Scilab Textbook Companion for Elementary Numerical Analysis: An Algorithmic Approach by S. D. Conte And C. de Boor [1]

Created by
Pravalika
B.Tech (pursuing)
Electronics Engineering
Visvesvaraya National Institute Of Technology
College Teacher
M. Devakar, VNIT Nagpur
Cross-Checked by
K. Suryanarayan and Prashant Dave, IITB

May 17, 2016

[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: Elementary Numerical Analysis: An Algorithmic Approach

Author: S. D. Conte And C. de Boor

Publisher: McGraw - Hill Companies

Edition: 3

Year: 1980

ISBN: 70124477

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

Chapter 1

Number systems and Errors

Scilab code Exa 1.1 number system

```

//Example (pg no.20)

   //the number %pi/4 is the value of infinite
      series
//sum(((-1)^i)/(2*i+1))=1-sum(2/(16*(j)^2)-1)
  // The sequence alpha1,alpha2,........is monotone-
     decreasing
  // to its limit %pi/4
  //  0< =(alpha)n - %pi/4 <= (1/(4*n+3))    n
     =1,2,.....
  // To calculate %pi/4 correct to within 10^(-6)
     using this sequence
  // we would need 10^(6)<=4*n+3
  n=(10^(6)-3)/4
       // roughly n=250,000
```

Chapter 2

Interpolation by polynomials

Scilab code Exa 2.1 shifted power form

```
//Example 2.1

// Given    p(6000)=1/3   , p(6001)=-2/3
// From p(x)=a0+a1*x  , by substituting x=6000 & x
   =6001
// we get equations a0+a1*(6000)=1/3 & a0+a1*(6001)
   =-2/3
//solving the above equations we get
a0=6000.3
a1=-1
deff('[y]=f(x)','y=6000.3-x')
f(6000)
f(6001)
// y=6000.3-x  , equation recovers only only the
    first digit of the
  // given function values,a loss of four decimal
      digits
  // remedy for such loss of significance is the use
       of SHIFTED POWER FORM
  //p(x)=a0 + a1*(x-c) + a2*(x-c)^2 + .......+ an*(x-
      c)^n
```

```
  //if we choose the center c to be 6000

  deff('[y]=p(x)','y=0.33333-(x-6000)')
  p(6000)
  p(6001)
```

Scilab code Exa 2.2 second degree interpolating polynomial

```
//Example 2.2

 K(1)=1.5709
 K(4)=1.5727
 K(6)=1.5751
 l0(3.5)=[(3.5-4)*(3.5-6)]/[(1-4)*(1-6)]
 l1(3.5)=[(3.5-1)*(3.5-6)]/[(4-1)*(4-6)]
 l2(3.5)=[(3.5-1)*(3.5-4)]/[(6-1)*(6-4)]
 K(3.5)=l0(3.5)*K(1)+l1(3.5)*K(4)+l2(3.5)*K(6);
 K(3.5)
```

Scilab code Exa 2.3 determine polynomial by newton formula

```
//Example 2.3
         //Using Newton formula

x0=1
x1=4
x2=6
 P2(1)=1.5709
 P2(4)=1.5727
 P2(6)=1.5751
 K1=[P2(1)-P2(4)]/(1-4)
 K2=[P2(4)-P2(6)]/(4-6)
 K3={K1-K2}/(1-6)
```

```
                // Where as K1 = K[1,4] , K2 =[4,6] ,K3 =
                    [1,4,6]
                deff('[y]=f(x)','y=P2(x0)+K1*(x-x0)+K3*(
                    x-x0)*(x-x1)')
                funcprot(0)
x=poly(0,"x")
y=P2(x0)+K1*(x-x0)+K3*(x-x0)*(x-x1)
x=3.5
f
```

Scilab code Exa 2.5 Newtons formula

```
//Example 2.5

 x0=1
 x1=4
 x2=6
 x3=0
  x4=3.5
   K(1)=1.5709
 K(4)=1.5727
 K(6)=1.5751
  P2(1)=1.5709
 P2(4)=1.5727
 P2(6)=1.5751

  p0=K(1)
  U0=1     //U0=U0(x')
 K1=[P2(1)-P2(4)]/(1-4)
 // Where as K1 = K[1,4]
 U1=(x4-x0)*U0        //U1=U1(x')
 p1=p0+U1*K1           //p1=p1(x')

 //adding the point x2=6
 K2=[P2(4)-P2(6)]/(4-6)
```

```
            // Where as K2 =K[4 ,6]

 K3={K1-K2}/(1-6)
            // Where as K1 = K[1 ,4]  , K2 =K[4 ,6]  ,K3 = K[1 ,4 ,6]
   U2=(x4-x0)*(x4-x1)          //U2=U2(x')
   p2=p1+U2*K3       //p2=p2(x')

   // to check error approximation for k(3.5) we add point x3=0
  // K(0)=1.5708=a
  // p2(0)=1.5708=K(0)
  a=1.5708
   K4=[P2(6)-a]/(6-0)
 //K4=K[x2,x3]=[6 ,0]
K5=-0.000001
 //K5=K[x0,x1,x2,x3]
 U3=(x4-x2)*(x4-x1)*(x4-x0)       //U3=U3(x')

 p3=p2+U3*K5
 //p3=p3(x')
```

Scilab code Exa 2.7 find N

```

//Example 2.7

deff('[y]=f(x)','y=(x)^(1/2)')
x0=1
x1=2

//abs(f(x')-p2(x')) <= ((x1-x0)^2)*M/8
//  abs(f(x')-p2(x')) <= 2*((h)^3)/(3*(3)^(1/3))

h=(5*((10)^(-8))*24*((3)^(1/2)))^(1/3)
```

```

//h is approximately 0.0128
//h=(x1-x0)/N

N={(x1-x0)/h}
//N is aproximately 79
```

Chapter 3

The solution of nonlinear equations

check Appendix AP 4 for dependency:

`bisect.sce`

check Appendix AP 6 for dependency:

`newt.sce`

check Appendix AP 7 for dependency:

`regulfalsi.sce`

check Appendix AP 3 for dependency:

`secantm.sce`

Scilab code Exa 3.1 Root finding

```
   // Example(3.1)

// finding roots using bisection method

     deff('[y]=f(x)','y=x-0.2*sin(x)-0.5')
```

```
     bisection(0.5,1.0,f)


//regula falsi method

deff('[y]=f(x)','y=x-0.2*sin(x)-0.5')
regularfalsi(0.5,1.0,f)

//secant method

deff('[y]=f(x)','y=x-0.2*sin(x)-0.5')
secant(0.5,1.0,f)


//newton rapson method


x=(0.5+1)/2
deff('[y]=f(x)','y=x-0.2*sin(x)-0.5')
deff ('[y]=g(x)','y=1-0.2*cos(x)')
x=newton(x,f,g)
```

check Appendix AP 4 for dependency:

`bisect.sce`

Scilab code Exa 3.2.a finding roots

```
//example(3.2a)


//bisection method

deff('[y]=f(x)','y=x^3-x-1')
bisection(1.0,1.5,f)

```

```
//regula falsi method

deff('[y]=f(x)','y=x^3-x-1')
regularfalsi(1.0,1.5,f)

//secant method

deff('[y]=f(x)','y=x^3-x-1')
secant(1.0,1.5,f)


//newton rapson method


x=(0.5+1)/2
deff('[y]=f(x)','y=x^3-x-1')
deff('[y]=g(x)','y=1-0.2*cos(x)')

newton(x,f,g)
```

check Appendix AP 5 for dependency:

`fixedp.sce`

check Appendix AP 6 for dependency:

`newt.sce`

check Appendix AP 7 for dependency:

`regulfalsi.sce`

Scilab code Exa 3.3 Fixed point iteration

```
//example(3.3)

```

```
                          //here f(x)=e^(-x)-sin(x) ,according
                               to fixed point iteration we take
                               g(x)=x=x+e^(-x)-sin(x);
                              // so, xn=g(xn)
deff('[y]=g(x)','y=x+(2.718)^(-x)-sin(x)')
x=0.6
for n=1:1:18
 g(x);
   x=g(x)
end
```

Scilab code Exa 3.4 Fixed point iteration

```
//example(3.4)

    //here f(x)=1.5*x-tan(x)-0.1=0, according to
       fixed point iteration we get x=(0.1+tan(x))
       /1.5
   //where g(x)=x=(0.1+tan(x))/1.5
   //& xn=g(xn)
deff('[y]=g(x)','y=(0.1+tan(x))/1.5')
x=0

for n=1:1:10
    g(x);
    x=g(x)
end
```

check Appendix AP 3 for dependency:

`secantm.sce`

Scilab code Exa 3.5 Secant method

```
//example(3.5)

deff ('[y]=f(x)','y=x^3-x-1')
secant(1.0,1.5,f)
```

Scilab code Exa 3.8.a Roots of a polynomial equation

```
//example(pg no.111)


//a,b & f are the modulus coeff of x^0,x^1,x^5
 c=[-6.8 10.8 -10.8 7.4 -3.7 1]
 a=6.8;
 b=10.8;
 f=1;
  n=5
 p5=poly(c,'x','coeff')
  p=n*a/b
  q=a/f^(1/n)
 roots(p4)

 xset('window',0);
x=-2:.01:2.5;                                                   //
   defining the range of x.
deff('[y]=f(x)','y=x^5-3.7*x^4+7.4*x^3-10.8*x
   ^2+10.8*x-6.8');                    //defining the
   cunction
y=feval(x,f);

a=gca();

a.y_location = "origin";

a.x_location = "origin";
```

```
plot(x,y)
    // instruction to plot the graph

title(' y = 8*x^3-12*x^2-2*x+3')
```

Scilab code Exa 3.8 Polynomial equations real roots

```
//example(3.8)



  //a,b & e are the modulus coeff of x^0,x^1,x^4
 c=[-1 1 -1 -1 1]
a=1;
b=1;
e=1;
 n=4
 p4=poly(c,'x','coeff')
  p=n*a/b
  q=(a/e)^(1/n)
 roots(p4)
      //from here we found that only 2 real roots,
         other two are complex roots
 xset('window',0);
x=-2:0.1:3;                                                     //
    defining the range of x.
deff('[y]=f(x)','y=x^4-x^3-x^2+x-1');
                    //defining the function
y=feval(x,f);

a=gca();

a.y_location = "origin";
```

```

a.x_location = "origin";
plot(x,y)
    // instruction to plot the graph

title(' y =x^4-x^3-x^2+x-1')
```

Scilab code Exa 3.10 Roots of polynomial equation

```
//example(3.10)

 c=[-3 1 0 1]]
 p3=poly(c,'x','coeff')
 roots(p3)
 //here
  xset('window',0);
x=-2:.01:2.5;                                                  //
    defining the range of x.
deff('[y]=f(x)','y=x^3+x-3');                                  //
    defining the cunction
y=feval(x,f);

a=gca();

a.y_location = "origin";

a.x_location = "origin";
plot(x,y)
    // instruction to plot the graph

title(' y =x^3+x-3')
```

Scilab code Exa 3.11 Roots of polynomial equation

```
//example(3.11)
c=[-6.8 10.8 -10.8 7.4 -3.7 1]
p5=poly(c,'y','coeff')
roots(p5)
```

Scilab code Exa 3.12 Roots of polynomial equation

```
//example(3.12)

c=[-5040 13068 -13132 6769 -1960 322 -28 1]
p7=poly(c,'y','coeff')
roots(p7)
```

Scilab code Exa 3.14 Roots of a polynomial equation

```
//example(3.14)

c=[-3 1 0 1]
p3=poly(c,'y','coeff')
roots(p3)
```

Scilab code Exa 3.15 Roots of a polynomial equation

```
//example(3.15)

```

```
c=[-6.8 10.8 -10.8 7.4 -3.7 1]
p5=poly(c,'x','coeff')
roots(p5)
```

Scilab code Exa 3.16 Roots of polynomial equation

```
//example(3.16)

c=[-5040 13068 -13132 6769 -1960 322 -28 1]
p7=poly(c,'x','coeff')
roots(p7)
```

Scilab code Exa 3.17.a Roots of a polynomial equation

```
//example(3.17)

 c=[51200 0 -39712 0 7392 0 -170 0 1 ]
 p8=poly(c,'x','coeff')
roots(p8)

 xset('window',0);
x=-11:01:11;                                                  //defining the range of x.
deff('[y]=f(x)','y=x^8-170*x^6+7392*x^4-39712*x^2+51200');                  //defining the cunction
y=feval(x,f);

a=gca();

a.y_location = "origin";

```

```
a.x_location = "origin";
plot(x,y)
    // instruction to plot the graph

title(' y =x^8-170*x^6+7392*x^4-39712*x^2+51200')
```

Chapter 4

Matrices and Systems of linear Equations

Scilab code Exa 4.1 matrix multiplication

```

//Example (pg no.130)

          A=[3 0 2;1 2 0;0 1 1]
          B=[2 1;0 1;1 0]
            A*B
```

Scilab code Exa 4.2 matrix multiplication

```
//Example (pg no.130)


       A=[2 1;1 3]
       B=[2 1;0 1]
       A*B
       B*A
```

```
//matrix multiplication is not commutative
            //so A*B!=B*A
```

Scilab code Exa 4.3 properties of matrices

```
//Example (pg no.133)

       A=[1 1;0 1]
       inv(A)
       B=[1 0;1 1]
       inv(B)
       A*B
       inv(A*B)
       inv(A)*inv(B)
                                                   //inv(A*B)=
                                                     inv(B)*inv
                                                     (A)
       inv(B)*inv(A)
                                                     //Hence inv
                                                       (A)*inv(
                                                       B) = inv
                                                       (A)*inv(
                                                       B)
        I=eye(3,3)
        C=(A*B)*(inv(A)*inv(B))
                                                    //C!=I
                                                     //so, inv(A
                                                       )*inv(B)
                                                        cannot
                                                       be the
                                                       inverse
                                                       of (A*B)
```

Scilab code Exa 4.4 matrix

```

//Example (pg no.136)

// x1 + 2(x2) = 3
//2(x1) + 4(x2) = 6

        A=[1 2;2 4]
            //coefficient matrix of above equations
        b=[3 6]'
        x=A\b
    //for corresponding homogenous system
             //  x1 + 2(x2) = 0
            //2(x1) + 4(x2) = 0
         A=[1 2;2 4]
             //coefficient matrix of above equations
        b=[0 0]'
        x=A\b
```

Scilab code Exa 4.6 determinant

```
//Example (pg no.140)

         A=[1 2;2 4]
         det(A)
                        // Here A is a singular
                           matrix i.e, det(A)=0
                        //inv(A)=(adj(A))/det(A)
                        //so A is not invertible
```

Scilab code Exa 4.7 matrix

```
//Example (pg no.144)

A = [1 2 3;4 5 6;7 8 9]
[L,U,E] = lu(A)
       // L is lower triangular matrix(mxn)
       // U is upper triangular matrix(mxmin(m,n))
      // E is permutation matrix(min(m,n)xn)
A*E
```

Scilab code Exa 4.8 Backward substitution

```
//example 4.1 (pg 149)

     //2x1 + 3x2 − x3 = 5
    //−2x2 − x3 = −7
   //−5x3 = −15

A = [2 3 -1;0 -2 -1;0 0 -5]
b = [5 -7 -15]'
a=[A b]
[nA,mA]=size(A)
n=nA

     //Backward substitution

x(3) = a(n,n+1)/a(n,n);

for i = n-1:-1:1
    sumk=0;
    for k=i+1:n
        sumk=sumk+a(i,k)*x(k);
    end
    x(i)=(a(i,n+1)-sumk)/a(i,i);
end
    disp(x)
```

Scilab code Exa 4.14 norm

```
//example(3.14)

 c=[-3 1 0 1]
 p3=poly(c,'y','coeff')
 roots(p3)
```

Scilab code Exa 4.16 determinant

```
//Example (pg no.186)


//(1)
         A=[1 4;2 3]
         det(A)

//(2)
         A=[4 1;3 2]
         det(A)
```

Scilab code Exa 4.17 determinant

```
//Example (pg no.186)

        A=[3.1 4;3.2 3]
        det(A)
```

Scilab code Exa 4.18 determinant

```
//Example (pg no.186)

          A=[1 2;2 2]
          B=[1 2;1 1]
          det(A)+det(B)
          C=[1 2; 3 3]
          det(C)
            //det(A)+det(B)=det(C)
```

Scilab code Exa 4.19 Eigen values

```
//Example(4.11) (pg no.191)

           B=[1 2 0;2 1 0;0 0 -1]
           lam = spec(B)
```

Scilab code Exa 4.20 Eigen values

```
//Example(4.14) (pg no.201)


B=[1 2 0;2 1 0;0 0 -1]
lam = spec(B)
norm(B)
        //Each eigen value of the matrix must have absolute value
        //      no bigger than the norm of that matrix
```

Scilab code Exa 4.21 determinant

```
//Example(4.15) (pgno 202)

B=[1 2 0;2 1 0;0 0 -1]
I=[1 0 0;0 1 0;0 0 1]
          //here we are taking lamda=a
          //det(B-a)*I=0 is characteristic equation
             to get lamda
deff('[y]=p(a)','p(a)=det(B-a)*I')
p(a)=0
```

Scilab code Exa 4.22 eigen values

```
//Example(4.16)(page no.203)

A=[4 -1 -1 -1;-1 4 -1 -1;-1 -1 4 -1;-1 -1 -1 4]
spec(A)
```

Chapter 5

systems of equations and unconstraned optimization

Scilab code Exa 5.1 gradient

```

//Example 5.1

deff('y=f(x)','y=((x1)^3)+((x2)^3)-2*((x1)^2)+3*((x2)^2)-8')
funcprot(0)
deff('y=g(x)','y=3*((x1)^2)-4*(x1)+3*((x2)^2)+6*(x2)')
       // f1=(df/dx1)(x)=0 , f2=(df/dx2)(x)=0
  deff('y=fp(x)','y=3*((x1)^2)-4*(x1)')
  deff('y=fpp(x)','y=3*((x2)^2)+6*(x2)')
       x1=poly(0,"x1")
 fp=(3*((x1)^2)-4*(x1))
 p=roots(fp)

 x2=poly(0,"x12")
 fpp=3*((x2)^2)+6*(x2)
 p=roots(fpp)
```

Scilab code Exa 5.2 Steep descent

```

//Example 5.2

 deff('[y]=f(x1,x2)','y=((x1)^3)+((x2)^3)-2*((x1)^2)+3*((x2)^2)-8')
   //x1=1 , x2=-1
    //(del)f(X(0))=[3*((x1)^2)-4*x1,3*((x2)^2)+6*x2]'=[-1,-3]'
    //Thus , in the first step of steep descent ,
    // we look for a minimum of the function
 funcprot(0)
deff('[y]= g(t)','y=((1+t)^3)+((-1+3*t)^3)-2*((1+t)^2)+3*((-1+3*t)^2)-8')
   //g'(t)=3*((1+t)^2)+3*3*((-1+3*t)^2)-4*(1+t)+3*2*(-1+3*t)
  t=poly(0,"t")
y=3*((1+t)^2)+3*3*((-1+3*t)^2)-4*(1+t)+3*2*3*(-1+3*t)
p=roots(y)
   // We choose the positive root t=1/3
   t=1/3;
   x1=1+t
   x2=-1+3*t
   a=3*((x1)^2)-4*x1
   b=3*((x2)^2)+6*x2
   funcprot(0)
 deff('[y]=fp(x1)','y=(3*((x1)^2)-4*(x1))')

  x1=poly(0,"x1")
 fp=(3*((x1)^2)-4*(x1))
 p=roots(fp)
```

Chapter 6

Approximation

Scilab code Exa 6.1 uniform aproximation

```
//Example 6.1

deff('[y]=f(x)','y=exp(x)')
x0=-1
x1=0
x2=1
//  F=f(x0,x1,x2)=f(-1,0,1)
F=f(x0)/[(x1-x0)*(x2-x0)]+f(x1)/[(x0-x1)*(x2-x1)]+f(
    x2)/[(x0-x2)*(x1-x2)]
   // W(-1,0,1)=2 and so, for a<= -1,1 <=b
  // |f[-1,0,1]|/2 <= dist(at infinity)(f,pi1)*****
  // dist(exp(x),pi1) >= 0.27154
```

Scilab code Exa 6.2 distance at infinity

```
//Example 6.2

 deff('[y]=f(x)','y =tan((%pi/4)*x)')
```

```

// on std interval −1 <= x <= 1 from pi3
// this is an odd function f(−x)=f(x)
n=4
p= (1/(2*(n+1)))*(f(1)-2*f(cos(%pi/(n+1)))+2*f(cos
   (2*%pi/(n+1)))-2*f(cos(3*%pi/(n+1)))+2*f(cos(4*
   %pi/(n+1)))-f(-1))
    // 0.00203 <= dist(at infinity)(f,pi4)=p=0.0041
```

Scilab code Exa 6.3 aproximation

```
//Example 6.3

deff('[y]=f(x)','y=exp(x)')
xset('window',0);
x=-1:.01:1;                // defining the range of
   x.
y=feval(x,f);

a=gca();

a.y_location = "origin";

a.x_location = "origin";
plot(x,y)                  // instruction to plot the
   graph



//     possible approximation
//         y = q1(x)

//     Let e(x) = exp(x) − [a0+a1*x]
//     q1(x) & exp(x) must be equal at two points in
   [−1,1], say at x1 & x2
```

```
//     sigma1 = max(abs(e(x)))
//     e(x1) = e(x2) = 0.
//     By another argument based on shifting the graph of y = q1(x),
//     we conclude that the maximum error sigma1 is attained at exactly 3 points.
//     e(-1) = sigma1
//     e(1) = sigma1
//     e(x3) = -sigma1
//     x1 < x3 < x2
//     Since e(x) has a relative minimum at x3, we have e'(x) = 0
//     Combining these 4 equations, we have..
//     exp(-1) - [a0-a1] = sigma1 --------------------(i)
//     exp(1) - [a0+a1] = p1 -------------------------(ii)
//     exp(x3) - [a0+a1*x3] = -sigma1 ----------------(iii)
//     exp(x3) - a1 = 0 ------------------------------(iv)

//     These have the solution

a1 = (exp(1) - exp(-1))/2
x3 = log(a1)
sigma1 = 0.5*exp(-1) + x3*(exp(1) - exp(-1))/4
a0 = sigma1 + (1-x3)*a1

x = poly(0,"x");
//     Thus,
q1 = a0 + a1*x

deff('[y1]=f(x)','y1=1.2643+1.1752*x')

xset('window',0);
x=-1:.01:1;                // defining the range of x.
```

```
y=feval(x,f);

a=gca();

a.y_location = "origin";

a.x_location = "origin";
plot(x,y)                    // instruction to plot the
   graph
```

Scilab code Exa 6.5 approximate

```
//Example 6.5

//xn=10+(n-1)/5
//Accordingly we choose
//phi1(x)=1 , phi2(x)=x ,phi3(x)=(x)^3
A=[6 63 662.2; 63 662.2 6967.8; 662.2 6967.8
   73393.5664]
norm(A,'inf')
x=[10.07 -2 0.099]'
A*x
norm(A*x,'inf')
norm(A*x)
a=(norm(x))/norm((A)^(-1))

//norm(A*x) >=norm(x)/norm((A)^(-1))
// norm(A^(-1),'inf') >= 7.8

cond(A)

//the condition number of A is much larger than
   10^5, so we take
deff('[y]=f(x)','y=10-2*x+(((x)^2))/2')
//f(x) is a polynomial of degree 2 ,F*(x) should be
```

```
    f(x) itself

c1=10
c2=-2
c3=0.1

// by using elimination algorithm (Gauss elimination
    ), we get
c1=9.99999997437
c2=-1.999999951
c3=0.099999976
 // by 14-decimal digit floating point arith metic
    method
 c1=6.035
 c2=-1.243
 c3=0.0639
 //calculation carried out in seven decimal digit
    floating point arithmetic
 c1=8.492
 c2=-1.712
 c3=0.0863
```

Scilab code Exa 6.11 polynomial of degree lessthan 3

```

//Example 6.11

x0=-1
x1=18
            //pi=<f,pi>
p0=integrate('exp(x)','x',x0,x1)

p1=integrate('x*exp(x)','x',x0,x1)

p2=integrate('(exp(x))*((x^2)-(1/3))','x',x0,x1)
```

```

p3=integrate('(exp(x))*((x^3)-3*x/5)','x',x0,x1)

//for legendre polynomials one can show
//si= <pi,pi> = 2/(2*i+1)
s0=2/(2*0+1)
s1=2/(2+1)
s2=2/(2*2+1)
s3=2/(2*3+1)

//di*=<f,pi>/si
//p*(x)=y=d0*1+d1*x+d2*(3/2)*((x^2)-(1/3))+d3*((x^3)
    -3*x/5)*(5/2)
//p*(x)=y=(p0/s0)*1+(p1/s1)*x+(p2/s2)*(3/2)*((x^2)
    -(1/3))+(p3/s3)*((x^3)-3*x/5)*(5/2)
poly(0,"x")
y=1.17552011*(1)+(1.103638324)*(x)+(0.3578143506)
    *(3/2)*((x^2)-(1/3))+(0.07045563367)*((x^3)-3*x
    /5)*(5/2)
//On (-1,1) ,this polynomial a maximum deviation
    from exp(x) of about 0.01
```

Scilab code Exa 6.12 Least squares approximation

```



/Example 6.12

//Least squares approximation

deff('[y]=f(x)','y=10-2*x+((x^2)/10)')
//we seek the polynomial of degree <= 2 which
    minimizes
//sum(n=1 to 9)[fn-p(xn)]^2
```

```
//we are dealing with scalar product with w(x)=1
P0(x)=1
//hence
s0=0;
B=0;
A1=0;
s1=0;
for n=1:1:6

s0=s0+1
B=[10+(n-1)/5]+B

A1=[10+[n-1]/5]*{[((n-1)/5)-0.5]^2}+A1

s1={[((n-1)/5)-0.5]^2}+s1

end
B0=B/s0

B1=A1/s1
C1=s1/s0

x=poly(0,"x")
y1=x-B0
x=poly(0,"x")
y2=((x-B0)^2)-0.1166667
//similarly calculate s2
s2=0.05973332
//p*(x)=(d0*)*P0(x)+(d1*)*P1(x)+(d2*)*P2(x)
//d0*=d0,d1*=d1,d2*=d2 are least squares
    approximation
//d0*=d0=sigma(n=1 to 6)[fn/6]        where fn=f(xn)

d0=0.03666667
d1=0.1
d2=0.0999999

x=poly(0,"x")
```

```
p=d0+d1*(x-B0)+d2*{[(x-B0)^2]-C1}
 //c1=c1*  ,c2=c2*,c3=c3*
 c1=9.99998
c2=-1.9999998
c3=0.0999999
```

Chapter 7

differentiation and integration

Scilab code Exa 7.1 integration

```
//I=integral(exp^(-x^2) dx)

deff('y=f(x)','y=exp(-(x^2))')
a=0,b=1
c=(a+b)/2
deff('y=g(x)','y=-2*x*exp(-(x^2))')
f(a)
f(b)
f(c)
g(a)
g(b)
g(c)
R=(b-a)*f(a)
M=(b-a)*f(c)
T=(b-a)*[f(a)+f(b)]/2
S=(b-a)*{f(a)+4*f(c)+f(b)}/6
CT=[(b-a)/2]*[f(a)+f(b)]+[(b-a^2)/12]*[g(a)-g(b)]
```

Scilab code Exa 7.8 adaptive quadrature

```
//Example 7.8

// True value of the integral
x0=0
x1=1
I=integrate('sqrt(x)','x',0,1)

//using adaptive quadrature based on simpsons rule

deff('[y]=f(x)','y=[(x)^(1/2)]')
x=1:1:10
 plot(x,f)

x2=(x0+x1)/2;
h=1/2
//considering the interval [x2,x1]=[1/2,1]

s=h/6.*{f(x2)+4*f((x2)+h/2)+f(x1)}
p=h/12*{f(x2)+4*f((x2)+h/4)+2*f((x2)+h/2)+4*f(x2+3*h
   /4)+f(x1)}
E=(1/15)*(p-s)
////Error criterion is clearly satisfied , hence we
   put value of p to SUM register to obtain partial
   approximation
//considering the interval  [x2,x1]=[0,1/2]

s1=h/6.*{f(x0)+4*f((x0)+h/2)+f(x2)}
p1=h/12.*{f(x0)+4*f((x0)+h/4)+2*f((x0)+h/2)+4*f(x0
   +3*h/4)+f(x2)}
E1=1/15.*[p1-s1]

// Here since error is not less than 0.00025 we have
    to divide interval[0,1/2] to [0,1/4]& [1/4,1/2]
h=1/4
//considering interval [1/4,1/2]

x3=1/4

```

```
s2=h/6.*{f(x3)+4*f((x3)+h/2)+f(x2)}
p2=h/12.*{f(x3)+4*f((x3)+h/4)+2*f((x3)+h/2)+4*f(x3+3*h/4)+f(x2)}
E2=1/15.*[p2-s2]

// E2 < (0.0005)/4
//Error criterion is clearly satisfied , hence we add value of p2 to SUM register to obtain partial approximation
sum=p+p2
funcprot(0)
//Applying again above basic formulas

//with h=1/4 , in interval [0.1/4]
// we get

s3=0.07975890
p3=0.08206578
E3=0.0001537922
 // Here since error is not less than 0.000125 we have to divide interval
// [0,1/4] in to [0,1/8]& [1/8,1/4] with h=1/8
h=1/8

// for interval [1/8,1/4]

s4=0.05386675
p4=0.05387027
E4=0.0000002346


// E4 < (0.0005)*h =(0.0005)/8 =0.0000625
//Error criterion is clearly satisfied , hence we add value of p4 to
//SUM register to obtain partial approximation
 sum=p+p2+p4
```

```
// consider interval [0,1/8]

s5=0.02819903
p5=0.02901464
E5=0.00005437

// E5 < 0.0000625

//Since the error test is passed on both intervals , we can add these values in to sUM register to get
sum=p+p2+p4+p5

// since the exact value of I is 0.666666666
abs(sum-I) <0.0005  // over the interval [0,1]
```

Chapter 8

THE SOLUTION OF DIFFERENTIAL EQUATIONS

Scilab code Exa 8.1 Taylor series

```
// Example 8.1

deff('[v]=f(x,y)','v=1-(y/x)')
funcprot(0)
deff('[v]=fp(x,y)','v=-(f(x)/x)+(y/(x^2))')
funcprot(0)
deff('[v]=fpp(x,y)','v=-(fp(x)/x)+2*(f(x)/(x^2))-2*(
   y/(x^3))')
funcprot(0)
deff('[v]=g(x,y)','v=-(fpp(x)/x)+3*(fp(x)/(x^2))-6*(
   f(x)/(x^3))+6*(y/(x^4))')
funcprot(0)
x1=2
y1=2
x=2.1
y2=y1+(x-2)*f(x1,y1)+((x-2)^2)*fp(x1,y1)/factorial
   (2)+((x-2)^3)*fpp(x1,y1)/factorial(3)+((x-2)^4)*g
```

```
(x1,y1)/factorial(4)
```

check Appendix AP 2 for dependency:

adamsbash.sce

Scilab code Exa 8.5 Adamsbashforth3

```
// Example 8.5

 deff('[v]=f(x,y)','v=x+y')

 [y,x]= adamsbashforth3(0,0,1,1/32,f)
```

check Appendix AP 1 for dependency:

modifiedEuler.sce

Scilab code Exa 8.6 Modified euler method

```
// Example 8.6

  // Modified Eulers method

  deff('[v]=f(x,y)','v=x-(1/y)')

 [y,x] = modifiedeuler(1,0,0.2,0.1,f)
```

Appendix

Scilab code AP 1 Modified euler method

```
function [u,t] = modifiedeuler(u0,t0,tn,h,f)

//modifiedeuler 1st order method solving ODE
//  du/dt = f(u,t), with initial
//conditions u=u0 at t=t0.  The
//solution is obtained for t = [t0:h:tn]
//and returned in u

umaxAllowed = 1e+100;

t = [t0:h:tn]; u = zeros(t); n = length(u); u(1) = u0;

for j = 1:n-1
    k1=h*f(t(j),u(j));
    k2=h*f(t(j)+h/2,u(j)+k1/2);
    u(j+1) = u(j) + k2;
   if u(j+1) > umaxAllowed then
           disp('Euler 1 - WARNING: underflow or overflow');
           disp('Solution sought in the following range:');
           disp([t0 h tn]);
          disp('Solution evaluated in the following range:');
          disp([t0 h t(j)]);
```

```
            n = j; t = t(1,1:n); u = u(1,1:n);
    break;
    end;
end;

endfunction
```

Scilab code AP 2 adamsbashforth

```

function [u,t] = adamsbashforth3(u0,t0,tn,h,f)

//adamsbashforth3 3rd order method solving ODE
//  du/dt = f(u,t), with initial
//conditions u=u0 at t=t0.  The
//solution is obtained for t = [t0:h:tn]
//and returned in u

umaxAllowed = 1e+100;

t = [t0:h:tn]; u = zeros(t); n = length(u); u(1) = u0;
for j = 1:n-1
if j<3 then
      k1=h*f(t(j),u(j));
    k2=h*f(t(j)+h,u(j)+k1);
    u(j+1) = u(j) + (k2+k1)/2;
end;

if j>=2 then
       u(j+2) = u(j+1) + (h/12)*(23*f(t(j+1),u(j+1))-16*f(t(j),u(j))+5*f(t(j-1),u(j-1)));
end;
end;
endfunction
```

Scilab code AP 3 secant mehod

```
function [x]=secant(a,b,f)
    N=100;              // define max. no. iterations
        to be performed
    PE=10^-4            // define tolerance for
       convergence
     for n=1:1:N        // initiating for loop
        x=a-(a-b)*f(a)/(f(a)-f(b));
        disp(x)
        if abs(f(x))<=PE then break; //checking for
           the required condition
        else a=b;
            b=x;
        end
     end
     disp(n," no. of iterations =") //
endfunction
```

Scilab code AP 4 bisection method

```
function x=bisection(a,b,f)
    N=100;                                       //
       define max. number of iterations
    PE=10^-4                                   //
       define tolerance
    if (f(a)*f(b) > 0) then
         error('no root possible f(a)*f(b) > 0')
            // checking if the decided range is
            containing a root
          abort;
    end;
    if(abs(f(a)) <PE) then
        error('solution at a')                     //
            seeing if there is an approximate root
           at a,
         abort;
    end;
    if(abs(f(b)) < PE) then                        //
       seeing if there is an approximate root at b,
```

```
    error('solution at b')
    abort;
    end;
    x=(a+b)/2
    for n=1:1:N                                    //
       initialising 'for' loop,
        p=f(a)*f(x)
        if p<0 then b=x ,x=(a+x)/2;
           //checking for the required conditions( f
           (x)*f(a)<0),
        else
             a=x
            x=(x+b)/2;
        end
        if abs(f(x))<=PE then break
           // instruction to come out of the loop
           after the required condition is achived,
        end
    end
    disp(n," no. of iterations =")
       // display the no. of iterations took to
       achive required condition,
endfunction
```

Scilab code AP 5 fixed point

```
function [x]=fixedp(x0,f)
//fixed-point iteration
N = 100; eps = 1.e-5; // define max. no. iterations
   and error
maxval = 10000.0;     // define value for divergence
a = x0;
while (N>0)
    xn = f(a);
    if(abs(xn-a)<eps)then
        x=xn
        disp(100-N);
        return(x);
```

```
    end;
    if (abs(f(x))>maxval)then
        disp(100-N);
        error('Solution diverges');
        abort;
    end;
    N = N - 1;
    xx = xn;
end;
error('No convergence');
abort;
//end function
```

Scilab code AP 6 newton method

```
function x=newton(x,f,fp)
    R=100;
    PE=10^-8;
    maxval=10^4;

    for n=1:1:R
        x=x-f(x)/fp(x);
        if abs(f(x))<=PE then break
        end
        if (abs(f(x))>maxval) then error('Solution
           diverges');
            abort
            break
        end
    end
    disp(n," no. of iterations =")
endfunction
```

Scilab code AP 7 Regular falsi method

```
function [x]=regularfalsi(a,b,f)
    N=100;
    PE=10^-5;
```

```
    for n=2:1:N
        x=a-(a-b)*f(a)/(f(a)-f(b));
        disp(x)
        if abs(f(x))<=PE then break;
        elseif (f(a)*f(x)<0) then b=x;
            else a=x;
        end
    end
    disp(n," no. of ittirations =")
endfunction
```