Scilab Textbook Companion for Numerical Methods by E. Balaguruswamy[1]

Created by
Arralli Prashanth
Numerical methods in Chemical Engineering
Chemical Engineering
IIT Guwahati
College Teacher
Dr. Prakash Kotecha
Cross-Checked by

July 31, 2019

[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 Methods

Author: E. Balaguruswamy

Publisher: Tata McGraw - Hill Education, New Delhi

Edition: 1

Year: 1999

ISBN: 9780074633113

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

Intorduction to Numerical Computing

Scilab code Exa 1.01 Theoritical Problem

```
1 //Example No. 1_01
2 //Pg No. 6
3 disp('Theoritical Problem')
4 disp('For Details go to page no. 6')
```

Chapter 3

Computer Codes and Arithmetic

Scilab code Exa 3.1 binary to decimal

```
1 //Example No. 3_01
2 //Binary to decimal
3 //Pg No. 45
4 clear ;close ; clc ;
5
6 b = '1101.1101'
7 v = strsplit(b,'.')    //splitting integral part and
      fraction part
8 integralp = str2code(v(1))//converting strings to
     numbers
9 fractionp = str2code(v(2))
10 li = length(integralp)  //lenght of integral part
11 lf = length(fractionp)  // and fractional part
12 di = 0 ;//Initializing integral part and decimal
     part
13 df = 0 ;
14 for i = 1:li
15     di = 2*di+integralp(i)
16 end
```

```
for i = lf:-1:1
    df = df/2 + fractionp(i)
end
df = df/2 ;
d = di + df ; //Integral and fractional parts
disp(d,'Decimal value = ')
```

Scilab code Exa 3.2 Hexadecimal to Decimal

```
//Example No. 3_02
//hexadecimal to decimal
//Pg No. 46
clear ; close ; clc ;

h = '12AF' ;
u = str2code(h)
u = abs(u)
n = length(u)
d = 0
for i = 1:n
    d = d*16 + u(i)
end
disp(d,'Decimal value = ')
//Using Scilab Function
d = hex2dec(h)
disp(d,'Using scilab function Decimal value = ')
```

Scilab code Exa 3.3 Decimal to Binary

```
//Example No. 3_03
//Decimal to Binary
//Pg No. 47
clear; close ; clc;
```

```
5
6 d = 43.375 ;
7 //Separating integral part and fractional parts
8 dint = floor(d)
9 dfrac = d - dint
10
11 //Integral Part
12 i = 1 ;
13 intp = dec2bin(dint)
14
15 //Fractional part
16 j = 1 ;
17 while dfrac ~= 0
18     fracp(j) = floor(dfrac*2)
19     dfrac = dfrac*2 - floor(dfrac*2)
20     j = j+1 ;
21 end
22 fracp = strcat(string(fracp))
23
24 b = strcat([intp,fracp],'.') //combining integral
     part and fractional part
25 disp(b,'Binary equivalent = ')
```

Scilab code Exa 3.4 Decimal to Octal

```
1 //Example No. 3_04
2 //Decimal to Octal
3 //Pg No. 48
4 clear ; close ; clc ;
5
6 d = 163 ;
7 oct = dec2oct(d)
8 disp(oct,'Octal number = ')
```

Scilab code Exa 3.5 Decimal to Binary

```
1  //Example No. 3_05
2  //Decimal to binary
3  //Pg No. 48
4  clear ; close ; clc ;
5
6  d = 0.65
7  j = 1 ;
8
9  while d ~= 0
10     fracp(j) = floor(d*2) //integral part of d*2
11     d = d*2 - floor(d*2)  //Fractional part of d*2
12     j = j+1 ;
13     decp(j-1) = d
14     p = 1
15
16     for i = 1:j-2
17         if abs(d - decp(i))< 0.001  then //Condition
               for terminating the recurring binary
              equivalent by
18             p = 0                          //finding
                 out if the new fractional part is
                 equal to any of the previous
                 fractonal parts
19             break
20         end
21     end
22
23     if p == 0 then
24         break
25     end
26
27 end
```

```
28 rec_p = fracp(i+1:j-1)      //Recurring part
29
30 rec_p = strcat(string(rec_p))
31 fracp = strcat(string(fracp))
32
33 disp(strcat( [fracp,rec_p] ),'Binary equivalent = ')
```

Scilab code Exa 3.6 Octal to Hexadecimal

```
1  //Example No. 3_06
2  //Octal to Hexadecimal
3  //Pg No. 49
4  clear ; close ; clc ;
5
6  oct = '243' ;
7  u = str2code(oct)
8  n = length(u)
9  for i = 1:n
10     b(i) = dec2bin(u(i)) //Converting each digit to
          binary equivalent
11     if length(b(i)) == 2 then        //making the
          binary equivalents into a groups of triplets
12         b(i) = strcat(['0',b(i)])
13     elseif length(b(i)) == 1
14         b(i) = strcat(['0','0',b(i)])
15     end
16 end
17 bin = strcat(b) //combining all the triplets
18 i = 1 ;
19 while length(bin) > 4
20     OtoH = strsplit(bin,length(bin)-4) //splitting
          the binary equivalent into groups of binary
          quadruplets
21     bin = OtoH(1)
22     h(i) = OtoH(2)
```

```
23      i = i+1
24  end
25  h(i) = bin ;
26  h = h($:-1:1)
27  h = bin2dec(h)
28  h = dec2hex(h)
29  h = strcat(h)
30
31  disp(h,'Hexadecimal equivalent of octal number 243
       is ')
```

Scilab code Exa 3.7 Hexadecimal to Octal

```
1  //Example No. 3_07
2  //Hexadecimal to Octal
3  //Pg No. 49
4  clear ; close ; clc ;
5
6  h = '39.B8' ;
7  h = strsplit(h,'.') //separating integral part and
       fractional part
8  cint = abs(str2code(h(1)))
9  cfrac = abs(str2code(h(2)))
10 bint = dec2bin(cint)
11 bfrac = dec2bin(cfrac)
12 bint = strcat(bint)
13 bfrac = strcat(bfrac)
14
15 //Integral Part
16 i = 1 ;
17 while length(bint) > 3
18      HtoO = strsplit(bint,length(bint)-3)
19      bint = HtoO(1)
20      oint(i) = HtoO(2)
21      i = i+1 ;
```

```
22 end
23 oint(i) = bint
24 oint =oint($:-1:1)
25 oint = bin2dec(oint)
26
27 //Fraction Part
28 i = 1 ;
29 while length(bfrac)> 3
30     HtoO = strsplit(bfrac,3)
31     bfrac = HtoO(2)
32     ofrac(i) = HtoO(1)
33     i = i+1
34 end
35 ofrac(i) = bfrac
36 ofrac = bin2dec(ofrac)
37
38 //Combining integral part and fraction part
39 oct = strcat([strcat(string(oint)),strcat(string(
      ofrac))],'.')
40 disp(oct,'Octal number equivalent of Hexadecimal
      number 39.B8 is ')
```

Scilab code Exa 3.8 Binary form of negative integers

```
1 //Example No. 3_08
2 //-ve Integer to binary
3 //Pg No. 50
4 clear ; close ; clc ;
5
6 negint = -13
7 posbin = dec2bin(abs(negint))
8 posbin = strcat(['0',posbin])
9 compl_1 = strsubst(posbin,'0','d')
10 compl_1 = strsubst(compl_1,'1','0')
11 compl_1 = strsubst(compl_1,'d','1')
```

```
12 compl_2 = dec2bin(bin2dec(compl_1) + 1)
13
14 disp(compl_2,'Binary equivalent of -13 is ')
```

Scilab code Exa 3.9 16 bit word representation

```
1  //Example No. 3_09
2  //Binary representation
3  //Pg No. 51
4  clear ;close ;clc ;
5
6  n = -32768
7  compl_32767 = dec2bin(bitcmp(abs(n)-1,16) + 1)
8  disp(compl_32767,'binary equivalent of -32767 is ')
9
10 n_1 = -1
11 dcomp = bitcmp(1,16)
12 compl_1 = dec2bin(dcomp+1)
13 disp(compl_1,'binary equivalent of -1 is ')
14 compl_32767_code = str2code(compl_32767)
15 compl_1_code = str2code(compl_1)
16 summ(1) = 1 //since -32768 is a negative number
17 c = 0
18 for i = 16:-1:2
19     summ(i) = compl_32767_code(i) + compl_1_code(i)+
       c
20     if summ(i) == 2 then
21         summ(i) = 0
22         c = 1
23     else
24         c = 0
25     end
26 end
27 binfinal = strcat(string(summ))
28 disp(binfinal,'Binary equivalent of -32768 in a 16
```

```
bit word is ')
```

Scilab code Exa 3.10 Floating Point Notation

```
1  //Example No. 3_10
2  //Floating Point Notation
3  //Pg No. 52
4  clear ; close ; clc ;
5
6  function [m,e] =float_notation(n)
7  m = n ;
8  for i = 1:16
9      if abs(m) >= 1 then
10         m = n/10^i
11         e = i
12     elseif abs(m) < 0.1
13         m = n*10^i
14         e = -i
15     else
16         if i == 1 then
17             e = 0
18         end
19         break ;
20     end
21 end
22 endfunction
23
24 [m,e] = float_notation(0.00596)
25 mprintf('\n 0.00596 is expressed as  %f*10^%i \n',m,
     e)
26 [m,e] = float_notation(65.7452)
27 mprintf('\n 65.7452 is expressed as  %f*10^%i \n',m,
     e)
28 [m,e] = float_notation(-486.8)
29 mprintf('\n -486.8 is expressed as  %f*10^%i \n',m,e
```

)

Scilab code Exa 3.11 Integer Arithmetic

```
1 //Example No. 3_11
2 //Interger Arithmetic
3 //Pg No. 53
4 clear ;close ;clc ;
5
6 disp(int(25 + 12))
7 disp(int(25 - 12))
8 disp(int(12 - 25))
9 disp(int(25*12))
10 disp(int(25/12))
11 disp(int(12/25))
```

Scilab code Exa 3.12 Integer Arithmetic

```
1 //Example No. 3_12
2 //Integer Arithmetic
3 //Pg No. 53
4 clear ;close ;clc ;
5 a = 5 ;
6 b = 7 ;
7 c = 3 ;
8 Lhs = int((a + b)/c)
9 Rhs = int(a/c) + int(b/c)
10 disp(Rhs,'a/c + b/c = ',Lhs,'(a+b)/c = ')
11 if Lhs ~= Rhs then
12     disp('The results are not identical.This is
          because the remainder of an integer division
          is always truncated')
13 end
```

Scilab code Exa 3.13 Floating Point Arithmetic Addition

```
1  //Example No. 3_13
2  //Floating Point Arithmetic
3  //Pg No. 54
4  clear ;close ;clc ;
5
6  fx = 0.586351 ;
7  Ex = 5 ;
8  fy = 0.964572 ;
9  Ey = 2 ;
10 [Ez,n] = max(Ex,Ey)
11 if n == 1 then
12     fy = fy*10^(Ey-Ex)
13     fz = fx + fy
14     if fz > 1 then
15         fz = fz*10^(-1)
16         Ez = Ez + 1
17     end
18     disp(fz,'fz = ',fy,'fy = ',Ez,'Ez = ')
19 else
20     fx = fx*10^(Ex - Ey)
21     fz = fx + fy
22     if fz > 1 then
23         fz = fz*10^(-1)
24         Ez = Ez + 1
25     end
26     disp(fz,'fz = ',fx,'fx = ',Ez,'Ez = ')
27 end
28 mprintf('\n z = %f E%i \n',fz,Ez)
```

Scilab code Exa 3.14 Floating Point Arithmetic Addition

```
//Example No. 3_14
//Floating Point Arithmetic
//Pg No. 54
clear ;close ;clc ;

fx = 0.735816 ;
Ex = 4 ;
fy = 0.635742 ;
Ey = 4 ;
[Ez,n] = max(Ex,Ey)
if n == 1 then
    fy = fy*10^(Ey-Ex)
    fz = fx + fy
    if fz > 1 then
        fz = fz*10^(-1)
        Ez = Ez + 1
    end
    disp(fz,'fz = ',fy,'fy = ',Ez,'Ez = ')
else
    fx = fx*10^(Ex - Ey)
    fz = fx + fy
    if fz > 1 then
        fz = fz*10^(-1)
        Ez = Ez + 1
    end
    disp(fz,'fz = ',fx,'fx = ',Ez,'Ez = ')
end
mprintf('\n z = %f E%i \n',fz,Ez)
```

Scilab code Exa 3.15 Floating Point Arithmetic Subtraction

```
//Example No. 3_15
//Floating Point Arithmetic
//Pg No. 54
clear ;close ;clc ;
```

```
5
6  fx = 0.999658 ;
7  Ex = -3 ;
8  fy = 0.994576 ;
9  Ey = -3 ;
10 Ez = max(Ex,Ey)
11 fy = fy*10^(Ey-Ex)
12 fz = fx - fy
13 disp(fz,'fz = ',Ez,'Ez = ')
14 mprintf('\n z = %f E%i \n',fz,Ez)
15 if fz < 0.1 then
16       fz = fz*10^6       //Since we are using 6
            significant digits
17       n = length(string(fz))
18       fz = fz/10^n
19       Ez = Ez + n - 6
20       mprintf('\n z = %f E%i (normalised) \n',fz,Ez)
21 end
```

Scilab code Exa 3.16 Floating Point Arithmetic Multiplication

```
1  //Example No. 3_16
2  //Floating Point Arithmetic
3  //Pg No. 55
4  clear ;close ;clc ;
5
6  fx = 0.200000 ;
7  Ex = 4 ;
8  fy = 0.400000 ;
9  Ey = -2 ;
10 fz = fx*fy
11 Ez = Ex + Ey
12 mprintf('\n fz = %f \n Ez = %i \n z = %f E%i \n',fz,
      Ez,fz,Ez)
13 if fz < 0.1 then
```

```
14      fz = fz*10
15      Ez = Ez - 1
16      mprintf('\n z = %f E%i (normalised) \n',fz,Ez)
17  end
```

Scilab code Exa 3.17 Floating Point Arithmetic division

```
1  //Example No. 3_17
2  //Floating Point Arithmetic
3  //Pg No. 55
4  clear ;close ;clc ;
5
6  fx = 0.876543 ;
7  Ex = -5 ;
8  fy = 0.200000 ;
9  Ey = -3 ;
10 fz = fx/fy
11 Ez = Ex - Ey
12 mprintf('\n fz = %f \n Ez = %i \n z = %f E%i \n',fz,
       Ez,fz,Ez)
13
14 if fz > 1 then
15      fz = fz/10
16      Ez = Ez + 1
17      mprintf('\n z = %f E%i (normalised) \n',fz,Ez)
18 end
```

Scilab code Exa 3.18 Errors in Arithmetic

```
1  //Example No. 3_18
2  //Floating Point Arithmetic
3  //Pg No. 56
4  clear ;close ;clc ;
```

```
5
6  fx = 0.500000 ;
7  Ex = 1 ;
8  fy = 0.100000 ;
9  Ey = -7 ;
10 [Ez,n] = max(Ex,Ey)
11 if n == 1 then
12     fy = fy*10^(Ey-Ex)
13     fz = fx + fy
14     if fz > 1 then
15         fz = fz*10^(-1)
16         Ez = Ez + 1
17     end
18     disp(fy,'fy = ',Ez,'Ez = ')
19 else
20     fx = fx*10^(Ex - Ey)
21     fz = fx + fy
22     if fz > 1 then
23         fz = fz*10^(-1)
24         Ez = Ez + 1
25     end
26     disp(fx,'fx = ',Ez,'Ez = ')
27 end
28 mprintf('\n fz = %f \n z = %f E%i \n',fz,fz,Ez)
```

Scilab code Exa 3.19 Errors in Arithmetic

```
1  //Example No. 3_19
2  //Floating Point Arithmetic
3  //Pg No. 56
4  clear ;close ;clc ;
5
6  fx = 0.350000 ;
7  Ex = 40 ;
8  fy = 0.500000 ;
```

```
Ey = 70 ;
fz = fx*fy
Ez = Ex + Ey
mprintf('\n fz = %f \n Ez = %i \n z = %f E%i \n',fz,
   Ez,fz,Ez)
if fz < 0.1 then
    fz = fz*10
    Ez = Ez - 1
    mprintf('\n z = %f E%i (normalised) \n',fz,Ez)
end
```

Scilab code Exa 3.20 Errors in Arithmetic

```
//Example No. 3_20
//Floating Point Arithmetic
//Pg No. 56
clear ;close ;clc ;

fx = 0.875000 ;
Ex = -18 ;
fy = 0.200000 ;
Ey = 95 ;
fz = fx/fy
Ez = Ex - Ey
mprintf('\n fz = %f \n Ez = %i \n z = %f E%i \n',fz,
   Ez,fz,Ez)

if fz > 1 then
    fz = fz/10
    Ez = Ez + 1
    mprintf('\n z = %f E%i (normalised) \n',fz,Ez)
end
```

Scilab code Exa 3.21 Errors in Arithmetic

```
1 //Example No. 3_21
2 //Floating Point Arithmetic
3 //Pg No. 57
4 clear ;close ;clc ;
5
6 fx = 0.500000 ;
7 Ex = 0 ;
8 fy = 0.499998 ;
9 Ey = 0 ;
10 Ez = 0 ;
11 fz = fx - fy
12 disp(fz,'fz = ',Ez,'Ez = ')
13 mprintf('\n z = %f E%i \n',fz,Ez)
14 if fz < 0.1 then
15      fz = fz*10^6
16      n  = length(string(fz))
17      fz = fz/10^n
18      Ez = Ez + n - 6
19      mprintf('\n z = %f E%i (normalised) \n',fz,Ez)
20 end
```

Scilab code Exa 3.22 Associative law of Addition

```
1 //Example No. 3_22
2 //Laws of Arithmetic
3 //Pg No. 57
4 clear ; close ; clc ;
5 function [fz,Ez] =add_sub(fx,Ex,fy,Ey) //addition
     and subtraction fuction
6 if fx*fy >= 0 then
7     //Addition
8     [Ez,n] = max(Ex,Ey)
9     if n == 1 then
```

```
10          fy = fy*10^(Ey-Ex)
11          fz = fx + fy
12          if fz > 1 then
13              fz = fz*10^(-1)
14              Ez = Ez + 1
15          end
16      else
17          fx = fx*10^(Ex - Ey)
18          fz = fx + fy
19          if fz > 1 then
20              fz = fz*10^(-1)
21              Ez = Ez + 1
22          end
23      end
24
25  else
26      //Subtraction
27      [Ez,n] = max(Ex,Ey)
28      if n == 1 then
29          fy = fy*10^(Ey-Ex)
30          fz = fx + fy
31          if abs(fz) < 0.1 then
32             fz = fz*10^6
33             fz = floor(fz)
34             nfz = length(string(abs(fz)))
35             fz = fz/10^nfz
36             Ez = nfz - 6
37          end
38      else
39          fx = fx*10^(Ex - Ey)
40          fz = fx + fy
41          if fz < 0.1 then
42             fz = fz*10^6
43             fz = int(fz)
44             nfz = length(string(abs(fz)))
45             fz = fz/10^nfz
46             Ez = nfz - 6
47          end
```

```
48      end
49  end
50  endfunction
51
52  fx = 0.456732
53  Ex = -2
54  fy = 0.243451
55  Ey = 0
56  fz = -0.24800
57  Ez = 0
58
59  [fxy,Exy] = add_sub(fx,Ex,fy,Ey)
60  [fxy_z,Exy_z] = add_sub(fxy,Exy,fz,Ez)
61  [fyz,Eyz] = add_sub(fy,Ey,fz,Ez)
62  [fx_yz,Ex_yz] = add_sub(fx,Ex,fyz,Eyz)
63  mprintf('fxy = %f\n Exy = %i \n fxy_z = %f\n Exy_z =
        %i \n fyz = %f \n Eyz = %i \n fx_yz = %f \n
       Ex_yz = %i \n',fxy,Exy,fxy_z,Exy_z,fyz,Eyz,fx_yz,
       Ex_yz)
64
65  if  fxy_z ~= fx_yz | Exy_z ~= Ex_yz then
66      disp('(x+y) + z ~= x + (y+z)')
67  end
```

Scilab code Exa 3.23 Associative law of Multiplication

```
1  //Example No. 3_23
2  //Associative law
3  //Pg No. 58
4  clear ; close ; clc ;
5  x = 0.400000*10^40
6  y = 0.500000*10^70
7  z = 0.300000*10^(-30)
8  disp('In book they have considered the maximum
       exponent can be only 99, since 110 is greater
```

```
    than 99 the result is erroneous')
disp((x*y)*z,'xy_z = ','but in scilab the this value
    is much larger than 110 so we get a correct
    result ')
disp(x*(y*z),'x_yz = ')
```

Scilab code Exa 3.24 Distributive law of Arithmetic

```
//Example No. 3_24
//Distributive law
//Pg No. 58
clear ;close ;clc ;

x = 0.400000*10^1 ;
fx = 0.400000
Ex = 1
y = 0.200001*10^0 ;
z = 0.200000*10^0 ;
x_yz = x*(y-z)
x_yz = x_yz*10^6
x_yz = floor(x_yz) //considering only six
    significant digits
n = length(string(x_yz))
fx_yz = x_yz/10^n
Ex_yz = n - 6
x_yz = fx_yz *10^Ex_yz
disp(x_yz,'x_yz = ')

fxy = fx*y
fxy = fxy*10^6
fxy = floor(fxy) //considering only six significant
    digits
n = length(string(fxy))
fxy = fxy/10^n
Exy = n - 6
```

```
26 xy = fxy * 10^Exy
27
28 fxz = fx*z
29 fxz = fxz*10^6
30 fxz = floor(fxz) //considering only six significant
      digits
31 n = length(string(fxz))
32 fxz = fxz/10^n
33 Exz = n - 6
34 xz = fxz * 10^Exz
35
36 xy_xz = xy - xz
37 disp(xy_xz,'xy_xz = ')
```

Chapter 4

Approximations and Errors in Computing

Scilab code Exa 4.1 Greatest Precision

```
1 //Example No. 4_01
2 //Greatest precision
3 //Pg No. 63
4 clear ; close ; clc ;
5
6 a = '4.3201'
7 b = '4.32'
8 c = '4.320106'
9 na = length(a)-strindex(a,'.')
10 mprintf('\n %s has a precision of 10^-%i\n',a,na)
11 nb = length(b)-strindex(b,'.')
12 mprintf('\n %s has a precision of 10^-%i\n',b,nb)
13 nc = length(c)-strindex(c,'.')
14 mprintf('\n %s has a precision of 10^-%i\n',c,nc)
15 [n,e] = max(na,nb,nc)
16 if e ==1 then
17     mprintf('\n The number with highest precision is
          %s\n',a)
18 elseif e == 2
```

```
19      mprintf('\n The number with highest precision is
            %s\n',b)
20  else
21      mprintf('\n The number with highest precision is
            %s\n',c)
22  end
```

Scilab code Exa 4.2 Accuracy of Numbers

```
1  //Example No. 4_02
2  //Accuracy of numbers
3  //Pg No. 63
4  clear ;close ;clc ;
5
6  function n = sd(x)
7      nd = strindex(x,'.')  //position of point
8      num = str2code(x)
9      if isempty(nd) & num(length(x)) == 0 then
10         mprintf('Accuracy is not specified\n')
11         n = 0 ;
12     else
13         if num(1)>= 1 & isempty(nd) then
14             n = length(x)
15         elseif num(1) >= 1 & ~isempty(nd) then
16                 n = length(x) - 1
17         else
18             for i = 1:length(x)
19                 if num(i) >= 1 & num(i) <= 9 then
20                     break
21                 end
22             end
23             n = length(x)- i + 1
24         end
25     end
26 endfunction
```

```
27 a = '95.763'
28 na = sd(a)
29 mprintf('%s has %i significant digits\n',a,na)
30 b = '0.008472'
31 nb = sd(b)
32 mprintf('%s has %i significant digits.The leading or
       higher order zeros are only place holders\n',b,
     nb)
33 c = '0.0456000'
34 nc = sd(c)
35 mprintf('%s has %i significant digits\n',c,nc)
36 d = '36'
37 nd = sd(d)
38 mprintf('%s has %i significant digits\n',d,nd)
39 e = '3600'
40 sd(e)
41 f = '3600.00'
42 nf = sd(f)
43 mprintf('%s has %i significant digits\n',f,nf)
```

Scilab code Exa 4.3 Addition in Binary form

```
1 //Example No. 4_03
2 //Pg No. 64
3 clear ; close ; clc ;
4
5 a = 0.1
6 b = 0.4
7 for i = 1:8
8     afrac(i) = floor(a*2)
9     a = a*2 - floor(a*2)
10    bfrac(i) = floor(b*2)
11    b = b*2 - floor(b*2)
12 end
13 afrac_s = '0' + '.' + strcat(string(afrac)) //string
```

```
    form binary equivalent of a i.e 0.1
bfrac_s = '0' + '.' + strcat(string(bfrac))
mprintf('\n 0.1_10 = %s \n 0.4_10 = %s \n ', afrac_s
    , bfrac_s)
for j = 8:-1:1
    summ(j) = afrac(j) + bfrac(j)
    if summ(j) > 1 then
        summ(j) = summ(j)-2
        afrac(j-1) = afrac(j-1) + 1
    end
end
summ_dec = 0
for k = 8:-1:1
    summ_dec = summ_dec + summ(k)
    summ_dec = summ_dec*1/2
end
disp(summ_dec,'sum =')
disp('Note : The answer should be 0.5, but it is not
    so.This is due to the error in conversion from
   decimal to binary form.')
```

Scilab code Exa 4.4 Rounding off

```
//Example No. 4_04
//Rounding-Off
//Pg No. 66
clear ; close ; clc ;

fx = 0.7526
E =3
gx = 0.835
d = E - (-1)
//Chopping Method
Approx_x = fx*10^E
Err = gx*10^(E-d)
```

```
13 mprintf('\n Chooping Method : \n Approximate x = %.4
   f*10^%i \n Error = %.4f \n ',fx,E,Err)
14 //Symmetric Method
15 if gx >= 0.5 then
16     Err = (gx -1)*10^(-1)
17     Approx_x = (fx + 10^(-d))*10^E
18 else
19     Approx_x = fx*10^E
20     Err = gx * 10^(E-d)
21 end
22 mprintf('\n Symmetric Rounding :\n Approximate x = %
   .4f*10^%i \n Error = %.4f \n ',fx + 10^(-d),E,Err
   )
```

Scilab code Exa 4.5 Truncation Error

```
1 //Example No. 4_05
2 //Truncation Error
3 //Pg No. 68
4 clear ; close ; clc ;
5
6 x = 1/5
7 //When first three terms are used
8 Trunc_err = x^3/factorial(3) + x^4/factorial(4) + x
   ^5/factorial(5) + x^6/factorial(6)
9 mprintf('\n a) When first three terms are used \n
   Truncation error = %.6E \n ',Trunc_err)
10
11 //When four terms are used
12 Trunc_err = x^4/factorial(4) + x^5/factorial(5) + x
   ^6/factorial(6)
13 mprintf('\n b) When first four terms are used \n
   Truncation error = %.6E \n ',Trunc_err)
14
15 //When Five terms are used
```

```
16 Trunc_err = x^5/factorial(5) + x^6/factorial(6)
17 mprintf('\n c) When first five terms are used \n
     Truncation error = %.6E \n ',Trunc_err)
```

Scilab code Exa 4.6 Truncation Error

```
1 //Example No. 4_06
2 //Truncation Error
3 //Pg No. 68
4 clear ; close ; clc ;
5
6 x = -1/5
7 //When first three terms are used
8 Trunc_err = x^3/factorial(3) + x^4/factorial(4) + x
     ^5/factorial(5) + x^6/factorial(6)
9 mprintf('\n a) When first three terms are used \n
     Truncation error = %.6E \n ',Trunc_err)
10
11 //When four terms are used
12 Trunc_err = x^4/factorial(4) + x^5/factorial(5) + x
     ^6/factorial(6)
13 mprintf('\n b) When first four terms are used \n
     Truncation error = %.6E \n ',Trunc_err)
14
15 //When Five terms are used
16 Trunc_err = x^5/factorial(5) + x^6/factorial(6)
17 mprintf('\n c) When first five terms are used \n
     Truncation error = %.6E \n ',Trunc_err)
```

Scilab code Exa 4.7 Absolute and Relative Error

```
1 //Example No. 4_07
2 //Absolute and Relative Errors
```

```
3 //Pg No. 71
4 clear ; close ; clc ;
5
6 h_bu_t = 2945 ;
7 h_bu_a = 2950 ;
8 h_be_t = 30 ;
9 h_be_a = 35 ;
10 e1 = abs(h_bu_t - h_bu_a)
11 e1_r = e1/h_bu_t
12 e2 = abs(h_be_t - h_be_a)
13 e2_r = e2/h_be_t
14 mprintf('\n For Building : \n Absolute error, e1 =
   %i \n Relative error , e1_r = %.2f percent \n ',
   e1,e1_r*100)
15 mprintf('\n For Beam : \n Absolute error, e2 = %i \
   n Relative error , e2_r = %.2G percent \n ',e2,
   e2_r*100)
```

Scilab code Exa 4.8 Machine Epsilon

```
1 //Example No. 4_08
2 //Machine Epsilon
3 //Pg No. 72
4 clear ; close ; clc ;
5
6 deff('q = Q(p)','q = 1 + (p-1)*log10(2)' )
7 p = 24
8 q = Q(p)
9 mprintf('q = %.1f \n We can say that the computer
   can store numbers with %i significant decimal
   digits \n ',q,q)
```

Scilab code Exa 4.9 Propagation of Error

```
1 //Example No. 4_09
2 //Propagation of Error
3 //Pg No. 75
4 clear ; close ; clc ;
5
6 x = 0.1234*10^4
7 y = 0.1232*10^4
8 d = 4
9 er_x = 10^(-d + 1)/2
10 er_y = 10^(-d + 1)/2
11 ex = x*er_x
12 ey = y*er_y
13 ez = abs(ex) + abs(ey)
14 er_z = abs(ez)/abs(x-y)
15
16 mprintf('\n |er_x| <= %.2f o/o\n |er_y| <= %.2fo/o \
   n ex = %.3f \n ey = %.3f \n |ez| = %.3f \n |er_z|
    = %.2fo/o \n',er_x *100,er_y*100,ex,ey,ez,er_z
   *100)
```

Scilab code Exa 4.10 Errors in Sequence of Computations

```
1 //Example No. 4_10
2 //Errors in Sequence of Computations
3 //Pg No. 77
4 clear ; close ; clc ;
5
6 x_a = 2.35 ;
7 y_a = 6.74 ;
8 z_a = 3.45 ;
9 ex = abs(x_a)*10^(-3+1)/2
10 ey = abs(y_a)*10^(-3+1)/2
11 ez = abs(z_a)*10^(-3+1)/2
12 exy = abs(x_a)*ey + abs(y_a)*ex
13 ew = abs(exy) + abs(ez)
```

```
14 mprintf('\n ex = %.5f \n ey = %.5f \n ez = %.5f \n
     exy = %.5f \n ew = %.5f \n',ex,ey,ez,exy,ew)
```

Scilab code Exa 4.11 Addition of Chain of Numbers

```
1  //Example No. 4_11
2  //Addition of Chain of Numbers
3  //Pg No. 77
4  clear ; close ; clc ;
5
6  x = 9678 ;
7  y = 678 ;
8  z = 78 ;
9  d = 4 ;  //length of mantissa
10 fx = x/10^4
11 fy = y/10^4
12 fu = fx + fy
13 Eu = 4
14 if fu >= 1 then
15     fu = fu/10
16     Eu = Eu + 1
17 end
18 //since length of mantissa is only four we need to
     maintain only four places in decimal, so
19 fu = floor(fu*10^4)/10^4
20 u = fu * 10^Eu
21 w = u + z
22 n = length(string(w))
23 w = floor(w/10^(n-4))*10^(n-4) //To maintain length
     of mantissa = 4
24 disp(w,'w = ')
25 True_w = 10444
26 ew = True_w - w
27 er_w = (True_w - w)/True_w
28 disp(er_w,'er ,w = ',ew,'ew = ',True_w,'True w = ')
```

Scilab code Exa 4.12 Addition of Chain of Numbers

```
1  //Example No. 4_12
2  //Addition of chain Numbers
3  //Pg No. 77
4  clear ; close ; clc ;
5
6  x = 9678 ;
7  y = 678 ;
8  z = 78 ;
9  d = 4 ;  //length of mantissa
10 n = max(length( string(y) ) , length(string(z)))
11 fy = y/10^n
12 fz = z/10^n
13 fu = fy + fz
14 Eu = n
15 if fu >= 1 then
16     fu = fu/10
17     Eu = Eu + 1
18 end
19 u = fu * 10^Eu
20 n = max(length( string(x) ) , length(string(u)))
21 fu = u/10^4
22 fx = x/10^4
23 fw = fu + fx
24 Ew = 4
25 if fw >= 1 then
26     fw = fw/10
27     Ew = Ew + 1
28 end
29 //since length of mantissa is only four we need to
     maintain only four places in decimal, so
30 fw = floor(fw*10^4)/10^4
31 w = fw*10^Ew
```

```
32 disp(w,'w = ')
33 True_w = 10444
34 ew = True_w - w
35 er_w = (True_w - w)/True_w
36 disp(er_w,'er,w = ',ew,'ew = ',True_w,'True w = ')
```

Scilab code Exa 4.13 Theoritical Problem

```
1 //Example No. 4_13
2 //Pg No. 78
3 disp('Theoritical Problem')
4 disp('For Details go to page no. 78')
```

Scilab code Exa 4.14 Absolute and Relative Errors

```
1 //Example No. 4_14
2 //Absolute & Relative Errors
3 //Pg No. 79
4 clear ; close ; clc ;
5
6 xa = 4.000
7 deff('f = f(x)','f = sqrt(x) + x')
8 //Assuming x is correct to 4 significant digits
9 ex = 0.5 * 10^(-4 + 1)
10 df_xa = derivative(f,4)
11 ef = ex * df_xa
12 er_f = ef/f(xa)
13 mprintf('\n ex = %.0E \n df(xa) = %.2f \n ef = %.2E
    \n er,f = %.2E \n', ex,df_xa,ef,er_f)
```

Scilab code Exa 4.15 Error Evaluation

```
1 //Example No. 4_15
2 //Error Evaluation
3 //Pg No. 80
4 clear ; close ; clc ;
5
6 x = 3.00 ;
7 y = 4.00 ;
8 deff('f = f(x,y)','f = x^2 + y^2 ')
9 deff('df_x = df_x(x)','df_x = 2*x')
10 deff('df_y = df_y(y)','df_y = 2*y')
11 ex = 0.005
12 ey = 0.005
13 ef = df_x(x)*ex + df_y(y)*ey
14 disp(ef,'ef = ')
```

Scilab code Exa 4.16 Condition and Stability

```
1 //Example No. 4_16
2 //Condition and Stability
3 //Pg No. 82
4 clear ; close ; clc ;
5
6 C1 = 7.00 ;
7 C2 = 3.00 ;
8 m1 = 2.00 ;
9 m2 = 2.01 ;
10 x = (C1 - C2)/(m2 - m1)
11 y = m1*((C1 - C2)/(m2 - m1)) + C1
12 disp(y,'y = ',x,'x = ')
13 disp('Changing m2 from 2.01 to 2.005')
14 m2 = 2.005
15 x = (C1 - C2)/(m2 - m1)
16 y = m1*((C1 - C2)/(m2 - m1)) + C1
```

```
17 mprintf('\n x = %i \n y = %i \n From the above
     results we can see that for small change in m2
     results in almost 100 percent change in the
     values of x and y.Therefore, the problem is
     absolutely ill-conditioned \n',x,y)
```

Scilab code Exa 4.17 Theoritical Problem

```
1 //Example No. 4_17
2 //Pg No. 83
3 disp('Theoritical Problem')
4 disp('For Details go to page no. 83')
```

Scilab code Exa 4.18 Difference of Square roots

```
1 //Example No. 4_18
2 //Difference of Square roots
3 //Pg No. 84
4 clear ; close ; clc ;
5
6 x = 497.0 ;
7 y = 496.0 ;
8 sqrt_x = sqrt(497)
9 sqrt_y = sqrt(496)
10 nx = length( string( floor( sqrt_x ) ) )
11 ny = length( string( floor( sqrt_y ) ) )
12 sqrt_x = floor(sqrt_x*10^(4-nx))/10^(4-nx)
13 sqrt_y = floor(sqrt_y*10^(4-ny))/10^(4-ny)
14 z1 = sqrt_x - sqrt_y
15 disp(z1,'z = sqrt(x) - sqrt(y)')
16 z2 = ( x -y)/(sqrt_x + sqrt_y)
17 if z2 < 0.1 then
18     z2 = z2*10^4
```

```
19      nz = length(string(floor(z2)))
20      z2 = floor(z2*10^(4-nz))/10^(8-nz)
21  end
22  disp( z2 , 'z = ( x-y )/( sqrt(x) + sqrt(y) )' )
```

Scilab code Exa 4.19 Theoritical Problem

```
1  //Example No. 4_19
2  //Pg No. 84
3  disp('Theoritical Problem')
4  disp('For Details go to page no. 84')
```

Scilab code Exa 4.20 Theoritical Problem

```
1  //Example No. 4_20
2  //Pg No. 85
3  disp('Theoritical Problem')
4  disp('For Details go to page no. 85')
```

Scilab code Exa 4.21 Induced Instability

```
1  //Example 4_21
2  //Pg No. 85
3  clear ; close ; clc ;
4
5  x = -10
6  T_act(1) = 1
7  T_trc(1) = 1
8  e_x_cal = 1
9  for i = 1:100
```

```
10      T_act(i+1) = T_act(i)*x/i
11      T_trc(i+1) = floor(T_act(i+1)*10^5)/10^5
12      TE(i) = abs(T_act(i+1)-T_trc(i+1))
13      e_x_cal = e_x_cal + T_trc(i+1)
14  end
15  e_x_act = exp(-10)
16  disp(e_x_act,'actual e^x = ',e_x_cal,'calculated e^x
        using roundoff = ',sum(TE),'Truncation Error = '
    )
17  disp('Here we can see the difference between
    calculated e^x and actual e^x this is due to
    trucation error (which is greater than final
    value of e^x ), so the roundoff error totally
    dominates the solution')
```

Chapter 6

Roots of Nonlinear Equations

Scilab code Exa 6.1 Possible initial guess values for roots

```
1 //Example No. 6_01
2 //Possible Initial guess values for roots
3 //Pg No. 126
4
5 clear ; close ; clc ;
6
7 A = [ 2 ; -8 ; 2 ; 12]; // Coefficients of x terms
    in the decreasing order of power
8 n = size(A);
9 x1 = -A(2)/A(1);
10 disp(x1,'The largest possible root is  x1 =')
11 disp(x1,'No root can be larger than the value')
12
13 x = sqrt((A(2)/A(1))^2 - 2*(A(3)/A(1))^2);
14
15 printf('\n all real roots lie in the interval (-%f,
    %f)\n',x,x)
16 disp('We can use these two points as initial guesses
     for the bracketing methods and one of them for
    open end methods')
```

Scilab code Exa 6.02 Theoritical Problem

```
1 //Example No. 6_02
2 //Pg No. 128
3 disp('Theoritical Problem')
4 disp('For Details go to page no. 128')
```

Scilab code Exa 6.3 Evaluating Polynomial using Horners rule

```
1 //Example No. 6_03
2 //Evaluating Polynomial using Horner's rule
3 //Pg No.
4 clear ; close ; clc ;
5
6 //Coefficients of x terms in the increasing order of
      power
7 A = [ 6 ; 1 ; -4 ; 1];
8 x = 2
9 [n,c] = size(A) ;
10 p(n) = A(n)
11 disp(p(n), 'p(4) = ')
12 for i = 1:n-1
13     p(n-i) = p(n-i+1)*x + A(n-i)
14     printf('\n p(%i)= %i\n',n-i,p(n-i))
15 end
16 mprintf('\n f(%i) = p(1) = %i',x,p(1))
```

Scilab code Exa 6.4 Bisection Method

```
1  //Example No. 6_04
2  //Root of a Equation Using Bisection Method
3  //Pg No. 132
4
5  clear ; close ; clc ;
6
7  //Coefficients in increasing order of power of x
      starting from 0
8  A = [-10 -4 1];
9  disp('First finding the interval that contains a
      root,this can be done by using Eq 6.10')
10 xmax = sqrt((A(2)/A(3))^2 - 2*(A(1)/A(3)))
11 printf('\n Both the roots lie in the interval (-%i,
      %i) \n',xmax,xmax)
12 x = -6:6
13 p = poly(A,'x','c')
14 fx = horner(p,x);
15 for i = 1:12
16     if fx(1,i)*fx(1,i+1) < 0 then
17         break ;
18     end
19 end
20 printf('\n The root lies in the interval (%i,%i)\n',
      x(1,i),x(1,i+1))
21 x1 = x(1,i);
22 x2 = x(1,i+1);
23 f1 = fx(1,i);
24 f2 = fx(1,i+1);
25 err = abs((x2-x1)/x2) ;
26 while err > 0.0001
27 x0 = (x1 + x2)/2 ;
28 f0 = horner(p,x0);
29 if f0*f1 < 0  then
30     x2 = x0
31     f2 = f0
32 elseif f0*f2 < 0
33     x1 = x0
34     f1 = f0
```

```
35 else
36     break
37 end
38 printf('\n the root lies in the interval (%f,%f)\n',
      x1,x2);
39 err = abs((x2-x1)/x2);
40 end
41 printf('\nthe approximate root is %f\n',x0)
```

Scilab code Exa 6.5 False Position Method

```
1  //Example No. 6_05
2  //False Position Method
3  //Pg No. 139
4  clear ; close ; clc ;
5
6  //Coefficients of polynomial in increasing order of
      power of x
7  A = [-2  -1  1];
8  x1 = 1 ;
9  x2 = 3 ;
10 fx = poly(A,'x','c');
11 for i = 1:15
12     printf('Iteration No. %i \n',i);
13     fx1 = horner(fx,x1);
14     fx2 = horner(fx,x2);
15     x0 = x1 - fx1*(x2-x1)/(fx2-fx1)
16     printf('x0 = %f \n',x0);
17     fx0 = horner(fx,x0);
18     if fx1*fx0 < 0 then
19         x2 = x0 ;
20     else
21         x1 = x0 ;
22     end
23 end
```

Scilab code Exa 6.06 Theoritical Problem

```
1 //Example No. 6_06
2 //Pg No. 146
3 disp('Theoritical Problem')
4 disp('For Details go to page no. 146')
```

Scilab code Exa 6.7 Newton Raphson Method

```
1 //Example No. 6_07
2 //Root of the Equation using Newton Raphson Method
3 //Pg No. 147
4 clear ; close ; clc ;
5
6 //Coefficients of polynomial in increasing order of
     power of x
7 A = [ 2  -3  1];
8 fx = poly(A,'x','c');
9 dfx = derivat(fx);
10
11 x(1) = 0 ;
12 for i = 1:10
13     f(i)  = horner(fx,x(i));
14     if f(i)~= 0 then
15         df(i) = horner(dfx,x(i));
16         x(i+1) = x(i) - f(i)/df(i) ;
17         printf('x%i = %f\n',i+1,x(i+1));
18     else
19         printf('Since f(%f) = 0, the root closer to
               the point x = 0 is %f \n',x(i),x(i) );
20         break
```

```
21      end
22  end
```

Scilab code Exa 6.8 Newton Raphson Method

```
1  //Example No. 6_08
2  //Root of the Equation using Newton Raphson Method
3  //Pg No. 151
4  clear ; close ; clc ;
5  //Coefficients of polynomial in increasing order of
       power of x
6  A = [ 6  1  -4  1 ];
7  fx = poly(A,'x','c');
8  dfx = derivat(fx);
9
10 x(1) = 5.0 ;
11 for i = 1:6
12      f(i)  = horner(fx,x(i));
13      if f(i)~= 0 then
14          df(i) = horner(dfx,x(i));
15          x(i+1) = x(i) - f(i)/df(i) ;
16          printf('x%i = %f\n',i+1,x(i+1));
17      end
18 end
19 disp('From the results we can see that number of
       correct digits approximately doubles with each
       iteration')
```

Scilab code Exa 6.9 Secant Method

```
1  //Example No. 6_09
2  //Root of the equation using SECANT Method
3  //Pg No. 153
```

```
clear ; close ; clc ;

//Coefficients of polynomial in increasing order of
   power of x
A = [ -10  -4  1];
x1 = 4 ;
x2 = 2 ;
fx = poly(A,'x','c')
for i = 1:6
    printf('\n For Iteration No. %i\n',i)
    fx1 = horner(fx,x1);
    fx2 = horner(fx,x2);
    x3 = x2 - fx2*(x2-x1)/(fx2-fx1) ;
    printf('\n x1 = %f\n x2 = %f \n fx1 = %f \n fx2
       = %f \n x3 = %f \n',x1,x2,fx1,fx2,x3) ;
    x1 = x2;
    x2 = x3;
end
disp('This can be still continued further for
   accuracy')
```

Scilab code Exa 6.10 Theoritical Problem

```
//Example No. 6_10
//Pg No. 155
disp('Theoritical Problem')
disp('For Details go to page no. 155')
```

Scilab code Exa 6.11 Fixed Point Method

```
//Example No. 6_11
//Fixed point method
//Pg No. 161
```

```
clear ; close ; clc ;

//Coefficients of polynomial in increasing order of
   power of x
A = [ -2  1  1 ];
B = [ 2 0 -1 ];
gx = poly(B,'x','c');
x(1) = 0 ;//initial guess x0 = 0
for i = 2:10
    x(i) = horner(gx,x(i-1));
    printf('\n x%i = %f\n',i-1,x(i))
    if (x(i)-x(i-1)) == 0 then
        printf('\n%f is root of the equation,since
           x%i - x%i = 0 \n',x(i),i-1,i-2)
        break
    end
end
//Changing initial guess x0 = -1
x(1) = -1 ;
for i = 2:10
    x(i) = horner(gx,x(i-1));
    printf('\nx%i = %f\n',i-1,x(i))
    if (x(i)-x(i-1)) == 0 then
        printf('\n %f is root of the equation,since
           x%i - x%i = 0',x(i),i-1,i-2)
        break
    end
end
```

Scilab code Exa 6.12 Fixed Point Method

```
//Example No. 6_12
//Fixed point method
//Pg No. 162
clear ; close ; clc ;
```

```
5
6  A = [ -5   0   1 ];
7  funcprot(0);
8  deff('x = g(x)','x = 5/x');
9  x(1) = 1 ;
10 printf('\n x0 = %f \n',x(1));
11 for i = 2:5
12     x(i) = feval(x(i-1),g);
13     printf(' x%i = %f \n',i-1,x(i))
14 end
15 //Defining g(x) in different way
16 deff('x = g(x)','x = x^2 + x - 5');
17 x(1) = 0;
18 printf('\n x0 = %f \n',x(1));
19 for i = 2:5
20     x(i) = feval(x(i-1),g);
21     printf(' x%i = %f \n',i-1,x(i))
22 end
23 //Third form of g(x)
24 deff('x = g(x)',' x = (x + 5/x)/2 ');
25 x(1) = 1;
26 printf('\n x0 = %f \n',x(1));
27 for i = 2:7
28     x(i) = feval(x(i-1),g);
29     printf(' x%i = %f \n',i-1,x(i))
30 end
```

Scilab code Exa 6.13 Fixed Point Method for non linear equations

```
1 //Example No. 6_13
2 //Solving System of non-linear equations using FIXED
      POINT METHOD
3 //Pg No. 169
4 clear ; close ; clc ;
5
```

```
6  printf(' x^2 - y^2 = 3 \n x^2 + x*y \n');
7  deff('x = f(x,y)','x = y + 3/(x+y)') ;
8  deff('y = g(x)','y = (6-x^2)/x') ;
9  x(1) = 1 ;
10 y(1) = 1 ;
11 printf('\n x0 = %f \n y0 = %f \n',x(1),y(1));
12 for i = 2:4
13     x(i) = feval(x(i-1),y(i-1),f);
14     y(i) = feval(x(i-1),g);
15     printf('\n x%i = %f \n y%i = %f \n',i-1,x(i),i
          -1,y(i));
16 end
```

Scilab code Exa 6.14 Newton Raphson Method for Non linear equations

```
1  //Example No. 6_14
2  //Solving System of Non-linear equations using
      Newton Raphson Method
3  //Pg No. 172
4  clear ; close ; clc ;
5
6  printf('x^2 + x*y = 6 \n x^2 - y^2 = 3 \n');
7  deff('f = F(x,y)','f = x^2 + x*y - 6' );
8  deff('g = G(x,y)','g = x^2 - y^2 -3');
9  deff('f1 = dFx(x,y)','f1 = 2*x + y');
10 deff('f2 = dFy(x,y)','f2 = y');
11 deff('g1 = dGx(x,y)','g1 = 2*x ');
12 deff('g2 = dGy(x,y)','g2 = -2*y');
13 x(1) = 1 ;
14 y(1) = 1 ;
15
16 for i = 2:3
17     Fval = feval(x(i-1),y(i-1),F);
18     Gval = feval(x(i-1),y(i-1),G);
19     f1 = feval(x(i-1),y(i-1),dFx);
```

```
20      f2 = feval(x(i-1),y(i-1),dFy);
21      g1 = feval(x(i-1),y(i-1),dGx);
22      g2 = feval(x(i-1),y(i-1),dGy);
23      D =  f1*g2 - f2*g1 ;
24
25      x(i) = x(i-1) - (Fval*g2 - Gval*f2)/D ;
26      y(i) = y(i-1) - (Gval*f1 - Fval*g1)/D ;
27      printf('\n x%i = %f \n y%i = %f \n',i-1,x(i),i
           -1,y(i))
28
29  end
```

Scilab code Exa 6.15 Synthetic Division

```
1  //Example No. 6_15
2  //Synthetic Division
3  //Pg No. 176
4  clear ; close ; clc ;
5
6  a = [-9 15 -7 1];
7  b(4) = 0 ;
8  for i = 3:-1:1
9      b(i) = a(i+1) + b(i+1)*3
10     printf('b%i = %f\n',i,b(i))
11 end
12     disp(poly(b,'x','c'),'Thus the polynomial is')
```

Scilab code Exa 6.16 Bairstow Method for Factor of polynomial

```
1  //Example No. 6_16
2  //Quadratic factor of a polynomial using Bairstow's
      Method
3  //Pg No. 187
```

```
clear ; close ; clc ;

a = [ 10 1 0 1];
n = length(a);
u = 1.8 ;
v = -1 ;

b(n) = a(n);
b(n-1) = a(n-1) + u*b(n);
c(n) = 0 ;
c(n-1) = b(n);

for i = n-2:-1:1
    b(i) = a(i) + u*b(i+1) + v*b(i+2) ;
    c(i) = b(i+1) + u*c(i+1) + v*c(i+2) ;
end
for i = n:-1:1
    printf('b%i = %f \n',i-1,b(i))
end
for i = n:-1:1
    printf('c%i = %f \n',i-1,b(i))
end

D = c(2)*c(2) - c(1)*c(3) ;
du = -1*(b(2)*c(2) - c(1)*c(3))/D ;
dv = -1*(b(1)*c(2) - b(2)*c(1))/D ;
u = u + du ;
v = v + du ;
printf('\n D = %f \n du = %f \n dv = %f \n u = %f\n
   v = %f \n',D,du,dv,u,v)
```

Scilab code Exa 6.17 Mullers Method for Leonards equation

```
//Example No. 6_17
//Solving Leonard's equation using MULLER'S Method
```

```
3  //Pg No. 197
4  clear ; close ; clc ;
5
6  deff('y = f(x)','y = x^3 + 2*x^2 + 10*x - 20') ;
7  x1 = 0 ;
8  x2 = 1 ;
9  x3 = 2 ;
10 for i = 1:10
11     f1 = feval(x1,f) ;
12     f2 = feval(x2,f) ;
13     f3 = feval(x3,f) ;
14     h1 = x1-x3 ;
15     h2 = x2-x3 ;
16     d1 = f1 - f3 ;
17     d2 = f2 - f3 ;
18     D = h1*h2*(h1-h2);
19     a0 = f3 ;
20     a1 = (d2*h1^2 - d1*h2^2)/D ;
21     a2 = (d1*h2 - d2*h1)/D ;
22     if abs(-2*a0/( a1 + sqrt( a1^2 - 4*a0*a2 ) )) <
          abs( -2*a0/( a1 - sqrt( a1^2 - 4*a0*a2 ) ))
23         h4 = -2*a0/(a1 + sqrt(a1^2 - 4*a0*a2));
24     else
25         h4 = -2*a0/(a1 - sqrt(a1^2 - 4*a0*a2))
26     end
27     x4 = x3 + h4 ;
28     printf('\n x1 = %f\n x2 = %f\n x3 = %f\n f1 = %f
          \n f2 = %f\n f3 = %f\n h1 = %f\n h2 = %f\n d1
           = %f\n d2 = %f\n a0 = %f\n a1 = %f\n a2 = %f
          \n h4 = %f\n x4 = %f\n ',x1,x2,x3,f1,f2,f3,h1
          ,h2,d1,d2,a0,a1,a2,h4,x4) ;
29     relerr = abs((x4-x3)/x4);
30     if relerr <= 0.00001
31         printf('root of the polynomial is x4 = %f',
              x4);
32         break
33     end
34     x1 = x2 ;
```

```
35        x2 = x3 ;
36        x3 = x4 ;
37
38
39  end
```

Chapter 7

Direct Solutions of Linear Equations

Scilab code Exa 7.1 Elimination Process

```
1  //Example No. 7_01
2  //Elimination Process
3  //Pg No. 211
4
5  clear ; close ; clc ;
6
7  A = [3 2 1 10; 2 3 2 14 ; 1 2 3 14];
8  A(2,:) = A(2,:) - A(1,:)*A(2,1)/A(1,1)
9  A(3,:) = A(3,:) - A(1,:)*A(3,1)/A(1,1)
10 disp(A)
11
12 A(3,:) = A(3,:) - A(2,:)*A(3,2)/A(2,2)
13 disp(A)
14
15 z = A(3,4)/A(3,3)
16 y = (A(2,4) - A(2,3)*z)/A(2,2)
17 x = (A(1,4) - A(1,2)*y - A(1,3)*z)/A(1,1)
18 disp(x,'x = ',y,'y = ',z,'z = ')
```

Scilab code Exa 7.2 Basic Gauss Elimination

```
1 //Example No. 7_02
2 //Basic Gauss Elimination
3 //Pg No. 214
4 clear ; close ; clc ;
5
6 A = [  3  6  1  ;  2  4  3  ;  1  3  2 ];
7 B = [16 13 9];
8 [ar1,ac1] = size(A);
9 Aug = [3 6 1 16 ; 2 4 3 13 ; 1 3 2 9]
10 for i = 2 : ar1
11     Aug(i,:) = Aug(i,:) - (Aug(i,1)/Aug(1,1))*Aug
       (1,:) ;
12 end
13 disp(Aug)
14 disp('since Aug(2,2) = 0 elimination is not possible
   ,so reordering the matrix')
15 Aug = Aug( [ 1 3 2],:);
16 disp(Aug)
17 disp('Elimination is complete and by back
   substitution the solution is\n')
18 disp('x3 = 1, x2 = 2 , x1 = 1 ')
```

Scilab code Exa 7.3 Gauss Elimination using Partial Pivoting

```
1 //Example No. 7_03
2 // Gauss Elimination using partial pivoting
3 // Pg No. 220
4 clear ; close ; clc ;
5
6 A = [   2  2  1   ;   4  2  3   ;   1  -1  1];
```

```
7 B = [  6   ;   4   ;  0  ];
8 [ ar , ac ] = size(A);
9 Aug = [   2  2  1  6   ;   4  2  3  4  ;  1  -1  1
     0 ];
10
11 for i = 1 : ar-1
12     [ p , m ] =  max(abs(Aug(i:ar,i)))
13     Aug(i:ar,:) = Aug([i+m-1  i+1:i+m-2  i  i+m:ar
          ],:);
14     disp(Aug)
15     for k = i+1 : ar
16         Aug(k,i:ar+1) = Aug(k,i:ar+1) - (Aug(k,i)/
              Aug(i,i))*Aug(i,i:ar+1);
17     end
18     disp(Aug)
19 end
20
21 //Back Substitution
22 X(ar,1) = Aug(ar,ar+1)/Aug(ar,ar)
23 for i = ar-1 : -1 : 1
24     X(i,1) = Aug(i,ar+1);
25     for j = ar : -1 : i+1
26         X(i,1) = X(i,1) - X(j,1)*Aug(i,j);
27     end
28     X(i,1) = X(i,1)/Aug(i,i);
29 end
30
31 printf('\n The solution can be obtained by back
     substitution \n x1 = %i \n x2 = %i \n x3 = %i \n'
     ,X(1,1),X(2,1),X(3,1))
```

Scilab code Exa 7.4 Gauss Jordan Elimination

```
1 //Example No. 7_04
2 //Gauss Jordan Elimination
```

```
3 //Pg No. 228
4 clear ; close ; clc ;
5
6 A = [  2  4  -6 ; 1  3  1 ; 2  -4  -2  ];
7 B = [  -8  ;  10  ;  -12  ];
8 [ar,ac] = size(A);
9 Aug = [  2  4  -6  -8  ;  1  3  1  10  ;  2  -4  -2
      -12  ];
10 disp(Aug)
11
12 for i = 1 : ar
13     Aug(i,i:ar+1) = Aug(i,i:ar+1)/Aug(i,i) ;
14     disp(Aug)
15     for k = 1 : ar
16         if k ~= i then
17             Aug(k,i:ar+1) = Aug(k,i:ar+1) - Aug(k,i)
                  *Aug(i,i:ar+1);
18         end
19     end
20     disp(Aug)
21 end
```

Scilab code Exa 7.5 DoLittle LU Decomposition

```
1 //Example No. 7_05
2 //DoLittle LU Decomposition
3 //Pg No. 234
4
5 clear ; close ; clc ;
6
7 A = [ 3  2  1  ;  2  3  2  ;  1  2  3  ];
8 B = [ 10  ;  14  ;  14  ];
9 [n , n] = size(A);
10
11 for i = 2:n
```

```
12      U(1,:) = A(1,:);
13      L(i,i) = 1 ;
14      if i ~= 1 then
15           L(i,1) = A(i,1)/U(1,1);
16      end
17  end
18
19  for j = 2:n
20      for i = 2:n
21
22          if i <= j then
23              U(i,j) = A(i,j);
24              for k = 1:i-1
25                  U(i,j) = U(i,j) - L(i,k)*U(k,j);
26              end
27              printf('\nU(%i,%i) = %f \n',i,j,U(i,j))
28
29          else
30              L(i,j) = A(i,j)
31              for k = 1:j-1
32                  L(i,j) = L(i,j) - L(i,k)*U(k,j);
33              end
34              L(i,j) = L(i,j)/U(j,j)  ;
35              printf('\n L(%i,%i) = %f \n',i,j,L(i,j))
36          end
37      end
38  end
39  disp(U,'U = ')
40  disp(L,'L = ')
41
42  //Forward Substitution
43  for i = 1:n
44      z(i,1) = B(i,1);
45      for j = 1:i-1
46          z(i,1) = z(i,1) - L(i,j)*z(j,1);
47      end
48      printf('\n z(%i) = %f \n',i,z(i,1))
49  end
```

```
50
51 //Back Substitution
52 for i = n : -1 : 1
53     x(i,1) = z(i,1);
54     for j = n : -1 : i+1
55         x(i,1) = x(i,1) - U(i,j)*x(j,1);
56     end
57     x(i,1) = x(i,1)/U(i,i);
58     printf('\n x(%i) = %f \n',i,x(i,1))
59 end
```

Scilab code Exa 7.6 Choleskys Factorisation

```
1  //Example No. 7_06
2  //Cholesky's Factorisation
3  //Pg No. 242
4
5  clear ; close ; clc ;
6
7  A = [ 1 2 3 ; 2 8 22  ; 3 22 82 ];
8  [n,n] = size(A);
9
10 for i = 1:n
11     for j = 1:n
12         if i == j then
13             U(i,i) = A(i,i)
14             for k = 1:i-1
15                 U(i,i) = U(i,i)-U(k,i)^2 ;
16             end
17             U(i,i) = sqrt(U(i,i));
18          elseif i < j
19              U(i,j) = A(i,j)
20              for k = 1:i-1
21                  U(i,j) = U(i,j) - U(k,i)*U(k,j);
22              end
```

```
23                 U(i,j) = U(i,j)/U(i,i)
24             end
25         end
26 end
27 disp(U)
```

Scilab code Exa 7.7 Ill Conditioned Systems

```
1 //Example No. 7_07
2 //Ill-Conditioned Systems
3 //Pg No. 245
4
5 clear ; close ; clc ;
6
7 A = [ 2  1 ; 2.001  1];
8 B = [ 25 ; 25.01 ];
9 x(1) = (25 - 25.01)/(2 - 2.001);
10 x(2) = (25.01*2 - 25*2.001)/(2*1 - 2.001*1);
11 printf('\n x(1) = %f \n x(2) = %f \n',x(1),x(2))
12 x(1) = (25 - 25.01)/(2 - 2.0005);
13 x(2) = (25.01*2 - 25*2.0005)/(2*1 - 2.0005*1);
14 printf('\n x(1) = %f \n x(2) = %f \n',x(1),x(2))
15 r = A*x-B
16 disp(x)
17 disp(r)
```

Chapter 8

Iterative Solution of Linear Equations

Scilab code Exa 8.1 Gauss Jacobi Iteration Method

```
1 //Example No. 8_01
2 //Gauss Jacobi
3 //Page No. 254
4 clear ; close ; clc ;
5
6 A = [ 2  1  1 ; 3  5  2 ; 2  1  4];
7 B = [ 5  ;  15  ;  8];
8 x1old = 0 ,x2old = 0 , x3old = 0 //intial assumption
      of x1,x2 & x3
9
10 disp('x1 = (5 - x2 - x3)/2 ')
11 disp('x2 = (15 - 3x1 - 2x3)/5 ')
12 disp('x3 = (8 - 2x1 - x2)/4')
13
14 for i = 1:4
15     printf('\n Iteration Number : %d\n',i)
16
17     x1 = (5 - x2old - x3old)/2 ;
18     x2 = (15 - 3*x1old - 2*x3old)/5 ;
```

```
19      x3 = (8 - 2*x1old - x2old)/4 ;
20
21      printf(' \n x1 = %f\n x2 = %f\n x3 = %f\n',x1,x2
           ,x3)
22
23      x1old = x1;
24      x2old = x2;
25      x3old = x3;
26
27  end
```

Scilab code Exa 8.2 Gauss Seidel Iterative Method

```
1  //Example No. 8_02
2  //Gauss Seidel
3  //Page No.261
4  clear ; close ; clc ;
5
6  A = [ 2  1  1 ; 3  5  2 ; 2  1  4];
7  B = [ 5  ;  15  ;  8];
8  x1old = 0 ,x2old = 0 , x3old = 0 //intial assumption
9
10 disp('(x1 = 5 - x2 - x3)/2 ')
11 disp('(x2 = 15 - 3x1 - 2x3)/5 ')
12 disp('(x3 = 8 - 2x1 - x2)/4')
13
14 for i = 1:2
15
16      printf('\n Iteration Number : %d',i)
17
18      x1 = (5 - x2old - x3old)/2 ;
19      x1old = x1;
20      x2 = (15 - 3*x1old - 2*x3old)/5 ;
21      x2old = x2;
22      x3 = (8 - 2*x1old - x2old)/4 ;
```

```
23      x3old = x3;
24
25       printf(' \n x1 = %f\n x2 = %f\n x3 = %f\n',x1,
           x2,x3)
26
27  end
```

Scilab code Exa 8.3 Gauss Seidel Iterative Method

```
1  //Example No. 8_03
2  //Gauss Seidel
3  //page no. 269
4  clear ; close ; clc ;
5
6  A = [ 3 1 ; 1 -3]
7  B = [ 5 ; 5 ]
8
9  disp('Using a matrix to display the results after
       each iteration, first row represents initial
       assumption')
10 X(1,1) = 0 , X(1,2) = 0 ;//initial assumption
11
12 maxit = 1000;//Maximum number of iterations
13 err = 0.0003 ;
14
15 disp('x1 = (5-x2)/3');
16 disp('x2 = (x1 - 5)/3');
17
18 for i = 2:maxit
19
20      X(i,1) = (5 - X(i-1,2))/3 ;
21      X(i,2) = (X(i,1) - 5)/3 ;
22
23      //Error Calculations
24      err1 =abs((X(i,1) - X(i-1,1))/X(i,1))
```

```
25      err2 =abs((X(i,2)- X(i-1,2))/X(i,2))
26
27      //Terminating Condition
28      if err >= err1 & err >= err2  then
29          printf('The system converges to the solution
                ( %f , %f ) in %d iterations\n',X(i,1),X
               (i,2),i-1 )
30          break
31      end
32
33  end
34  //calcution of true error i.e. difference between
       final result and results from each iteration
35  trueerr1 = abs(X(:,1) - X(i,1)*ones(i,1)) ;
36  trueerr2 = abs(X(:,2) - X(i,2)*ones(i,1)) ;
37
38  //displaying final results
39  D = [  X  trueerr1   trueerr2] ;
40  disp(D)
```

Scilab code Exa 8.4 Gauss Seidel Iterative Method

```
1  //Example No. 8_04
2  //Gauss Seidel
3  //Page No.261
4  clear ; close ; clc ;
5
6  A = [ 1 -3 ; 3 1 ];
7  B = [ 5  ;  5 ];
8  x1old = 0 ,x2old = 0  //intial assumption
9
10 disp('x1 = 5 + 3*x2 ')
11 disp('x2 = 5 - 3*x1 ')
12
13 for i = 1:3
```

```
14
15      x1 = 5 + 3*x2old ;
16      x1old = x1;
17      x2 = 5 - 3*x1old ;
18      x2old = x2;
19
20       printf('\n Iteration : %i  x1 = %i and x2 = %i\
           n',i,x1,x2)
21
22  end
23  disp('It is clear that the process do not converge
       towards the solution, rather it diverges.')
```

Chapter 9

Curve Fitting Interpolation

Scilab code Exa 9.1 Polynomial Forms

```
1 //Example No. 9_01
2 //Pg No.277
3 clear ; close ; clc ;
4
5 printf('solving linear equations \n a0 + 100a1 = 3/7
      \n a0 + 101a1 = -4/7 \n we get,\n');
6 C = [ 1 100 ; 1 101]
7 p = [ 3/7 ; -4/7]
8 a = C\p
9 printf('\n a0 = %f \n a1 = %f \n',a(1),a(2));
10 x = poly(0,'x') ;
11 px = a(1) + a(2)*x
12 p100 = horner(px,100)
13 p101 = horner(px,101)
14 printf('\n p(100) = %f \n p(101) = %f\n',p100,p101)
```

Scilab code Exa 9.2 Shifted Power form

```
1  //Example No. 9_02
2  //Page No. 278
3  clear ; close ; clc ;
4
5  C = [ 1 100-100 ; 1 101-100]
6  p = [ 3/7 ; -4/7]
7  a = C\p
8  printf('\n a0 = %f \n a1 = %f \n',a(1),a(2));
9  x = poly(0,'x') ;
10 px = a(1) + a(2)*(x - 100)
11 p100 = horner(px,100)
12 p101 = horner(px,101)
13 printf('\n p(100) = %f \n p(101) = %f\n',p100,p101)
```

Scilab code Exa 9.3 Linear Interpolation

```
1  //Example No. 9_03
2  //Page No. 280
3  clear ; close ; clc ;
4
5  x = 1:5
6  f = [1 1.4142 1.7321 2 2.2361]
7  n = 2.5
8  for i = 1:5
9      if n <= x(i) then
10         break ;
11     end
12 end
13 printf('%f lies between points %i and %i',n,x(i-1),x
       (i))
14 f2_5 = f(i-1) + ( n - x(i-1) )*( f(i) - f(i-1) )/( x
       (i) - x(i-1) )
15 err1 = 1.5811 - f2_5
16 disp(f2_5,'f(2.5) = ')
17 disp(err1,'error1 = ')
```

```
18 disp('The correct answer is 1.5811.The difference
     between results is due to use of a linear model
     to a nonlinear use')
19 disp('repeating the procedure using x1 = 2 and x2 =
     4')
20 x1 = 2
21 x2 = 4
22 f2_5 = f(x1) + ( 2.5 - x1 )*( f(x2) - f(x1) )/( x2 -
      x1 )
23 err2 = 1.5811 - f2_5
24 disp(err2,'error2 = ')
25 disp(f2_5,'f(2.5) = ')
26 disp('NOTE- The increase in error due to the
     increase in the interval between the
     interpolating data')
```

Scilab code Exa 9.4 Lagrange Interpolation

```
1  //Example No. 9_04
2  //Lagrange Interpolation- Second order
3  //Pg No. 282
4  clear ; close ; clc ;
5
6  X = [ 1 2 3 4 5]
7  Fx = [ 1 1.4142 1.7321 2 2.2361];
8  X = X(2:4)
9  Fx = Fx(2:4)
10 x0 = 2.5
11 x = poly(0,'x')
12 p = 0
13 for i = 1:3
14     L(i) = 1
15     for j = 1:3
16         if j == i then
17             continue ;
```

```
18          else
19              L(i) = L(i)*( x - X(j) )/( X(i) - X(j) )
                    ;
20          end
21      end
22      p = p + Fx(i)*L(i)
23  end
24  L0 = horner(L(1),2.5) ;
25  L1 = horner(L(2),2.5) ;
26  L2 = horner(L(3),2.5) ;
27  p2_5 = horner(p,2.5) ;
28  printf('For x = 2.5 we have,\n L0(2.5) = %f \n L1
    (2.5) = %f \n L2(2.5) = %f \n p(2.5) = %f \n',L0,
    L1,L2,p2_5)
29
30  err = sqrt(2.5) - p2_5 ;
31  printf('The error is %f',err);
```

Scilab code Exa 9.5 Lagrange Interpolation

```
1  //Example No. 9_05
2  //Lagrange Interpolation
3  //Pg No. 283
4  clear ; close ; clc ;
5
6  i = [ 0 1 2 3 ]
7  X = [ 0 1 2 3 ]
8  Fx = [ 0 1.7183 6.3891 19.0855 ]
9  x = poly(0,'x');
10 n = 3 //order of lagrange polynomial
11 p = 0
12 for i = 1:n+1
13     L(i) = 1
14     for j = 1:n+1
15         if j == i then
```

```
16              continue ;
17          else
18              L(i) = L(i)*( x - X(j) )/( X(i) - X(j) )
                    ;
19          end
20      end
21      p = p + Fx(i)*L(i)
22  end
23  disp("The Lagrange basis polynomials are")
24  for i = 1:4
25          disp(string(L(i)))
26  end
27  disp("The interpolation polynomial is ")
28  disp(string(p))
29  disp('The interpolation value at x = 1.5 is ' )
30  p1_5 = horner(p,1.5);
31  e1_5 = p1_5 + 1 ;
32  disp(e1_5,'e^1.5 = ',p1_5);
```

Scilab code Exa 9.6 Newton Interpolation

```
1  //Example No. 9_06
2  //Newton Interpolation - Second order
3  //Pg No. 288
4  clear ; close ; clc ;
5
6  i = [ 0 1 2 3]
7  X = 1:4
8  Fx = [ 0 0.3010 0.4771 0.6021]
9  X = 1:3
10 Fx = Fx(1:3)
11 x = poly(0,'x');
12 A = Fx'
13 for i = 2:3
14     for j = 1:4-i
```

```
15          A(j,i) = ( A(j+1,i-1)-A(j,i-1) )/(X(j+i-1)-X
               (j)) ;
16      end
17  end
18  printf('The coefficients of the polynomial are,\n a0
       = %.4G \n a1 = %.4G \n a2 = %.4G \n',A(1,1),A
       (1,2),A(1,3))
19  p = A(1,1);
20  for i = 2:3
21      p = p +A(1,i)* prod(x*ones(1,i-1) - X(1:i-1));
22  end
23  disp(string(p))
24  p2_5 = horner(p,2.5)
25  printf('p(2.5) = %.4G \n',p2_5)
```

Scilab code Exa 9.7 Newton Divided Difference Interpolation

```
1  //Example No. 9_07
2  //Newton Divided Difference Interpolation
3  //Pg No. 291
4  clear ; close ; clc ;
5
6  i = 0:4
7  X = 1:5
8  Fx = [ 0 7 26 63 124];
9  x = poly(0,'x');
10 A = [ i' X' Fx']
11 for i = 4:7
12       for j = 1:8-i
13          A(j,i) = ( A(j+1,i-1)-A(j,i-1) )/(X(j+i-3)-X
               (j)) ;
14      end
15 end
16 disp(A)
17 p = A(1,3);
```

```
18 p1_5(1) = p ;
19 for i = 4:7
20     p = p +A(1,i)* prod(x*ones(1,i-3) - X(1:i-3));
21     p1_5(i-2) = horner(p,1.5);
22 end
23 printf('p0(1.5) = %f \n p1(1.5) = %f \n p2(1.5) = %f
       \n p3(1.5) = %f \n p4(1.5) = %f \n',p1_5(1),p1_5
   (2),p1_5(3),p1_5(4),p1_5(5));
24 disp(p1_5(5),'The function value at x = 1.5 is')
```

Scilab code Exa 9.8 Newton Gregory Forward Difference Formula

```
1  //Example No. 9_08
2  //Newton-Gregory forward difference formula
3  //Pg No. 297
4  clear ; close ; clc ;
5
6  X = [ 10 20 30 40 50]
7  Fx = [ 0.1736 0.3420 0.5000 0.6428 0.7660]
8  x = poly(0,'x');
9  A = [X' Fx'];
10 for i = 3:6
11      A(1:7-i,i) = diff(A(1:8-i,i-1))
12 end
13 disp(A)
14 x0 = X(1);
15 h = X(2) - X(1) ;
16 x1 = 25
17 s = (x1 - x0)/h ;
18 p(1) = Fx(1);
19 for j = 1:4
20     p(j+1) = p(j) + prod(s*ones(1,j)-[0:j-1])*A(1,j
          +2)/factorial(j)
21 end
22 printf('p1(s) = %.4G \n p2(s) = %.4G \n p3(s) = %.4G
```

```
    \n p4(s) = %.4G \n',p(2),p(3),p(4),p(5))
23 printf(' Thus sin(%d) = %.4G \n ',x1,p(5))
```

Scilab code Exa 9.9 Newton Backward Difference Formula

```
1  //Example No. 9_09
2  //Newton Backward difference formula
3  //Pg No. 299
4  clear ;close ;clc ;
5
6  X = [ 10 20 30 40 50]
7  Fx = [ 0.1736 0.3420 0.5000 0.6428 0.7660]
8  x = poly(0,'x');
9  A = [X' Fx'];
10 for i = 3:6
11      A(i-1:5,i) = diff(A(i-2:5,i-1))
12 end
13 disp(A);
14 xn = X(5);
15 h = 10 ;
16 xuk = 25;
17 s = (xuk - xn)/h ;
18 disp(s,'s = ');
19 p(1) = Fx(5)
20 for j = 1:4
21     p(j+1) = p(j) + prod(s*ones(1,j)-[0:j-1])*A(5,j
       +2)/factorial(j)
22 end
23 printf('\n\n p1(s) = %.4G \n p2(s) = %.4G \n p3(s) =
    %.4G \n p4(s) = %.4G \n',p(2),p(3),p(4),p(5))
24 printf(' Thus sin(%d) = %.4G \n ',xuk,p(5))
```

Scilab code Exa 9.10 Splines

```
1  //Example No. 9_10
2  //Splines
3  //Pg No. 301
4  clear ; close ; clc ;
5
6  x = poly(0,'x');
7  function isitspline(f1,f2,f3,x0,x1,x2,x3)
8      n1 = degree(f1),n2 = degree(f2),n3 = degree(f3)
9      n = max(n1,n2,n3)
10     f1_x1 = horner(f1,x1)
11     f2_x1 = horner(f2,x1)
12     f2_x2 = horner(f2,x2)
13     f3_x2 = horner(f3,x2)
14     if n ==1 & f1_x1 == f2_x1 & f2_x2 == f3_x2 then
15         printf('The piecewise polynomials are
               continuous and f(x) is a linear spline')
16     elseif f1_x1 == f2_x1 & f2_x2 == f3_x2
17         for i = 1:n-1
18             df1 = derivat(f1)
19             df2 = derivat(f2)
20             df3 = derivat(f3)
21             df1_x1 = horner(df1,x1)
22             df2_x1 = horner(df2,x1)
23             df2_x2 = horner(df2,x2)
24             df3_x2 = horner(df3,x2)
25             f1 = df1,  f2 = df2,  f3 = df3
26             if df1_x1 ~= df2_x1 | df2_x2 ~= df3_x2
                  then
27                 printf('The %ith derivative of
                      polynomial is not continuours',i)
28                 break
29             end
30         end
31         if i == n-1 & df1_x1 == df2_x1 & df2_x2 ==
              df3_x2 then
32             printf('The polynomial is continuous and
                   its derivatives from 1 to %i are
                   continuous, f(x) is a %ith degree
```

```
                polynomial',i,i+1)
33          end
34      else
35              printf('The polynomial is not continuous
                ')
36      end
37
38  endfunction
39  n = 4 , x0 = -1 , x1 = 0, x2 = 1 , x3 = 2
40  f1 = x+1 ;
41  f2 = 2*x + 1 ;
42  f3 = 4 - x ;
43  disp('case 1')
44  isitspline(f1,f2,f3,x0,x1,x2,x3)
45  n = 4, x0 = 0 , x1= 1 , x2 = 2 , x3 = 3
46  f1 = x^2 + 1 ;
47  f2 = 2*x^2 ;
48  f3 = 5*x - 2 ;
49  disp('case 2')
50  isitspline(f1,f2,f3,x0,x1,x2,x3)
51  n = 4, x0 = 0, x1 = 1, x2 = 2, x3 = 3
52  f1 = x,
53  f2 = x^2 - x + 1,
54  f3 = 3*x - 3
55  disp('case 3')
56  isitspline(f1,f2,f3,x0,x1,x2,x3)
```

Scilab code Exa 9.11 Cubic Spline Interpolation

```
1  //Example No. 9_11
2  //Cubic Spline Interpolation
3  //Pg No. 306
4  clear ; close ; clc ;
5
6  X = [ 4 9 16]
```

```
7  Fx = [ 2 3 4]
8  n = length(X)
9  h = diff(X)
10 disp(h)
11 x = poly(0,'x');
12 A(1) = 0;
13 A(n) = 0;
14
15 //Since we do not know only a1 = A(2) we just have
      one equation which can be solved directly without
       solving tridiagonal matrix
16 A(2) = 6*( ( Fx(3) - Fx(2) )/h(2) - ( Fx(2) - Fx(1)
      )/h(1) )/( 2*( h(1) + h(2) ) );
17 disp(A(2),'a1 = ');
18 xuk = 7;
19 for i = 1:n-1
20     if xuk <= X(i+1) then
21         break;
22     end
23 end
24 u = x*ones(1,2) - X(i:i+1)
25 s = (  A(2)*( u(i)^3 - ( h(i)^2 )*u(i))/6*h(i)  ) +
      ( Fx(i+1)*u(i)- Fx(i)*u(i+1) )/h(i);
26 disp(s,'s(x) = ');
27 s_7 = horner(s,xuk);
28 disp(s_7,'s(7)')
```

Scilab code Exa 9.12 Cubic Spline Interpolation

```
1 //Example No. 9_12
2 //Cubic Spline Interpolation
3 //Pg No. 313
4 clear ; close ;clc ;
5
6 X = 1:4 ;
```

```
Fx = [ 0.5 0.3333 0.25 0.2]
n = length(X)
h = diff(X)
disp(h)
x = poly(0,'x');
A(1) = 0;
A(n) = 0;
//Forming Tridiagonal Matrix
//take make diagonal below main diagonal be 1 , main diagonal is 2 and diagonal above main diagonal is 3
diag1 = h(2:n-2);
diag2 = 2*(h(1:n-2)+h(2:n-1));
diag3 = h(2:n-2);
TridiagMat = diag(diag1,-1)+diag(diag2)+diag(diag3,1)
disp(TridiagMat);
D = diff(Fx);
D = 6*diff(D./h);
disp(D)
A(2:n-1) = TridiagMat\D'
disp(A)
xuk = 2.5;
for i = 1:n-1
    if xuk <= X(i+1) then
        break;
    end
end
u = x*ones(1,2) - X(i:i+1)
s = (  A(i)*( h(i+1)^2*u(2) - u(2)^2 )/( 6*h(i+1) ) ) + (  A(i+1)*( u(1)^3 - ( h(i)^2 )*u(1))/6*h(i) ) + ( Fx(i+1)*u(1)- Fx(i)*u(2) )/h(i);
disp(s,'s(x) = ');
s2_5 = horner(s,xuk);
disp(s2_5,'s(2.5)')
```

Chapter 10

Curve Fitting Regression

Scilab code Exa 10.1 Fitting a Straight line

```
1  //Example No. 10_01
2  //Fitting a Straight Line
3  //Pg No. 326
4  clear ;close ;clc ;
5
6  x = poly(0,'x')
7  X = 1:5
8  Y = [ 3 4 5 6 8 ];
9  n = length(X);
10 b = ( n*sum(X.*Y) - sum(X)*sum(Y) )/( n*sum(X.*X) -
       (sum(X))^2 )
11 a = sum(Y)/n - b*sum(X)/n
12 disp(b,'b = ')
13 disp(a,'a = ')
14 y = a + b*x
```

Scilab code Exa 10.2 Fitting a Power Function Model to given data

```
1  //Example No. 10_02
2  //Fitting a Power-Function model to given data
3  //Pg No. 331
4  clear ;close ;clc ;
5
6  x = poly(0,'x');
7  X = 1:5
8  Y = [ 0.5 2 4.5 8 12.5 ]
9  Xnew = log(X)
10 Ynew = log(Y)
11 n = length(Xnew)
12 b = ( n*sum(Xnew.*Ynew) - sum(Xnew)*sum(Ynew) )/( n*
      sum(Xnew.*Xnew) - ( sum(Xnew) )^2 )
13 lna = sum(Ynew)/n - b*sum(Xnew)/n
14 a = exp(lna)
15 disp(b,'b = ')
16 disp(lna,'lna = ')
17 disp(a,'a = ')
18 printf('\n The power function equation obtained is \
      n y = %.4Gx^%.4G',a,b);
```

Scilab code Exa 10.3 Fitting a Straight line using Regression

```
1  //Example No. 10_03
2  //Pg No. 332
3  clear ;close ;clc ;
4
5  time = 1:4
6  T = [ 70 83 100 124 ]
7  t = 6
8  Fx = exp(time/4)
9  n = length(Fx)
10 Y = T ;
11 b = ( n*sum(Fx.*Y) - sum(Fx)*sum(Y) )/( n*sum(Fx.*Fx
      ) - (sum(Fx))^2 )
```

```
12 a = sum(Y)/n - b*sum(Fx)/n
13 disp(b,'b = ')
14 disp(a,'a = ')
15 printf('The relationship between T and t is \n T = %
   .4G*e^(t/4) + %.4G \n',b,a)
16 deff('T = T(t)','T = b*exp(t/4) + a ')
17 T_6 = T(6)
18
19 disp(T_6,'The temperature at t = 6 is')
```

Scilab code Exa 10.4 Curve Fitting

```
1  //Example No. 10_04
2  //Curve Fitting
3  //Pg NO. 335
4  clear ; close ; clc ;
5
6  x = 1:4 ;
7  y = [6 11 18 27 ];
8  n = length(x) //Number of data points
9  m = 2+1       //Number of unknowns
10 disp('Using CA = B form , we get')
11 for j = 1:m
12     for k = 1:m
13         C(j,k) = sum(x.^(j+k-2))
14     end
15     B(j) = sum( y.*( x.^( j-1 ) ) )
16 end
17 disp(B,'B = ',C,'C = ')
18 A = inv(C)*B
19 disp(A,'A = ')
20 printf('Therefore the least sqaures polynomial is\n
     y = %i + %i*x + %i*x^2 \n',A(1),A(2),A(3))
```

Scilab code Exa 10.5 Plane Fitting

```
1  //Example No. 10_05
2  //Plane Fitting
3  //Pg No. 342
4  clear ; close ; clc ;
5
6  x = 1:4
7  z = 0:3
8  y = 12:6:30
9  n = length(x) //Number of data points
10 m = 3         //Number of unknowns
11 G = [ ones(1,n) ; x ; z]
12 H = G'
13 C = G*H
14 B = y*H
15 D = C\B'
16 disp(C,B)
17 disp(D)
18 mprintf('\n The regression plane is \n y = %i + %f*x
       + %i*z ',D(1),D(2),D(3))
```

Chapter 11

Numerical Differentiation

Scilab code Exa 11.1 First order Forward Difference

```
1 //Example No. 11_01
2 //First order forward difference
3 //Pg No. 348
4 clear ;close ;clc ;
5
6 x = poly(0,"x");
7 deff('F = f(x)','F = x^2');
8 deff('DF = df(x,h)','DF = (f(x+h)-f(x))/h');
9 dfactual = derivat(f(x));
10 h = [0.2 ; 0.1 ; 0.05 ; 0.01 ]
11 for i = 1:4
12     y(i) = df(1,h(i));
13     err(i) = y(i) - horner(dfactual,1)
14 end
15 table = [h y err];
16 disp(table)
```

Scilab code Exa 11.2 Three Point Formula

```
1  //Example No. 11_02
2  //Three-Point Formula
3  //Pg No. 350
4  clear ;close ;clc ;
5
6  x = poly(0,"x");
7  deff('F = f(x)','F = x^2');
8  deff('DF = df(x,h)','DF = (f(x+h)-f(x-h))/(2*h)');
9  dfactual = derivat(f(x));
10 h = [0.2 ; 0.1 ; 0.05 ; 0.01 ]
11 for i = 1:4
12     y(i) = df(1,h(i));
13     err(i) = y(i) - horner(dfactual,1)
14 end
15 table = [h y err];
16 disp(table)
```

Scilab code Exa 11.3 Error Analysis

```
1  //Example No. 11_03
2  //Pg No. 353
3  close ;clear ;clc ;
4
5  x = 0.45;
6  deff('F = f(x)','F = sin(x)');
7  deff('DF = df(x,h)','DF = (f(x+h) - f(x))/h');
8  dfactual =  cos(x);
9  h = 0.01:0.005:0.04;
10 n = length(h);
11 for i = 1:n
12     y(i) = df(x,h(i))
13     err(i) = y(i) - dfactual ;
14 end
15 table = [ h' y err];
16 disp(table)
```

```
17 //scilab uses 16 significant digits so the bound for
       roundoff error is 0.5*10^(-16)
18 e = 0.5*10^(-16)
19 M2 = max(sin(x+h));
20 hopt = 2*sqrt(e/M2);
21 disp(hopt,'hopt = ',M2,'M2 = ')
```

Scilab code Exa 11.4 Approximate Second Derivative

```
1 //Example No. 11_04
2 //Approximate second derivative
3 //Pg No. 354
4 clear ;close ;clc ;
5
6 x = 0.75;
7 h = 0.01;
8 deff('F = f(x)','F = cos(x)');
9 deff('D2F = d2f(x,h)','D2F = ( f(x+h) - 2*f(x) + f(x
      -h) )/h^2 ');
10 y = d2f(0.75,0.01);
11 d2fexact = -cos(0.75)
12 err = d2fexact - y ;
13 disp(err,'err = ',d2fexact,'d2fexact = ',y,'y = ')
```

Scilab code Exa 11.5 Differentiation of Tabulated Data

```
1 //Example No. 11_05
2 //Differentiation of tabulated data
3 //Pg No. 358
4 clear ;close ;clc ;
5
6 T = 5:9 ;
7 s = [10  14.5  19.5  25.5  32 ];
```

```
h = T(2)-T(1);
n = length(T)
function V = v(t)
    if find(T == t) == 1 then
        V = [ -3*s( find(T == t) ) + 4*s( find(T==(t
            +h)) ) - s( find( T == (t+2*h) ) ) ]/(2*h
            ) //Three point forward difference
            formula
    elseif find(T == t) == n
        V = [ 3*s( find(T == t) ) - 4*s( find(T==(t-
            h)) ) + s( find( T == (t-2*h) ) ) ]/(2*h)
             //Backward difference formula
    else
        V = [ s( find(T == (t+h)) ) - s( find(T == (
            t-h)) ) ]/(2*h) //Central difference
            formula
    end
endfunction

v_5 = v(5)
v_7 = v(7)
v_9 = v(9)

disp(v_9,'v(9) = ',v_7,'v(7) = ',v_5,'v(5) = ')
```

Scilab code Exa 11.6 Three Point Central Difference Formula

```
//Example No. 11_06
//Three Point Central Difference formula
//Pg No. 359
clear ;close ;clc ;

T = 5:9 ;
s = [10  14.5  19.5  25.5  32 ];
h = T(2)-T(1);
```

```
deff('A = a(t)','A = [ s( find( T == (t+h) ) ) - 2*s
    ( find( T == t) ) + s( find( T == (t-h) ) ) ]/h^2
    ')
a_7 = a(7)

disp(a_7,'a(7) = ')
```

Scilab code Exa 11.7 Second order Derivative

```
//Example No. 11_7
//Pg No. 359
clear ; close ; clc ;

h = 0.25 ;
//y''(x) = e^(x^2)
//y(0) = 0 , y(1) = 0
// y''(x) = y(x+h) - 2*y(x) + y(x-h)/h^2  = e^(x^2)
//( y(x + 0.25) - 2*y(x) + y(x-0.25) )/0.0625 = e^(x
    ^2)
//y(x+0.25) - 2*y(x) + y(x - 0.25) = 0.0624*e^(x^2)
//y(0.5) - 2*y(0.25) + y(0) = 0.0665
//y(0.75) - 2*y(0.5) + y(0.25) = 0.0803
//y(1) - 2*y(0.75) + y(0.5) = 0.1097
//given y(0) = y(1) = 0
//
//0 + y2 - 2y1 = 0.06665
//y3 - 2*y2 + y1 = 0.0803
//-2*y3 + y2 + 0 = 0.1097
//Therefore
A = [0 1 -2 ; 1 -2 1 ; -2 1 0 ]
B = [ 0.06665 ; 0.0803 ; 0.1097 ]
C = A\B
mprintf('solving the above equations we get \n\n y1
    = y(0.25) = %f \n y2 = y(0.5) = %f \n y3 = y
    (0.75) = %f \n ',C(3),C(2),C(1))
```

Scilab code Exa 11.8 Richardsons Extrapolation Technique

```
1  //Example No. 9_01
2  //Richardson's Extrapolation Technique
3  //Pg No. 362
4  clear ;close ;clc ;
5
6  x = -0.5:0.25:1.5
7  h = 0.5 ;
8  r = 1/2 ;
9
10 deff('F = f(x)','F = exp(x)');
11 deff('D = D(x,h)' ,'D = [ f(x + h) - f(x-h) ]/(2*h)
       ' );
12 deff('df = df(x,h,r)','df = [D(x,r*h) - r^2*D(x,h)
       ]/(1-r^2)');
13
14 df_05 = df(0.5,0.5,1/2);
15 disp(df_05,'richardsons technique -  df(0.5) = ',D
       (0.5,0.25),'D(rh) = D(0.25) = ',D(0.5,0.5),'D
       (0.5) = ')
16 dfexact = derivative(f,0.5)
17 disp(dfexact,'Exact df(0.5) = ')
18 disp('The result by richardsons technique is much
       better than other results')
19
20 //r = 2
21 disp(df(0.5,0.5,2),'df(x) = ',D(0.5,2*0.5),'D(rh) =
       ','for r = 2')
```

Chapter 12

Numerical Integration

Scilab code Exa 12.1 Trapezoidal Rule

```
1 //Example No. 12_01
2 //Trapezoidal Rule
3 //Pg No. 373
4 clear ;close ;clc ;
5
6 x = poly(0,"x");
7 deff('F = f(x)','F = x^3 + 1');
8
9 //case(a)
10 a = 1;
11 b = 2 ;
12 h = b - a ;
13 It = (b-a)*(f(a)+f(b))/2
14 d2f = derivat(derivat( f(x) ))
15 Ett = h^3*horner(d2f,2)/12
16 Iexact = intg(1,2,f)
17 Trueerror = It - Iexact
18 disp(Trueerror,'True error = ',Iexact,'Iexact = ',
     Ett,'Ett = ',It,'It = ','case(a)')
19 disp('Here Error bound is an overestimate of true
     error')
```

```
20
21 //case(b)
22 a = 1;
23 b = 1.5 ;
24 h = b - a ;
25 It = (b-a)*(f(a)+f(b))/2
26 Ett = h^3*horner(d2f,1.5)/12
27 Iexact = intg(1,1.5,f)
28 Trueerror = It - Iexact
29 disp(Trueerror,'True error = ',Iexact,'Iexact = ',
      Ett,'Ett = ',It,'It = ','case(b)')
```

Scilab code Exa 12.2 Trapezoidal Rule

```
1 //Example No. 12_02
2 //Tapezoidal rule
3 //Pg No. 376
4 clear ;close ;clc ;
5
6 deff('F = f(x)','F = exp(x)');
7 a = -1 ;
8 b = 1 ;
9
10 //case(a)
11 n = 2
12 h = (b-a)/n
13 I = 0
14 for i = 1:n
15     I = I + f(a+(i-1)*h)+f(a+i*h);
16 end
17 I = h*I/2 ;
18 disp(I,'intergral for case(a),Ia = ')
19
20 //case(b)
21 n = 4
```

```
22 h = (b-a)/n
23 I = 0
24 for i = 1:n
25     I = I + f(a+(i-1)*h)+f(a+i*h);
26 end
27 I = h*I/2 ;
28 Iexact = 2.35040
29 disp('n = 4 case is better than n = 2 case',Iexact,'
   exact integral,Iexact = ',I,'intergral for case(b
   ),Ib = ')
```

Scilab code Exa 12.3 Simpons 1 by 3 rule

```
1 //Example No. 12_03
2 //Simpon's 1/3 rule
3 //Pg No. 381
4 clear ;close ;clc ;
5
6 funcprot(0) //To avoid warning message for defining
   function f(x) twice
7 //case(a)
8 deff('F = f(x)','F = exp(x)');
9 a = -1;
10 b = 1;
11 h = (b-a)/2
12 x1 = a+h
13 Is1 = h*( f(a) + f(b) + 4*f(x1) )/3
14 disp(Is1,'Integral for case(a) , Is1 = ',h,'h = ')
15
16 //case(b)
17 deff('F = f(x)','F = sqrt(sin(x))');
18 a = 0
19 b = %pi/2
20 h = (b-a)/2
21 x1 = a+h
```

```
Is1 = h*( f(a) + f(b) + 4*f(x1) )/3
disp(Is1,'Integral for case(b),Is1 = ',h,'h = ')
```

Scilab code Exa 12.4 Simpons 1 by 3 rule

```
//Example No. 12_04
//Simpon's 1/3 rule
//Pg No.382
clear ;close ;clc ;

deff('F = f(x)','F = sqrt( sin(x) )');
x0 = 0 ;
xa = %pi/2 ;

//case(a) n = 4
n = 4 ;
h = ( xa-x0 )/n
I = 0
for i = 1:n/2
    I = I + f(x0 + (2*i-2)*h) + 4*f(x0 + (2*i-1)*h)
       + f(x0 + 2*i*h) ;
end
I = h*I/3
disp(I,'Integral value for n = 4 is',h,'h = ')

//case(b) n = 6
n = 6
h = ( xa-x0 )/n
I = 0
for i = 1:n/2
    I = I + f(x0 + (2*i-2)*h) + 4*f(x0 + (2*i-1)*h)
       + f(x0 + 2*i*h) ;
end
I = h*I/3
disp(I,'Integral value for n = 6 is',h,'h = ')
```

Scilab code Exa 12.5 Simpsons 3 by 8 rule

```
1 //Example No. 12_05
2 //Simpson's 3/8 rule
3 //Pg No. 386
4 clear ;close ;clc ;
5
6 funcprot(0)
7 //case(a)
8 deff('F = f(x)','F = x^3 + 1');
9 a = 1 ;
10 b = 2 ;
11 h = (b-a)/3
12 x1 = a + h
13 x2 = a + 2*h
14 Is2 = 3*h*( f(a) + 3*f(x1) + 3*f(x2) + f(b) )/8 ;
15 disp(Is2,'Integral of x^3 +1 from 1 to 2 is')
16 //case(b)
17 deff('F = f(x)','F = sqrt( sin(x) )');
18 a = 0 ;
19 b = %pi/2 ;
20 h = (b-a)/3
21 x1 = a + h
22 x2 = a + 2*h
23 Is2 = 3*h*( f(a) + 3*f(x1) + 3*f(x2) + f(b) )/8 ;
24 disp(Is2,'Integral of sqrt(sin(x)) from 0 to %pi/2
   is')
```

Scilab code Exa 12.6 Booles Five Point Formula

```
1 //Example No. 12_06
```

```
2 //Booles's Five-Point formula
3 //Pg No. 387
4 clear ;close ;clc ;
5
6 deff('F = f(x)','F = sqrt( sin(x) )')
7 x0 = 0;
8 xb = %pi/2 ;
9 n = 4 ;
10 h = (xb - x0)/n
11 Ib = 2*h*(7*f(x0) + 32*f(x0+h) + 12*f(x0 + 2*h) +
      32*f(x0+3*h) + 7*f(x0+4*h))/45;
12 disp(Ib,'Ib = ')
```

Scilab code Exa 12.7 Romberg Integration Formula

```
1 //Example No. 12_07
2 //Romberg Integration formula
3 //Pg No. 391
4 clear ;close ;clc ;
5
6 deff('F = f(x)','F = 1/x');
7 //since we can't have (0,0) element in matrix we
     start with (1,1)
8 a = 1 ;
9 b = 2 ;
10 h = b-a ;
11 R(1,1) = h*(f(a)+f(b))/2
12 disp(R(1,1),'R(0,0) = ')
13 for i = 2:3
14     h(i) = (b-a)/2^(i-1)
15     s = 0
16     for k = 1:2^(i-2)
17         s = s + f(a + (2*k - 1)*h(i));
18     end
19     R(i,1) = R(i-1,1)/2 + h(i)*s;
```

```
20      printf('\nR(%i,0) = %f \n',i-1,R(i,1))
21  end
22  for j = 2:3
23      for i = j:3
24          R(i,j) = (4^(j-1)*R(i,j-1) - R(i-1,j-1))
                /(4^(j-1)-1);
25          printf('\nR(%i,%i) = %f \n',i-1,j-1,R(i,j))
26      end
27  end
```

Scilab code Exa 12.8 Two Point Gauss Legefre Formula

```
1  //Example No. 12_08
2  //Two Point Gauss -Legedre formula
3  //Pg No. 397
4  clear ;close ;clc ;
5
6  deff('F = f(x)','F = exp(x)');
7  x1 = -1/sqrt(3)
8  x2 = 1/sqrt(3)
9  I = f(x1) + f(x2)
10 disp(I,'I = ',x2,'x2 = ',x1,'x1 = ')
```

Scilab code Exa 12.9 Gaussian Two Point Formula

```
1  //Example No. 12_09
2  //Gaussian two point formula
3  //Pg No. 398
4  clear ;close ;clc ;
5
6  a = -2 ;
7  b = 2 ;
8  deff('F = f(x)','F = exp(-x/2)')
```

```
A = (b-a)/2
B = (a+b)/2
C = (b-a)/2
deff('G = g(z)','G = exp(-1*(A*z+B)/2)')
w1 = 1 ;
w2 = 1 ;
z1 = -1/sqrt(3)
z2 = 1/sqrt(3)
Ig = C*( w1*g(z1) + w2*g(z2) )
printf('g(z) = exp(-(%f*z + %f)/2) \n C = %f \n Ig =
    %f \n',A,B,C,Ig)
```

Scilab code Exa 12.10 Gauss Legendre Three Point Formula

```
//Example No. 9_01
//Gauss-Legendre Three-point formula
//Pg No. 400
clear ;close ;clc ;

a = 2 ;
b = 4 ;
A = (b-a)/2
B = (b+a)/2
C = (b-a)/2
deff('G = g(z)','G = (A*z + B)^4 + 1')
w1 = 0.55556 ;
w2 = 0.88889 ;
w3 = 0.55556 ;
z1 = -0.77460;
z2 = 0 ;
z3 = 0.77460 ;
Ig = C*( w1*g(z1) + w2*g(z2) + w3*g(z3) )
printf('g(z) = (%f*z + %f)^4 + 1 \n C = %f \n Ig =
   %f \n',A,B,C,Ig)
```

Chapter 13

Numerical Solution of Ordinary Differential Equations

Scilab code Exa 13.1 Taylor Method

```
1 //Example No. 13_01
2 //Taylor method
3 //Pg No. 414
4 clear ; close ; clc ;
5
6 deff('F = f(x,y)','F = x^2 + y^2')
7 deff('D2Y = d2y(x,y)','D2Y = 2*x + 2*y*f(x,y)');
8 deff('D3Y = d3y(x,y)','D3Y = 2 + 2*y*d2y(x,y) + 2*f(
    x,y)^2');
9 deff('Y = y(x)','Y = 1 + f(0,1)*x + d2y(0,1)*x^2/2 +
     d3y(0,1)*x^3/6');
10 disp(y(0.25),'y(0.25) = ')
11 disp(y(0.5),'y(0.5) = ')
```

Scilab code Exa 13.2 Recursive Taylor Method

```
1  //Example No. 13_02
2  //Recursive Taylor Method
3  //Pg No. 415
4  clear ; close ; clc ;
5
6  deff('F = f(x,y)','F = x^2 + y^2')
7  deff('D2Y = d2y(x,y)','D2Y = 2*x + 2*y*f(x,y)');
8  deff('D3Y = d3y(x,y)','D3Y = 2 + 2*y*d2y(x,y) + 2*f(
       x,y)^2');
9  deff('D4Y = d4y(x,y)','D4Y = 6*f(x,y)*d2y(x,y) + 2*y
       *d3y(x,y) ');
10 h = 0.2 ;
11 deff('Y = y(x,y)','Y = y + f(x,y)*h + d2y(x,y)*h^2/2
       + d3y(x,y)*h^3/6 + d4y(x,y)*h^4/factorial(4)');
12 x0 = 0;
13 y0 = 0 ;
14 for i = 1:2
15      y_(i) = y(x0,y0)
16     printf(' Iteration-%i\n\n dy(0) = %f\n d2y(0) =
          %f\n d3y(0) = %f\n d4y(0) = %f\n ',i,f(x0,y0),
          d2y(x0,y0),d3y(x0,y0),d4y(x0,y0))
17      x0 = x0 + i*h
18      y0 = y_(i)
19     printf('y(0) = %f\n\n',y_(i))
20 end
```

Scilab code Exa 13.3 Picards Method

```
1  //Example No. 13_3
2  //Picard's Method
3  //Pg No. 417
4  clear ; close ; clc ;
5  funcprot(0)
6  //y'(x) = x^2 + y^2,y(0) = 0
7  //y(1) = y0 + integral(x^2 + y0^2,x0,x)
```

```
8  //y(1) = x^3/3
9  //y(2) = 0 + integral(xY2 + y1^2,x0,x)
10 //      = integral(x^2 + x^6/9,0,x) = x^3/3 + x^7/63
11 //therefore y(x) = x^3 /3 + x^7/63
12 deff('Y = y(x)','Y = x^3/3 + x^7/63 ')
13 disp(y(1),'y(1) = ',y(0.2),'y(0.2) = ',y(0.1),'y
       (0.1) = ','for dy(x) = x^2 + y^2 the results are
       ')
14
15 //y'(x) = x*e^y, y(0) = 0
16 //y0 = 0 , x0 = 0
17 //Y(1) = 0 + integral(x*e^0,0,x) = x^2/2
18 //y(2) = 0 + integral( x*e^( x^2/2 ) ,0,x) = e^(x
       ^2/2)-1
19 //therefore y(x) = e^(x^2/2) - 1
20 deff('Y = y(x)','Y = exp(x^2/2) - 1 ')
21 disp(y(1),'y(1) = ',y(0.2),'y(0.2) = ',y(0.1),'y
       (0.1) = ',,'for dy(x) = x*e^y the results are ')
```

Scilab code Exa 13.4 Eulers Method

```
1  //Example No. 13_04
2  //Euler's Method
3  //Pg No. 417
4  clear ; close ; clc ;
5
6  deff('DY = dy(x)','DY = 3*x^2 + 1')
7  x0 = 1
8  y(1) = 2 ;
9  //h = 0.5
10 h = 0.5
11 mprintf('for h = %f\n',h)
12 for i = 2 : 3
13     y(i) = y(i-1) + h*dy(x0+(i-2)*h)
14     mprintf('y(%f) = %f\n',x0+(i-1)*h,y(i))
```

```
15 end
16 //h = 0.25
17 h = 0.25
18 mprintf('\nfor h = %f\n',h)
19 for i = 2 : 5
20     y(i) = y(i-1) + h*dy(x0+(i-2)*h)
21     mprintf('y(%f) = %f\n',x0+(i-1)*h,y(i))
22 end
```

Scilab code Exa 13.5 Error Estimation in Eulers Method

```
1  //Example No. 13_05
2  //Error estimation in Euler's Method
3  //Pg No. 422
4  clear ; close ; clc ;
5
6  deff('DY = dy(x)','DY = 3*x^2 + 1')
7  deff('D2Y = d2y(x)','D2Y = 6*x')
8  deff('D3Y = d3y(x)','D3Y = 6')
9  deff('exacty = exacty(x)','exacty = x^3 + x')
10 x0 = 1
11 y(1) = 2
12 h = 0.5
13 for i = 2 : 3
14     x(i-1) = x0 + (i-1)*h
15     y(i) = y(i-1) + h*dy(x0+(i-2)*h)
16     mprintf('\n Step %i \n x(%i) = %f\n y(%f) = %f\n
           ',i-1,i-1,x(i-1),x(i-1),y(i))
17     Et(i-1) = d2y(x0+(i-2)*h)*h^2/2 +  d3y(x0+(i-2)*
           h)*h^3/6
18     mprintf('\n Et(%i) = %f\n',i-1,Et(i-1))
19     truey(i-1) = exacty(x0+(i-1)*h)
20     gerr(i-1) = truey(i-1) - y(i)
21 end
22
```

```
23 table = [x y(2:3) truey Et gerr]
24 disp(table,'   x       Est y    true y     Et
      Globalerr')
```

Scilab code Exa 13.6 Heuns Method

```
1  //Example No. 13_06
2  //Heun's Method
3  //Pg No. 427
4  clear ; close ;clc ;
5
6  deff('F = f(x,y)','F = 2*y/x ')
7  deff('exacty = exacty(x)','exacty = 2*x^2')
8  x(1) = 1 ;
9  y(1) = 2 ;
10 h = 0.25 ;
11 //Euler's Method
12 disp('EULERS METHOD')
13 for i = 2:5
14     x(i) = x(i-1) + h ;
15     y(i) = y(i-1) + h*f(x(i-1),y(i-1));
16     mprintf('y(%f) = %f \n ',x(i),y(i))
17 end
18 eulery = y
19 //Heun's Method
20 disp('HEUNS METHOD')
21 for i = 2:5
22     m1 = f(x(i-1),y(i-1)) ;
23     ye(i) = y(i-1) + h*f(x(i-1),y(i-1));
24     m2 = f(x(i),ye(i)) ;
25     y(i) = y(i-1) + h*(m1 + m2)/2
26    mprintf('\nIteration %i \n m1 = %f\n ye(%f) = %f
        \n m2 = %f \n y(%f) = %f \n',i-1,m1,x(i),ye(i)
        ,m2,x(i),y(i))
27 end
```

```
truey = exacty(x) ;
table = [x eulery y truey ] ;
disp(table,'   x      Eulers    Heuns       Analytical'
   )
```

Scilab code Exa 13.7 Polygon Method

```
//Example No. 13_07
//Polygon Method
//Pg NO. 433
clear ; close ; clc ;
deff('F = f(x,y)','F = 2*y/x ')
x(1) = 1 ;
y(1) = 2 ;
h = 0.25 ;
for i = 2:3
    x(i) = x(i-1) + h ;
    y(i) = y(i-1) + h*f(  x(i-1)+ h/2   ,  y(i-1) + h
       *f( x(i-1)   ,  y(i-1) )/2  );
    mprintf('y(%f) = %f \n ',x(i),y(i))
end
```

Scilab code Exa 13.8 Classical Runge Kutta Method

```
//Example No. 13_08
//Classical Runge Kutta Method
//Pg No. 439
clear ; close ; clc ;

deff('F = f(x,y)','F = x^2 + y^2');
h = 0.2
x(1) = 0 ;
y(1) = 0 ;
```

```
10
11 for i = 1:2
12     m1 = f(  x(i)   ,   y(i)   ) ;
13     m2 = f(  x(i) + h/2   ,   y(i) + m1*h/2   ) ;
14     m3 = f(  x(i) + h/2   ,   y(i) + m2*h/2   ) ;
15     m4 = f(  x(i) + h   ,   y(i) + m3*h   ) ;
16     x(i+1) = x(i) + h ;
17     y(i+1) = y(i) +  (m1 + 2*m2 + 2*m3 + m4)*h/6 ;
18
19     mprintf('\nIteration - %i\n m1 = %f\n m2 = %f \n
           m3 = %f \n m4 = %f \n y(%f) = %f \n',i,m1,m2
          ,m3,m4,x(i+1),y(i+1))
20 end
```

Scilab code Exa 13.9 Optimum Step size

```
1  //Example No. 13_09
2  //Optimum step size
3  //Pg No. 444
4  clear ; close ; clc ;
5
6  x = 0.8 ;
7  h1 = 0.05 ;
8  y1 = 5.8410870 ;
9  h2 = 0.025 ;
10 y2 = 5.8479637 ;
11
12 //d = 4
13 h = ((h1^4 - h2^4)*10^(-4)/(2*(y2 - y1)))^(1/4)
14 disp(h,'h = ','for four decimal places')
15
16 //d = 6
17 h = ((h1^4 - h2^4)*10^(-6)/(2*(y2 - y1)))^(1/4)
18 disp(h,'h = ','for six decimal places')
19 disp('Note-We can use h = 0.01 for four decimal
```

```
places and h = 0.004 for six decimal places')
```

Scilab code Exa 13.10 Milne Simpson Predictor Corrector Method

```
1  //Example No. 13_10
2  //Milne-Simpson Predictor-Corrector method
3  //Pg NO. 446
4  clear;close;clc;
5
6  deff('F = f(x,y)','F = 2*y/x')
7  x0 = 1 ;
8  y0 = 2 ;
9  h = 0.25 ;
10 //Assuming y1 ,y2 and y3(required for milne-simpson
     formula) are estimated using Fourth- order Runge
     kutta method
11 x1 = x0 + h
12 y1 = 3.13 ;
13 x2 = x1 + h
14 y2 = 4.5 ;
15 x3 = x2 + h
16 y3 = 6.13 ;
17 //Milne Predictor formula
18 yp4 = y0 + 4*h*(2*f(x1,y1) - f(x2,y2) + 2*f(x3,y3))
     /3
19 x4 = x3 + h
20 fp4 = f(x4,yp4) ;
21 disp(fp4,'fp4 = ',yp4,'yp4 = ')
22 //Simpson Corrector formula
23 yc4 = y2 + h*( f(x2,y2) + 4*f(x3,y3) + fp4)/3
24 f4 = f(x4,yc4)
25 disp(f4,'f4 = ',yc4,'yc4 = ')
26
27 yc4 = y2 + h*( f(x2,y2) + 4*f(x3,y3) + f4)/3
28 disp(yc4 ,'yc4 = ')
```

```
29 disp('Note- the exact solution is y(2) = 8')
```

Scilab code Exa 13.11 Adams Bashforth Moulton Method

```
1  //Example No. 13_11
2  //Adams-Bashforth-Moulton Method
3  //Pg NO. 446
4  clear;close;clc;
5
6  deff('F = f(x,y)','F = 2*y/x')
7  x0 = 1 ;
8  y0 = 2 ;
9  h = 0.25 ;
10 x1 = x0 + h
11 y1 = 3.13 ;
12 x2 = x1 + h
13 y2 = 4.5 ;
14 x3 = x2 + h
15 y3 = 6.13 ;
16 //Adams Predictor formula
17 yp4 = y3 + h*(55*f(x3,y3) - 59*f(x2,y2) + 37*f(x1,y1
      ) - 9*f(x0,y0))/24
18 x4 = x3 + h
19 fp4 = f(x4,yp4)
20 disp(fp4,'fp4 = ',yp4,'yp4 = ','Adams Predictor
      formula')
21 //Adams Corrector formula
22 yc4 = y3 + h*( f(x1,y1) - 5*f(x2,y2) + 19*f(x3,y3) +
       9*fp4 )/24
23 f4 = f(x4,yc4)
24 disp(f4,'f4 = ',yc4,'yc4 = ','Adams Corrector
      formula')
25
26 yc4 = y3 + h*( f(x1,y1) - 5*f(x2,y2) + 19*f(x3,y3) +
       9*f4 )/24
```

```
27 disp(yc4 ,'refined-yc4 = ')
```

Scilab code Exa 13.12 Milne Simpson Method Using Modifier

```
1  //Example No. 13_12
2  //Milne-Simpson Method using modifier
3  //Pg No. 453
4  clear ; close ; clc ;
5
6  deff('F = f(y)','F = -y^2 ')
7  x = [ 1 ; 1.2 ; 1.4 ; 1.6 ] ;
8  y = [ 1 ; 0.8333333 ; 0.7142857 ; 0.625 ] ;
9  h = 0.2 ;
10
11 for i = 1:2
12     yp = y(i) + 4*h*( 2*f( y(i+1) ) - f( y(i+2) ) +
          2*f( y(i+3) ) )/3
13     fp = f(yp) ;
14     yc = y( i+2) + h*(f( y(i+2) ) + 4*f( y(i+3) ) +
          fp )/3 ;
15     Etc = -(yc - yp)/29
16     y(i+4) = yc + Etc
17     mprintf('\n y%ip = %f\n f%ip = %f \n y%ic = %f \
          n Modifier Etc = %f \n Modified y%ic = %f \n'
          ,i+3,yp,i+3,fp,i+3,yc,Etc,i+3,y(i+4))
18 end
19 exactanswer = 0.5 ;
20 err = exactanswer - y(6) ;
21 disp(err,'error = ')
```

Scilab code Exa 13.13 System of Differential Equations

```
1  //Example No. 13_13
```

```
2 //System of differetial Equations
3 //Pg No. 455
4 clear ; close ; clc ;
5
6 deff('F1 = f1(x,y1,y2)','F1 = x + y1 + y2 ')
7 deff('F2 = f2(x,y1,y2)','F2 = 1 + y1 + y2 ')
8
9 x0 = 0 ;
10 y10 = 1 ;
11 y20 = -1 ;
12 h = 0.1 ;
13 m1(1) = f1( x0 , y10 , y20 )
14 m1(2) = f2( x0 , y10 , y20 )
15 m2(1) = f1( x0+h , y10 + h*m1(1) , y20 + h*m1(2) )
16 m2(2) = f2( x0+h , y10 + h*m1(1) , y20 + h*m1(2) )
17 m(1) = (m1(1) + m2(1))/2
18 m(2) = (m1(2) + m2(2))/2
19
20 y1_0_1 = y10 + h*m(1)
21 y2_0_1 = y20 + h*m(2)
22
23 mprintf('m1(1) = %f\n m1(2) = %f\n m2(1) = %f\n m2
   (2) = %f\n m(1) = %f\n m(2) = %f\n y1(0.1) = %f\n
    y2(0.1) = %f\n',m1(1),m1(2),m2(1),m2(2),m(1),m
   (2),y1_0_1,y2_0_1)
```

Scilab code Exa 13.14 Higher Order Differential Equations

```
1 //Example No. 13_14
2 //Higher Order Differential Equations
3 //Pg No. 457
4 clear ; close ; clc ;
5
6 x0 = 0
7 y10 = 0
```

```
y20 = 1
h = 0.2
m1(1) = y20 ;
m1(2) = 6*x0 + 3*y10 - 2*y20
m2(1) = y20 + h*m1(2)
m2(2) = 6*(x0+h) + 3*(y10 + h*m1(1)) - 2*(y20 + h*m1
   (2))
m(1) = (m1(1) + m2(1))/2
m(2) = (m1(2) + m2(2))/2

y1_0_2 = y10 + h*m(1)
y2_0_2 = y20 + h*m(2)

mprintf('m1(1) = %f\n m1(2) = %f\n m2(1) = %f\n m2
   (2) = %f\n m(1) = %f\n m(2) = %f\n y1(0.1) = %f\n
    y2(0.1) = %f\n',m1(1),m1(2),m2(1),m2(2),m(1),m
   (2),y1_0_2,y2_0_2)
```

Chapter 14

Boundary Value and Eigenvalue Problems

Scilab code Exa 14.1 Shooting Method

```
1  //Example No. 14_01
2  //Shooting Method
3  //Pg No. 467
4  clear ; close ; clc ;
5
6  function [B,y] = heun(f,x0,y0,z0,h,xf)
7      x(1) = x0 ;
8      y(1) = y0 ;
9      z(1) = z0 ;
10     n = (xf - x0)/h
11     for i = 1:n
12         m1(1) = z(i) ;
13         m1(2) = f(x(i),y(i))
14         m2(1) = z(i) + h*m1(2)
15         m2(2) = f(x(i)+h,y(i)+h*m1(1))
16         m(1) = (m1(1) + m2(1))/2
17         m(2) = ( m1(2) + m2(2) )/2
18         x(i+1) = x(i) + h
19         y(i+1) = y(i) + h*m(1)
```

```
20          z(i+1) = z(i) + h*m(2)
21      end
22      B = y(n+1)
23  endfunction
24
25  deff('F = f(x,y)','F = 6*x')
26  x0 = 1 ;
27  y0 = 2 ;
28  h = 0.5 ;
29  z0 = 2
30  M1 = z0
31  xf = 2
32  B = 9
33  [B1,y] = heun(f,x0,y0,z0,h,xf)
34  disp(B1,'B1 = ')
35  if B1 ~= B then
36      disp('Since B1 is less than B , let z(1) = y(1)
          = 4*(M2)')
37      z0 = 4
38      M2 = z0
39      [B2,y] = heun(f,x0,y0,z0,h,xf)
40      disp(B2,'B2 = ')
41      if B2 ~= B then
42          disp('Since B2 is larger than B ,let us have
              third estimate of z(1) = M3 ')
43          M3 = M2 - (B2 - B)*(M2 - M1)/(B2 - B1)
44          z0 = M3
45          [B3,y] = heun(f,x0,y0,z0,h,xf)
46          disp(y,'The solution is ',B3,'B3 = ')
47      end
48  end
```

Scilab code Exa 14.2 Finite Difference Method

```
1  //Example No. 14_02
```

```
//Finite Difference Method
//Pg No. 470
clear ; close ; clc ;

deff('D2Y = d2y(x)','D2Y = exp(x^2)')
x_1 = 0;
y_0 = 0 ;
y_1  = 0 ;
h = 0.25
xf = 1
n = (xf-x_1)/h
for i = 1:n-1
    A(i,:) = [1  -2   1]
    B(i,1) = exp((x_1 + i*h)^2)*h^2
end
A(1,1) = 0 ; //since we know y0 and y4
A(3,3) = 0 ;
A(1,1:3) = [ A(1,2:3) 0] //rearranging terms
A(3,1:3) = [ 0 A(3,1:2)]
C = A\B //Solution of Equations
mprintf(' \n The solution is \n y1 = y(0.25) = %f \n
    y2 = y(0.5) = %f \n y3 = y(0.75) = %f \n ',C(1),
   C(2),C(3))
```

Scilab code Exa 14.3 Eigen Vectors

```
//Example No. 14_03
//Eigen Vectors
//Pg No. 473
clear ; close ; clc ;

A = [8 -4 ; 2 2 ] ;
lamd = poly(0,'lamd')
p = det(A - lamd*eye())
root = roots(p)
```

```
10 mprintf('\n The roots are \n lamda1 = %f \n lamda2 =
       %f \n ',root(1),root(2))
11 A1 = A - root(1)*eye()
12 X1 = [-1*A1(1,2)/A1(1,1) ; 1]
13 disp(X1,'X1 = ')
14 A2 = A - root(2)*eye()
15 X2 = [-1*A2(1,2)/A2(1,1) ; 1]
16 disp(X2,'X2 = ')
```

Scilab code Exa 14.4 Fadeev Leverrier Method

```
1 //Example No. 14_04
2 //Fadeev - Leverrier method
3 //Pg No. 474
4 clear ; close ; clc ;
5
6 A = [ -1 0 0 ; 1 -2 3 ; 0 2 -3 ]
7 [r,c] = size(A)
8 A1 = A
9 p(1) = trace(A1)
10 for i = 2:r
11     A1 = A*( A1 - p(i-1)*eye())
12     p(i) = trace(A1)/i
13     mprintf('\nA%i = ',i)
14     disp(A1)
15     mprintf('\np%i = %f\n',i,p(i))
16 end
17 x = poly(0,'x');
18 p = p($:-1:1)
19 polynomial = poly([-p ; 1],'x','coeff')
20 disp(polynomial,'Charateristic polynomial is')
```

Scilab code Exa 14.5 Eigen Vectors

```
//Example No. 14_05
//Eigen Vectors
//Pg No. 476

clear ; close ; clc ;

A = [ -1 0 0 ; 1 -2 3 ; 0 2 -3]
[evectors,evalues] = spec(A)
for i = 1:3
    mprintf('\n Eigen vector - %i \n for lamda%i =
       %f \n X%i = ',i,i,evalues(i,i),i)
    evectors(:,i) = evectors(:,i)/evectors(2,i)
    disp(evectors(:,i))
end
```

Scilab code Exa 14.6 Power Method

```
//Example No. 14_06
//Power method
//Pg No. 478
clear ; close ; clc ;

A = [ 1 2 0 ; 2 1 0 ; 0 0 -1 ]
X(:,1) = [0 ; 1 ; 0]
for i = 1:7
    Y(:,i) = A*X(:,i)
    X(:,i+1) = Y(:,i)/max(Y(:,i))
end
disp('  0       1       2       3       4
              5             6             7 ','
   Iterations')
disp(X,'X = ',[[%nan ;%nan ;%nan] Y ],'Y = ')
```

Chapter 15

Solution of Partial Differential Equations

Scilab code Exa 15.1 Elliptic Equations

```
1  //Example No. 15_01
2  //Elliptic Equations
3  //Pg No. 488
4  clear ; close ; clc ;
5
6  l = 15
7  h = 5
8  n = 1 + 15/5
9  f(1,1:4) = 100 ;
10 f(1:4,1) = 100 ;
11 f(4,1:4) = 0 ;
12 f(1:4,4) = 0 ;
13
14 //At point 1 :  f2 + f3 - 4f1 + 100 + 100 = 0
15 //At point 2 :  f1 + f4 - 4f2 + 100 +   0 = 0
16 //At point 3 :  f1 + f4 - 4f3 + 100 +   0 = 0
17 //At point 4 :  f2 + f3 - 4f4 +   0 +   0 = 0
18 //
19 //Final Equations are
```

```
20 //          -4f1 + f2 + f3 + 0 = -200
21 //            f1 - 4f2 + 0 + f4 = -100
22 //            f1 + 0 - 4f3 + f4 = -100
23 //            0  + f2 + f3 - 4f4 = 0
24 A = [ -4 1 1 0 ; 1 -4 0 1 ; 1 0 -4 1 ; 0 1 1 -4 ]
25 B = [-200 ; -100 ; -100 ; 0]
26 C = A\B
27 mprintf('\n The solution of the system is \n f1 = %i
       \n f2 = %i \n f3 = %i \n f4 = %f \n ',C(1),C(2),
     C(3),C(4))
```

Scilab code Exa 15.2 Liebmanns Iterative Method

```
1  //Example No. 15_02
2  //Liebmann's Iterative method
3  //Pg No. 489
4  clear ; close ; clc ;
5
6  f(1,1:4) = 100 ;
7  f(1:4,1) = 100 ;
8  f(4,1:4) = 0 ;
9  f(1:4,4) = 0 ;
10 f(3,3) = 0
11 for n = 1:5
12     for i = 2:3
13         for j = 2:3
14             if n == 1 & i == 2 & j == 2 then
15                 f(i,j) = ( f(i+1,j+1) + f(i-1,j-1) +
                       f(i-1,j+1) + f(i+1,j-1) )/4
16             else
17                 f(i,j) = ( f(i+1,j) + f(i-1,j) + f(i
                      ,j+1) + f(i,j-1) )/4
18             end
19         end
20     end
```

```
21      A(2:5,n) = [f(2,2);f(2,3) ;  f(3,2) ;  f(3,3) ]
22  end
23  A(1,1:5) = 0:4
24  disp(A,'First row of below matrix represents
       iteration number')
```

Scilab code Exa 15.3 Poissons Equation

```
1  //Example No. 15_03
2  //Poisson's Equation
3  //Pg No. 490
4  clear ; close ; clc ;
5
6  //D2f = 2*x^2 * y^2
7  // f = 0
8  // h = 1
9  //Point 1 : 0 + 0 + f2 + f3 - 4f1 = 2(1)^2 * 2^2
10 //           f2 + f3 - 4f1 = 8
11 //Point 2 : 0 + 0 + f1 + f4 -4f2 = 2*(2)^2*2^2
12 //            f1 - 4f2 = f4 = 32
13 //Point 3 : 0 + 0 + f1 + f4 - 4f4 = 2*(1^2)*1^2
14 //            f1 -4f3 + f4 = 2
15 //Point 4 : 0 + 0 + f2 + f3 - 4f4 = 2* 2^2 * 1^2
16 //             f2 + f3 - 4f4 = 8
17 //Rearranging the equations
18 //            -4f1 + f2 + f3 = 8
19 //             f1 - 4f2 + f4 = 32
20 //             f1 - 4f3 + f4 = 2
21 //              f2 + f3 - 4f4 = 8
22 A = [ -4 1 1 0 ; 1 -4 0 1 ; 1 0 -4 1 ; 0 1 1 -4]
23 B = [ 8 ; 32 ; 2 ; 8 ]
24 C = A\B ;
25 mprintf('The solution is \n f1 = %f \n f2 = %f \n f3
     = %f \n f4 = %f \n ', C(1),C(2),C(3),C(4))
```

Scilab code Exa 15.4 Gauss Siedel Iteration

```
1 //Example No. 15_04
2 //Gauss-Seidel Iteration
3 //Pg No. 491
4 clear ; close ; clc ;
5
6 f2 = 0
7 f3 = 0
8 for i = 1:4
9     f1 = (f2 + f3 - 8)/4
10    f4 = f1
11    f2 = (f1 + f4 -32)/4
12    f3 = (f1 + f4 - 2)/4
13    mprintf('\nIteration %i\n f1 = %f,     f2 = %f,
          f3 = %f,     f4 = %f\n',i,f1,f2,f3,f4)
14 end
```

Scilab code Exa 15.5 Initial Value Problems

```
1 //Example No. 15_05
2 //Initial Value Problems
3 //Pg No. 494
4 clear ; close ; clc ;
5
6 h = 1 ;
7 k = 2 ;
8 tau = h^2/(2*k)
9 for i = 2:4
10    f(1,i) = 50*( 4 - (i-1) )
11 end
12 f(1:7,1) = 0 ;
```

```
f(1:7,5) = 0 ;
for j = 1:6
    for i = 2:4
        f(j+1,i) = ( f(j,i-1) + f(j,i+1) )/2
    end
end
disp(f,'The final results are ')
```

Scilab code Exa 15.6 Crank Nicholson Implicit Method

```
//Example No. 15_06
//Crank-Nicholson Implicit Method
//Pg No. 497
clear ; close ; clc ;

h = 1 ;
k = 2 ;
tau = h^2/(2*k)
for i = 2:4
    f(1,i) = 50*( 4 - (i-1) )
end
f(1:5,1) = 0 ;
f(1:5,5) = 0 ;
A = [4  -1  0 ; -1  4  -1 ; 0  -1  4]
for j = 1:4
    for i = 2:4
        B(i-1,1) = f(j,i-1) + f(j,i+1)
    end
    C = A\B
    f(j+1,2) = C(1)
    f(j+1,3) = C(2)
    f(j+1,4) = C(3)
end
disp(f,'The final solution using crank nicholson
   implicit method is ')
```

Scilab code Exa 15.7 Hyperbolic Equations

```
1  //Example No. 15_07
2  //Hyperbolic Equations
3  //Pg No. 500
4  clear ; close ;clc ;
5
6  h = 1
7  Tbyp = 4
8  tau = sqrt(h^2 /4)
9  r = 1+(2.5 - 0)/tau
10 c = 1+(5 - 0)/h
11 for i = 2:c-1
12     f(1,i) = (i-1)*(5 - (i-1) )
13 end
14 f(1:r,1) = 0
15 f(1:r,c) = 0
16 for i = 2:c-1
17     g(i) = 0
18     f(2,i) = (f(1,i+1) + f(1,i-1))/2 + tau*g(i)
19 end
20 for j = 2:r-1
21     for i = 2:c-1
22         f(j+1,i) = -f(j-1,i) + f(j,i+1) + f(j,i-1)
23     end
24 end
25 disp(f,'The values estimated are ')
```