1,64-k);
end
d = izigzag(f);
y = idct2(d);
```

o DCT Image Block Processing

```
%pblks2
% DCT block processing, of an image, retaining k coefficients
% in each 8 by 8 zizaged block.
% All remaining coefficients are either set to zero (R = 0)
% or the first 2R+1 of them are approximated using:
% Linear or quadratic interpolation or moment preserving.
% MSE =Mean Square Error. PSNR = Peak Signal-to-Noise Ratio
clear;
clc;
clf;

fn = input('Enter Image File Name: ','s');
[a,map] = bmpread(fn);
X = ind2gray(a,map);
[N,M] = size(X);
R = 4;
k = 8;

subplot(2,2,1); imshow(X,256);
pause;


Y = blkproc(X,[8 8],'dctmi',k,0);
subplot(2,2,2); imshow(Y,256);
MSE = sum(sum((X - Y).^2))/(N*M);
PSNR = 10 * log10(255*255/MSE);
disp(MSE); disp(PSNR);
pause;

Y = blkproc(X,[8 8],'dctmi',k,R);
subplot(2,2,3); imshow(Y,256);
MSE = sum(sum((X - Y).^2))/(N*M);
PSNR = 10 * log10(255*255/MSE);
disp(MSE); disp(PSNR);
pause;

Y = blkproc(X,[8 8],'dctmmp',k,R);
subplot(2,2,4); imshow(Y,256);
MSE = sum(sum((X - Y).^2))/(N*M);
PSNR = 10 * log10(255*255/MSE);
disp(MSE); disp(PSNR);
```

- Linear MP Piecewise Approximation for 2-D Surface block diagram is shown in Figure I.

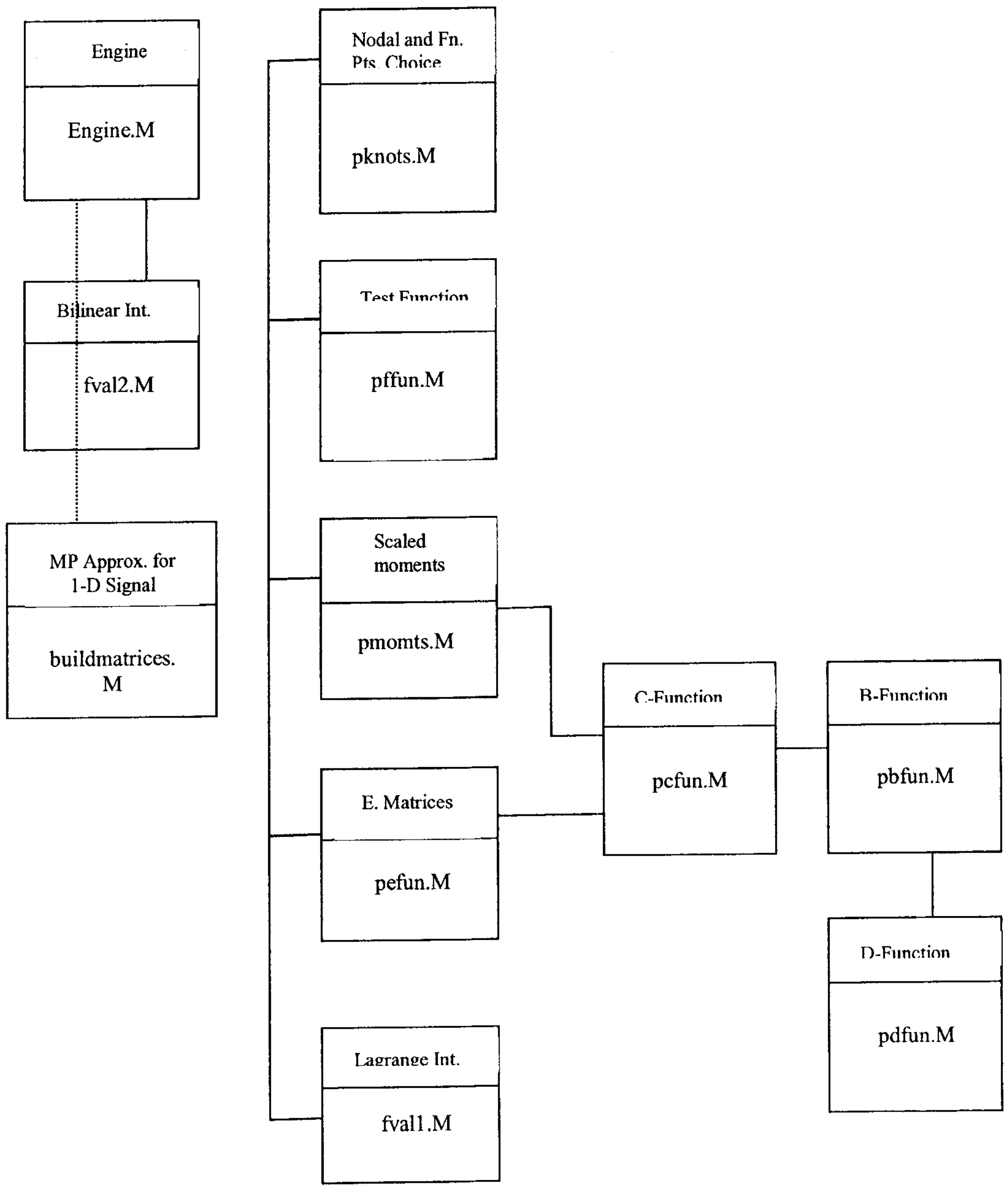


Fig.I. Matlab Code Block Diagram for 2-D Surface

E. Linear Piecewise Approximation for 2-D Surface

- Engine

```
clc;
clear;

%Initializations
%nb = input('Enter no. of blocks: ');
nb = 1;
%R  = input('Enter Knot Sampling Rate: ');
R = 20;
%nk = input('Enter no. of Knot points: '); more than 6 causes a deformation
in results
nk = 4;
d = 0.1;
kw = 5;          %noise level
x0 = 0;

%Initializations based on calculations
n = R*(nk-1);

for i = 1:n+1
    x(i) = (i-1)*d;
end
y = x;

for i = 1:nk
    xk(i) = (i-1)*R*d;
```

```
end
yk = xk;


[X,Y] = meshgrid(x,y);
[XK,YK] = meshgrid(xk,yk);
[rt,ct] =  size(X);


%Generating Plain Surface
Rxy = sqrt(X.^2+Y.^2) + eps;
Z0 = sin(Rxy)./Rxy;
% Z0 = pi*(Rxy.^2)./Rxy;
% Z0 = peaks(rt);


[Lx,mx] = size(X);
[Ly,my] = size(Y);


r = rand(Lx,my);
for k = 0:kw


  w = 0.1*k;
  ww(k+1) = w;


  Z = sin(Rxy)./Rxy + r*w;
  %Z = pi*(Rxy.^2)./Rxy + r*w;
  %Z = peaks(rt) + r*w;


    % ZI Knot Points Surface over the original surface
    in = 1;
    for i = 1:nk
        if (i == 1)
            in = 1;
        else
```

```
            in = 1 + (i-1)*R;
        end
        for j = 1:nk
            if (j == 1)
                ou = 1;
            else
              ou = 1 + (j-1)*R;
          end
          ZI(i,j) = Z0(in,ou);
      end
end

% IXK Knot points surface over the noisy surface
in = 1;
for i = 1:nk
    if (i == 1)
        in = 1;
    else
        in = 1 + (i-1)*R;
    end
    for j = 1:nk
        if (j == 1)
            ou = 1;
        else
            ou = 1 + (j-1)*R;
        end
        IXK(i,j) = Z(in,ou);
    end
end

I = interp2(X,Y,Z,XK,YK,'linear');
```

```
%This is the part responsible for building U

tempY = Y';

tempZ = Z';

for i = 1:n+1

    if (i == 1)

    [GF,G2] = buildmatrices(w,yk(1),d,R,nk,n,tempY(i,:),tempZ(i,:),yk);

    elseif (i == 2)

            index = 1 + (i-1)*R;

    [GF,G2] = buildmatrices(w,yk(1),d,R,nk,n,tempY(i,:),tempZ(i,:),yk);


    else

            index = (i-1)*R;

    [GF,G2] = buildmatrices(w,yk(1),d,R,nk,n,tempY(i,:),tempZ(i,:),yk);


    end

    U(:,i) = GF';

    G2a(i,:) = G2';

end


for i = 1:n+1

    if (i == 1)

            [GF,G2] = buildmatrices(w,xk(1),d,R,nk,n,X(i,:),Z(i,:),xk);

    elseif (i == 2)

            index = 1 + (i-1)*R;

            [GF,G2] = buildmatrices(w,xk(1),d,R,nk,n,X(i,:),Z(i,:),xk);


    else

            index = (i-1)*R;

            [GF,G2] = buildmatrices(w,xk(1),d,R,nk,n,X(i,:),Z(i,:),xk);


    end
    W(i,:) = GF;

    G2b(i,:) = G2';

end


index = 1;

for i = 1:nk

    for j = 1:nk

        Gb(i,:) = G2b(index,:);

    end

    if (i == 1)

        index = 1 + (i-1)*R;

    else

        index = (i-1)*R;

    end

end


index = 1;

for i = 1:nk

    for j = 1:nk

        Ga(i,:) = G2a(index,:);

    end

    if (i == 1)

        index = 1 + (i-1)*R;

    else

        index = (i-1)*R;

    end

end


%GI Knot points surface on the interpolation result

GI = fval2(xk,yk,R,d,nk,I);

%Knot points surface on the MP result

Temp = Ga + Gb;
```

```
    G = 0.5*Temp;

    Temp = W+U;

    V=0.5*Temp;

    % Calculating MSE

    msef = mean((Z0 - V) .^2);

    msei = mean((Z0 - GI) .^2);

    ef(k+1) = mean(msef);

    ei(k+1) = mean(msei);

end


       % The resulting surfaces.

       % mesh(X,Y,Z0);

       % figure, mesh(X,Y,Z);

       % figure, mesh(X,Y,V);

       % figure, mesh(X,Y,GI);


       % At Nodal Points

       % figure, mesh(XK,YK,ZI);

       % figure, mesh(XK,YK,IXK);

       % figure, mesh(XK,YK,G);

       % figure, mesh(XK,YK,I);


       %Percentage of Error Graph

       figure, plot(ww,ef,'r',ww,ei,'b');

       xlabel('Noise Level'); ylabel('MSE');

       title('Piecewise Linear Approximation');
```

- Build Matrices

```
function [GF,G2] = buildmatrices(w,x0,d,R,nk,n,X,F,XK);


clc;


s = nk - 1;

c = X(n+1);

YK = XK/c;

[SIG,M] = pmomts(X,F,c,nk);

E = pefun(YK,s);

G2 = pinv(E)*SIG;

GF = fval1(X,XK,R,G2);
```

- Bilinear Interpolation Function

```
function Resultx = fval2(xk,yk,R,d,nk,I);


point_n = R+1;

n = R*(nk-1);

e = 0.001;

Resultx = [];

for i = 1:nk-1

    Resulty = [];

    for j = 1:nk-1


        x1 = xk(i); x2 = xk(i+1);
```

```
            y1 = yk(j); y2 = yk(j+1);
            z11 = I(i,j); z12 = I(i,j+1);
            z21 = I(i+1,j); z22 = I(i+1,j+1);

            x = linspace(x1, x2, point_n);
            y = linspace(y1, y2, point_n);

            [xx, yy] = meshgrid(x, y);

            p1 = (xx-x1)/(x2-x1);
            p2 = 1-p1;
            q1 = (yy-y1)/(y2-y1);
            q2 = 1-q1;
            zz = z11*(p2.*q2) + z12*(p2.*q1) + z21*(p1.*q2) + z22*(p1.*q1);
            if (j == 1)
                Resulty = [zz(:,1:R)];
            elseif (j == (nk-1))
                Resulty = [Resulty zz];
            else
                Resulty = [Resulty zz(:,1:R)];
            end
        end
        if (i == 1)
            Resultx = [Resulty(1:R,:)];
        elseif (i == (nk-1))
            Resultx = [Resultx;Resulty];
        else
            Resultx = [Resultx;Resulty(1:R,:)];
        end
    end
```

Appendix C.

Bilinear MP Derivation

Appendix C: Bilinear MP Piecewise Approximation Derivation for 2-D Surface

Assuming that between the nodal points x_a, y_b, x_c and y_d the function is piecewise linear, such that x lies between x_a and x_c while y lies between y_b and y_d; we use Bilinear Interpolation formula

$$z_{xy} = (1-\frac{x-x_a}{x_c-x_a})(1-\frac{y-y_b}{y_d-y_b})z_{ab} + (\frac{x-x_a}{x_c-x_a})(1-\frac{y-y_b}{y_d-y_b})z_{cb} + (\frac{x-x_a}{x_c-x_a})(\frac{y-y_b}{y_d-y_b})z_{cd}$$
$$+(1-\frac{x-x_a}{x_c-x_a})(\frac{y-y_b}{y_d-y_b})z_{ad} \qquad \text{(Eq. a)}$$

to compute the kth scaled moment over that region:

$$\mu_{p+q}((a,b),(c,d)) = \int_b^d\int_a^c x^p y^q z_{xy} dx.dy \qquad \text{(Eq. b)}$$

With $D(t,s,m,n) = \int_b^d\int_a^c \frac{t^m s^n}{(t_c - t_a)(s_d - s_b)} dt.ds$ then the evaluation of the integral gives

$$\mu_{m+n}((i,j),(i+1,j+1)) = z_{i,j}bn(j+1,i+1,m+1,n+1) + z_{i,j+1}bp(j,i+1,m+1,n+1)$$
$$+ z_{i+1,j}bn(j,i,m+1,n+1) - z_{i+1,j}bp(j+1,i,m+1,n+1) \qquad \text{(Eq. c)}$$

Where bn(u,v,k,l) = D(t,s,k,l) - s_u D(t,s,k,l-1) - t_vD(t,s,k-1,l) + $t_v s_u$D(t,s,k-1,l-1)

And bn(u,v,k,l) = D(t,s,k,l) + s_u D(t,s,k,l-1) + t_vD(t,s,k-1,l) - $t_v s_u$D(t,s,k-1,l-1)

For nodal points $\{z_{i,j}$ i=0,1...x & j=0,1...y}, then the total scaled moments for the x*y segments will be:

$$\mu_{m+n} = \sum_j \sum_i C_{ij}(m,n) * z_{ij} \text{, i=0,1,...,x \& j=0,1,...,y \& m+n = 0,1,...,x}$$

Where $C_{ij}(m,n)$ is a function defined as follows:

$$C_{ij}(m,n) = \begin{cases} \text{bn(j+1, i+1, m+1, n+1)} & \text{for i=0, j=0} \\ \text{bp(j-1, i+1, m+1, n+1) + bn(j+1, i+1, m+1, n+1)} & \text{for i=0, j=1...y-1} \\ \text{bp(j-1, i+1, m+1, n+1)} & \text{for i=0, j=y} \\ \text{bn(j+1, i+1, m+1, n+1) - bp(j+1, i -1, m+1, n+1)} & \text{for i =1...x-1, j=0} \\ \text{- bp(j+1, i -1, m+1, n+1)} & \text{for i=x, j=0} \\ \text{bn(j-1, i -1,m+1,n+1) + bp(j-1, i+1, m+1, n+1)} & \text{for i=1...x-1,j=y} \\ \text{bn(j-1, i -1,m+1,n+1)} & \text{for i=x, j=y} \\ \text{bn(j-1, i -1,m+1,n+1) - bp(j+1, i -1, m+1, n+1)} & \text{for i=x, j=1...y-1} \\ \text{bn(j-1, i -1,m+1,n+1) - bp(j+1, i -1, m+1, n+1) +} & \\ \text{bp(j-1, i+1, m+1, n+1) - bn(j+1, i+1, m+1, n+1)} & \text{for i=1...x-1, 1...y-1} \end{cases} \quad \text{Eq. d}$$