CHEBYSHEV APPROXIMATION OF DISCRETE POLYNOMIALS AND SPLINES

FAUZIAHANIM BINTI CHE SEMAN

A thesis submitted as a partial fulfillment
of requirements for the award of the
Degree of Master of Engineering (Electrical)

Department of Electrical Engineering
Faculty of Engineering
Kolej Universiti Teknologi Tun Hussein Onn

SEPTEMBER 2004

WITH THE MOST BLESSFUL AND GRACEFUL OF ALLAH

ACKNOWLEDGEMENT

I would like to thank my advisor, Professor ShahNor b Basri, for his guidance at each stage of this work, moral support through the years and thorough editing of the manuscript. In many ways, I am indebted to him.

I would also like to thank Associate Professor Siti Fauzeyah and Ms Fazita for editing the thesis.

I could not thank enough my parents Che Seman Che Muda and Nor Azizah Embong, my siblings Ijah, Udin, Sidah, Irah, Na, Yie, Mie, Hakim, Muiz, Kak Cik and Fikah for their love and faith in me.

Most of all, my dearest thanks must go to my fiance Mohd Zaharudin and my friends Kak Ila, Kak Nie, Kak Liya, Kak Zu, Kak Sha, Kak Ziana, Kak Fizah and Kak Herda

ABSTRACT

This work is concerned with the approximation of discrete data using polynomials and splines based on the Chebyshev approximation criterion. Five algorithms are proposed in this work to implement the Chebyshev approximation criterion. These algorithms use either cubic splines or Lagrange polynomials to construct the approximation function. The efficiency of each algorithm developed is evaluated on the basis of the number of iterations and extreme points required for convergence, and the magnitude of the errors generated. One measure of efficiency is the minimum number of knots required to capture the full behavior of the data, and the size of the error between the actual data and its approximation. These knots are the extreme points that determine how well the fitting function approximates the actual data. Since a critical step in the approximation is identifying the extreme points, in this work we have proposed a novel procedure for finding the set of extreme points for the incoming discrete data efficiently. The procedure developed in this work use polynomials and cubic splines to construct the approximation function. The output of the algorithm is a set of extreme points that can be used to construct a minimal error approximation function. A total of five algorithms based on the Lagrange polynomials and cubic splines have been developed in this work to identify the extreme points. The efficiency of each algorithm is analysed in terms of computation time and the magnitude of the errors generated. In real environment, it is hope that this theoretical work can be applied to actual data and solves problems which occur in data processing.

ABSTRAK

Projek ini meneliti penghampiran data diskret menggunakan kriteria penghampiran Chebyshev. Lima algoritma dicadangkan bagi mengimplementasikan kriteria penghampiran Chebyshev. Kesemua algoritma menggunakan sama ada kaedah gelugur kubus ataupun polinomial Lagrange bagi membina fungsi penghampiran. Kecekapan setiap algoritma yang dibina diuji melalui bilangan lelaran dan titik ekstrem yang diperlukan untuk penumpuan, serta magnitud ralat yang dihasilkan. Pengukuran kecekapan turut melibatkan bilangan minimum knot yang diperlukan bagi mewakili keseluruhan data, dan saiz ralat di antara data sebenar dan data penghampiran. Knot-knot ini adalah titik ekstrem yang menentukan sejauh mana fungsi pemasangan menghampiri data sebenar. Oleh kerana langkah yang paling kritikal adalah mengenalpasti titik ekstrem, projek ini mencadangkan prosedur baru bagi mendapatkan set titik ekstrem dari data diskret secara efisien. Prosedur dibina menggunakan polinomial dan gelugur kubus untuk membina fungsi penghampiran. Output algoritma ini adalah satu set titik ekstrem yang boleh digunakan untuk membina fungsi penghampiran dengan ralat yang minimum. Lima algoritma berdasarkan polinomial Lagrangean dan gelugur kubus dibina untuk mendapatkan titik ekstrem. Kecekapan setiap algoritma dianalisis berpandukan masa pengiraan dan magnitud ralat. Dalam situasi sebenar, hasil kajian secara teori ini diharapkan boleh digunakan pada data sebenar dan menyelesaikan permasalahan yang berlaku di dalam pemprosesan data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

f(t)	-	Continuous Function
$\Phi(t)$	-	Basis Function
g(A,t)	-	Approximate Function
e(t)	-	Error Function
δ	-	Chebyshev error norm
D	-	Determinant
S	-	Spline function
T	-	Chebyshev polynomial

LIST OF APPENDICES

CHAPTER I

INTRODUCTION

1.1 Background

The phenomenal advancements that have been achieved in electronics technology in the last few decades have brought about new hardware and possibilites. Concomitant with these advancements is the explosive growth of computing power available to engineers and scientists alike. With the widespread availability of cheap data storage and processing hardware, new and more powerful signal processing techniques are being explored to take advantage of these developments. Conversely, many of the techniques and algorithms that are currently being investigated required for hardware with huge data storage and computing capabilities.

The approximation concept has a close relation with filter application. Like a approximation, a filter will remove the unimportant of incoming data and only allow the important data to proceed to the next stage of a system. For example, a bandwidth of a human voice can rise up to 8 kHz but in telephone system, the bandwidth of voice data is limited to 4 kHz[1]. This limitation cause the voice of customers in telephone line sounds

different with conversation in a real situation. However, with a lower quality of a system, the information from the voice still can be understood by the customers. A Chebyshev filter is a well known application that adapts the Chebyshev approximation [2]. An ideal filter introduces problems such as requirement of a large size of bandwidth and harmonic response in digital signal processing. Two discontinuities of pass band and stop band are joined by adapting Chebyshev approximation to provide a similar response of an ideal filter. Data approximation can be extended to filter design (see for example, references [2], [3], [4]); this work, however, does not cover the the use of either the Chebyshev nor the Lagrange approximation to filter design.

A number of works have been reported in the literature on the use of polynomials and splines to implement the Chebyshev approximation for modelling of incoming discrete data [5], [6], [7]. Data collected by sensors readings or non polynomials discrete functions which required curve fitting need to be approximated or interpolated by polynomials and splines to be analyzed for further studies. In general, curve fitting problems occur in signal processing, graphics, statistical analysis and in geometric modeling.

Figure 1.1 shows a block diagram representation of digital speech processing using Chebyshev approximation to remove unimportant data. The signal that passes the stage of Chebyshev approximation model block can represent the entire data by using less number of data.

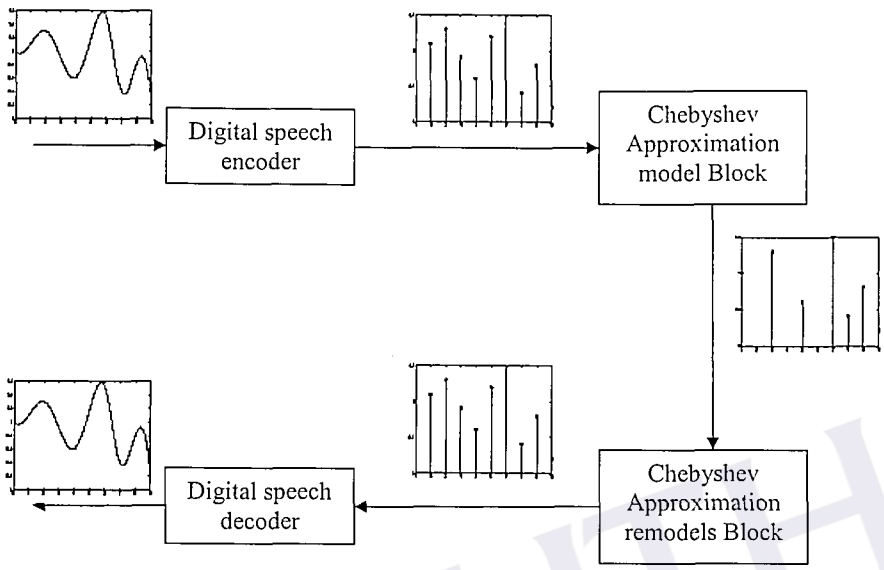


Figure 1.1: Application of Chebyshev approximation

Reconstruction of the incoming discrete data can be achieved in either of two ways; namely, interpolation, and approximation [3]. In the interpolation method, an interpolation function is determined which passes through the actual values of the discrete function.

The interpolation function should pass the exact values of the discrete function. The interpolation function is applied in an estimation of the values between two discrete data where the connectivity between two knots is not known.

Conversely, an approximation finds a curve that passes near a set of knots, which is used to express whole set of incoming dicrete data with a much lesser number of data points.

An approximation needs certain number of knots as a control points, known as extreme points where the set of extreme points of data will perform approximate data similar to actual data. However, the approximation creates a new element that differentiates the correct data from the approximation data known as error. The approximation methods always involve errors and errors is measured for the determination of appropriate approximation. Error occurs when there is a difference between true values and approximate values [8]. The approximation must minimize the error with respect to actual data. The various distances are based on norms such as least squared norm, Euclidean norm or infinity norm [3]. However, in wide area of data, Chebyshev error norm usually perform better because the norm shows the maximum error which occurs in an interval of data.

The characteristics of function of polynomials and splines make it suitable to interpolate data in certain conditions. Polynomial is easy to manipulate and presents a good approximation of data patterns but can wildly oscillate during large intervals especially for polynomial orders that are bigger than three [2]. The better option is to use spline in the approximation. Spline has good computational properties such as compact representation and computational stability. Splines have most attractive properties because they can be divided into several segments which can prevent the function from oscillating too far [9]. Another important consideration is the spline representations are more numerically stable and computationally efficient. Splines require a lower order than polynomial to fit the same performance of polynomial. Nevertheless, using splines as the solving methods in Chebyshev approximation will cause a non-linear approximation system. The approximation will be more complicated and complex. Therefore, it is important to analyze the characteristics of approximations function in order to have the best approximation of data.

1.2 Problem Statement

An approximation conveys a simpler function with less number of data or knots to express the entire real data. The approximation creates errors that differentiate the actual data from the approximate data. The approximation can save on the computation required for high level signal processing, which is directly related to the number of the knots [10].

Many engineering applications involve signal processing when analyzing the incoming discrete data such as in the robotics motion design [5]. The resulting motion or moving frames of the robot can be represented by polynomials and splines. The unimportant or non critical data movement can be approximated to reduce memory space. The model of a frame of speech data also can be achieved using spline [1]. The speech model involves data that is determined by frequencies. The high peaks of the frequencies become the critical data and should be considered in the approximation. The approximation of data is related to data compression.

The critical part of the approximation is to identify the critical knots from a set of data [8]. These knots are known as the extreme points that can control the approximation function to perform a similar approximate data according to actual data. The effectiveness of the approximation function is determined by the minimum number of knots which describes the behavior of data and present less error between real data and approximation data.

There are two approaches which are commonly used in an approximation of data. The first approach of getting the extreme points is by removing the unimportant knots and known as data reduction [11]. The knot is removed eventually until the approximate data gives an approximation within an acceptable error bound. Data reduction has better performance to approximate in random behavior. The second approach begins with essentially no knots and builds up the approximating function by adding knot. The knot is

added eventually until the approximation satisfies the performance according to the actual data. This second approach is suitable to approximate a well behaved data.

This project proposed a new approach that does not involves data reduction or data increment as described previously in the paragraph. The discrete data are well distributed on a given interval and the connectivity between the knots is not known. The project initializes several number of knots and algorithms that adapt the Chebyshev approximation are proposed to acquire a set of knots. These knots are known as extreme points. The algorithms will iterate until a set of the extreme points is converged. The approximation will introduce errors. The approximation which uses different number of extreme points will be analyzed.

1.3 Scope and Objectives

The research fulfills the objectives that includes to model a chebyshev approximation in representation of the two dimensional incoming discrete data. The data forms in linear. The research also differentiates the polynomial approximation and spline approximation in terms of total extreme points, the computation time and the generated error. Since polynomials and splines have their own characteristic in interpolation, hence of they cannot simply be used to represent data.

The research proposes a procedure of the Chebyshev approximation by using the polynomials and splines as an approximate function. Outcome of the algorithm is a set of extreme points that can represent an approximate data which has similar response due to the actual data. Several algorithms that use polynomial and spline are developed to identify the extreme points and the results are evaluated according to the behavior of

polynomial and spline. Results of the algorithms are analyzed in terms of computation time and the generated error.

The algorithms of Chebyshev approximation that use a fixed knot polynomial and spline are developed in this project. The fixed knot means that the given knot or data location cannot be changed by the approximation process. The approximation only involves two dimensional data. Since the incoming data is distributed in discrete domain, the analysis during the approximation only considers the data on the grid sampling. Total numbers of the extreme points are fixed during the initialization of the extreme points. The effectiveness of the approximation function is determined in term of the generated error and the error of the algorithm is analyzed in least square sense.

In general, curve fitting problems occur in signal processing, graphics, statistical analysis and in geometric modeling. The high volume of incoming data is overwhelming to memory space and needs an efficient approximation algorithm. The approximation function will approximate the actual data by using less numbers of data but creates error. In approximation data, the most important part is the identification of the correct extreme points, otherwise the approximation function does not generate the best approximation of data. Thus, the algorithms are developed to identify a set of the extreme points and generate the best approximation of the incoming data.

Chapter II is a literature review on reseach related to the author's work. Also included is a summary of some the works that has been carried out elsewhere on filter design and multidimensional and experimental data analysis.

Chapter III discusses the related theoretical background for Chebyshev approximation on a continuous domain. Most of the theories hold for discrete domain approximation. Also, the general description of the Reméz Exchange Algorithm is

provided in this chapter. The chapter also discusses the numerical background of polynomials and splines. At the end of the chapter, the relation between the continuous and discrete data is discussed.

Chapter IV proposes several algorithms that are developed in the project and the algorithms are important to generate a good procedure of approximation data. The chapter also discusses about the software development. The MATLAB code is programmed to realize the outcome of the algorithms.

Chapter V discusses the results of the comparison between of the algorithms proposed in Chapter IV. The analyses involve the number of total iteration where the extreme points are converged and the absolute error. Then the relation of the results is discussed with the performance of the equipment which are processing time and memory space.

Chapter VI concludes the discussion with a summary of the results obtained. Also discussed are the possible applications of the results. Finally, several recommendations are suggested for further research on this theme.

CHAPTER II

LITERATURE REVIEW

2.0 Introduction

This chapter reviews the works that has been published in the literature which are related to this project's scope. The literature review includes both the theoretical and experimental works. These works include that on filter design, manipulation of multidimensional data, and experimental data analysis.

2.1 General Background

Many well known applications deal with discrete data that need curve fitting to closely express the whole behaviour of data [12], [13], [14] with less data. For example, fitting smooth curve is one of the important themes on pattern recognition and data analysis.

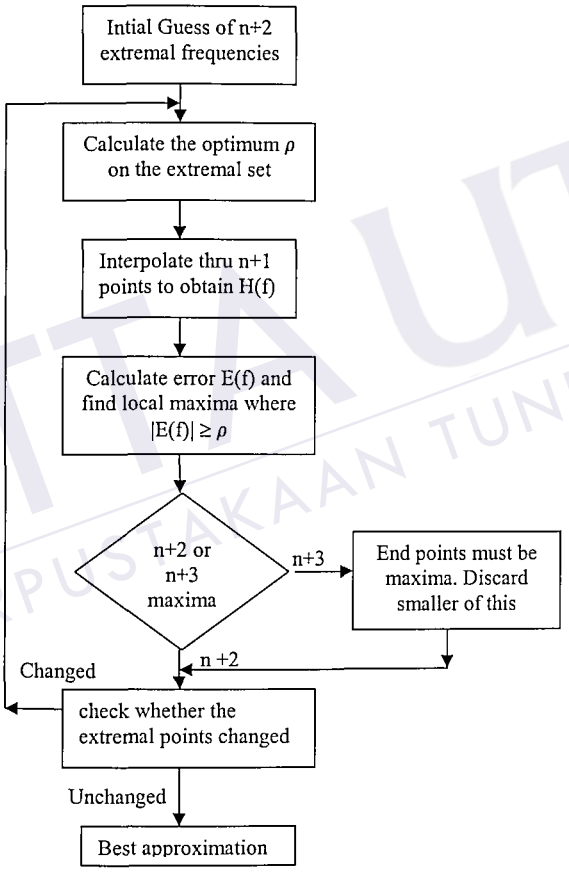
One measure of the effectiveness of the fitting function is the minimum number of knots required to capture the full behavior of the data, and the size of the absolute error between the actual data and its approximation. These knots, or extreme points, determine how well the fitting function approximates the actual data. Since a critical step of the approximation procedure is to identify the extreme points, many researchers have developed their own algorithms to suit the different incoming data types. These include, among others, data reduction methods that progressively reduce the number of knots required, until the limiting error tolerance between the approximating function and actual data is reached. Other methods start with no knots and progressively build up the approximating function by adding more knots until the error tolerance is reached.

Many data fitting algorithms have been developed, based on different goodness-of-fit measures. One popular goodness-of fit criterion is based on the minimisation of the mean-square error. Another equally popular goodness-of-fit measure is the Chebyshev approximation criterion which based on the minimisation of the maximum value of the difference between the original curve and its approximation function. The Chebyshev approximation criterion has been routinely used since the 1960s in modern digital filter design techniques. Filters designed based the Chebyshev criterion produce responses approximating closely those of ideal filters and hence circumvent the appearance of spurious harmonic responses in digital signal processing. Crousel and Neirynck [2] developed an approximation of the ideal filter characteristic as a ratio of polynomials from the Chebyshev criterion. Parks and McClellan [4] applied the Chebyshev approximation criterion to design finite-length impulse response filters. Data fitting using the least square approximation criterion has been done Haruki and Horiuchi [7] using splines.

The curve-fitting procedure involves evaluating *distance* between data points and a prospective fitting function. Mizuta [12] has developed an algorithm that is based on Newton-Raphson to evaluate the exact distance between the data points and a prospective curve. Kitson [11] introduced three steps of data reduction strategy which are significant

APPENDIX A

A.1 Parks and McClellan design algorithm



APPENDIX B

Matlab Code

B.1 Algorithm-1

```
n= 0:1:100;
fn= sin(2*pi*(n-50)/100);
iteration=0;
ce=cell(1,20);
tabser=cell(1,20);

% initial extremal points

en=[ 0 10 20 30 40 100];
while iteration<=20
fen= sin(2*pi*(en-50)/100);
p=polyfit(en,fen,5);
f = polyval(p,n);

figure(iteration+1);
plot(en,fen,'o',n,f); grid on;hold on;
plot(n,fn,'m')

error=1;
%while iteration>=0 & error>=0.01
    errortemp=f-fn;
    abserror=max (abs(errortemp));
    tabser{iteration+1}=abserror;
    num=length(en);
    numb=1;

     % correct the values of the extreme points by adapting Chebyshev
approximation
    while numb<=num                                % adding interpolating
values with error
         fen(numb)=fen(numb)+abserror;
        if((numb)==length(en))
            break
        else
        fen(numb+1)=fen(numb+1)-abserror;
        numb=numb+2;
    end
    end

    % find the error function between approximate data and actual data

    ji=lagrange(n,en,fen);                         % finding new extremal
points
    dip=ji-fn;
```

```
    beza=diff(dip);
    nume=1;
    extrem=zeros(1,length(en));
    bilang=2;
    extrem(1)=en(1);
    extrem(length(en))=en(length(en));

     % obtain the extreme points by differentiating the error function
    while nume<=length(beza)-1
        check=beza(nume+1).*beza(nume);
       if check<=0
           extrem(bilang)=nume;
           bilang=bilang+1;
       end
       nume=nume+1;
   end

   %update and save a new set of the extreme points

   en=extrem;
   ce{iteration+1}=extrem;


  % the extreme converge or not
  % if extremal points converged, iteration stopped
  % otherwise, continue the procedure to find the extreme points


if (iteration>2 & ce{iteration}==ce{iteration-1})
   break

  % stopping criteria
  % check whether the extreme points keep iterating at the same values
  % if the extreme points keep iterating at the same sets, iteration
stopped
  % otherwise,continue the procedure to find the extreme points


   elseif (iteration>4 & ce{iteration}==ce{iteration-2})
      break

  elseif (iteration>6 &ce{iteration}==ce{iteration-3})
 break

elseif (iteration>8 &ce{iteration}==ce{iteration-4})
    break

 % else, iteration continue
else
       iteration=iteration+1;
end
end
```

B.2 Algorithm-2

```
n= 0:1:100;
fn= sin(2*pi*(n-50)/100);
iteration=0;
ce=cell(1,10);

% initial extremal points
en=[ 0 10 20 30 40 100];

while iteration<=10
fen= sin(2*pi*(en-50)/100);
f=spline(en,fen,n);
figure(iteration+1);
plot(en,fen,'o',n,f); grid on;hold on;
plot(n,fn,'m')

error=1;

    errortemp=f-fn;
    abserror=max (abs(errortemp));
    tabser{iteration+1}=abserror;
    num=length(en);
    numb=1;

    % correct the values of the extreme points by adapting Chebyshev
approximation
  while numb<=num                          % adding interpolating
values with error
        fen(numb)=fen(numb)+abserror;
        if((numb)==length(en))
            break
        else
        fen(numb+1)=fen(numb+1)-abserror;
        numb=numb+2;
    end
    end

    % find the error function between approximate data and actual data

    ji=lagrange(n,en,fen);                    % finding new extremal
points
    dip=ji-fn;
    beza=diff(dip);
    nume=1;
    extrem=zeros(1,length(en));
    bilang=2;
    extrem(1)=en(1);
    extrem(length(en))=en(length(en));

   % obtain the extreme points by differentiating the error function

    while nume<=length(beza)-1
```

```
        check=beza(nume+1).*beza(nume);
        if check<=0
            extrem(bilang)=nume;
            bilang=bilang+1;
        end
        nume=nume+1;
    end
    %update and save a new set of the extreme points

    en=extrem;
    ce{iteration+1}=extrem;


  % the extreme converge or not
  % if extremal points converged, iteration stopped
  % otherwise, continue the procedure to find the extreme points


if (iteration>2 & ce{iteration}==ce{iteration-1})
   break

  % stopping criteria
  % check whether the extreme points keep iterating at the same values
  % if the extreme points keep iterating at the same sets, iteration
stopped
  % otherwise,continue the procedure to find the extreme points


   elseif (iteration>4 & ce{iteration}==ce{iteration-2})
      break

   elseif (iteration>6 &ce{iteration}==ce{iteration-3})
 break

elseif (iteration>8 &ce{iteration}==ce{iteration-4})
    break

 % else, iteration continue
else
       iteration=iteration+1;
end
end
```

B.3 Algorithm-3

```
%define the actual data

n= 0:1:100;
fn= sin(2*pi*(n-50)/100);
iteration=0;
ce=cell(1,12);
tabser=cell(1,12);

  % initial extremal points
en=[ 0 10 20 30 40 100];
while iteration<=12
fen= sin(2*pi*(en-50)/100);
f=lagrange(n,en,fen);
figure(iteration+1);
plot(en,fen,'o',n,f); grid on;hold on;
plot(n,fn,'m')

error=1;
%while iteration>=0 & error>=0.01
    errortemp=f-fn;
    abserror=max (abs(errortemp));
    tabser{iteration+1}=abserror;
    num=length(en);
    numb=1;


    % correct the values of the extreme points by adapting Chebyshev
approximation

    while numb<=num                          % adding interpolating
values with error
        fen(numb)=fen(numb)+abserror;
        if((numb)==length(en))
            break
        else
        fen(numb+1)=fen(numb+1)-abserror;
        numb=numb+2;
    end
    end

    % find the error function between approximate data and actual data

    ji=lagrange(n,en,fen);                   % finding new extremal
points
    dip=ji-fn;
    beza=diff(dip);
    nume=1;
    extrem=zeros(1,length(en));
    bilang=2;
    extrem(1)=en(1);
    extrem(length(en))=en(length(en));
```

```
  % obtain the extreme points by differentiating the error function

    while nume<=length(beza)-1
        check=beza(nume+1).*beza(nume);
       if check<=0
           extrem(bilang)=nume;
           bilang=bilang+1;
       end
       nume=nume+1;
   end
   en=extrem;
   ce{iteration+1}=extrem;


  % the extreme converge or not
  % if extremal points converged, iteration stopped
  % otherwise, continue the procedure to find the extreme points


if (iteration>2 & ce{iteration}==ce{iteration-1})
   break

  % stopping criteria
  % check whether the extreme points keep iterating at the same values
  % if the extreme points keep iterating at the same sets, iteration
stopped
  % otherwise,continue the procedure to find the extreme points


   elseif (iteration>4 & ce{iteration}==ce{iteration-2})
      break

   elseif (iteration>6 &ce{iteration}==ce{iteration-3})
 break

elseif (iteration>8 &ce{iteration}==ce{iteration-4})
    break

 % else, iteration continue
else
       iteration=iteration+1;
end
end
```

B.4 Algorithm-4

```
n= 0:1:100;
fn= sin(2*pi*(n-50)/100);
iteration=0;
ce=cell(1,30);tabser=cell(1,30);

   % initial extremal points
en=[ 0 10 20 30 40 100];

% approximate the points using cubic spline
% display the approximate function

while iteration<=20

fen= sin(2*pi*(en-50)/100);
f=lagrange(n,en,fen);
figure(iteration+1);
plot(en,fen,'o',n,f); grid on;hold on;
plot(n,fn,'m')


error=1;

% estimate the absolute error

    errortemp=f-fn;
    abserror=max (abs(errortemp));
    tabser{iteration+1}=abserror;
    num=length(en);
    numb=1;


  % correct the values of the extreme points by adapting Chebyshev
approximation

    while numb<=num                              % adding interpolating
values with error
         fen(numb)=fen(numb)+abserror;
        if((numb)==length(en))
            break
        else
        fen(numb+1)=fen(numb+1)-abserror;
        numb=numb+2;
    end
    end

    % find the error function between approximate data and actual data

    ji=spline(en,fen,n);                        % finding new extremal
points
    dip=ji-fn;
    beza=diff(dip);
    nume=1;
    extrem=zeros(1,length(en));
    bilang=2;
    extrem(1)=en(1);
```

```
    extrem(length(en))=en(length(en));

    % obtain the extreme points by differentiating the error function

    while nume<=length(beza)-1
        check=beza(nume+1).*beza(nume);
       if check<=0
           extrem(bilang)=nume;
           bilang=bilang+1;
       end
       nume=nume+1;
   end

      %update and save a new set of the extreme points

   en=extrem;
   ce{iteration+1}=extrem;


  % the extreme converge or not
  % if extremal points converged, iteration stopped
  % otherwise, continue the procedure to find the extreme points


if (iteration>2 & ce{iteration}==ce{iteration-1})
   break

  % stopping criteria
  % check whether the extreme points keep iterating at the same values
  % if the extreme points keep iterating at the same sets, iteration
stopped
  % otherwise,continue the procedure to find the extreme points


   elseif (iteration>4 & ce{iteration}==ce{iteration-2})
      break

   elseif (iteration>6 &ce{iteration}==ce{iteration-3})
 break

elseif (iteration>8 &ce{iteration}==ce{iteration-4})
    break

 % else, iteration continue
else
        iteration=iteration+1;
end
end
```

B.5 Algorithm-5

```
n= 0:1:100;
fn= sin(2*pi*(n-50)/100);
iteration=1;
ce=cell(1,20);
tabser=cell(1,20);

% initial extreme points
en=[ 0 10 20 30 40 100];

while iteration<=30

% approximate the points using cubic spline
% display the approximate function

fen= sin(2*pi*(en-50)/100);
f=spline(en,fen,n);
figure(iteration);
plot(en,fen,'o',n,f); grid on;hold on;
plot(n,fn,'m')


error=1;


% estimate the absolute error

    errortemp=f-fn;
    abserror=max (abs(errortemp));
    tabser{iteration}=abserror;
    num=length(en);
    numb=1;

 % correct the values of the extreme points by adapting Chebyshev
approximation

    while numb<=num                                % adding interpolating
values with error
        fen(numb)=fen(numb)+abserror;
        if((numb)==length(en))
            break
        else
        fen(numb+1)=fen(numb+1)-abserror;
        numb=numb+2;
    end
end

% find the error function between approximate data and actual data


    ji=spline(en,fen,n);                          % finding new extremal
points
```

```
    dip=ji-fn;
    beza=diff(dip);
    nume=1;
    extrem=zeros(1,length(en));
    bilang=2;
    extrem(1)=en(1);
    extrem(length(en))=en(length(en));


% obtain the extreme points by differentiating the error function

    while nume<=length(beza)-1
        check=beza(nume+1).*beza(nume);
       if check<=0
           extrem(bilang)=nume;
           bilang=bilang+1;
       end
       nume=nume+1;
   end

   %update and save a new set of the extreme points

   en=extrem;
   ce{iteration}=extrem;


  % the extreme converge or not
  % if extremal points converged, iteration stopped
  % otherwise, continue the procedure to find the extreme points


if (iteration>2 & ce{iteration}==ce{iteration-1})
   break

  % stopping criteria
  % check whether the extreme points keep iterating at the same values
  % if the extreme points keep iterating at the same sets, iteration
stopped
  % otherwise,continue the procedure to find the extreme points


   elseif (iteration>4 & ce{iteration}==ce{iteration-2})
      break

   elseif (iteration>6 &ce{iteration}==ce{iteration-3})
 break

elseif (iteration>8 &ce{iteration}==ce{iteration-4})
    break

 % else, iteration continue
else
       iteration=iteration+1;
end
end
```

APPENDIX B

Matlab Code

B.1 Algorithm-1

```
n= 0:1:100;
fn= sin(2*pi*(n-50)/100);
iteration=0;
ce=cell(1,20);
tabser=cell(1,20);

% initial extremal points

en=[ 0 10 20 30 40 100];
while iteration<=20
fen= sin(2*pi*(en-50)/100);
p=polyfit(en,fen,5);
f = polyval(p,n);

figure(iteration+1);
plot(en,fen,'o',n,f); grid on;hold on;
plot(n,fn,'m')

error=1;
%while iteration>=0 & error>=0.01
    errortemp=f-fn;
    abserror=max (abs(errortemp));
    tabser{iteration+1}=abserror;
    num=length(en);
    numb=1;

     % correct the values of the extreme points by adapting Chebyshev
approximation
    while numb<=num                                % adding interpolating
values with error
         fen(numb)=fen(numb)+abserror;
        if((numb)==length(en))
            break
        else
        fen(numb+1)=fen(numb+1)-abserror;
        numb=numb+2;
    end
    end

    % find the error function between approximate data and actual data

    ji=lagrange(n,en,fen);                         % finding new extremal
points
    dip=ji-fn;
```