Scilab Textbook Companion for Numerical Analysis by I. Jacques And C. Judd[1]

Created by
Pragya Chordia And Shubham Mittal
Int. MSc. App. Math. (Pursuing)
Mathematics
IIT Roorkee
College Teacher
Prof. Roshan Lal
Cross-Checked by
Santosh Kumar, IITB

August 10, 2013

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: Numerical Analysis

Author: I. Jacques And C. Judd

Publisher: Chapman And Hall

Edition: 1

Year: 1987

ISBN: 0-412-27950-9

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes

List of Figures

Chapter 1

Introduction

Scilab code Exa 1.1 Illustrating big errors caused by small errors

```
//Illustrating that a small error in data provided
   can result in big errors.
//with original equations
//X+Y=2 & X+1.01Y=2.01
clear;
clc;
close();
A=[1 1;1 1.01];
B=[2 2.01]';
x=A\B;
disp(x,'Solutions are :')
x=linspace(-0.5,1.5);
y1=2-x;
y2=(2.01-x)/1.01;
subplot(2,1,1);
plot(x,y1)
plot(x,y2,'r')
xtitle('plot of correct equations','x axis','y axis'
   )
//with the equations having some error in data
//X+Y=2 & X+1.01Y=2.02
```

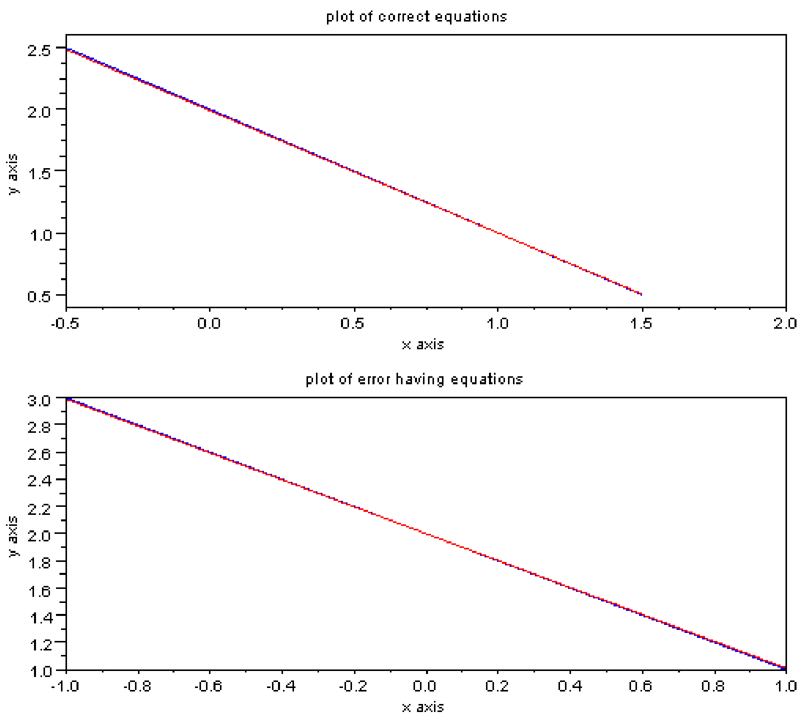

Figure 1.1: Illustrating big errors caused by small errors

```
A=[1 1;1 1.01];
B=[2 2.02]';
x=A\B;
disp(x,'Solutions are :')
subplot(2,1,2);
x=linspace(-1,1);
y1=2-x;
y2=(2.02-x)/1.01;
plot(x,y1)
plot(x,y2,'r')
xtitle('plot of error having equations','x axis','y
    axis')
```

Scilab code Exa 1.4 Calculating Induced instability through deflation method

```
//illustrating the induced instability through the
   deflation method of polynomial factorisation.
clear;
clc;
close();
x=poly(0,'x');
p3=x^3-13*x^2+32*x-20;//Given Polynomial
roots(p3)
//suppose that an estimate of its largest zero is
   taken as 10.1.Now devide p3 by (x-10.1)
p2=x^2-2.9*x+2.71;//the quotient
roots(p2)
disp('induced a large error in roots')
```

Chapter 2

Linear Algebric Equation

Scilab code Exa 2.1 Illutrates the effect of the partial pivoting

```
//Illutrates the effect of the partial pivoting using 3 significant //figure arithmetic with rounding
//first done without pivoting and then with partial pivoting
clear;
close();
clc;
A=[0.610,1.23,1.72;1.02,2.15,-5.51;-4.34,11.2,-4.25];
B=[0.792;12.0;16.3];
C=[A,B];
format('v',10);
n=3;
for k=1:n-1
  for i=k+1:n
    c=C(i,k)/C(k,k);
    for j=k:n+1
      C(i,j)=C(i,j)-c*C(k,j);
    end
```

```
  end
end
x3=C(3,4)/C(3,3);
x2=(C(2,4)-C(2,3)*x3)/C(2,2);
x1=(C(1,4)-C(1,3)*x3-C(1,2)*x2)/C(1,1);
disp([x1,x2,x3],'Answers without partial pivoting :')


C=[A,B];
format('v',5);
n=3;
for k=1:n-1
  m = max(abs(A(:,k)));
  for l=k:n
    if C(l,k)==m
      temp = C(l,:);
      C(l,:)= C(k,:);
      C(k,:)=temp;
      break;
    end
  end
  for i=k+1:n
    c=C(i,k)/C(k,k);
    for j=k:n+1
      C(i,j)=C(i,j)-c*C(k,j);
    end
  end
end
x3=C(3,4)/C(3,3);
x2=(C(2,4)-C(2,3)*x3)/C(2,2);
x1=(C(1,4)-C(1,3)*x3-C(1,2)*x2)/C(1,1);
disp([x1,x2,x3],'Answers using partial pivoting : ')
```

Scilab code Exa 2.2 Decomposition in LU form

```
//Illustrates the decomposition of every matrix into product of lower //and upper triangular matrix if diagonal elements of any one is '1' //then L and U could uniquely be determined.
clear;
close();
clc;
format('v',5);
A = {4,-2,2;4,-3,-2;2,3,-1];
L(1,1)=1;L(2,2)=1;L(3,3)=1;
for i=1:3
  for j=1:3
    s=0;
    if j>=i
      for k=1:i-1
        s=s+L(i,k)*U(k,j);
      end
      U(i,j)=A(i,j)-s;
    else
      for k=1:j-1
        s=s+L(i,k)*U(k,j);
      end
      L(i,j)=(A(i,j)-s)/U(j,j);
    end
  end
end
disp(L,'L =')
disp(U,'U =')
```

Scilab code Exa 2.3 LU factorization method for solving the system of equation

```
//Applying LU factorization method for solving the system of equation

clear;
close();
clc;
format('v',5);
A = {4,-2,2;4,-3,-2;2,3,-1];
for l=1:3
  L(l,l)=1;
end
for i=1:3
  for j=1:3
    s=0;
    if j>=i
      for k=1:i-1
        s=s+L(i,k)*U(k,j);
      end
      //disp(s,'sum :');
      U(i,j)=A(i,j)-s;
    else
      //s=0;
      for k=1:j-1
        s=s+L(i,k)*U(k,j);
      end
      L(i,j)=(A(i,j)-s)/U(j,j);
    end
  end
end
b=[6;-8;5];
c=L\b;
x=U\c;
disp(x,'Solution of equations :')
```

Scilab code Exa 2.4 LU factorisation method for solving the system of equation

```
//Application of LU factorisation method for solving
    the system of equation
//In this case A(1,1)=0 so to avoid the division by
   0 we will have to interchange the rows.

clear;
close();
clc;
format('v',5);
A = {2,2,-2,4;0,1,5,3;1,5,7,-10;-1,1,6,-5];
for l=1:4
  L(l,l)=1;
end
for i=1:4
  for j=1:4
    s=0;
    if j>=i
      for k=1:i-1
        s=s+L(i,k)*U(k,j);
      end
      //disp(s,'sum :');
      U(i,j)=A(i,j)-s;
    else
      //s=0;
      for k=1:j-1
        s=s+L(i,k)*U(k,j);
      end
      L(i,j)=(A(i,j)-s)/U(j,j);
    end
  end
end
b=[4;-6;14;0];
c=L\b;
x=U\c;
disp(x,'Solution of equations :')
```

Scilab code Exa 2.5 Choleski decomposition

```
//Solving the problem using Choleski decomposition
//Decomposition of a matrix "A" to L and L'

clear;
close();
clc;
format('v',7)
A = [4,2,-2;2,10,2;-2,2,3];
n = 3;
for i = 1:n
    for j = 1:i
        s=0;
        if i==j
            for k = 1:j-1
                s=s+L(j,k)*L(j,k);
            end
            L(j,j)= sqrt(A(j,j)-s);
        else
            for k = 1:j-1
                s=s+L(i,k)*L(j,k);
            end
            L(i,j)= (A(i,j)-s)/L(j,j);
        end
    end
end
U = L';
disp(L,'Lower triangular matrix is :')
disp(U,'Upper triangular matrix is :')
```

Scilab code Exa 2.6 Jacobi method

```
//Solving the problem using Jacobi method
//the first case in converging and the 2nd is
   diverging ..... drawback
//of jacobi method
//the ans is correct to 2D place

clear;
close();
clc;
format('v',7);
x1=[0,0];
x2=[0,0];
x3=[0,0];
x1(1,2)=0.2*(6-2*x2(1,1)+x3(1,1));
x2(1,2)=0.16667*(4-x1(1,1)+3*x3(1,1));
x3(1,2)=0.25*(7-2*x1(1,1)-x2(1,1));
i=1;
while (abs(x1(1,1)-x1(1,2))>0.5*10^-2 | abs(x2(1,1)-
   x2(1,2))>0.5*10^-2 | abs(x3(1,1)-x3(1,2))
   >0.5*10^-2 )
    x1(1,1)=x1(1,2);
    x2(1,1)=x2(1,2);
    x3(1,1)=x3(1,2);
    x1(1,2)=0.2*(6-2*x2(1,1)+x3(1,1));
    x2(1,2)=0.16667*(4-x1(1,1)+3*x3(1,1));
    x3(1,2)=0.25*(7-2*x1(1,1)-x2(1,1));
    i=i+1;
end
disp([x1; x2; x3],'Answers are :')
disp(i,'Number of Iterations :')
```

```
x1=[0,0];
x2=[0,0];
x3=[0,0];
x1(1,2)=4-6*x2(1,1)+3*x3(1,1);
x2(1,2)=0.5*(6-5*x1(1,1)+x3(1,1));
x3(1,2)=0.25*(7-2*x1(1,1)-x2(1,1));
i=1;
while (abs(x1(1,1)-x1(1,2))>0.5*10^-2 | abs(x2(1,1)-x2(1,2))>0.5*10^-2 | abs(x3(1,1)-x3(1,2))>0.5*10^-2 )
    x1(1,1)=x1(1,2);
    x2(1,1)=x2(1,2);
    x3(1,1)=x3(1,2);
    x1(1,2)=(4-6*x2(1,1)+3*x3(1,1));
    x2(1,2)=0.5*(6-5*x1(1,1)+x3(1,1));
    x3(1,2)=0.25*(7-2*x1(1,1)-x2(1,1));
    i=i+1;
end
disp([x1; x2; x3],'Answers are :')
disp(i,'Number of Iterations :')
```

Scilab code Exa 2.7 Gauss Seidel method

```
//the problem is solved using Gauss-Seidel method
//it is fast convergent as compared to jacobi method

clear;
close();
clc;
format('v',7);
x1=[0,0];
```

```
x2=[0,0];
x3=[0,0];
x1(1,2)=0.2*(6-2*x2(1,1)+x3(1,1));
x2(1,2)=0.16667*(4-x1(1,2)+3*x3(1,1));
x3(1,2)=0.25*(7-2*x1(1,2)-x2(1,2));
i=1;
while (abs(x1(1,1)-x1(1,2))>0.5*10^-2 | abs(x2(1,1)-x2(1,2))>0.5*10^-2 | abs(x3(1,1)-x3(1,2))>0.5*10^-2 )
    x1(1,1)=x1(1,2);
    x2(1,1)=x2(1,2);
    x3(1,1)=x3(1,2);
    x1(1,2)=0.2*(6-2*x2(1,1)+x3(1,1));
    x2(1,2)=0.16667*(4-x1(1,2)+3*x3(1,1));
    x3(1,2)=0.25*(7-2*x1(1,2)-x2(1,2));
    i=i+1;
end
disp([x1; x2; x3],'Answers are :')
disp(i,'Number of Iterations :')
```

Scilab code Exa 2.8 Successive over relaxation method

```
//The method used to solve is SOR(Successive over-relaxation) method
//only marginal improvement is possible for this pasticular system since
//Gauss-Seidel iteration itself converges quite rapidly

clear;
close();
clc;
format('v',7);
```

```
x1=[0,0];
x2=[0,0];
x3=[0,0];
w =0.9;
x1(1,2)=x1(1,1)+0.2*w*(6-5*x1(1,1)-2*x2(1,1)+x3(1,1));
x2(1,2)=x2(1,1)+0.16667*w*(4-x1(1,2)-6*x2(1,1)+3*x3(1,1));
x3(1,2)=x3(1,1)+0.25*w*(7-2*x1(1,2)-x2(1,2)-4*x3(1,1));
i=1;
while (abs(x1(1,1)-x1(1,2))>0.5*10^-2 | abs(x2(1,1)-x2(1,2))>0.5*10^-2 | abs(x3(1,1)-x3(1,2))>0.5*10^-2 )
    x1(1,1)=x1(1,2);
    x2(1,1)=x2(1,2);
    x3(1,1)=x3(1,2);
    x1(1,2)=x1(1,1)+0.2*w*(6-5*x1(1,1)-2*x2(1,1)+x3(1,1));
    x2(1,2)=x2(1,1)+0.16667*w*(4-x1(1,2)-6*x2(1,1)+3*x3(1,1));
    x3(1,2)=x3(1,1)+0.25*w*(7-2*x1(1,2)-x2(1,2)-4*x3(1,1));
    i=i+1;
end
disp([x1; x2; x3],'Answers are:')
disp(i,'Number of Iterations :')
```

Scilab code Exa 2.9 Gauss Seidel and SOR method

```
//Solving four linear system of equations with Gauss-Seidel and SOR method
//the convergence is much faster in SOR method
```

```
clear;
close();
clc;
format('v',7);
x1=[0,0];
x2=[0,0];
x3=[0,0];
x4=[0,0];
x1(1,2)=-0.33333*(1-x2(1,1)-3*x4(1,1));
x2(1,2)=0.16667*(1-x1(1,2)-x3(1,1));
x3(1,2)=0.16667*(1-x2(1,2)-x4(1,1));
x4(1,2)=-0.33333*(1-3*x1(1,2)-x3(1,2));
i=1;
while (abs(x1(1,1)-x1(1,2))>0.5*10^-2 | abs(x2(1,1)-x2(1,2))>0.5*10^-2 | abs(x3(1,1)-x3(1,2))>0.5*10^-2 | abs(x4(1,1)-x4(1,2))>0.5*10^-2)
    x1(1,1)=x1(1,2);
    x2(1,1)=x2(1,2);
    x3(1,1)=x3(1,2);
    x4(1,1)=x4(1,2);
    x1(1,2)=-0.33333*(1-x2(1,1)-3*x4(1,1));
    x2(1,2)=0.16667*(1-x1(1,2)-x3(1,1));
    x3(1,2)=0.16667*(1-x2(1,2)-x4(1,1));
    x4(1,2)=-0.33333*(1-3*x1(1,2)-x3(1,2));
    i=i+1;
end
disp([x1; x2; x3; x4],'Answers are:')
disp(i,'Number of Iterations :')
w=1.6;
x1=[0,0];
x2=[0,0];
x3=[0,0];
x4=[0,0];
x1(1,2)=x1(1,1)-0.33333*w*(1+3*x1(1,1)-x2(1,1)-3*x4(1,1));
```

```
x2(1,2)=x2(1,1)+0.16667*w*(1-x1(1,2)-6*x2(1,2)-x3(1,1));
x3(1,2)=x3(1,1)+0.16667*w*(1-x2(1,2)-6*x3(1,2)-x4(1,1));
x4(1,2)=x4(1,1)-0.33333*w*(1-3*x1(1,2)-x3(1,2)+3*x4(1,1));
i=1;
while (abs(x1(1,1)-x1(1,2))>0.5*10^-2 | abs(x2(1,1)-x2(1,2))>0.5*10^-2 | abs(x3(1,1)-x3(1,2))>0.5*10^-2 | abs(x4(1,1)-x4(1,2))>0.5*10^-2)
    x1(1,1)=x1(1,2);
    x2(1,1)=x2(1,2);
    x3(1,1)=x3(1,2);
    x4(1,1)=x4(1,2);
    x1(1,2)=x1(1,1)-0.33333*w*(1+3*x1(1,1)-x2(1,1)-3*x4(1,1));
    x2(1,2)=x2(1,1)+0.16667*w*(1-x1(1,2)-6*x2(1,2)-x3(1,1));
    x3(1,2)=x3(1,1)+0.16667*w*(1-x2(1,2)-6*x3(1,2)-x4(1,1));
    x4(1,2)=x4(1,1)-0.33333*w*(1-3*x1(1,2)-x3(1,2)+3*x4(1,1));
    i=i+1;
end
disp([x1; x2; x3; x4],'Answers are :')
disp(i,'Number of Iterations :')
```

Chapter 3

Non linear algebric equations

Scilab code Exa 3.1 Bisection Method

```
//Bisection Method
clc;
clear;
close();
format('v',9);
b(1)=1;a(1)=0;k=5;
deff('[fx]=bisec(x)','fx =(x+1).^2.*exp(x.^2-2)-1');
x = linspace(0,1);
plot(x,((x+1).^2).*(exp(x.^2-2))-1);
//in interval [0,1]changes its sign thus has a root
//k = no of decimal place of accuracy
//a = lower limit of interval
//b = upper limit of interval
//n = no of iterations required
n = log2((10^k)*(b-a));
n = ceil(n);
disp(n,'Number of iterations : ')
for i = 1:n-1
    N(i) = i;
    c(i) = (a(i)+b(i))/2;
    bs(i) = bisec(c(i));
```

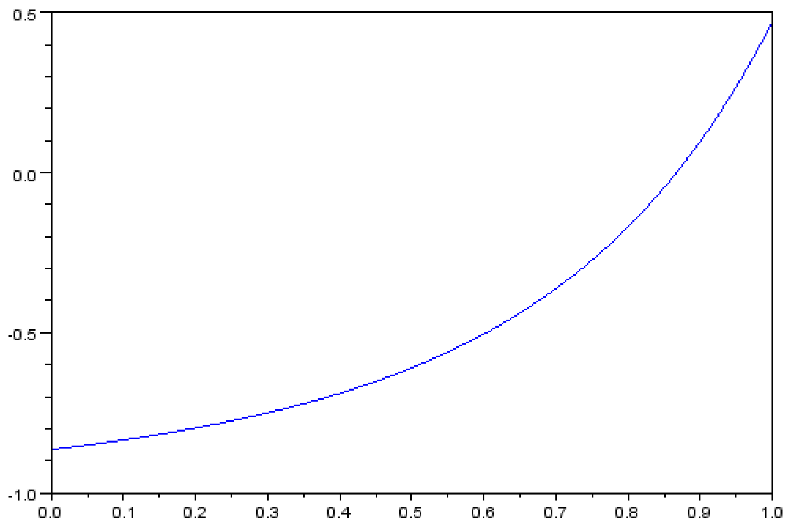

Figure 3.1: Bisection Method

```
    if (bisec(b(i))*bisec(c(i))<0)
        a(i+1)=c(i);
        b(i+1)=b(i);
    else
        b(i+1)=c(i);
        a(i+1)=a(i);
    end
end
N(i+1)=i+1;
c(i+1) = (a(i+1)+b(i+1))/2;
bs(i+1) = bisec(c(i));
ann = [N a b c bs];
disp(ann , 'The Table : ');
disp(c(i),'The root of the function is : ')
```

Scilab code Exa 3.2 False positioning method

```
//The solution using false position method
clc;
clear;
close();
b(1)=1;a(1)=0;k=5;i=1;
format('v',9);
deff('[fx]=bisec(x)','fx =(x+1)^2*exp(x^2-2)-1');
x = linspace(0,1);
plot(x,((x+1).^2).*(exp(x.^2-2))-1);
//in interval [0,1]changes its sign thus has a root
//k = no of decimal place of accuracy
//a = lower limit of interval
//b = upper limit of interval
c(i) = (a(i)*bisec(b(i))-b(i)*bisec(a(i)))/(bisec(b(
   i))-bisec(a(i)));
bs(1)=bisec(c(1));
N(1) = 1;
a(i+1)=c(i);
b(i+1)=b(i);
while abs(bisec(c(i)))>(0.5*(10^-k))
    i = i+1;
    N(i)=i;
    c(i) = (a(i)*bisec(b(i))-b(i)*bisec(a(i)))/(
       bisec(b(i))-bisec(a(i)));
    bs(i) = bisec(c(i));
    if (bisec(b(i))*bisec(c(i))<0)
        a(i+1)=c(i);
        b(i+1)=b(i);
    else
        b(i+1)=c(i);
```

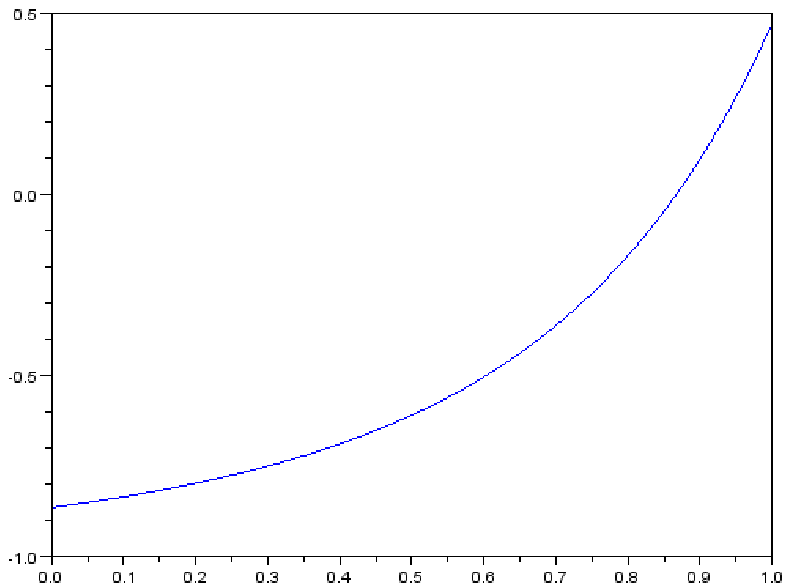

Figure 3.2: False positioning method

```
        a(i+1)=a(i);
    end
end
a(10)=[];b(10)=[];
ann = [N a b c bs];
disp(ann , 'The Table : ');
disp('The root of the function is bracketed by
   [0.647116 1] ');
```

Scilab code Exa 3.3 fixed point iteration method

```
//We have quadratic equation x^2-2*x-8=0 with roots
    -2 and 4
//for solving it we use fixed point iteration method
     for that we rearrange it in 3 ways.
//first way x=(2*x+8)^(1/2)
//here x0 is chosen arbitrarily

clear;
clc;
close();
format('v',5)
funcprot(0);
deff('[fixed_point]=fx(x)','fixed_point=(2*x+8)^0.5'
    )
x0=5;
while abs(x0-fx(x0))>0.5*10^(-2)
  x0=fx(x0);
end
disp(x0,'root is :')

//second way x=(2*x+8)/x

format('v',5)
funcprot(0);
deff('[fixed_point]=fx(x)','fixed_point=(2*x+8)/x')
x0=5;
while abs(x0-fx(x0))>0.5*10^(-2)
  x0=fx(x0);
end
disp(x0,'root is :')

//third way x=(x^2-8)/2

format('v',10)
funcprot(0);
deff('[fixed_point]=fx(x)','fixed_point=(x^2-8)/2')
```

```
x0=5;
for i=1:5
  x0=fx(x0);
  disp(x0,'value is :')
end
disp(x0,'As you can see that the root is not
   converging.So this method is not applicable.')
```

Scilab code Exa 3.4 Type of convergence

```
//checking for the convergence and divergence of
   different functions we are getting after
   rearrangement of the given quadratic equation x
   ^2-2*x-8=0.
//after first type of arrangement we get a function
   gx=(2*x+8)^(1/2).for this we have..

clear;
clc;
close();
alpha=4;
I=alpha-1:alpha+1;//required interval
deff('[f1]=gx(x)','f1=(2*x+8)^(1/2)');
deff('[f2]=diffgx(x)','f2=(2*x+8)^(-0.5)');
x=linspace(3,5);
subplot(2,1,1);
plot(x,(2*x+8)^(1/2))
plot(x,x)
x0=5;
if diffgx(I)>0
  disp('Errors in two consecutive iterates are of
     same sign so convergence is monotonic')
end
```

```
if abs(diffgx(x0))<1
  disp('So this method converges')
end

//after second type of arrangement we get a function
    gx=(2*x+8)/x.for this we have..

deff('[f1]=gx(x)','f1=(2*x+8)/x');
deff('[f2]=diffgx(x)','f2=(-8)/(x^2)');
x=linspace(1,5);
for i=1:100
  y(1,i)=2+8/x(1,i);
end
subplot(2,1,2);
plot(x,y)
plot(x,x)
x0=5;
if diffgx(I)<0
  disp('Errors in two consecutive iterates are of
     opposite sign so convergence is oscillatory')
end
if abs(diffgx(x0))<1
  disp('So this method converges')
end
```

Scilab code Exa 3.5 Newton Method

```
//Newton's Method
//the first few iteration converges quikcly in
   negative root as compared to positive root
clc;
```

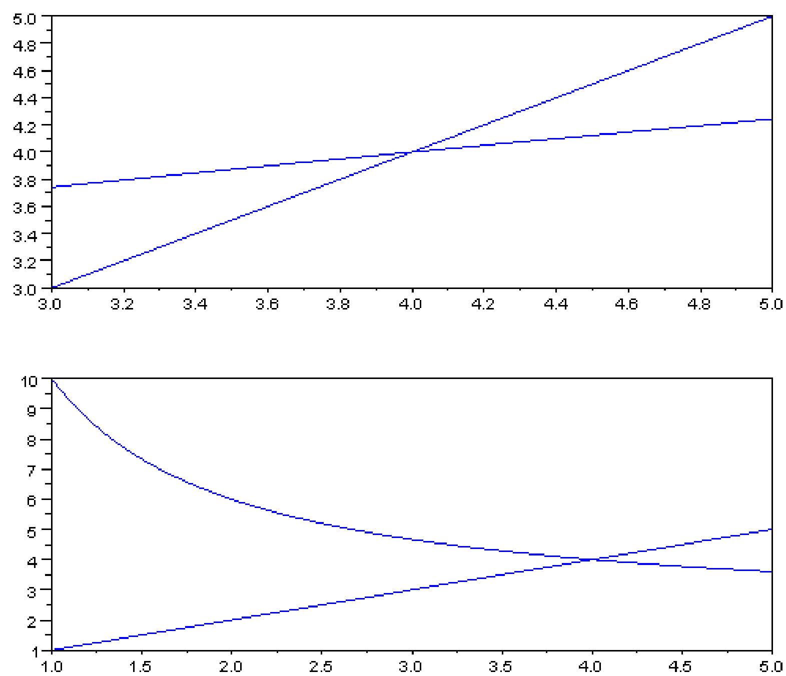

Figure 3.3: Type of convergence

```
clear;
close();
funcprot(0);
format('v',9);
deff('[Newton]=fx(x)','Newton=exp(x)-x-2');
deff('[diff]=gx(x)','diff=exp(x)-1');
x = linspace(-2.5,1.5);
plot(x,exp(x)-x-2)
//from the graph the function has 2 roots
//considering the initial negative root -10
x1 = -10;
x2 = x1-fx(x1)/gx(x1);
i=0;
while abs(x1-x2)>(0.5*10^-7)
    x1=x2;
    x2 = x1-fx(x1)/gx(x1);
    i=i+1;
end
disp(i,'Number of iterations : ')
disp(x2,'The negative root of the function is : ')


//considering the initial positive root 10
x1 = 10;
x2 = x1-fx(x1)/gx(x1);
i=0;
while abs(x1-x2)>(0.5*10^-7)
    x1=x2;
    x2 = x1-fx(x1)/gx(x1);
    i=i+1;
end
disp(i,'Number of iteration : ')
disp(x2,'The positive root of the function is : ')
//number of iterations showing fast and slow
   convergent
```

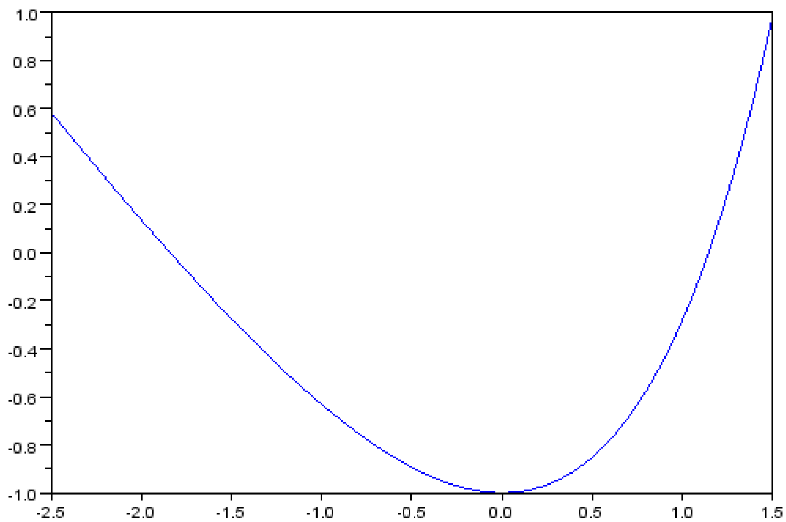

Figure 3.4: Newton Method

Scilab code Exa 3.6 Secant Method

```
//Secant Method
//the first few iteration converges quikcly in
   negative root as compared to positive root
clc;
clear;
close();
funcprot(0);
format('v',9);
deff('[Secant]=f(x)','Secant=exp(x)-x-2');
```

```
x = linspace(0,1.5);
subplot(2,1,1);
plot(x,exp(x)-x-2);
plot(x,0);
//from the graph the function has 2 roots
//considering the initial negative root −10
x0 = -10
x1 = -9;
x2 = (x0*f(x1)-x1*f(x0))/(f(x1)-f(x0));
i=0;
while abs(x1-x2)>(0.5*10^-7)
    x0=x1;
    x1=x2;
    x2 = (x0*f(x1)-x1*f(x0))/(f(x1)-f(x0));
    i=i+1;
end
disp(i,'Number of iterations : ')
disp(x2,'The negative root of the function is : ')


//considering the initial positive root 10
subplot(2,1,2);
x = linspace(-2.5,0);
plot(x,exp(x)-x-2);
plot(x,0);
x0 = 10
x1 = 9;
x2 = (x0*f(x1)-x1*f(x0))/(f(x1)-f(x0));
i=0;
while abs(x1-x2)>(0.5*10^-7)
    x0=x1;
    x1=x2;
    x2 = (x0*f(x1)-x1*f(x0))/(f(x1)-f(x0));
    i=i+1;
end
disp(i,'Number of iteration : ')
disp(x2,'The positive root of the function is : ')
//number of iterations showing fast and slow
```

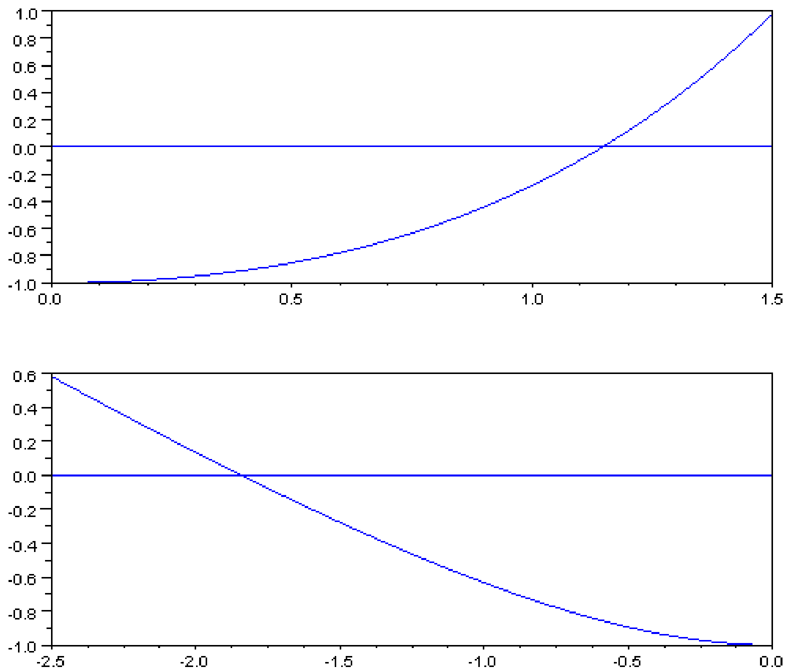

Figure 3.5: Secant Method

```
   convergent

format('v',6)
//Order of secant method (p)
 p = log(31.52439)/log(8.54952);
 disp(p,'Order of Secant Method : ')
```

Scilab code Exa 3.7 System of Non Linear Equations

```
//Non-Linear Equation
clc;
clear;
close();
funcprot(0);
format('v',9);
i = 1;
deff('[func1]=f(x,y)','func1=x^2+y^2-4');
deff('[func2]=g(x,y)','func2=2*x-y^2');
deff('[difffx]=fx(x)','difffx=2*x');
deff('[difffy]=fy(y)','difffy=2*y');
deff('[diffgx]=gx(x)','diffgx=2');
deff('[diffgy]=gy(y)','diffgy=-2*y');
x1(i)=1;y1(i)=1;
J = [fx(x1(i)),fy(y1(i));gx(x1(i)),gy(y1(i))];
n=[x1(i);y1(i)]-inv(J)*[f(x1(i),y1(i));g(x1(i),y1(i)
    )];
x2(i)=n(1,1);y2(i)=n(2,1);
N(1)=i-1;
while (abs(x2(i)-x1(i))>0.5*10^-7) | (abs(y2(i)-y1(i
    ))>0.5*10^-7)
    i=i+1;
    N(i)=i-1;
    x1(i)=x2(i-1);
    y1(i)=y2(i-1);
    J = [fx(x1(i)),fy(y1(i));gx(x1(i)),gy(y1(i))];
    n=[x1(i);y1(i)]-inv(J)*[f(x1(i),y1(i));g(x1(i),
        y1(i))];
    x2(i)=n(1,1);y2(i)=n(2,1);
end
N(i+1)=i;
x1(i+1) = x2(i);
y1(i+1) = y2(i);
n = [N x1 y1];
disp(n,'The value of n x and y :')
```

Scilab code Exa 3.8 System of Non Linear Equations

```
//Non-Linear Equation
clc;
clear;
close();
funcprot(0);
format('v',9);
deff('[func1]=f(x1,x2)','func1=-2.0625*x1+2*x2-0.0625*x1^3+0.5');
deff('[func2]=g(x1,x2,x3)','func2=x3-2*x2+x1-0.0625*x2^3+0.125*x2*(x3-x1)');
deff('[func3]=h(x2,x3,x4)','func3=x4-2*x3+x2-0.0625*x3^3+0.125*x3*(x4-x2)');
deff('[func4]=k(x3,x4)','func4=-1.9375*x4+x3-0.0625*x4^3-0.125*x3*x4+0.5');
//define the corresponding partial differenciation of the function = 16
deff('[difffx1]=fx1(x1)','difffx1=-2.0625-3*0.0625*x1^2');
deff('[difffx2]=fx2(x2)','difffx2=2');

deff('[diffgx1]=gx1(x2)','diffgx1=1-0.125*x2');
deff('[diffgx2]=gx2(x1,x2,x3)','diffgx2=-2-3*0.0625*x2^2+0.125*(x3-x1)');
deff('[diffgx3]=gx3(x2)','diffgx3=1+0.125*x2');

deff('[diffhx2]=hx2(x3)','diffhx2=1-0.125*x3');
deff('[diffhx3]=hx3(x3,x4)','diffhx3=-2-0.0625*3*x3^2+0.125*x4');
deff('[diffhx4]=hx4(x3)','diffhx4 = 1+0.125*x3');
```

```
deff('[diffkx3]=kx3(x4)','diffkx3=1-0.125*x4');
deff('[diffkx4]=kx4(x3,x4)','diffkx4=-1.9375-3*0.0625*x4^2-0.125*x3');

x = [1.5 1.25 1.0 0.75]';
for i=1:6
    N(i)=i-1;
    x1(i) = x(1);x2(i)=x(2);x3(i) = x(3);x4(i)=x(4);
    J = [fx1(x(1)),fx2(x(2)),0,0;gx1(x(2)),gx2(x(1),x(2),x(3)),gx3(x(2)),0;0,hx2(x(3)),hx3(x(3),x(4)),hx4(x(3));0,0,kx3(x(4)),kx4(x(3),x(4))];
    x = x - inv(J)*[f(x(1),x(2));g(x(1),x(2),x(3));h(x(2),x(3),x(4));k(x(3),x(4))];
end
n = [N x1 x2 x3 x4];
disp(n,'The values of N x1 x2 x3 x4 respectively : ');
```

Chapter 4

Eigenvalues and eigenvectors

Scilab code Exa 4.1 Power Method of finding largest Eigen value

```
//The Power Method of finding largest Eigen value of
    given matrix
clear;
clc;
close();
A=[3 0 1;2 2 2;4 2 5];    //Given Matrix
u0=[1 1 1]';     //Intial vector
v=A*u0;
a=max(u0);
while abs(max(v)-a)>0.05     //for accuracy
  a=max(v);
  u0=v/max(v);
  v=A*u0;
end
format('v',4);
disp(max(v),'Eigen value :')
format('v',5);
disp(u0,'Eigen vector :')
```

Scilab code Exa 4.2 Power Method of finding largest Eigen value

```
//The Power Method of finding largest Eigen value of
    given matrix
clear;
clc;
close();
A=[3 0 1;2 2 2;4 2 5];
new_A=A-7*eye(3,3);    //Given Matrix
u0=[1 1 1]';      //Intial vector
v=new_A*u0;
a=max(abs(u0));
while abs(max(abs(v))-a)>0.005     //for accuracy
  a=max(abs(v));
  u0=v/max(abs(v));
  v=new_A*u0;
end
format('v',5);
disp(max(v),'Eigen value :')
format('v',5);
disp(u0,'Eigen vector :')
```

Scilab code Exa 4.3 Convergence of Inverse Iteration

```
//Convergence of Inverse Iteration
clc;
clear;
close();
```

```
format('v',4);
A = [3 0 1;2 2 2; 4 2 5];
e1 = 7.00;
e2 = 1.02;
p = sum(diag(A))-e1-e2;
disp(A, 'A = ');
A = A - p*eye(3,3);
disp(A,'A-1.98I = ');
L = [1 0 0; 0.50 1 0; 0.26 0.52 1];
U = [4 2 3.02; 0 -.98 0.49; 0 0 -.03];
disp(L,U,'The decomposition of A - 1.98I (L,U): ');
u = [1,1,1]';
I = inv(U)*inv(L);
for i = 1:3
    v = inv(U)*inv(L)*u;
    disp(max(v),v,u,i-1,'The values of s u(s) v(s+1)
        and max(v(s+1)) : ');
    u = v./max(v);
end
disp(u,'The Eigen Vector : ');
ev = p+1/max(v);
disp(ev,'The approx eigen value  :');
```

Scilab code Exa 4.4 Deflation

```
//Deflation
clc;
clear;
close();
A = [10 -6 -4; -6 11 2; -4 2 6];
P = [1 0 0;-1 1 0;-0.5 0 1];
disp(P,A,'The A and the P(transformation matrix) are
    : ');
```

```
B = inv(P)*A*P;
disp(B,'Hence B = ')
C = B;
C(1,:) = [];
C(:,1) = [];
disp(C,'The deflated matrix : ');
Y = spec(C);
disp(Y,'The matrix A therefore has  eigen values : '
   );
e1 = [1/3,1,-1/2]';
e2 = [2/3,1,1]';
disp(e1,e2,'The eigen values of B are : ');
x1 = P*e1;
x2 = P*e2;
disp(3/2.*x1,3/2.*x2,'The eigen vextors of the
   orginal matrix A : ')
```

Scilab code Exa 4.5 Threshold serial Jacobi Method

```
//Threshold serial Jacobi Method
//taking threshold values 0.5 and 0.05
clc;
clear;
close();
format('v',9);
A = [3 0.4 5;0.4 4 0.1;5 0.1 -2];
//for first cycle |0.4|<0.5 trasnformation is
   omitted
//|5|>0.5 a zero is created at (1,3)
//by taking the rotation matrix P1=[c 0 s; 0 1 0;-s
   0 c]; where c=cos and s=sin
//O is theta
p=1;q=3;
```

```
O = 0.5*atan(2*A(p,q)/(A(q,q)-A(p,p)));
P1 = [cos(O) 0 sin(O);0 1 0;-sin(O) 0 cos(O)];
A1 = A;
A2 = inv(P1)*A*P1;
//as all the off-diagonals < 0.5 the first cycle is complete
disp(diag(A2),'The eigen values for case 1 : ')

//second cycle for 0.05
count =0;
EV = P1;
for i=1:3
    for j=i+1:3
        if A2(i,j)>0.05 then
            p=i;q=j;
            O = 0.5*atan(2*A2(p,q)/(A2(q,q)-A2(p,p)));
            c = cos(O);
            s = sin(O);
            P = eye(3,3);
            P(p,p)=c;
            P(q,q)=c;
            P(p,q)=s;
            P(q,p)=-s;
            A = inv(P)*A2*P;
            disp(EV,'value of P*')
            EV = EV * P;
            count = count+1;
        end
    end
end
//eigen values are the diagonal elements of A and the column of P gives eigen vectors
disp(diag(A),'Eigen values : ')
disp(EV,'Correspoding eigen vectors : ')
```

Scilab code Exa 4.6 The Gerchgorin circle

```
//The Gerchgorin circle
clc;
clear;
close();
format('v',9);
x = [0:.1:14];
plot2d(0,0,-1,"031"," ",[0,-5,14,5]);
plot(x,0);
A = [5 1 0;-1 3 1;-2 1 10];
disp(A,'A = ');
for i=1:3
   disp(A(i,i),'Centers are : ');
    radius = 0;
    for j=1:3
        if j~=i then
            radius = radius + abs(A(i,j));
        end
    end
    disp(radius,'Radius : ');
    xarc(A(i,i)-radius,radius,2*radius,2*radius
       ,0,360*64);
end
disp('The figure indicates that 2 of the eigenvalues
    of A lie inside the intersected region of 2
   circles , and the remaining eigen value in the
   other circle.');
```

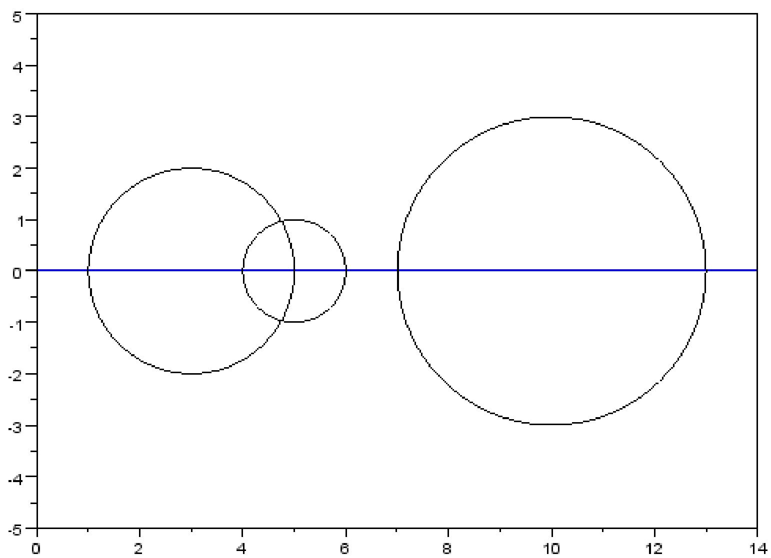

Figure 4.1: The Gerchgorin circle

Scilab code Exa 4.7 Sturm sequence property

```
//Sturm sequence property
clc;
clear;
close();
C=[2,4,0,0;4,10,3,0;0,3,9,-1;0,0,-1,5];
//find the eigen vClues lying (0,5)
p=0;

f(1)=1;
f(2)=C(1,1)-p;
count = 0;
if f(1)*f(2)>=0 then
    count = 1;
end
for r=3:5
    br=C(r-2,r-1);
    f(r)=-br^2*f(r-2)+(C(r-1,r-1)-p)*f(r-1);
    if f(r)*f(r-1)>=0 then
        count = count+1;
    end
end
disp(f,'Sturm sequences')
disp(count,'Number of eigen values strickly greater
    than 0 : ')

p=5;
f(1)=1;
f(2)=C(1,1)-p;
count1 = 0;
if f(1)*f(2)>=0 then
```

```
    count1 = 1;
end
for r=3:5
    br=C(r-2,r-1);
    f(r)=-br^2*f(r-2)+(C(r-1,r-1)-p)*f(r-1);
    if f(r)*f(r-1)>=0 then
        count1 = count1+1;
    end
end
disp(f,'Sturm sequences')
disp(count1,'Number of eigen values strickly greater
    than 5 : ')
disp(count-count1,'Number of eigen values between 0
   and 5 : ')
```

Scilab code Exa 4.8 Gerschgorins first theorem

```
//Gerschgorin's first theorem
clc;
clear;
close();
//find the eigen values lying [0,4] with an error of
    0.25
//taking p at mid point of the interval
C=[2,-1,0;-1,2,-1;0,-1,1];
p=2;

f(1)=1;
f(2)=C(1,1)-p;
count = 0;
if f(1)*f(2)>0 then
    count = 1;
end
```

```
for r=3:4
    br=C(r-2,r-1);
    f(r)=-br^2*f(r-2)+(C(r-1,r-1)-p)*f(r-1);
    if f(r)*f(r-1)>0 then
        count = count+1;
//     elseif f(r-1)==0 && f(r-1)*          ?????? check for sign when f(r)=zero
    end
end
disp(f,'Sturm sequences')
disp(count,'Number of eigen values strickly greater than 2 : ')

p=1;
f(1)=1;
f(2)=C(1,1)-p;
count1 = 0;
if f(1)*f(2)>0 then
    count1 = 1;
end
for r=3:4
    br=C(r-2,r-1);
    f(r)=-br^2*f(r-2)+(C(r-1,r-1)-p)*f(r-1);
    if f(r)*f(r-1)>0 then
        count1 = count1+1;
    end
end
disp(f,'Sturm sequences')
disp(count1,'Number of eigen values strickly greater than 1 : ')

p=1.5;
f(1)=1;
f(2)=C(1,1)-p;
count2 = 0;
if f(1)*f(2)>0 then
    count2 = 1;
end
```

```
for r=3:4
    br=C(r-2,r-1);
    f(r)=-br^2*f(r-2)+(C(r-1,r-1)-p)*f(r-1);
    if f(r)*f(r-1)>0 then
        count2 = count2+1;
    end
end
disp(f,'Sturm sequences')
disp(count2,'Number of eigen values strickly greater than 1.5 : ')
disp(p+0.25,'Eigen value lying between [1.5,2] ie with an error of 0.25 is : ')
```

Scilab code Exa 4.9 Givens Method

```
//Given's Method
//reduce A1 to tridiagonal form
clc;
clear;
close();
format('v',7);
A1 = [2 -1 1 4;-1 3 1 2;1 1 5 -3;4 2 -3 6];
disp(A1,'A = ')
// zero is created at (1,3)
//by taking the rotation matrix X1=[c 0 s; 0 1 0;-s 0 c]; where c=cos and s=sin
//O is theta

count =0;
for i=1:(4-2)
    for j=i+2:4
        if abs(A1(i,j))>0 then
            p=i+1;q=j;
```

```
            O = -atan(A1(p-1,q)/(A1(p-1,p)));
            c = cos(O);
            s = sin(O);
            X = eye(4,4);
            X(p,p)=c;
            X(q,q)=c;
            X(p,q)=s;
            X(q,p)=-s;

            A1 = X'*A1*X;
            disp(A1, 'Ai = ');
            disp(X ,'X = ');
            disp(O, 'Theta = ');
            count = count+1;
        end
    end
end
disp(A1,'Reduced A1 to trigonal matrix is : ')
```

Scilab code Exa 4.10 Householder Matrix

```
//Householder Matrix
clc;
clear;
close();
format('v',7);
e = [1;0;0];
x = [-1;1;4];
disp(e , 'e = ');
disp(x , 'x = ');
//considering the positive k according to sign
   convention
k = sqrt(x'*x);
```

```
disp(k,'k = ');
u = x - k*e;
disp(u,'u = ');
Q = eye(3,3) - 2*u*u'/(u'*u);
disp(Q,'Householder Matrix : ')
```

Scilab code Exa 4.11 Householder methods

```
//Householder methods
clc;
clear;
close();
format('v',7);
A = [2 -1 1 4;-1 3 1 2;1 1 5 -3;4 2 -3 6];
disp(A, 'A = ');
n=4;
for r=1:n-2
    x = A(r+1:n,r);
    f = eye(n-r,n-r);
    e = f(:,1)
    I = eye(r,r);
    O(1:n-r,r) = 0;
    //calculating Q
    k = sqrt(x'*x);
    u = x - k*e;
    Q = eye(n-r,n-r) - 2*u*u'/(u'*u);
    //substituting in P
    P(1:r,1:r)= I;
    P(r+1:n,1:r)=0;
    P(1:r,r+1:n)=0;
    P(r+1:n,r+1:n)=Q;
    A = P*A*P;
    disp(A,Q,P,'The P Q and A matrix are ; ')
```

```
end
C = A;
disp(C,'The tridiagonal matrix by householder method
    is : ')
```

Scilab code Exa 4.12 stable LR method

```
//stable LR method
clc;
clear;
close();
format('v',7);
A = [2 1 3 1;-1 2 2 1;1 0 1 0;-1 -1 -1 1];
disp(A, 'A = ');
for i = 1:6
   [L,R,P]= lu(A);
   A = R*P*L;
   disp(A,R,L,'The L R and A matrix are : ');
end
disp(A,'The (1,1) and (4,4) elements have converged
   to real eigenvalues')
X = [A(2,2) A(2,3);A(3,2) A(3,3)];
E = spec(X);
disp(E,'Although submatrix themselves are not
   converging their eigen values converges.')
```

Scilab code Exa 4.13 Orthogonal decomposition QR method

```
//Orthogonal decomposition − QR method
```

```
//reduce A to tridiagonal form
clc;
clear;
close();
format('v',7);
A1 = [1 4 2;-1 2 0;1 3 -1];
disp(A1, 'A = ');
// zero is created in lower triangle
//by taking the rotation matrix X1=[c s 0;-s c 0;0 0
    1]; where c=cos and s=sin
//O is theta

Q = eye(3,3);
for i=2:3
    for j=1:i-1
        p=i;q=j;
        O = -atan(A1(p,q)/(A1(q,q)));
        c = cos(O);
        s = sin(O);
        X = eye(3,3);
        X(p,p)=c;
        X(q,q)=c;
        X(p,q)=-s;
        X(q,p)=s;
        A1 = X'*A1;
        Q = Q*X;
        disp(A1,X,'The X and A matrix : ');
    end
end
R = A1;
disp(R,Q,'Hence the original matrix can be
   decomposed as : ')
```

Scilab code Exa 4.14 Reduction to upper Hessenberg form

```
//Redduction to upper Hessenberg form
clc;
clear;
close();
format('v',7);
A1 = [4 2 1 -3;2 4 1 -3;3 2 2 -3;1 2 1 0];
disp(A1, 'A = ' );
//the element with largest modulus below diagonal in
    first column need to be at the top and then
   similarly for column 2
A1=gsort(A1,'lr');
temp = A1(:,3);
A1(:,3) = A1(:,2);
A1(:,2) = temp;
M1 = eye(4,4);
M1(3,2) = A1(3,1)/A1(2,1);
M1(4,2) = A1(4,1)/A1(2,1);
A2 = inv(M1)*A1*M1;
disp(A2,M1, 'M1 and A2 : ')
A2=gsort(A2,'lr');
temp = A2(:,3);
A2(:,3) = A2(:,4);
A2(:,4) = temp;
M2 = eye(4,4);
M2(4,3) = A2(4,2)/A2(3,2);
A3 = inv(M2)*A2*M2;
disp(M2,'M2 = ');
disp(A3,'Upper Hessenberg Matrix :')


//for i=2:n
//     M =eye(4,4);
//      for j=i+1:n
//          M(j,i) = A(j,)
//      end
//end
```

Scilab code Exa 4.15 Redduction to upper Hessenberg form and calculating eigen values

```
//Redduction to upper Hessenberg form and
   calculating eigen values
clc;
clear;
close();
format('v',7);
A1 = [4 2 1 -3;2 4 1 -3;3 2 2 -3;1 2 1 0];
//the element with largest modulus below diagonal in
    first column need to be at the top and then
   similarly for column 2
A1=gsort(A1,'lr');
temp = A1(:,3);
A1(:,3) = A1(:,2);
A1(:,2) = temp;
M1 = eye(4,4);
M1(3,2) = A1(3,1)/A1(2,1);
M1(4,2) = A1(4,1)/A1(2,1);
A2 = inv(M1)*A1*M1;

A2=gsort(A2,'lr');
temp = A2(:,3);
A2(:,3) = A2(:,4);
A2(:,4) = temp;
M2 = eye(4,4);
M2(4,3) = A2(4,2)/A2(3,2);
A3 = inv(M2)*A2*M2;
H = A3;
disp(H,'Upper Hessenberg Matrix :')
l =0;
```

```
for i=4:-1:1
    K =H(1:i,1:i);
    while abs(K(i,i)-l)>0.005
        l=K(i,i);
        [Q,R]=qr(K-K(i,i)*eye(i,i));
        K = R*Q + K(i,i)*eye(i,i);
    end
    l = 0;
    EV(i) = K(i,i);
end
disp(EV,'Eigen Values : ')
```

Chapter 5

Methods of approximation theory

Scilab code Exa 5.1 Lagranges Method of interpolation

```
//Construction of the quadratic interpolating polynomial to the function f(x)=ln(x) by using Lagrange's Method of interpolation.

close();
clear;
clc;
xi = linspace(2,3,3);
format('v',10);
y = [0.69315 0.91629 1.09861];
x = poly(0,'x');

//Following are the polynomials

L0 = (x-xi(2))*(x-xi(3))/((xi(1)-xi(2))*(xi(1)-xi(3)));
L1 = (x-xi(1))*(x-xi(3))/((xi(2)-xi(1))*(xi(2)-xi(3)));
L2 = (x-xi(1))*(x-xi(2))/((xi(3)-xi(1))*(xi(3)-xi(2)
```

```
));
p2 = L0*y(1) + L1*y(2) + L2*y(3);
disp(p2 , 'The Required Polynomial : ')

//Showing the difference between actual and obtained value
format('v',8);
disp(log(2.7),'Actual Value of Polynomial at x=2.7')
disp(horner(p2,2.7),'Obtained Value of Polynomial at x=2.7')

err = log(2.7)-horner(p2,2.7);
disp(err , 'Error in approximation : ')
```

Scilab code Exa 5.2 Theoritical bound on error

```
//Theoritical bound on error
//it needs Symbolic Toolbox
//cd ~\Desktop\maxima_symbolic;
//exec 'symbolic.sce'
clc;
clear;
close();
syms x;
fx = log(x);
n = 2;
x0 = 2;
x1 = 2.5;
x2 = 3;
diff1_fx = diff(fx,x);
diff2_fx = diff(diff1_fx,x);
diff3_fx = diff(diff2_fx,x);
//so fx satisfies the continuity conditions on [2,3]
```

```
x= poly(0,'x');
eta = linspace(2,3,100);
//fx-p2x is equal to
func = (x-2)*(x-2.5)*(x-3)*2/(factorial(3)*eta^3);
min_func = (x-2)*(x-2.5)*(x-3)*2/(factorial(3)*min(eta)^3);
disp(min_func , 'func will be less than or equal to');
x = 2.7;
max_error = abs(horner(min_func,x));
disp(max_error , 'Error does not exceed :');
```

Scilab code Exa 5.3 Divided difference

```
//Divided difference for the functin = ln(x)
clc;
clear;
close();
format('v',9);
x = [1 1.5 1.75 2];
fx = [0 0.40547 0.55962 0.69315];
fab(1) = (fx(2)-fx(1))/(x(2)-x(1));
fab(2) = (fx(3)-fx(2))/(x(3)-x(2));
fab(3) = (fx(4)-fx(3))/(x(4)-x(3));
fabc(1)= (fab(2)-fab(1))/(x(3)-x(1));
fabc(2)= (fab(3)-fab(2))/(x(4)-x(2));
fabcd(1)= (fabc(2)-fabc(1))/(x(4)-x(1));
disp(fx',fab,fabc,fabcd,'Divided difference columns : ')

//We can redraw the table, the existing entries does not change
x(5)=1.1;
```

```
fx(5)=0.09531;
fab(4) = (fx(5)-fx(4))/(x(5)-x(4));
fabc(3)= (fab(4)-fab(3))/(x(5)-x(3));
fabcd(2)= (fabc(3)-fabc(2))/(x(5)-x(2));
fabcde(1)=(fabcd(2)-fabcd(1))/(x(5)-x(1));
disp(fx',fab,fabc,fabcd,fabcde,'Divided difference columns after addition of an entry : ')
```

Scilab code Exa 5.4 Polynomial Interpolation Divided Differnce form

```
//Polynomial Interpolation: Divided Differnce form
clc;
clear;
close();
format('v',8);
x = [1 1.5 1.75 2];
fx = [0 0.40547 0.55962 0.69315];
fab(1) = (fx(2)-fx(1))/(x(2)-x(1));
fab(2) = (fx(3)-fx(2))/(x(3)-x(2));
fab(3) = (fx(4)-fx(3))/(x(4)-x(3));
fabc(1)= (fab(2)-fab(1))/(x(3)-x(1));
fabc(2)= (fab(3)-fab(2))/(x(4)-x(2));
fabcd(1)= (fabc(2)-fabc(1))/(x(4)-x(1));

x(5)=1.1;
fx(5)=0.09531;
fab(4) = (fx(5)-fx(4))/(x(5)-x(4));
fabc(3)= (fab(4)-fab(3))/(x(5)-x(3));
fabcd(2)= (fabc(3)-fabc(2))/(x(5)-x(2));
fabcde(1)=(fabcd(2)-fabcd(1))/(x(5)-x(1));
disp(fabcde,fabcd,fabc,fab,fx','Divided difference columns : ')

```

```
x1 = poly(0,'x1');
p3x = fx(1)+fab(1)*(x1-x(1))+fabc(1)*(x1-x(1))*(x1-x(2))+fabcd(1)*(x1-x(1))*(x1-x(2))*(x1-x(3));
p3=horner(p3x,1.3);
disp(p3,'The interpolated value at 1.3 using p3(x) is : ')

p4x = p3x + fabcde(1)*(x1-x(1))*(x1-x(2))*(x1-x(3))*(x1-x(4));
p4=horner(p4x,1.3);
disp(p4,'The interpolated value at 1.3 using p4(x) is : ')
```

Scilab code Exa 5.5 Construction of Forward Difference Table

```
//Construction of Forward Difference Table
close();
clear;
clc;
x = poly(0,'x');
fx = (x-1)*(x+5)/((x+2)*(x+1));
xi = linspace(0.0,0.8,9);
x0 = 0;
h = 0.1;
format('v',9);
// values of function at different xi's
fi = horner(fx , xi);
// First order difference
for j = 1:8
  delta1_fi(j) = fi(j+1) - fi(j);
end
// Second order difference
for j = 1:7
```

```
  delta2_fi(j) = delta1_fi(j+1) - delta1_fi(j);
end
// Third order difference
for j = 1:6
  delta3_fi(j) = delta2_fi(j+1) - delta2_fi(j);
end
// Fourth order difference
for j = 1:5
  delta4_fi(j) = delta3_fi(j+1) - delta3_fi(j);
end

disp(fi , 'Values of f(x) : ')
disp(delta1_fi , 'First Order Difference :')
disp(delta2_fi , 'Second Order Difference :')
disp(delta3_fi , 'Third Order Difference :')
disp(delta4_fi , 'Fourth Order Difference :')
```

Scilab code Exa 5.6 Illustration of Newtons Forward Difference Formula

```
//Illustration of Newton's Forward Difference Formula
close();
clear;
clc;
x = poly(0,'x');
fx = (x-1)*(x+5)/((x+2)*(x+1));
xi = linspace(0.0,0.8,9);
x0 = 0;
h = 0.1;
format('v',9);
// values of function at different xi's
f0 = horner(fx , xi);
// First order difference
```

```
for j = 1:8
  delta1_f0(j) = f0(j+1) - f0(j);
end
// Second order difference
for j = 1:7
  delta2_f0(j) = delta1_f0(j+1) - delta1_f0(j);
end
// Third order difference
for j = 1:6
  delta3_f0(j) = delta2_f0(j+1) - delta2_f0(j);
end
// Fourth order difference
for j = 1:5
  delta4_f0(j) = delta3_f0(j+1) - delta3_f0(j);
end
//Calculating p4(0.12)
//x0+s*h=0.12
s = (0.12-x0)/h;
p4 = f0(1) + delta1_f0(1)*s + delta2_f0(1)*s*(s-1)/
    factorial(2) + delta3_f0(1)*s*(s-1)*(s-2)/
    factorial(3) + delta4_f0(1)*s*(s-1)*(s-2)*(s-3)/
    factorial(4);
disp(p4 , 'Value of p4(0.12)');
//exact value of f(0.12) is -1.897574 so error
err = p4--1.897574;
disp(err , 'Error in estimation');
```

Scilab code Exa 5.7 Illustration of Central Difference Formula

```
//Illustration of Central Difference Formula
close();
clear;
clc;
```

```
xi = 0:0.2:1.2;
fi = sin(xi);
x0 = 0;
h = 0.2;
format('v',8);
// First order difference
delta1_fi = diff(fi);
// Second order difference
delta2_fi = diff(delta1_fi);
// Third order difference
delta3_fi = diff(delta2_fi);
// Fourth order difference
delta4_fi = diff(delta3_fi);
//Fifth order difference
delta5_fi = diff(delta4_fi);
//Sixth order difference
delta6_fi = diff(delta5_fi);
disp(fi , 'Values of f(x) : ')
disp(delta1_fi , 'First Order Difference :')
disp(delta2_fi , 'Second Order Difference :')
disp(delta3_fi , 'Third Order Difference :')
disp(delta4_fi , 'Fourth Order Difference :')
disp(delta5_fi , 'Fifth Order Difference :')
disp(delta6_fi , 'Sixth Order Difference :')
//Calculating p2(0.67)
xm = 0.6;
x = 0.67;
s = (x-xm)/0.2;
p2 = fi(4) + {s*(delta1_fi(3)+delta1_fi(4))/2} + s*s*(delta2_fi(3))/2;
disp(p2 , 'Value of p2(0.67) : ');
//Calculating p4(0.67)
p4 = p2 + s*(s*s-1)*(delta3_fi(3)+delta3_fi(2))/12 + s*s*(s*s-1)*delta4_fi(2)/24;
disp(p4 , 'Value of p4(0.67) : ');
//Exact value of sin(0.67) is 0.62099 so error in estimation
err = 0.62099-0.62098;
```

```
disp(err , 'Error in estimation : ');
```

Scilab code Exa 5.8 Hermite Interpolation

```
//Hermite Interpolation
clc;
clear;
close();
format('v',9);
funcprot(0);
deff('[LL0]=L0(x)','LL0= 2*x^2-11*x+15');
deff('[LL1]=L1(x)','LL1= -4*x^2+20*x-24');
deff('[LL2]=L2(x)','LL2= 2*x^2-9*x+10');
deff('[LL0d]=L0d(x)','LL0d= 4*x-11');
deff('[LL1d]=L1d(x)','LL1d= -8*x+20');
deff('[LL2d]=L2d(x)','LL2d= 4*x-9');

disp('In this case n = 2. The legranges polynomial and their derivative . ');
disp('L0(x)=2*x^2-11*x+15   L1(x)= -4*x^2+20x-24   L2(x)=2x^2-9x+10');
disp('L0d(x)=4*x-11   L1d(x)= -8*x+20   L2d(x)=4*x-9');

disp('ri(x) = [1-2(x-xi)Lid(xi)][Li(x)]^2     si(x) = (x-xi)[Li(x)]^2');

deff('[rr0]=r0(x)','rr0=(1-2*(x-2)*L0d(2))*(L0(x))^2');
deff('[rr1]=r1(x)','rr1=(1-2*(x-2.5)*L1d(2.5))*(L1(x))^2');
deff('[rr2]=r2(x)','rr2=(1-2*(x-3)*L2d(3))*(L2(x))^2');
```

```
deff('[ss0]=s0(x)','ss0=(x-2)*L0(x)^2');
deff('[ss1]=s1(x)','ss1=(x-2.5)*L1(x)^2');
deff('[ss2]=s2(x)','ss2=(x-3)*L2(x)^2');

y = [log(2) log(2.5) log(3)];
yd = [0.500000 0.400000 0.333333];

deff('[H5]=H(x)','H5=r0(x)*y(1)+r1(x)*y(2)+r2(x)*y(3)+s0(x)*yd(1)+s1(x)*yd(2)+s2(x)*yd(3)');
y2 = H(2.7);
disp(y2,'The calculated value of y(2.7):');
act = log(2.7);
disp(act,'The exact value is of y(2.7): ');
err = act - y2;
disp(err,'The error is :');
```

Scilab code Exa 5.9 Hermite cubic Interpolation

```
//Hermite cubic Interpolation
clc;
clear;
close();
format('v',9);
funcprot(0);

x0 = -2;x1 = 0;x2 = 1;
y0 = 3;y1 = 1;y2 = -2;
y0d = -1;y1d = 0;y1d = 1;
h0 = 2;
h1 = 1;

deff('[H3_0]=H30(x)','H3_0=y0*((x-x1)^2/h0^2+2*(x-x0
```

```
)*(x-x1)^2/h0^3)+y1*((x-x0)^2/h0^2-2*(x-x1)*(x-x0)^2/h0^3)+y0d*(x-x0)*(x-x1)^2/h0^2+y1d*((x-x1)*(x-x0)^2)/h0^2');
deff('[H3_1]=H31(x)','H3_1=y1*((x-x2)^2/h1^2+2*(x-x1)*(x-x2)^2/h1^3)+y2*((x-x1)^2/h1^2-2*(x-x2)*(x-x1)^2/h1^3)+y1d*(x-x1)*(x-x2)^2/h1^2+y2d*((x-x2)*(x-x1)^2)/h1^2');

disp ('H(x) = x^3/4+x^2+1        on   -2<=x<=0');
disp ('        7*x^3-10*x^2+1    on    0<=x<=1');
```

Scilab code Exa 5.10 Illustration cubic spline interpolation with equal difference

```
//Illustration cubic spline interpolation with equal difference
//It needs Symbolic Toolbox
clc;
clear;
close();
x = -1:1;
fx = x^4;
y = fx;
function y = myfunction(x)
 y = x^4;
endfunction
diff_y = derivative(myfunction, x');
diff_y0 = diff_y(1);
diff_y2 = diff_y(9);
//cd ~/Desktop/maxima_symbolic
//exec symbolic.sce
syms a0 b0 c0 d0;
x = poly(0,'x');
```

```
s0x = a0+b0*x+c0*x^2+d0*x^3;
syms a1 b1 c1 d1;
s1x = a1+b1*x+c1*x^2+d1*x^3;
diff1_s0x = diff(s0x,x);
diff2_s0x = diff(diff1_s0x,x);
diff1_s1x = diff(s1x,x);
diff2_s1x = diff(diff1_s1x,x);
//from condition(ii)
x = -1;
eval(s0x,x);
//it gives equation a0-b0+c0-d0=1
x=1;
eval(s1x,x);
//it gives equation a1+b1+c1+d1=1
x = 0;
eval(s0x,x);
//it gives equation a0=0
eval(s1x,x);
//it gives equation a1=0
//from condition(iii)
x=0;
eval(diff1_s0x,x);
eval(diff1_s1x,x);
//it gives b0=b1;
//from condition(iv)
eval(diff2_s0x);
eval(diff2_s1x);
//it gives 2*c0=2*c1
//Applying boundary conditions
x=-1;
eval(diff1_s0x);
//it gives b0-2*c0+3*d0=-4
x=1;
eval(diff1_s1x);
//it gives b1+2*c1+3*d1=4
//Matrix form for the equations
A=[1 -1 1 -1 0 0 0 0;
1 0 0 0 0 0 0 0;
```

```
0 0 0 0 1 0 0 0;
0 0 0 0 1 1 1 1;
0 1 0 0 0 -1 0 0;
0 0 1 0 0 0 -1 0;
0 1 -2 3 0 0 0 0;
0 0 0 0 0 1 2 3];
C=[1 0 0 1 0 0 -4 4];
B = inv(A)*C';
//it implies
a0=0;b0=0;c0=-1;d0=-2;a1=0;b1=0;c1=-1;d1=2;
//for -1<=x<=0
x=poly(0,'x');
sx = eval(s0x);
disp(sx , 'for -1<=x<=0 sx =' );
//for 0<=x<=1
sx = eval(s1x);
disp(sx , 'for 0<=x<=1 sx =' );
```

Scilab code Exa 5.11 Illustration cubic spline interpolation with unequal difference

```
//Illustration cubic spline interpolation with
   unequal difference
clc;
clear;
close();
//with free boundary conditions
xi = [0 1 3 3.5 5];
yi = [1.00000 0.54030 -0.98999 -0.93646 0.28366];
n = 4;
h0 = xi(2)-xi(1);
h1 = xi(3)-xi(2);
h2 = xi(4)-xi(3);
```

```
h3 = xi(5)-xi(4);
//After imposing free boundary conditions the matrix we get
A = [2 1 0 0 0;
1 3 1/2 0 0;
0 1/2 5 2 0;
0 0 2 16/3 2/3;
0 0 0 2/3 4/3];
C = [-1.37910 ; -2.52682 ; -0.50536 ; 2.26919 ; 1.62683] ;
format('v',8);
B = inv(A)*C;
//it gives
diff1_y0 = -0.33966;
diff1_y1 = -0.69978;
diff1_y2 = -0.17566;
diff1_y3 = 0.36142;
diff1_y4 = 1.03941;
//cubic polynomial for 3<=x<=3.5
x = poly(0,'x')
s2x = yi(3)*[{(x-3.5)*(x-3.5)/(0.5*0.5)}+{2*(x-3)*(x-3.5)*(x-3.5)/(0.5*0.5*0.5)}] + yi(4)*[{(x-3)*(x-3)/(0.5*0.5)}-{2*(x-3.5)*(x-3)*(x-3)/(0.5*0.5*0.5)}] +  diff1_y2*{(x-3)*(x-3.5)*(x-3.5)/(0.5*0.5)} + diff1_y3*{(x-3.5)*(x-3)*(x-3)/(0.5*0.5)};
x = 3.14159;
disp(horner(s2x,x) , 'value of s2x at 3.14159 : ');
//with clamped boundary conditions
diff1_y0 = -sin(0);
diff1_y4 = -sin(5);
//matrix form
A = [3 0.5 0;0.5 5 2 ; 0 2 16/3];
C = [-2.52682 ; -0.50536 ; 1.62991];
B = inv(A)*C;
//it gives
diff1_y1 = -0.81446;
diff1_y2 = -0.16691;
```

```
diff1_y3 = 0.36820;
s2x = yi(3)*[{(x-3.5)*(x-3.5)/(0.5*0.5)}+{2*(x-3)*(x
    -3.5)*(x-3.5)/(0.5*0.5*0.5)}] + yi(4)*[{(x-3)*(x
    -3)/(0.5*0.5)}-{2*(x-3.5)*(x-3)*(x-3)
    /(0.5*0.5*0.5)}] +  diff1_y2*{(x-3)*(x-3.5)*(x
    -3.5)/(0.5*0.5)} + diff1_y3*{(x-3.5)*(x-3)*(x-3)
    /(0.5*0.5)};
x = 3.14159;
disp(horner(s2x,x) , 'value of s2x at 3.14159 : ');
```

Scilab code Exa 5.12 Alternating way of constructing cubic splines

```
//Alternating way of constructing cubic splines
clc;
clear;
close();
//from example 5.11
xi = [0 1 3 3.5 5];
yi = [1.00000 0.54030 -0.98999 -0.93646 0.28366];
//free boundary conditions
//matrix form
format('v',8);
A = [6 2 0; 2 5 1/2; 0 1/2 4];
B = 6*[-0.30545 ; 0.87221 ; 0.70635];
C = inv(A)*B;
c1 = C(1);
c2 = C(2);
c3 = C(3);
x = poly(0,'x');
s2x = c2*(3.5-x)*(3.5-x)*(3.5-x)/(6*0.5) + c3*(x-3)
    *(x-3)*(x-3)/(6*0.5) + {yi(3)/0.5+0.5*c2/6}*(3.5-
    x) + {yi(4)/0.5 + 0.5*c3/6}*(x-3);
x = 3.14159;
```

```
val = horner(s2x,x)*(-1.00271)/(-0.90705);
disp(val , 'value of s2x at 3.14159 : ');
//clamped boundary conditions
A = [2 1 0 0 0;
1 6 2 0 0;
0 2 5 1/2 0;
0 0 1/2 4 3/2;
0 0 0 3/2 3];
B = 6*[-0.45970; -0.30545 ; 0.87221 ; 0.70635; 0.14551];
C = inv(A)*B;
c0 = C(1);
c1 = C(2);
c2 = C(3);
c3 = C(4);
c4 = C(5);
s2x = c2*(3.5-x)*(3.5-x)*(3.5-x)/(6*0.5) + c3*(x-3)*(x-3)*(x-3)/(6*0.5) + {yi(3)/0.5+0.5*c2/6}*(3.5-x) + {yi(4)/0.5 + 0.5*c3/6}*(x-3);
x = 3.14159;
val = horner(s2x,x)*(-1.00227)/(-0.91030);
disp(val , 'value of s2x at 3.14159 : ');
```

Scilab code Exa 5.13 Linear Least square aproximation method

```
//Linear Least square aproximation method
clc;
clear;
close();
xi = [-5 -3 1 3 4 6 8];
yi = [18 7 0 7 16 50 67];
wi = [1 1 1 1 20 1 1];
format('v',7);
```

```
//Representation of equation in matrix form
W = [sum(wi) sum(wi.*xi); sum(wi.*xi) sum(wi.*xi.*xi)];
Y = [sum(wi.*yi); sum(wi.*yi.*xi)];
A = inv(W)*Y;
a0 = A(1);
a1 = A(2);
x = poly(0,'x');
p1x = a1*x + a0;
disp(p1x, 'The approximating polynomial is :');
x = linspace(-5,8,1000);
p1x = a1*x + a0;
subplot(2,1,1);
plot(x,p1x);
plot(xi,yi,'o');

wi = [1 1 1 1 1 1 1];
//Representation of equation in matrix form
W = [sum(wi) sum(wi.*xi); sum(wi.*xi) sum(wi.*xi.*xi)];
Y = [sum(wi.*yi); sum(wi.*yi.*xi)];
A = inv(W)*Y;
a0 = A(1);
a1 = A(2);
x = poly(0,'x');
p1x = a1*x + a0;
disp(p1x, 'The approximating polynomial is :')
x = linspace(-5,8,1000);
p1x = a1*x + a0;
subplot(2,1,2);
plot(x,p1x);
plot(xi,yi,'o');
```

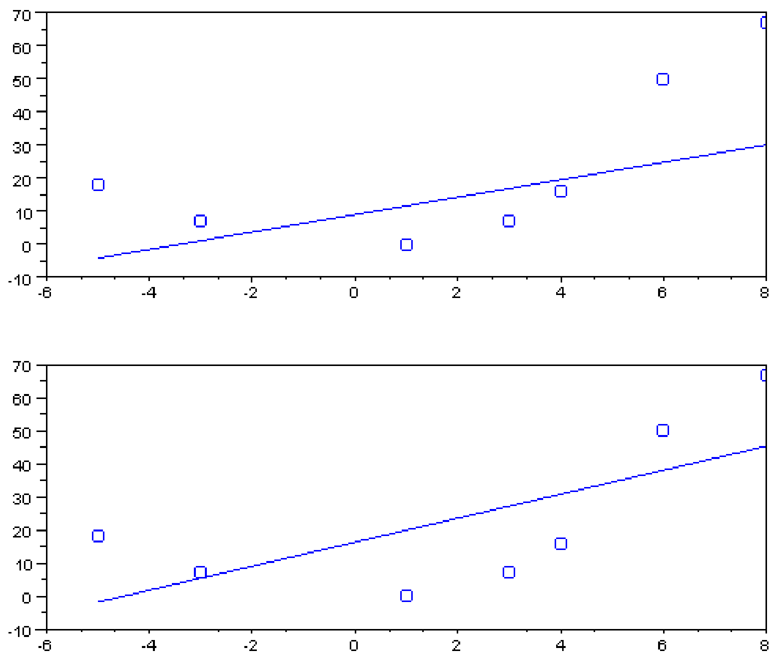

Figure 5.1: Linear Least square aproximation method

Scilab code Exa 5.14 Quadratic Least square aproximation method

```
//Quadratic Least square aproximation method
clc;
clear;
close();
xi = [-5 -3 1 3 4 6 8];
yi = [18 7 0 7 16 50 67];
wi = [1 1 1 1 20 1 1];
format('v',7);
//Representation of equation in matrix form
W = [sum(wi) sum(wi.*xi) sum(wi.*xi.*xi); sum(wi.*xi
    ) sum(wi.*xi.*xi) sum(wi.*xi.*xi.*xi); sum(wi.*xi
    .*xi) sum(wi.*xi.*xi.*xi) sum(wi.*xi.*xi.*xi.*xi)
    ];
Y = [sum(wi.*yi); sum(wi.*yi.*xi); sum(wi.*xi.*xi.*
    yi)];
A = inv(W)*Y;
a0 = A(1);
a1 = A(2);
a2 = A(3);
x = poly(0,'x');
p1x = a2*x^2 + a1*x + a0;
disp(p1x, 'The approximating polynomial is :');
x = linspace(-5,8,1000);
p1x = a2*x^2 + a1*x + a0;
plot(x,p1x);
plot(xi,yi,'o');
```

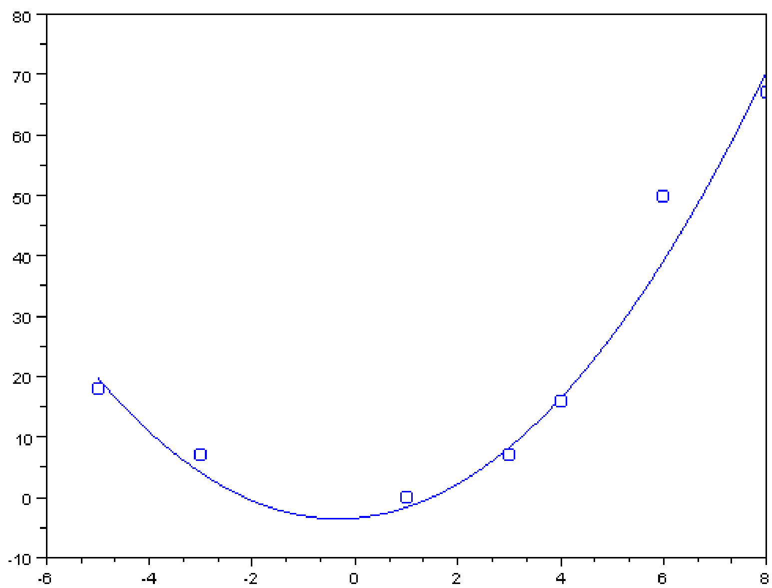

Figure 5.2: Quadratic Least square aproximation method

Scilab code Exa 5.15 Least square aproximation method with exponential functions

```
//Least square aproximation method with exponential functions
clc;
clear;
close();
xi = [0 0.25 0.4 0.5];
yi = [9.532 7.983 4.826 5.503];
wi = ones(1,4);
//data corresponding to linearised problem
Xi = [0 0.25 0.4 0.5];
Yi = [2.255 2.077 1.574 1.705];
wi = ones(1,4);
format('v',6);
//Representation of equation in matrix form
W = [sum(wi) sum(wi.*xi); sum(wi.*xi) sum(wi.*xi.*xi)];
Y = [sum(wi.*Yi); sum(wi.*Yi.*Xi)];
C = inv(W)*Y;
A = C(1);
B = C(2);
a = exp(2.281);
b = B;
disp(a, 'a = ');
disp(b, 'b = ');
//So the non linear system becomes
disp('9.532-a+7.983*exp(0.25*b)-a*exp(0.5*b)+4.826*exp(0.4*b)-a*exp(0.8*b)+5.503*exp(0.5*b)-a*exp(b) = 0');
disp('1.996*a*exp(0.25*b) -0.25*a*a*exp(0.5*b)+1.930*
```

```
    a*exp(0.4*b)-0.4*a*a*exp(0.8*b)+2.752*a*exp(0.5*b
    )-0.5*a*a*exp(b) = 0');
//Applying Newtons Method we get
a = 9.731;
b = -1.265;
disp(a , 'a = ');
disp(b , ' b = ');
```

Scilab code Exa 5.16 Least square approximation to continuous functions

```
//Least square approximation to continuous functions
clc;
clear;
close();
format('v',8);
funcprot(0);
deff('[g]=f(x,y)','g= -y^2/(1+x)');
disp('approximation of e^x on [0,1] with a uniform
    weight w(x)=1')
a11 = integrate('1','x',0,1);
a12 = integrate('x','x',0,1);
a13 = integrate('x*x','x',0,1);
a14 = integrate('x^3','x',0,1);
a21 = integrate('x','x',0,1);
a22 = integrate('x^2','x',0,1);
a23 = integrate('x^3','x',0,1);
a24 = integrate('x^4','x',0,1);
a31 = integrate('x^2','x',0,1);
a32 = integrate('x^3','x',0,1);
a33 = integrate('x^4','x',0,1);
a34 = integrate('x^5','x',0,1);
a41 = integrate('x^3','x',0,1);
a42 = integrate('x^4','x',0,1);
```

```
a43 = integrate('x^5','x',0,1);
a44 = integrate('x^6','x',0,1);

c1 = integrate('exp(x)','x',0,1);
c2 = integrate('x*exp(x)','x',0,1);
c3 = integrate('x^2*exp(x)','x',0,1);
c4 = integrate('x^3*exp(x)','x',0,1);

A = [a11 a12 a13 a14;a21 a22 a23 a24;a31 a32 a33 a34;a41 a42 a43 a44];
C = [c1;c2;c3;c4];
ann = inv(A)*C;
disp(ann, 'The coefficients a0,a1,a2,a3 are respectively :  ');

deff('[px]=p3(x)','px=ann(4)*x^3+ann(3)*x^2+ann(2)*x+ann(1)');
x = [0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0]';
e = exp(x);
p = p3(x);
err = e-p;
ann = [x e p err];

disp(ann,'Displaying the value of x exp(x) p3(x) exp(x)-p3(x) :');
plot(x,err);
plot(x,0);
```

Scilab code Exa 5.17 Gram Schmidt process for finding orthogonal functions

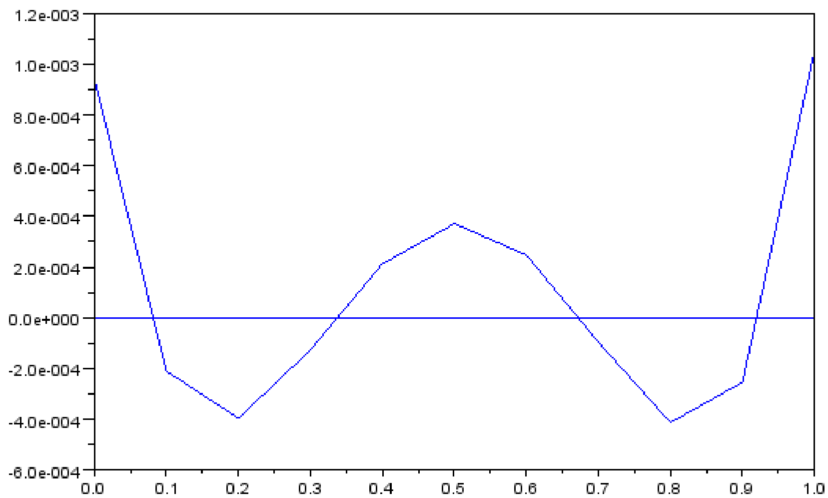

Figure 5.3: Least square approximation to continuous functions

```
//Gram - Schmidt process for finding orthogonal functions
clc;
clear;
close();
format('v',8);
funcprot(0);

disp('The orthogonal functions : ')
x = poly(0,'x');
ph0 = 1;

disp(ph0 , 'phi0(x) = ');
K1_0 = -integrate('x','x',0,1)/integrate('ph0^2','x',0,1);
ph1 = x + K1_0*ph0;
disp(ph1 , 'phi1(x) = ');

K2_0 = -integrate('x^2*ph0','x',0,1)/integrate('ph0
```

```
^2','x',0,1);
disp(K2_0 ,'K(2,0) = ');
K2_1 = -integrate('x^2*(x-.5)','x',0,1)/integrate('(x-.5)^2','x',0,1);
disp(K2_1 ,'K(2,1) = ');
ph2 = x^2 + K2_0*ph0 + K2_1*ph1;
disp(ph2 , 'phi2(x) = ');

K3_0 = -integrate('x^3*ph0','x',0,1)/integrate('ph0^2','x',0,1);
disp(K3_0 ,'K(3,0) = ');
K3_1 = -integrate('x^3*(x-.5)','x',0,1)/integrate('(x-.5)^2','x',0,1);
disp(K3_1 ,'K(3,1) = ');
K3_2 = -integrate('x^3*(x^2-x+1/6)','x',0,1)/integrate('(x^2-x+1/6)^2','x',0,1);
disp(K3_2 ,'K(3,2) = ');
ph3 = x^3 + K3_0*ph0 + K3_1*ph1 + K3_2*ph2;
disp(ph3 , 'phi3(x) = ');
```

Scilab code Exa 5.18 Gram Schmidt process for cubic polynomial least squares approx

```
//Gram - Schmidt process for cubic polynomial least squares approx
clc;
clear;
close();
format('v',8);
funcprot(0);

disp('The orthogonal functions : ')
x = poly(0,'x');
```

```
ph0 = 1;

disp(ph0 , 'phi0(x) = ');
K1_0 = -integrate('x','x',0,1)/integrate('ph0^2','x',0,1);
ph1 = x + K1_0*ph0;
disp(ph1 , 'phi1(x) = ');

K2_0 = -integrate('x^2*ph0','x',0,1)/integrate('ph0^2','x',0,1);
disp(K2_0 ,'K(2,0) = ');
K2_1 = -integrate('x^2*(x-.5)','x',0,1)/integrate('(x-.5)^2','x',0,1);
disp(K2_1 ,'K(2,1) = ');
ph2 = x^2 + K2_0*ph0 + K2_1*ph1;
disp(ph2 , 'phi2(x) = ');

K3_0 = -integrate('x^3*ph0','x',0,1)/integrate('ph0^2','x',0,1);
disp(K3_0 ,'K(3,0) = ');
K3_1 = -integrate('x^3*(x-.5)','x',0,1)/integrate('(x-.5)^2','x',0,1);
disp(K3_1 ,'K(3,1) = ');
K3_2 = -integrate('x^3*(x^2-x+1/6)','x',0,1)/integrate('(x^2-x+1/6)^2','x',0,1);
disp(K3_2 ,'K(3,2) = ');
ph3 = x^3 + K3_0*ph0 + K3_1*ph1 + K3_2*ph2;
disp(ph3 , 'phi3(x) = ');

deff('[y]=f(x)','y= exp(x)');
deff('[phi0]=ph_0(x)','phi0= horner(ph0,x)');
deff('[phi1]=ph_1(x)','phi1= horner(ph1,x)');
deff('[phi2]=ph_2(x)','phi2= horner(ph2,x)');
deff('[phi3]=ph_3(x)','phi3= horner(ph3,x)');
a0 = integrate('f(x)*ph_0(x)','x',0,1)/integrate('ph_0(x)^2','x',0,1);
disp(a0,'a0 = ');
a1 = integrate('f(x)*ph_1(x)','x',0,1)/integrate('
```

```
    ph_1(x)^2','x',0,1);
disp(a1,'a1 = ');
a2 = integrate('f(x)*ph_2(x)','x',0,1)/integrate('
    ph_2(x)^2','x',0,1);
disp(a2,'a2 = ');
a3 = integrate('f(x)*ph_3(x)','x',0,1)/integrate('
    ph_3(x)^2','x',0,1);
disp(a3,'a3 = ');

p3 = a0*ph0 + a1*ph1 + a2*ph2 +a3*ph3;
disp(p3 , 'p3(x)');
```

Chapter 6

Numerical Differntiation and Integration

Scilab code Exa 6.1 Numerical Differentiation

```
//Numerical Differentiation
clc;
clear;
close();
format('v',9);
deff('[y]=f(x)','y=exp(-x)');

x0 = ones(:,8);
h = [1 .2 .1 .02 .01 .002 .001 .0002];
x1 = 1+h;
f0 = f(x0);
f1 = f(x1);
dif = (f1-f0)./h;
max_trun_err = exp(-1).*h/2;
act_err = abs(- exp(-1)-dif);
answer = [h' f0' f1' dif' max_trun_err' act_err'];
disp(answer,'     h          f0               f1          f1
   -f0/h      he^-1        |Actual Error|');
x = (0:.0002:3);
```

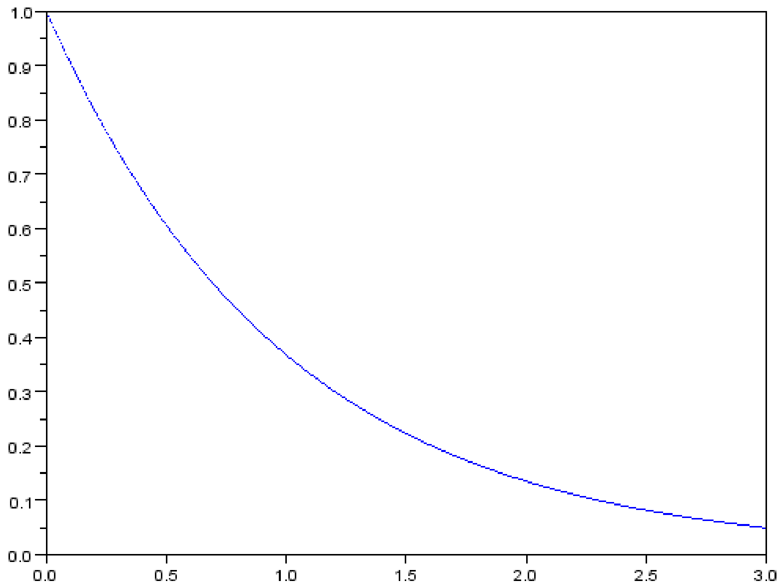

Figure 6.1: Numerical Differentiation

```
plot(x,f(x));
```

Scilab code Exa 6.2 Numerical Differentiation

```
//Numerical Differentiation
clc;
clear;
close();
format('v',9);
```

```
deff('[y]=f(x)','y=exp(-x)');
h = [1 .2 .1 .02 .01 .002 .001 .0002];
x0 = 1 - h;
x1 = ones(:,8);
x2 = 1+h;
f0 = f(x0);
f1 = f(x1);
f2 = f(x2);
dif = (f2-f0)./(2*h);
max_trun_err = exp(h-1).*h^2/6;
act_err = abs(- exp(-1)-dif);
answer = [h' f0' f2' dif' max_trun_err' act_err'];
disp(answer,'    h         f0              f2         f2-f0/2h   h^2*exp(h-1)/6  |Actual Error|');
disp('truncation error does not exceed h^2*exp(h-1)/6')
x = (0:.0002:3);
plot(x,f(x));
```

Scilab code Exa 6.3 Numerical Integration

```
//Numerical Integration
clc;
clear;
close();
format('v',9);
funcprot(0);
deff('[y]=f(x)','y=x*cos(x)');

rec = %pi * f(0)/4;
disp(rec,'Retangular Rule : ');
```

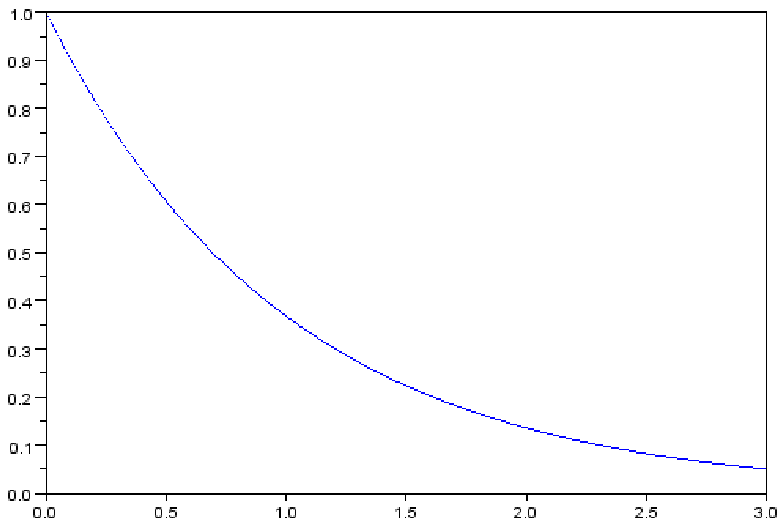

Figure 6.2: Numerical Differentiation

```

trap = %pi*(f(0)+f(%pi/4))/8;
disp(trap,'Trapezoidal Rule : ');

sip = %pi*(f(0)+4*f(%pi/8)+f(%pi/4))/(3*8);
disp(sip,'Simpson''s Rule : ');

sip38 = %pi*3*(f(0)+3*f(%pi/12)+3*f(%pi/6)+f(%pi/4))/(12*8);
disp(sip38,'Simpson''s 3/8 Rule : ');

exact = integrate('x*cos(x)','x',0,%pi/4);
disp(exact,'The exact value of intergation is :');
err = exact - rec;
err(2) = exact - trap;
err(3) = exact - sip;
err(4) = exact - sip38;
disp(err,'thus corresponding errors are : ');
```

Scilab code Exa 6.4 Numerical Integration

```
//Newton Cotes formula
clc;
clear;
close();
format('v',9);
funcprot(0);
disp('Integral 0 to PI/4 x*cos dx');
disp('based on open Newton-Cotes formulas ');

deff('[y]=f(x)','y=x*cos(x)');

k = [0 1 2 3]

a = 0;
b = %pi/4;
h = (ones(:,4)*(b-a))./(k+2);
x0 = a+h;
xk = b-h;

k(1) = 2*h(1)*f(h(1));
disp(k(1),'k=0');

k(2) = 3*h(2)*(f(h(2))+f(2*h(2)))/2;
disp(k(2),'k=1');

k(3) = 4*h(3)*(2*f(h(3))-f(2*h(3))+2*f(3*h(3)))/3;
disp(k(3),'k=2');

k(4) = 5*h(4)*(11*f(h(4))+f(2*h(4))+f(3*h(4))+11*f
    (4*h(4)))/24;
```

```
disp(k(4),'k=3');

exact = integrate('x*cos(x)','x',0,%pi/4);
disp(exact,'The exact value of intergation is :');
exact = ones(:,4)*exact;
err = exact-k;
disp(err','thus corresponding errors are : ');
```

Scilab code Exa 6.5 Trapezoidal Rule

```
//Trapezoidal Rule
clc;
clear;
close();
format('v',10);
funcprot(0);
disp('Integral 0 to 2 e^x dx');
disp('based on trapezoidal rule ');

deff('[y]=f(x)','y=exp(x)');

n = [1 2 4 8];

a = 0;
b = 2;
h = (ones(:,4)*(b-a))./n;

t(1) = h(1)*(f(a)+f(b))/2;
disp(t(1),'n=1');

t(2) = h(2)*(f(a)+f(b)+2*f(h(2)))/2;
disp(t(2),'n=2');

```

```
t(3) = h(3)*(f(a)+f(b)+2*(f(h(3))+f(2*h(3))+f(3*h(3)))))/2;
disp(t(3),'n=4');

t(4) = h(4)*(f(a)+f(b)+2*(f(h(4))+f(2*h(4))+f(3*h(4))+f(4*h(4))+f(5*h(4))+f(6*h(4))+f(7*h(4))))/2;
disp(t(4),'n=8');

exact = integrate('exp(x)','x',0,2);
disp(exact,'The exact value of intergation is :');
exact = ones(4)*exact;
err = exact-t;
disp(err,'thus corresponding errors are : ');
```

Scilab code Exa 6.6 Simpson Rule

```
//Simpson Rule
clc;
clear;
close();
format('v',10);
funcprot(0);

deff('[y]=f(x)','y=exp(x)');

n = [1 2 4];

a = 0;
b = 2;
h = (ones(:,3)*(b-a))./(2*n);

s(1) = h(1)*(f(a)+f(b)+4*f(h(1)))/3;
disp(s(1),'n=1');
```

```
s(2) = h(2)*(f(a)+f(b)+2*f(2*h(2))+4*(f(h(2))+f(3*h
    (2))))/3;
disp(s(2),'n=2');

s(3) = h(3)*(f(a)+f(b)+2*(f(2*h(3))+f(4*h(3))+f(6*h
    (3)))+4*(f(h(3))+f(3*h(3))+f(5*h(3))+f(7*h(3))))
    /3;
disp(s(3),'n=4');

exact = integrate('exp(x)','x',0,2);
disp(exact,'The exact value of intergation is :');
exact = ones(3)*exact;
err = exact-s;
disp(err,'thus corresponding errors are : ');
```

Scilab code Exa 6.7 Rombergs Interpolation

```
//Romberg's Interpolation
clc;
clear;
close();
exec('C:\Users\Pragya\Desktop\scilab\trap.sci', -1);
format('v',10);
funcprot(0);
deff('[y]=f(x)','y=exp(x)');
a = 0;
b = 2;

t(1,1)=trap(f,a,b,0,0);
disp(t(1,1),'T(0,0) : ');

t(2,1)=trap(f,a,b,1,0);
```

```
disp(t(2,1),'T(1,0) : ');

t(3,1)=trap(f,a,b,2,0);
disp(t(3,1),'T(2,0) : ');

t(4,1)=trap(f,a,b,3,0);
disp(t(4,1),'T(3,0) : ');

t(2,2)=trap(f,a,b,1,1);
disp(t(2,2),'T(1,1) : ');

t(3,2)=trap(f,a,b,2,1);
disp(t(3,2),'T(2,1) : ');

t(4,2)=trap(f,a,b,3,1);
disp(t(4,2),'T(3,1) : ');

t(3,3)=trap(f,a,b,2,2);
disp(t(3,3),'T(2,2) : ');

t(4,3)=trap(f,a,b,3,2);
disp(t(4,3),'T(3,2) : ');

t(4,4)=trap(f,a,b,3,3);
disp(t(4,4),'T(3,3) : ');

disp(t,'The corresponding Romberg Table is : ');
```

Scilab code Exa 6.8 Rombergs Method

```
//Romberg's Method
clc;
clear;
```

```
close();
exec('C:\Users\Pragya\Desktop\scilab\trap.sci', -1);
format('v',10);
funcprot(0);
deff('[y]=f(x)','y=exp(x)');
a = 0;
b = 2;

t(1,1)=trap(f,a,b,0,0);
disp(t(1,1),'T(0,0) : ');

t(2,1)=(t(1,1)+2*1*f(1))/2;
disp(t(2,1),'T(1,0) : ');

t(3,1)=(t(2,1)+f(1/2)+f(3/2))/2;
disp(t(3,1),'T(2,0) : ');

t(4,1)=(t(3,1)+.5*(f(1/4)+f(3/4)+f(5/4)+f(7/4)))/2;
disp(t(4,1),'T(3,0) : ');
```

Scilab code Exa 6.9 Simpsons Adaptive Quatrature

```
//Simpson's Adaptive Quatrature
clc;
clear;
close();
format('v',10);
funcprot(0);
deff('[y]=f(x)','y=exp(x)');
a = 0.5;
b = 1;
h = (b-a)/2;
S1 = h*(f(a)+4*f((a+b)/2)+f(b))/3;
```

```
disp(S1,'S1 : ');

S2 = h*(f(a)+4*f((3*a+b)/4)+2*f((a+b)/2)+4*f((a+3*b)/4)+f(b))/6;
disp(S2,'S2 : ');

err = abs(S2-S1)/15;
disp(err,'An estimate of the error in S2 is : ' );

act = integrate('exp(x)','x',.5,1)
act_err = abs(act-S2);
disp(act_err,'The Actual error in S2 is : ');
```

Scilab code Exa 6.10 Simpsons Adaptive Quatrature

```
//Simpson's Adaptive Quatrature
clc;
clear;
close();
format('v',7);
funcprot(0);
deff('[y]=f(x)','y=exp(-3*x)*sin(3*x)');
e = 0.0005;
a = 0;
b = %pi;
h = (b-a)/2;

S1 = h*(f(a)+4*f((a+b)/2)+f(b))/3;
disp(S1,'S1 : ');
S2 = h*(f(a)+4*f((3*a+b)/4)+2*f((a+b)/2)+4*f((a+3*b)/4)+f(b))/6;
disp(S2,'S2 : ');

```

```
err = abs(S2-S1)/15;
disp(err,'|S2-S1|>15e so [0.%pi] must be subdivided ');

a = (a+b)/2;
h = (b-a)/2;
S1 = h*(f(a)+4*f((a+b)/2)+f(b))/3;
disp(S1,'S1 : ');
S2 = h*(f(a)+4*f((3*a+b)/4)+2*f((a+b)/2)+4*f((a+3*b)/4)+f(b))/6;
disp(S2,'S2 : ');
s = S2;
disp (abs(S2-S1),'|S2-S1|<15e/2 ');

b = a;
a = 0;
h = (b-a)/2;

S1 = h*(f(a)+4*f((a+b)/2)+f(b))/3;
disp(S1,'S1 : ');
S2 = h*(f(a)+4*f((3*a+b)/4)+2*f((a+b)/2)+4*f((a+3*b)/4)+f(b))/6;
disp(S2,'S2 : ');

err = abs(S2-S1)/15;
disp(err,'|S2-S1|>15e so interval must be subdivided ');

a = (a+b)/2;
h = (b-a)/2;
S1 = h*(f(a)+4*f((a+b)/2)+f(b))/3;
disp(S1,'S1 : ');
S2 = h*(f(a)+4*f((3*a+b)/4)+2*f((a+b)/2)+4*f((a+3*b)/4)+f(b))/6;
disp(S2,'S2 : ');
s = s+S2;
disp (abs(S2-S1),'|S2-S1|<15e/4 ');
```

```
b = a;
a = 0;
h = (b-a)/2;

S1 = h*(f(a)+4*f((a+b)/2)+f(b))/3;
disp(S1,'S1 : ');
S2 = h*(f(a)+4*f((3*a+b)/4)+2*f((a+b)/2)+4*f((a+3*b)
   /4)+f(b))/6;
disp(S2,'S2 : ');

err = abs(S2-S1)/15;
disp(err,'|S2-S1|>15e so interval must be subdivided
    ' );

a = (a+b)/2;
h = (b-a)/2;
S1 = h*(f(a)+4*f((a+b)/2)+f(b))/3;
disp(S1,'S1 : ');
S2 = h*(f(a)+4*f((3*a+b)/4)+2*f((a+b)/2)+4*f((a+3*b)
   /4)+f(b))/6;
disp(S2,'S2 : ');
s = s+S2;
disp (abs(S2-S1),'|S2-S1|<15e/8 ');

b = a;
a = 0;
h = (b-a)/2;

S1 = h*(f(a)+4*f((a+b)/2)+f(b))/3;
disp(S1,'S1 : ');
S2 = h*(f(a)+4*f((3*a+b)/4)+2*f((a+b)/2)+4*f((a+3*b)
   /4)+f(b))/6;
disp(S2,'S2 : ');
disp (abs(S2-S1),'|S2-S1|<15e/8 ');
s = s+S2;
disp(s);
```

Scilab code Exa 6.11 Gaussian Quadrature Rule

```
//Gaussian Quadrature Rule
clc;
clear;
close();
format('v',10);
funcprot(0);
disp('Integral 0 to 1 f(x)dx');
b = 1;
a = 0;
x = poly(0,'x');
p = x^2-x+1/6;
x1 = roots(p);
A = [1 1;x1'];
//X = [c0;c1];
B = [(b-a);(b^2-a^2)/2];
X = inv(A)*B;
 disp (X,'Are the c1,c2 constants : ');
 disp (x1,'Are the corresponding roots (x1,x2) : ');
 disp ('c0*f(x0)+c1*f(x1)');
```

Scilab code Exa 6.12 Gaussian Quadrature Rule

```
//Gaussian Quadrature Rule
clc;
clear;
close();
```

```
format('v',10);
funcprot(0);
disp('Integral 0 to 2 exp(x)dx');
deff('[y]=f(t)','y=exp(t+1)');
b = 1;
a = -1;
x = poly(0,'x');
p = x^4 - 6*x^2/7+3/35;
x1 = roots(p);
A = [1 1 1 1;x1';(x1.^2)';(x1.^3)'];
B = [(b-a);(b^2-a^2)/2;(b^3-a^3)/3;(b^4-a^4)/4];
C = inv(A)*B;
I = C(1)*f(x1(1))+C(2)*f(x1(2))+C(3)*f(x1(3))+C(4)*f
    (x1(4));
disp(I,'Calculated integration : ');
exact = integrate('exp(x)','x',0,2);
disp(exact,'The exact value of intergation is :');
err = exact - I ;
disp(err,'Error : ' );
```

Chapter 7

Ordinary Differential Eqautions Initial value problem

Scilab code Exa 7.1 Eulers Method

```
//Euler's Method
clc;
clear;
close();
format('v',8);
funcprot(0);
deff('[g]=f(x,y)','g= -y^2/(1+x)');
y = 1;
x = 0;
h = 0.05;
while x<0.2
    y = y - 0.05*y^2/(1+x);
    x = x + h;
    disp(y,x,'Value of y at x :');
end
disp(y,'The calculated value of y(0.2):');
x = 0.2;
act = 1/(1+log(1+x));
disp(act,'The exact value is of y(0.2): ');
```

```
err = act - y;
disp(err,'The error is :');
```

Scilab code Exa 7.2 Eulers trapezoidal predictor corrector pair

```
//Euler's trapezoidal predictor-corrector pair
clc;
clear;
close();
format('v',8);
funcprot(0);
deff('[g]=f(x,y)','g= -y^2/(1+x)');
y = 1;
x = 0;
h = 0.05;
i=0;
while x<0.2
    y0 = y - 0.05*y^2/(1+x);
    disp(y0,'The Y0 :')
    y1 = y - h*(y^2/(1+x)+y0^2/(1+x+h))/2;
    disp(y1,'The Y1 :')
    y2 = y - h*(y^2/(1+x)+y1^2/(1+x+h))/2;
    disp(y2,'The Y2 :')
    y = y2;
    x = x + h;
end
disp(y2,'The calculated value of y(0.2):');
x = 0.2;
act = 1/(1+log(1+x));
disp(act,'The exact value is of y(0.2): ');
err = act - y2;
disp(err,'The error is :');
```

Scilab code Exa 7.3 Mid point formula

```
//Mid-point formula
clc;
clear;
close();
format('v',8);
funcprot(0);
deff('[g]=f(x,y)','g= -y^2/(1+x)');
y0 = 1;
y1 = 0.95335;
x = 0.05;
h = 0.05;
i=0;
while x<0.2
    y2 = y0 - 0.1*y1^2/(1+x);
    disp(y2,'The Y :')
    y0 = y1;
    y1 = y2;
    x = x + h;
end
disp(y2,'The calculated value of y(0.2):');
x = 0.2;
act = 1/(1+log(1+x));
disp(act,'The exact value is of y(0.2): ');
err = act - y2;
disp(err,'The error is :');
```

Scilab code Exa 7.4 Illustraion of Taylor Series for approximation

```
//Illustraion of Taylor Series for approximation
//It needs symbolic toolbox
clc;
clear;
close();
cd ~/Desktop/maxima_symbolic;
exec symbolic.sce
y0 = 1;
x0 = 0;
y1_0 = -y0^2/(1+x0);
y2_0 = (2*y0^3+y0^2)/((1+x0)^2);
y3_0 = -(6*y0^4 + 6*y0^3 + 2*y0^2)/((1+x0)^3);
//similarly
y4_0 = 88;
y5_0 = -694;
y6_0 = 6578;
y7_0 = -72792;
syms r h;
format('v',10);
yxr = 1 - r*h + (y2_0*(r*h)^2)/factorial(2) - (y3_0
    *(r*h)^3)/factorial(3) + (y4_0*(r*h)^4)/factorial
    (4) - (y5_0*(r*h)^5)/factorial(5) +(y6_0*(r*h)^6)
    /factorial(6) - (y7_0*(r*h)^7)/factorial(7);
yxr_5d = 1 - r*h + (y2_0*(r*h)^2)/factorial(2) + (
    y3_0*(r*h)^3)/factorial(3) + (y4_0*(r*h)^4)/
    factorial(4);
h = 0.05;
r = 1;
yx1 = eval(yxr_5d);
format('v',8);
disp(dbl(yx1), 'Value when r = 1 :');

syms r h;
format('v',10);
yxr = 1 - r*h + (y2_0*(r*h)^2)/factorial(2) - (y3_0
    *(r*h)^3)/factorial(3) + (y4_0*(r*h)^4)/factorial
```

```
    (4) - (y5_0*(r*h)^5)/factorial(5) +(y6_0*(r*h)^6)
    /factorial(6) - (y7_0*(r*h)^7)/factorial(7);
yxr_5d = 1 - r*h + (y2_0*(r*h)^2)/factorial(2) + (
    y3_0*(r*h)^3)/factorial(3) + (y4_0*(r*h)^4)/
    factorial(4) + (y5_0*(r*h)^5)/factorial(5) ;
h = 0.05;
r = 2;
yx1 = eval(yxr_5d);
format('v',8);
disp(dbl(yx1), 'Value when r = 2 :')
```

Scilab code Exa 7.5 3 Step Adams Bashforth and 2 step Adam Moulton formula

```
// 3-Step Adams - Bashforth and 2- step Adam-Moulton
     formula
clc;
clear;
close();
format('v',8);
funcprot(0);
disp('Integral 0 to 2 exp(x)dx');
deff('[yd]=f(x,y)','yd = -y^2/(1+x)');

y0 = 1;
x0 = 0;
h = 0.05;
x1 = x0+h;
x2 = x1+h;
y2 = 0.91298;
y1 = 0.95348;
for i = 1:2
    yn = y2 + h*(23*f(x2,y2)-16*f(x1,y1)+5*f(x0,y0))
```

```
    /12;
    disp(yn,'yn(0) = ');
    yn_i = yn;
    yn_i = y2 + h*(5*f(x2+h,yn_i)+8*f(x2,y2)-f(x1,y1
       ))/12;
    disp(yn_i , 'yn(i)');
    yn_i = y2 + h*(5*f(x2+h,yn_i)+8*f(x2,y2)-f(x1,y1
       ))/12;
    disp(yn_i , 'yn(i)');
    y0 = y1;y1 = y2;y2 = yn_i;
    x0 = x1;x1 = x2;x2 = x2+h;
end
x = 0.2 ;
act = 1/(1+log(1+x));
disp(act,'The exact value is of y(0.2): ');
err = act - y2;
disp(err,'The error is :');
```

Scilab code Exa 7.10 Runge Kutta Methods

```
// Runge- Kutta Methods
clc;
clear;
close();
format('v',8);
funcprot(0);
disp('Integral 0 to 2 exp(x)dx');
deff('[t]=f(x,y)','t=-y^2/(1+x)');
yn = 1;
xn = 0;
h = 0.05;
for i = 1:4
    k1 = f(xn,yn);
```

```
    k2 = f(xn+0.5*h,yn+.5*h*k1);
    k3 = f(xn+0.5*h,yn+.5*h*k2);
    k4 = f(xn+h,yn+h*k3);
    yn_1 = yn + h*(k1+2*k2+2*k3+k4)/6;
    n = i-1;
    ann(:,i) = [n k1 k2 k3 k4 yn_1]';
    yn = yn_1;
    xn = xn+h;
end

disp(ann,'Calculated integration : ');
```

Scilab code Exa 7.11 Eulers Methods

```
// Euler's Methods
clc;
clear;
close();
format('v',8);
funcprot(0);
disp('Integral 0 to 2 exp(x)dx');
deff('[ud]=f(u,v)','ud=u^2-2*u*v');
deff('[vd]=g(x,u,v)','vd=u*x+u^2*sin(v)');
un = 1;
vn = -1;
xn = 0;
h = 0.05;
for i = 1:2
    un_1 = un + h*(f(un,vn));
    disp(un_1);
    vn_1 = vn + h*(g(xn,un,vn));
    disp(vn_1);
    vn = vn_1;
```

```
    un = un_1;
    xn = xn + h;
end
ann = [un vn];
disp(ann,'Calculated U2 n V2 values : ');
```

Scilab code Exa 7.12 Eulers trapezoidal predictor corrector pair

```
// Euler's trapezoidal predictor-corrector pair
clc;
clear;
close();
format('v',8);
funcprot(0);
disp('Integral 0 to 2 exp(x)dx');
deff('[ud]=f(u,v)','ud=u^2-2*u*v');
deff('[vd]=g(x,u,v)','vd=u*x+u^2*sin(v)');
un = 1;
vn = -1;
xn = 0;
h = 0.05;
for i = 1:2
    un_1p = un + h*(f(un,vn));
    disp(un_1p);
    vn_1p = vn + h*(g(xn,un,vn));
    disp(vn_1p);
    un_1c = un + h*(f(un,vn)+f(un_1p,vn_1p))/2;
    disp(un_1c);
    vn_1c = vn + h*(g(xn,un,vn)+g(xn+h,un_1p,vn_1p))/2;
    disp(vn_1c);
    un_1cc = un + h*(f(un,vn)+f(un_1c,vn_1c))/2;
    disp(un_1cc);
```

```
    vn_1cc = vn + h*(g(xn,un,vn)+g(xn+h,un_1c,vn_1c)
       )/2;
    disp(vn_1cc);
    vn = vn_1cc;
    un = un_1cc;
    xn = xn + h;
end
ann = [un vn];
disp(ann,'Calculated U2 n V2 values : ');
```

Scilab code Exa 7.13 4 Stage Runge Kutta method

```
// 4-Stage Runge-Kutta method
clc;
clear;
close();
format('v',8);
funcprot(0);
disp('Integral 0 to 2 exp(x)dx');
deff('[ud]=f(u,v)','ud=u^2-2*u*v');
deff('[vd]=g(x,u,v)','vd=u*x+u^2*sin(v)');
un = 1;
vn = -1;
xn = 0;
h = 0.05;
for i = 1:2
    k1 = f(un,vn);
    disp(k1);
    l1 = g(xn,un,vn);
    disp(l1);
    k2 = f(un+.5*h*k1,vn+.5*h*l1) ;
    disp(k2);
    l2 = g(xn+.5*h,un+.5*h*k1,vn+.5*h*l1) ;
```

```
    disp(l2);
    k3 = f(un+.5*h*k2,vn+.5*h*l2) ;
    disp(k3);
    l3 = g(xn+.5*h,un+.5*h*k2,vn+.5*h*l2) ;
    disp(l3);
    k4 = f(un+h*k3,vn+h*l3);
    disp(k4);
    l4 = g(xn+h,un+h*k3,vn+h*l3);
    disp(l4);
    un_1 = un + h*(k1+2*k2+2*k3+k4)/6;
    disp(un_1,'u(n+1) : ');
    vn_1 = vn + h*(l1+2*l2+2*l3+l4)/6;
    disp(vn_1,'v(n+1) : ');
    un = un_1;
    vn = vn_1;
    xn = xn +h;
end
ann = [un vn];
disp(ann,'Calculated U2 n V2 values : ');
```

Chapter 8

Ordinary Differential Eqautions boundary value problem

Scilab code Exa 8.1 The finite difference method

```
//The finite difference method
clc;
clear;
close();
format('v',7);
funcprot(0);
disp('Integral 0 to 2 exp(x)dx');
deff('[pp]=p(x)','pp=x');
deff('[qq]=q(x)','qq=-3');
deff('[rr]=r(x)','rr=exp(x)');
y0 = 1;
yn = 2;
x = [.2 .4 .6 .8 1];
h = 0.2;
A = [-2-h^2*q(x(1)) 1-h*p(x(1))/2 0 0;1+h*p(x(2))/2
    -2-h^2*q(x(2)) 1-h*p(x(2))/2 0;0 1+h*p(x(3))/2
    -2-h^2*q(x(3)) 1-h*p(x(3))/2;0 0 1+h*p(x(4))/2
    -2-h^2*q(x(4))];
disp(A,'A');
```

```
c = [h^2*r(x(1))-(1+h*p(x(1))/2)*y0;h^2*r(x(2));h^2*
    r(x(3));h^2*r(x(4))-(1-h*p(x(4))/2)*yn];
Y = inv(A)*c;
disp(Y','The respective values of y1,y2,y3,y4 : ');
```