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

August 10, 2013

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

Book Description

Title: Numerical 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

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

Chapter 3

Computer Codes and Arithmetic

Scilab code Exa 3.1 binary to decimal

```
//Example No. 3_01
//Binary to decimal
//Pg No. 45
clear ;close ; clc ;

b = '1101.1101'
v = strsplit(b,'.')     //splitting integral part and
    fraction part
integralp = str2code(v(1))//converting strings to
   numbers
fractionp = str2code(v(2))
li = length(integralp)  //lenght of integral part
lf = length(fractionp)  // and fractional part
di = 0 ;//Initializing integral part and decimal
   part
df = 0 ;
for i = 1:li
    di = 2*di+integralp(i)
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;
```

```

d = 43.375 ;
//Separating integral part and fractional parts
dint = floor(d)
dfrac = d - dint

//Integral Part
i = 1 ;
intp = dec2bin(dint)

//Fractional part
j = 1 ;
while dfrac ~= 0
    fracp(j) = floor(dfrac*2)
    dfrac = dfrac*2 - floor(dfrac*2)
    j = j+1 ;
end
fracp = strcat(string(fracp))

b = strcat([intp,fracp],'.') //combining integral
   part and fractional part
disp(b,'Binary equivalent = ')
```

Scilab code Exa 3.4 Decimal to Octal

```
//Example No. 3_04
//Decimal to Octal
//Pg No. 48
clear ; close ; clc ;

d = 163 ;
oct = dec2oct(d)
disp(oct,'Octal number = ')
```

Scilab code Exa 3.5 Decimal to Binary

```
//Example No. 3_05
//Decimal to binary
//Pg No. 48
clear ; close ; clc ;

d = 0.65
j = 1 ;

while d ~= 0
    fracp(j) = floor(d*2) //integral part of d*2
    d = d*2 - floor(d*2)  //Fractional part of d*2
    j = j+1 ;
    decp(j-1) = d
    p = 1

    for i = 1:j-2
        if abs(d - decp(i))< 0.001  then //Condition for terminating the recurring binary equivalent by
            p = 0                         //finding out if the new fractional part is equal to any of the previous fractonal parts
            break
        end
    end

    if p == 0 then
        break
    end

end
```

```
rec_p = fracp(i+1:j-1)      //Recurring part

rec_p = strcat(string(rec_p))
fracp = strcat(string(fracp))

disp(strcat( [fracp,rec_p] ),'Binary equivalent = ')
```

Scilab code Exa 3.6 Octal to Hexadecimal

```
//Example No. 3_06
//Octal to Hexadecimal
//Pg No. 49
clear ; close ; clc ;

oct = '243' ;
u = str2code(oct)
n = length(u)
for i = 1:n
    b(i) = dec2bin(u(i)) //Converting each digit to binary equivalent
    if length(b(i)) == 2 then        //making the binary equivalents into a groups of triplets
      b(i) = strcat(['0',b(i)])
    elseif length(b(i)) == 1
      b(i) = strcat(['0','0',b(i)])
    end
end
bin = strcat(b) //combining all the triplets
i = 1 ;
while length(bin) > 4
    OtoH = strsplit(bin,length(bin)-4) //splitting the binary equivalent into groups of binary quadruplets
    bin = OtoH(1)
    h(i) = OtoH(2)
```

```
    i = i+1
end
h(i) = bin ;
h = h($:-1:1)
h = bin2dec(h)
h = dec2hex(h)
h = strcat(h)

disp(h,'Hexadecimal equivalent of octal number 243
   is ')
```

Scilab code Exa 3.7 Hexadecimal to Octal

```
//Example No. 3_07
//Hexadecimal to Octal
//Pg No. 49
clear ; close ; clc ;

h = '39.B8' ;
h = strsplit(h,'.') //separating integral part and
   fractional part
cint = abs(str2code(h(1)))
cfrac = abs(str2code(h(2)))
bint = dec2bin(cint)
bfrac = dec2bin(cfrac)
bint = strcat(bint)
bfrac = strcat(bfrac)

//Integral Part
i = 1 ;
while length(bint) > 3
    HtoO = strsplit(bint,length(bint)-3)
    bint = HtoO(1)
    oint(i) = HtoO(2)
    i = i+1 ;
```

```
end
oint(i) = bint
oint =oint($:-1:1)
oint = bin2dec(oint)

//Fraction Part
i = 1 ;
while length(bfrac)> 3
    HtoO = strsplit(bfrac,3)
    bfrac = HtoO(2)
    ofrac(i) = HtoO(1)
    i = i+1
end
ofrac(i) = bfrac
ofrac = bin2dec(ofrac)

//Combining integral part and fraction part
oct = strcat([strcat(string(oint)),strcat(string(ofrac))],'.')
disp(oct,'Octal number equivalent of Hexadecimal number 39.B8 is ')
```

Scilab code Exa 3.8 Binary form of negative integers

```
//Example No. 3_08
//-ve Integer to binary
//Pg No. 50
clear ; close ; clc ;

negint = -13
posbin = dec2bin(abs(negint))
posbin = strcat(['0',posbin])
compl_1 = strsubst(posbin,'0','d')
compl_1 = strsubst(compl_1,'1','0')
compl_1 = strsubst(compl_1,'d','1')
```

```
compl_2 = dec2bin(bin2dec(compl_1) + 1)

disp(compl_2,'Binary equivalent of -13 is ')
```

Scilab code Exa 3.9 16 bit word representation

```
//Example No. 3_09
//Binary representation
//Pg No. 51
clear ;close ;clc ;

n = -32768
compl_32767 = dec2bin(bitcmp(abs(n)-1,16) + 1)
disp(compl_32767,'binary equivalent of -32767 is ')

n_1 = -1
dcomp = bitcmp(1,16)
compl_1 = dec2bin(dcomp+1)
disp(compl_1,'binary equivalent of -1 is ')
compl_32767_code = str2code(compl_32767)
compl_1_code = str2code(compl_1)
summ(1) = 1 //since -32768 is a negative number
c = 0
for i = 16:-1:2
    summ(i) = compl_32767_code(i) + compl_1_code(i)+
       c
    if summ(i) == 2 then
        summ(i) = 0
        c = 1
    else
        c = 0
    end
end
binfinal = strcat(string(summ))
disp(binfinal,'Binary equivalent of -32768 in a 16
```

```
bit word is ')
```

Scilab code Exa 3.10 Floating Point Notation

```
//Example No. 3_10
//Floating Point Notation
//Pg No. 52
clear ; close ; clc ;

function [m,e] =float_notation(n)
m = n ;
for i = 1:16
    if abs(m) >= 1 then
        m = n/10^i
        e = i
    elseif abs(m) < 0.1
        m = n*10^i
        e = -i
    else
        if i == 1 then
            e = 0
        end
        break ;
    end
end
endfunction

[m,e] = float_notation(0.00596)
mprintf('\n 0.00596 is expressed as  %f*10^%i \n',m,
    e)
[m,e] = float_notation(65.7452)
mprintf('\n 65.7452 is expressed as  %f*10^%i \n',m,
    e)
[m,e] = float_notation(-486.8)
mprintf('\n -486.8 is expressed as  %f*10^%i \n',m,e
```

)

Scilab code Exa 3.11 Integer Arithmetic

```
//Example No. 3_11
//Interger Arithmetic
//Pg No. 53
clear ;close ;clc ;

disp(int(25 + 12))
disp(int(25 - 12))
disp(int(12 - 25))
disp(int(25*12))
disp(int(25/12))
disp(int(12/25))
```

Scilab code Exa 3.12 Integer Arithmetic

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

Scilab code Exa 3.13 Floating Point Arithmetic Addition

```
//Example No. 3_13
//Floating Point Arithmetic
//Pg No. 54
clear ;close ;clc ;

fx = 0.586351 ;
Ex = 5 ;
fy = 0.964572 ;
Ey = 2 ;
[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 3.14 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 ;
```

```
fx = 0.999658 ;
Ex = -3 ;
fy = 0.994576 ;
Ey = -3 ;
Ez = max(Ex,Ey)
fy = fy*10^(Ey-Ex)
fz = fx - fy
disp(fz,'fz = ',Ez,'Ez = ')
mprintf('\n z = %f E%i \n',fz,Ez)
if fz < 0.1 then
     fz = fz*10^6       //Since we are using 6 significant digits
     n = length(string(fz))
     fz = fz/10^n
     Ez = Ez + n - 6
     mprintf('\n z = %f E%i (normalised) \n',fz,Ez)
end
```

Scilab code Exa 3.16 Floating Point Arithmetic Multiplication

```
//Example No. 3_16
//Floating Point Arithmetic
//Pg No. 55
clear ;close ;clc ;

fx = 0.200000 ;
Ex = 4 ;
fy = 0.400000 ;
Ey = -2 ;
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 3.17 Floating Point Arithmetic division

```
//Example No. 3_17
//Floating Point Arithmetic
//Pg No. 55
clear ;close ;clc ;

fx = 0.876543 ;
Ex = -5 ;
fy = 0.200000 ;
Ey = -3 ;
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 3.18 Errors in Arithmetic

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

```

fx = 0.500000 ;
Ex = 1 ;
fy = 0.100000 ;
Ey = -7 ;
[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(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(fx,'fx = ',Ez,'Ez = ')
end
mprintf('\n fz = %f \n z = %f E%i \n',fz,fz,Ez)
```

Scilab code Exa 3.19 Errors in Arithmetic

```
//Example No. 3_19
//Floating Point Arithmetic
//Pg No. 56
clear ;close ;clc ;

fx = 0.350000 ;
Ex = 40 ;
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

```
//Example No. 3_21
//Floating Point Arithmetic
//Pg No. 57
clear ;close ;clc ;

fx = 0.500000 ;
Ex = 0 ;
fy = 0.499998 ;
Ey = 0 ;
Ez = 0 ;
fz = fx - fy
disp(fz,'fz = ',Ez,'Ez = ')
mprintf('\n z = %f E%i \n',fz,Ez)
if fz < 0.1 then
     fz = fz*10^6
     n  = length(string(fz))
     fz = fz/10^n
     Ez = Ez + n - 6
     mprintf('\n z = %f E%i (normalised) \n',fz,Ez)
end
```

Scilab code Exa 3.22 Associative law of Addition

```
//Example No. 3_22
//Laws of Arithmetic
//Pg No. 57
clear ; close ; clc ;
function [fz,Ez] =add_sub(fx,Ex,fy,Ey) //addition and subtraction fuction
if fx*fy >= 0 then
    //Addition
    [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
    else
        fx = fx*10^(Ex - Ey)
        fz = fx + fy
        if fz > 1 then
            fz = fz*10^(-1)
            Ez = Ez + 1
        end
    end

else
    //Subtraction
    [Ez,n] = max(Ex,Ey)
    if n == 1 then
        fy = fy*10^(Ey-Ex)
        fz = fx + fy
        if abs(fz) < 0.1 then
           fz = fz*10^6
           fz = floor(fz)
           nfz = length(string(abs(fz)))
           fz = fz/10^nfz
           Ez = nfz - 6
        end
    else
        fx = fx*10^(Ex - Ey)
        fz = fx + fy
        if fz < 0.1 then
           fz = fz*10^6
           fz = int(fz)
           nfz = length(string(abs(fz)))
           fz = fz/10^nfz
           Ez = nfz - 6
        end
```

```
        end
    end
endfunction

fx = 0.456732
Ex = -2
fy = 0.243451
Ey = 0
fz = -0.24800
Ez = 0

[fxy,Exy] = add_sub(fx,Ex,fy,Ey)
[fxy_z,Exy_z] = add_sub(fxy,Exy,fz,Ez)
[fyz,Eyz] = add_sub(fy,Ey,fz,Ez)
[fx_yz,Ex_yz] = add_sub(fx,Ex,fyz,Eyz)
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)

if  fxy_z ~= fx_yz | Exy_z ~= Ex_yz then
    disp('(x+y) + z ~= x + (y+z)')
end
```

Scilab code Exa 3.23 Associative law of Multiplication

```
//Example No. 3_23
//Associative law
//Pg No. 58
clear ; close ; clc ;
x = 0.400000*10^40
y = 0.500000*10^70
z = 0.300000*10^(-30)
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
```

```
xy = fxy * 10^Exy

fxz = fx*z
fxz = fxz*10^6
fxz = floor(fxz) //considering only six significant digits
n = length(string(fxz))
fxz = fxz/10^n
Exz = n - 6
xz = fxz * 10^Exz

xy_xz = xy - xz
disp(xy_xz,'xy_xz = ')
```

Chapter 4

Approximations and Errors in Computing

Scilab code Exa 4.1 Greatest Precision

```
//Example No. 4_01
//Greatest precision
//Pg No. 63
clear ; close ; clc ;

a = '4.3201'
b = '4.32'
c = '4.320106'
na = length(a)-strindex(a,'.')
mprintf('\n %s has a precision of 10^-%i\n',a,na)
nb = length(b)-strindex(b,'.')
mprintf('\n %s has a precision of 10^-%i\n',b,nb)
nc = length(c)-strindex(c,'.')
mprintf('\n %s has a precision of 10^-%i\n',c,nc)
[n,e] = max(na,nb,nc)
if e ==1 then
    mprintf('\n The number with highest precision is %s\n',a)
elseif e == 2
```

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

Scilab code Exa 4.2 Accuracy of Numbers

```
//Example No. 4_02
//Accuracy of numbers
//Pg No. 63
clear ;close ;clc ;

function n = sd(x)
    nd = strindex(x,'.')  //position of point
    num = str2code(x)
    if isempty(nd) & num(length(x)) == 0 then
        mprintf('Accuracy is not specified\n')
        n = 0 ;
    else
        if num(1)>= 1 & isempty(nd) then
            n = length(x)
        elseif num(1) >= 1 & ~isempty(nd) then
                n = length(x) - 1
        else
            for i = 1:length(x)
                if num(i) >= 1 & num(i) <= 9 then
                    break
                end
            end
            n = length(x)- i + 1
        end
    end
endfunction
```

```
a = '95.763'
na = sd(a)
mprintf('%s has %i significant digits\n',a,na)
b = '0.008472'
nb = sd(b)
mprintf('%s has %i significant digits.The leading or higher order zeros are only place holders\n',b,nb)
c = '0.0456000'
nc = sd(c)
mprintf('%s has %i significant digits\n',c,nc)
d = '36'
nd = sd(d)
mprintf('%s has %i significant digits\n',d,nd)
e = '3600'
sd(e)
f = '3600.00'
nf = sd(f)
mprintf('%s has %i significant digits\n',f,nf)
```

Scilab code Exa 4.3 Addition in Binary form

```
//Example No. 4_03
//Pg No. 64
clear ; close ; clc ;

a = 0.1
b = 0.4
for i = 1:8
    afrac(i) = floor(a*2)
    a = a*2 - floor(a*2)
    bfrac(i) = floor(b*2)
    b = b*2 - floor(b*2)
end
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)
```

```
mprintf('\n Chooping Method : \n Approximate x = %.4f*10^%i \n Error = %.4f \n ',fx,E,Err)
//Symmetric Method
if gx >= 0.5 then
    Err = (gx -1)*10^(-1)
    Approx_x = (fx + 10^(-d))*10^E
else
    Approx_x = fx*10^E
    Err = gx * 10^(E-d)
end
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

```
//Example No. 4_05
//Truncation Error
//Pg No. 68
clear ; close ; clc ;

x = 1/5
//When first three terms are used
Trunc_err = x^3/factorial(3) + x^4/factorial(4) + x^5/factorial(5) + x^6/factorial(6)
mprintf('\n a) When first three terms are used \n Truncation error = %.6E \n ',Trunc_err)

//When four terms are used
Trunc_err = x^4/factorial(4) + x^5/factorial(5) + x^6/factorial(6)
mprintf('\n b) When first four terms are used \n Truncation error = %.6E \n ',Trunc_err)

//When Five terms are used
```

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

Scilab code Exa 4.6 Truncation Error

```
//Example No. 4_06
//Truncation Error
//Pg No. 68
clear ; close ; clc ;

x = -1/5
//When first three terms are used
Trunc_err = x^3/factorial(3) + x^4/factorial(4) + x
    ^5/factorial(5) + x^6/factorial(6)
mprintf('\n a) When first three terms are used \n
    Truncation error = %.6E \n ',Trunc_err)

//When four terms are used
Trunc_err = x^4/factorial(4) + x^5/factorial(5) + x
    ^6/factorial(6)
mprintf('\n b) When first four terms are used \n
    Truncation error = %.6E \n ',Trunc_err)

//When Five terms are used
Trunc_err = x^5/factorial(5) + x^6/factorial(6)
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

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

```
//Pg No. 71
clear ; close ; clc ;

h_bu_t = 2945 ;
h_bu_a = 2950 ;
h_be_t = 30 ;
h_be_a = 35 ;
e1 = abs(h_bu_t - h_bu_a)
e1_r = e1/h_bu_t
e2 = abs(h_be_t - h_be_a)
e2_r = e2/h_be_t
mprintf('\n For Building : \n Absolute error ,  e1 = %i \n Relative error , e1_r = %.2f percent \n ',e1,e1_r*100)
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

```
//Example No. 4_08
//Machine Epsilon
//Pg No. 72
clear ; close ; clc ;

deff('q = Q(p)','q = 1 + (p-1)*log10(2)' )
p = 24
q = Q(p)
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

```
//Example No. 4_09
//Propagation of Error
//Pg No. 75
clear ; close ; clc ;

x = 0.1234*10^4
y = 0.1232*10^4
d = 4
er_x = 10^(-d + 1)/2
er_y = 10^(-d + 1)/2
ex = x*er_x
ey = y*er_y
ez = abs(ex) + abs(ey)
er_z = abs(ez)/abs(x-y)

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

```
//Example No. 4_10
//Errors in Sequence of Computations
//Pg No. 77
clear ; close ; clc ;

x_a = 2.35 ;
y_a = 6.74 ;
z_a = 3.45 ;
ex = abs(x_a)*10^(-3+1)/2
ey = abs(y_a)*10^(-3+1)/2
ez = abs(z_a)*10^(-3+1)/2
exy = abs(x_a)*ey + abs(y_a)*ex
ew = abs(exy) + abs(ez)
```

```
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

```
//Example No. 4_11
//Addition of Chain of Numbers
//Pg No. 77
clear ; close ; clc ;

x = 9678 ;
y = 678 ;
z = 78 ;
d = 4 ;  //length of mantissa
fx = x/10^4
fy = y/10^4
fu = fx + fy
Eu = 4
if fu >= 1 then
    fu = fu/10
    Eu = Eu + 1
end
//since length of mantissa is only four we need to
    maintain only four places in decimal, so
fu = floor(fu*10^4)/10^4
u = fu * 10^Eu
w = u + z
n = length(string(w))
w = floor(w/10^(n-4))*10^(n-4) //To maintain length
    of mantissa = 4
disp(w,'w = ')
True_w = 10444
ew = True_w - w
er_w = (True_w - w)/True_w
disp(er_w,'er,w = ',ew,'ew = ',True_w,'True w = ')
```

Scilab code Exa 4.12 Addition of Chain of Numbers

```
//Example No. 4_12
//Addition of chain Numbers
//Pg No. 77
clear ; close ; clc ;

x = 9678 ;
y = 678 ;
z = 78 ;
d = 4 ;  //length of mantissa
n = max(length( string(y) ) , length(string(z)))
fy = y/10^n
fz = z/10^n
fu = fy + fz
Eu = n
if fu >= 1 then
    fu = fu/10
    Eu = Eu + 1
end
u = fu * 10^Eu
n = max(length( string(x) ) , length(string(u)))
fu = u/10^4
fx = x/10^4
fw = fu + fx
Ew = 4
if fw >= 1 then
    fw = fw/10
    Ew = Ew + 1
end
//since length of mantissa is only four we need to
   maintain only four places in decimal, so
fw = floor(fw*10^4)/10^4
w = fw*10^Ew
```

```
disp(w,'w = ')
True_w = 10444
ew = True_w - w
er_w = (True_w - w)/True_w
disp(er_w,'er,w = ',ew,'ew = ',True_w,'True w = ')
```

Scilab code Exa 4.13 Theoritical Problem

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

Scilab code Exa 4.14 Absolute and Relative Errors

```
//Example No. 4_14
//Absolute & Relative Errors
//Pg No. 79
clear ; close ; clc ;

xa = 4.000
deff('f = f(x)','f = sqrt(x) + x')
//Assuming x is correct to 4 significant digits
ex = 0.5 * 10^(-4 + 1)
df_xa = derivative(f,4)
ef = ex * df_xa
er_f = ef/f(xa)
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

```
//Example No. 4_15
//Error Evaluation
//Pg No. 80
clear ; close ; clc ;

x = 3.00 ;
y = 4.00 ;
deff('f = f(x,y)','f = x^2 + y^2 ')
deff('df_x = df_x(x)','df_x = 2*x')
deff('df_y = df_y(y)','df_y = 2*y')
ex = 0.005
ey = 0.005
ef = df_x(x)*ex + df_y(y)*ey
disp(ef,'ef = ')
```

Scilab code Exa 4.16 Condition and Stability

```
//Example No. 4_16
//Condition and Stability
//Pg No. 82
clear ; close ; clc ;

C1 = 7.00 ;
C2 = 3.00 ;
m1 = 2.00 ;
m2 = 2.01 ;
x = (C1 - C2)/(m2 - m1)
y = m1*((C1 - C2)/(m2 - m1)) + C1
disp(y,'y = ',x,'x = ')
disp('Changing m2 from 2.01 to 2.005')
m2 = 2.005
x = (C1 - C2)/(m2 - m1)
y = m1*((C1 - C2)/(m2 - m1)) + C1
```

```
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

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

Scilab code Exa 4.18 Difference of Square roots

```
//Example No. 4_18
//Difference of Square roots
//Pg No. 84
clear ; close ; clc ;

x = 497.0 ;
y = 496.0 ;
sqrt_x = sqrt(497)
sqrt_y = sqrt(496)
nx = length( string( floor( sqrt_x ) ) )
ny = length( string( floor( sqrt_y ) ) )
sqrt_x = floor(sqrt_x*10^(4-nx))/10^(4-nx)
sqrt_y = floor(sqrt_y*10^(4-ny))/10^(4-ny)
z1 = sqrt_x - sqrt_y
disp(z1,'z = sqrt(x) - sqrt(y)')
z2 = ( x -y)/(sqrt_x + sqrt_y)
if z2 < 0.1 then
    z2 = z2*10^4
```

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

Scilab code Exa 4.19 Theoritical Problem

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

Scilab code Exa 4.20 Theoritical Problem

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

Scilab code Exa 4.21 Induced Instability

```
//Example 4_21
//Pg No. 85
clear ; close ; clc ;

x = -10
T_act(1) = 1
T_trc(1) = 1
e_x_cal = 1
for i = 1:100
```

```
    T_act(i+1) = T_act(i)*x/i
    T_trc(i+1) = floor(T_act(i+1)*10^5)/10^5
    TE(i) = abs(T_act(i+1)-T_trc(i+1))
    e_x_cal = e_x_cal + T_trc(i+1)
end
e_x_act = exp(-10)
disp(e_x_act,'actual e^x = ',e_x_cal,'calculated e^x
    using roundoff = ',sum(TE),'Truncation Error = '
   )
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

```
//Example No. 6_01
//Possible Initial guess values for roots
//Pg No. 126

clear ; close ; clc ;

A = [ 2 ; -8 ; 2 ; 12]; // Coefficients of x terms
    in the decreasing order of power
n = size(A);
x1 = -A(2)/A(1);
disp(x1,'The largest possible root is  x1 =')
disp(x1,'No root can be larger than the value')

x = sqrt((A(2)/A(1))^2 - 2*(A(3)/A(1))^2);

printf('\n all real roots lie in the interval (-%f,
    %f)\n',x,x)
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

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

Scilab code Exa 6.3 Evaluating Polynomial using Horners rule

```
//Example No. 6_03
//Evaluating Polynomial using Horner's rule
//Pg No.
clear ; close ; clc ;

//Coefficients of x terms in the increasing order of
    power
A = [ 6 ; 1 ; -4 ; 1];
x = 2
[n,c] = size(A) ;
p(n) = A(n)
disp(p(n), 'p(4) = ')
for i = 1:n-1
    p(n-i) = p(n-i+1)*x + A(n-i)
    printf('\n p(%i)= %i\n',n-i,p(n-i))
end
mprintf('\n f(%i) = p(1) = %i',x,p(1))
```

Scilab code Exa 6.4 Bisection Method

```
//Example No. 6_04
//Root of a Equation Using Bisection Method
//Pg No. 132

clear ; close ; clc ;

//Coefficients in increasing order of power of x starting from 0
A = [-10 -4 1];
disp('First finding the interval that contains a root,this can be done by using Eq 6.10')
xmax = sqrt((A(2)/A(3))^2 - 2*(A(1)/A(3)))
printf('\n Both the roots lie in the interval (-%i,%i) \n',xmax,xmax)
x = -6:6
p = poly(A,'x','c')
fx = horner(p,x);
for i = 1:12
    if fx(1,i)*fx(1,i+1) < 0 then
        break ;
    end
end
printf('\n The root lies in the interval (%i,%i)\n',x(1,i),x(1,i+1))
x1 = x(1,i);
x2 = x(1,i+1);
f1 = fx(1,i);
f2 = fx(1,i+1);
err = abs((x2-x1)/x2) ;
while err > 0.0001
x0 = (x1 + x2)/2 ;
f0 = horner(p,x0);
if f0*f1 < 0  then
    x2 = x0
    f2 = f0
elseif f0*f2 < 0
    x1 = x0
    f1 = f0
```

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

Scilab code Exa 6.5 False Position Method

```
//Example No. 6_05
//False Position Method
//Pg No. 139
clear ; close ; clc ;

//Coefficients of polynomial in increasing order of
   power of x
A = [-2  -1  1];
x1 = 1 ;
x2 = 3 ;
fx = poly(A,'x','c');
for i = 1:15
    printf('Iteration No. %i \n',i);
    fx1 = horner(fx,x1);
    fx2 = horner(fx,x2);
    x0 = x1 - fx1*(x2-x1)/(fx2-fx1)
    printf('x0 = %f \n',x0);
    fx0 = horner(fx,x0);
    if fx1*fx0 < 0 then
        x2 = x0 ;
    else
        x1 = x0 ;
    end
end
```

Scilab code Exa 6.06 Theoritical Problem

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

Scilab code Exa 6.7 Newton Raphson Method

```
//Example No. 6_07
//Root of the Equation using Newton Raphson Method
//Pg No. 147
clear ; close ; clc ;

//Coefficients of polynomial in increasing order of
   power of x
A = [ 2  -3  1];
fx = poly(A,'x','c');
dfx = derivat(fx);

x(1) = 0 ;
for i = 1:10
    f(i)  = horner(fx,x(i));
    if f(i)~= 0 then
        df(i) = horner(dfx,x(i));
        x(i+1) = x(i) - f(i)/df(i) ;
        printf('x%i = %f\n',i+1,x(i+1));
    else
        printf('Since f(%f) = 0, the root closer to
           the point x = 0 is %f \n',x(i),x(i) );
        break
```

```
    end
end
```

Scilab code Exa 6.8 Newton Raphson Method

```
//Example No. 6_08
//Root of the Equation using Newton Raphson Method
//Pg No. 151
clear ; close ; clc ;
//Coefficients of polynomial in increasing order of power of x
A = [ 6  1  -4  1 ];
fx = poly(A,'x','c');
dfx = derivat(fx);

x(1) = 5.0 ;
for i = 1:6
    f(i)  = horner(fx,x(i));
    if f(i)~= 0 then
        df(i) = horner(dfx,x(i));
        x(i+1) = x(i) - f(i)/df(i) ;
        printf('x%i = %f\n',i+1,x(i+1));
    end
end
disp('From the results we can see that number of correct digits approximately doubles with each iteration')
```

Scilab code Exa 6.9 Secant Method

```
//Example No. 6_09
//Root of the equation using SECANT Method
//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 ;
```

```
A = [ -5  0  1 ];
funcprot(0);
deff('x = g(x)','x = 5/x');
x(1) = 1 ;
printf('\n x0 = %f \n',x(1));
for i = 2:5
    x(i) = feval(x(i-1),g);
    printf(' x%i = %f \n',i-1,x(i))
end
//Defining g(x) in different way
deff('x = g(x)','x = x^2 + x - 5');
x(1) = 0;
printf('\n x0 = %f \n',x(1));
for i = 2:5
    x(i) = feval(x(i-1),g);
    printf(' x%i = %f \n',i-1,x(i))
end
//Third form of g(x)
deff('x = g(x)',' x = (x + 5/x)/2 ');
x(1) = 1;
printf('\n x0 = %f \n',x(1));
for i = 2:7
    x(i) = feval(x(i-1),g);
    printf(' x%i = %f \n',i-1,x(i))
end
```

Scilab code Exa 6.13 Fixed Point Method for non linear equations

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

```

```
printf(' x^2 - y^2 = 3 \n x^2 + x*y \n');
deff('x = f(x,y)','x = y + 3/(x+y)') ;
deff('y = g(x)','y = (6-x^2)/x') ;
x(1) = 1 ;
y(1) = 1 ;
printf('\n x0 = %f \n y0 = %f \n',x(1),y(1));
for i = 2:4
    x(i) = feval(x(i-1),y(i-1),f);
    y(i) = feval(x(i-1),g);
    printf('\n x%i = %f \n y%i = %f \n',i-1,x(i),i
       -1,y(i));
end
```

Scilab code Exa 6.14 Newton Raphson Method for Non linear equations

```
//Example No. 6_14
//Solving System of Non-linear equations using
   Newton Raphson Method
//Pg No. 172
clear ; close ; clc ;

printf('x^2 + x*y = 6 \n x^2 - y^2 = 3 \n');
deff('f = F(x,y)','f = x^2 + x*y - 6' );
deff('g = G(x,y)','g = x^2 - y^2 -3');
deff('f1 = dFx(x,y)','f1 = 2*x + y');
deff('f2 = dFy(x,y)','f2 = y');
deff('g1 = dGx(x,y)','g1 = 2*x ');
deff('g2 = dGy(x,y)','g2 = -2*y');
x(1) = 1 ;
y(1) = 1 ;

for i = 2:3
    Fval = feval(x(i-1),y(i-1),F);
    Gval = feval(x(i-1),y(i-1),G);
    f1 = feval(x(i-1),y(i-1),dFx);
```

```
    f2 = feval(x(i-1),y(i-1),dFy);
    g1 = feval(x(i-1),y(i-1),dGx);
    g2 = feval(x(i-1),y(i-1),dGy);
    D =  f1*g2 - f2*g1 ;

    x(i) = x(i-1) - (Fval*g2 - Gval*f2)/D ;
    y(i) = y(i-1) - (Gval*f1 - Fval*g1)/D ;
    printf('\n x%i = %f \n y%i = %f \n',i-1,x(i),i
       -1,y(i))

end
```

Scilab code Exa 6.15 Synthetic Division

```
//Example No. 6_15
//Synthetic Division
//Pg No. 176
clear ; close ; clc ;

a = [-9 15 -7 1];
b(4) = 0 ;
for i = 3:-1:1
    b(i) = a(i+1) + b(i+1)*3
    printf('b%i = %f\n',i,b(i))
end
    disp(poly(b,'x','c'),'Thus the polynomial is')
```

Scilab code Exa 6.16 Bairstow Method for Factor of polynomial

```
//Example No. 6_16
//Quadratic factor of a polynomial using Bairstow's
   Method
//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
```

```
//Pg No. 197
clear ; close ; clc ;

deff('y = f(x)','y = x^3 + 2*x^2 + 10*x - 20') ;
x1 = 0 ;
x2 = 1 ;
x3 = 2 ;
for i = 1:10
    f1 = feval(x1,f) ;
    f2 = feval(x2,f) ;
    f3 = feval(x3,f) ;
    h1 = x1-x3 ;
    h2 = x2-x3 ;
    d1 = f1 - f3 ;
    d2 = f2 - f3 ;
    D = h1*h2*(h1-h2);
    a0 = f3 ;
    a1 = (d2*h1^2 - d1*h2^2)/D ;
    a2 = (d1*h2 - d2*h1)/D ;
    if abs(-2*a0/( a1 + sqrt( a1^2 - 4*a0*a2 ) )) <
       abs( -2*a0/( a1 - sqrt( a1^2 - 4*a0*a2 ) ))
       then
        h4 = -2*a0/(a1 + sqrt(a1^2 - 4*a0*a2));
    else
        h4 = -2*a0/(a1 - sqrt(a1^2 - 4*a0*a2))
    end
    x4 = x3 + h4 ;
    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) ;
    relerr = abs((x4-x3)/x4);
    if relerr <= 0.00001 then
        printf('root of the polynomial is x4 = %f',
           x4);
        break
    end
```

```
    x1 = x2 ;
    x2 = x3 ;
    x3 = x4 ;


end
```

Chapter 7

Direct Solutions of Linear Equations

Scilab code Exa 7.1 Elimination Process

```
//Example No. 7_01
//Elimination Process
//Pg No. 211

clear ; close ; clc ;

A = [3 2 1 10; 2 3 2 14 ; 1 2 3 14];
A(2,:) = A(2,:) - A(1,:)*A(2,1)/A(1,1)
A(3,:) = A(3,:) - A(1,:)*A(3,1)/A(1,1)
disp(A)

A(3,:) = A(3,:) - A(2,:)*A(3,2)/A(2,2)
disp(A)

z = A(3,4)/A(3,3)
y = (A(2,4) - A(2,3)*z)/A(2,2)
x = (A(1,4) - A(1,2)*y - A(1,3)*z)/A(1,1)
disp(x,'x = ',y,'y = ',z,'z = ')
```

Scilab code Exa 7.2 Basic Gauss Elimination

```
//Example No. 7_02
//Basic Gauss Elimination
//Pg No. 214
clear ; close ; clc ;

A = [ 3 6 1 ; 2 4 3 ; 1 3 2 ];
B = [16 13 9];
[ar1,ac1] = size(A);
Aug = [3 6 1 16 ; 2 4 3 13 ; 1 3 2 9]
for i = 2 : ar1
    Aug(i,:) = Aug(i,:) - (Aug(i,1)/Aug(1,1))*Aug
       (1,:) ;
end
disp(Aug)
disp('since Aug(2,2) = 0 elimination is not possible
   ,so reordering the matrix')
Aug = Aug( [ 1 3 2],:);
disp(Aug)
disp('Elimination is complete and by back
   substitution the solution is\n')
disp('x3 = 1, x2 = 2 , x1 = 1 ')
```

Scilab code Exa 7.3 Gauss Elimination using Partial Pivoting

```
//Example No. 7_03
// Gauss Elimination using partial pivoting
// Pg No. 220
clear ; close ; clc ;

A = [ 2 2 1 ; 4 2 3 ; 1 -1 1];
```

```
B = [  6   ;   4   ;  0  ];
[ ar , ac ] = size(A);
Aug = [   2  2  1  6   ;   4  2  3  4  ;  1  -1  1
   0 ];

for i = 1 : ar-1
    [ p , m ] =  max(abs(Aug(i:ar,i)))
    Aug(i:ar,:) = Aug([i+m-1  i+1:i+m-2  i  i+m:ar
       ],:);
    disp(Aug)
    for k = i+1 : ar
        Aug(k,i:ar+1) = Aug(k,i:ar+1) - (Aug(k,i)/
           Aug(i,i))*Aug(i,i:ar+1);
    end
    disp(Aug)
end

//Back Substitution
X(ar,1) = Aug(ar,ar+1)/Aug(ar,ar)
for i = ar-1 : -1 : 1
    X(i,1) = Aug(i,ar+1);
    for j = ar : -1 : i+1
        X(i,1) = X(i,1) - X(j,1)*Aug(i,j);
    end
    X(i,1) = X(i,1)/Aug(i,i);
end

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

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

```
//Pg No. 228
clear ; close ; clc ;

A = [  2  4  -6 ; 1  3  1 ; 2  -4  -2  ];
B = [  -8  ;  10  ;  -12  ];
[ar,ac] = size(A);
Aug = [  2  4  -6  -8  ;  1  3  1  10  ;  2  -4  -2
    -12  ];
disp(Aug)

for i = 1 : ar
    Aug(i,i:ar+1) = Aug(i,i:ar+1)/Aug(i,i) ;
    disp(Aug)
    for k = 1 : ar
        if k ~= i then
            Aug(k,i:ar+1) = Aug(k,i:ar+1) - Aug(k,i)
               *Aug(i,i:ar+1);
        end
    end
    disp(Aug)
end
```

Scilab code Exa 7.5 DoLittle LU Decomposition

```
//Example No. 7_05
//DoLittle LU Decomposition
//Pg No. 234

clear ; close ; clc ;

A = [ 3  2  1  ;  2  3  2  ;  1  2  3  ];
B = [ 10  ;  14  ;  14  ];
[n , n] = size(A);

for i = 2:n
```

```
    U(1,:) = A(1,:);
    L(i,i) = 1 ;
    if i ~= 1 then
         L(i,1) = A(i,1)/U(1,1);
    end
end

for j = 2:n
    for i = 2:n

        if i <= j then
            U(i,j) = A(i,j);
            for k = 1:i-1
                U(i,j) = U(i,j) - L(i,k)*U(k,j);
            end
            printf('\nU(%i,%i) = %f \n',i,j,U(i,j))

        else
            L(i,j) = A(i,j)
            for k = 1:j-1
                L(i,j) = L(i,j) - L(i,k)*U(k,j);
            end
            L(i,j) = L(i,j)/U(j,j)  ;
            printf('\n L(%i,%i) = %f \n',i,j,L(i,j))
        end
    end
end
disp(U,'U = ')
disp(L,'L = ')

//Forward Substitution
for i = 1:n
    z(i,1) = B(i,1);
    for j = 1:i-1
        z(i,1) = z(i,1) - L(i,j)*z(j,1);
    end
    printf('\n z(%i) = %f \n',i,z(i,1))
end
```

```

//Back Substitution
for i = n : -1 : 1
    x(i,1) = z(i,1);
    for j = n : -1 : i+1
        x(i,1) = x(i,1) - U(i,j)*x(j,1);
    end
    x(i,1) = x(i,1)/U(i,i);
    printf('\n x(%i) = %f \n',i,x(i,1))
end
```

Scilab code Exa 7.6 Choleskys Factorisation

```
//Example No. 7_06
//Cholesky's Factorisation
//Pg No. 242

clear ; close ; clc ;

A = [ 1 2 3 ; 2 8 22  ; 3 22 82 ];
[n,n] = size(A);

for i = 1:n
    for j = 1:n
        if i == j then
            U(i,i) = A(i,i)
            for k = 1:i-1
                U(i,i) = U(i,i)-U(k,i)^2 ;
            end
            U(i,i) = sqrt(U(i,i));
         elseif i < j
             U(i,j) = A(i,j)
             for k = 1:i-1
                 U(i,j) = U(i,j) - U(k,i)*U(k,j);
             end
```

```
                U(i,j) = U(i,j)/U(i,i)
            end
        end
end
disp(U)
```

Scilab code Exa 7.7 Ill Conditioned Systems

```
//Example No. 7_07
//Ill-Conditioned Systems
//Pg No. 245

clear ; close ; clc ;

A = [ 2  1 ; 2.001  1];
B = [ 25 ; 25.01 ];
x(1) = (25 - 25.01)/(2 - 2.001);
x(2) = (25.01*2 - 25*2.001)/(2*1 - 2.001*1);
printf('\n x(1) = %f \n x(2) = %f \n',x(1),x(2))
x(1) = (25 - 25.01)/(2 - 2.0005);
x(2) = (25.01*2 - 25*2.0005)/(2*1 - 2.0005*1);
printf('\n x(1) = %f \n x(2) = %f \n',x(1),x(2))
r = A*x-B
disp(x)
disp(r)
```

Chapter 8

Iterative Solution of Linear Equations

Scilab code Exa 8.1 Gauss Jacobi Iteration Method

```
//Example No. 8_01
//Gauss Jacobi
//Page No. 254
clear ; close ; clc ;

A = [ 2  1  1 ; 3  5  2 ; 2  1  4];
B = [ 5  ;  15  ;  8];
x1old = 0 ,x2old = 0 , x3old = 0 //intial assumption
    of x1,x2 & x3

disp('x1 = (5 - x2 - x3)/2 ')
disp('x2 = (15 - 3x1 - 2x3)/5 ')
disp('x3 = (8 - 2x1 - x2)/4')

for i = 1:4
    printf('\n Iteration Number : %d\n',i)

    x1 = (5 - x2old - x3old)/2 ;
    x2 = (15 - 3*x1old - 2*x3old)/5 ;
```

```
    x3 = (8 - 2*x1old - x2old)/4 ;

    printf(' \n x1 = %f\n x2 = %f\n x3 = %f\n',x1,x2
       ,x3)

    x1old = x1;
    x2old = x2;
    x3old = x3;

end
```

Scilab code Exa 8.2 Gauss Seidel Iterative Method

```
//Example No. 8_02
//Gauss Seidel
//Page No.261
clear ; close ; clc ;

A = [ 2  1  1 ; 3  5  2 ; 2  1  4];
B = [ 5  ;  15  ;  8];
x1old = 0 ,x2old = 0 , x3old = 0 //intial assumption

disp('(x1 = 5 - x2 - x3)/2 ')
disp('(x2 = 15 - 3x1 - 2x3)/5 ')
disp('(x3 = 8 - 2x1 - x2)/4')

for i = 1:2

    printf('\n Iteration Number : %d',i)

    x1 = (5 - x2old - x3old)/2 ;
    x1old = x1;
    x2 = (15 - 3*x1old - 2*x3old)/5 ;
    x2old = x2;
    x3 = (8 - 2*x1old - x2old)/4 ;
```

```
    x3old = x3;

     printf(' \n x1 = %f\n x2 = %f\n x3 = %f\n',x1,
        x2,x3)

end
```

Scilab code Exa 8.3 Gauss Seidel Iterative Method

```
//Example No. 8_03
//Gauss Seidel
//page no. 269
clear ; close ; clc ;

A = [ 3 1 ; 1 -3]
B = [ 5 ; 5 ]

disp('Using a matrix to display the results after
   each iteration, first row represents initial
   assumption')
X(1,1) = 0 , X(1,2) = 0 ;//initial assumption

maxit = 1000;//Maximum number of iterations
err = 0.0003 ;

disp('x1 = (5-x2)/3');
disp('x2 = (x1 - 5)/3');

for i = 2:maxit

    X(i,1) = (5 - X(i-1,2))/3 ;
    X(i,2) = (X(i,1) - 5)/3 ;

    //Error Calculations
    err1 =abs((X(i,1) - X(i-1,1))/X(i,1))
```

```
    err2 =abs((X(i,2)- X(i-1,2))/X(i,2))

    //Terminating Condition
    if err >= err1 & err >= err2  then
        printf('The system converges to the solution
            ( %f , %f ) in %d iterations\n',X(i,1),X
           (i,2),i-1 )
        break
    end

end
//calcution of true error i.e. difference between
   final result and results from each iteration
trueerr1 = abs(X(:,1) - X(i,1)*ones(i,1)) ;
trueerr2 = abs(X(:,2) - X(i,2)*ones(i,1)) ;

//displaying final results
D = [  X  trueerr1   trueerr2] ;
disp(D)
```

Scilab code Exa 8.4 Gauss Seidel Iterative Method

```
//Example No. 8_04
//Gauss Seidel
//Page No.261
clear ; close ; clc ;

A = [ 1 -3 ; 3 1 ];
B = [ 5  ;  5 ];
x1old = 0 ,x2old = 0  //intial assumption

disp('x1 = 5 + 3*x2 ')
disp('x2 = 5 - 3*x1 ')

for i = 1:3
```

```

    x1 = 5 + 3*x2old ;
    x1old = x1;
    x2 = 5 - 3*x1old ;
    x2old = x2;

     printf('\n Iteration : %i  x1 = %i and x2 = %i\n',i,x1,x2)

end
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

```
//Example No. 9_01
//Pg No.277
clear ; close ; clc ;

printf('solving linear equations \n a0 + 100a1 = 3/7
    \n a0 + 101a1 = -4/7 \n we get,\n');
C = [ 1 100 ; 1 101]
p = [ 3/7 ; -4/7]
a = C\p
printf('\n a0 = %f \n a1 = %f \n',a(1),a(2));
x = poly(0,'x') ;
px = a(1) + a(2)*x
p100 = horner(px,100)
p101 = horner(px,101)
printf('\n p(100) = %f \n p(101) = %f\n',p100,p101)
```

Scilab code Exa 9.2 Shifted Power form

```
//Example No. 9_02
//Page No. 278
clear ; close ; clc ;

C = [ 1 100-100 ; 1 101-100]
p = [ 3/7 ; -4/7]
a = C\p
printf('\n a0 = %f \n a1 = %f \n',a(1),a(2));
x = poly(0,'x') ;
px = a(1) + a(2)*(x - 100)
p100 = horner(px,100)
p101 = horner(px,101)
printf('\n p(100) = %f \n p(101) = %f\n',p100,p101)
```

Scilab code Exa 9.3 Linear Interpolation

```
//Example No. 9_03
//Page No. 280
clear ; close ; clc ;

x = 1:5
f = [1 1.4142 1.7321 2 2.2361]
n = 2.5
for i = 1:5
    if n <= x(i) then
        break ;
    end
end
printf('%f lies between points %i and %i',n,x(i-1),x
   (i))
f2_5 = f(i-1) + ( n - x(i-1) )*( f(i) - f(i-1) )/( x
   (i) - x(i-1) )
err1 = 1.5811 - f2_5
disp(f2_5,'f(2.5) = ')
disp(err1,'error1 = ')
```

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

Scilab code Exa 9.4 Lagrange Interpolation

```
//Example No. 9_04
//Lagrange Interpolation- Second order
//Pg No. 282
clear ; close ; clc ;

X = [ 1 2 3 4 5]
Fx = [ 1 1.4142 1.7321 2 2.2361];
X = X(2:4)
Fx = Fx(2:4)
x0 = 2.5
x = poly(0,'x')
p = 0
for i = 1:3
    L(i) = 1
    for j = 1:3
        if j == i then
            continue ;
```

```
        else
            L(i) = L(i)*( x - X(j) )/( X(i) - X(j) ) ;
        end
    end
    p = p + Fx(i)*L(i)
end
L0 = horner(L(1),2.5) ;
L1 = horner(L(2),2.5) ;
L2 = horner(L(3),2.5) ;
p2_5 = horner(p,2.5) ;
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)

err = sqrt(2.5) - p2_5 ;
printf('The error is %f',err);
```

Scilab code Exa 9.5 Lagrange Interpolation

```
//Example No. 9_05
//Lagrange Interpolation
//Pg No. 283
clear ; close ; clc ;

i = [ 0 1 2 3 ]
X = [ 0 1 2 3 ]
Fx = [ 0 1.7183 6.3891 19.0855 ]
x = poly(0,'x');
n = 3 //order of lagrange polynomial
p = 0
for i = 1:n+1
    L(i) = 1
    for j = 1:n+1
        if j == i then
```

```
            continue ;
        else
            L(i) = L(i)*( x - X(j) )/( X(i) - X(j) ) ;
        end
    end
    p = p + Fx(i)*L(i)
end
disp("The Lagrange basis polynomials are")
for i = 1:4
        disp(string(L(i)))
end
disp("The interpolation polynomial is ")
disp(string(p))
disp('The interpolation value at x = 1.5 is ' )
p1_5 = horner(p,1.5);
e1_5 = p1_5 + 1 ;
disp(e1_5,'e^1.5 = ',p1_5);
```

Scilab code Exa 9.6 Newton Interpolation

```
//Example No. 9_06
//Newton Interpolation - Second order
//Pg No. 288
clear ; close ; clc ;

i = [ 0 1 2 3]
X = 1:4
Fx = [ 0 0.3010 0.4771 0.6021]
X = 1:3
Fx = Fx(1:3)
x = poly(0,'x');
A = Fx'
for i = 2:3
    for j = 1:4-i
```

```
        A(j,i) = ( A(j+1,i-1)-A(j,i-1) )/(X(j+i-1)-X(j)) ;
    end
end
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))
p = A(1,1);
for i = 2:3
    p = p +A(1,i)* prod(x*ones(1,i-1) - X(1:i-1));
end
disp(string(p))
p2_5 = horner(p,2.5)
printf('p(2.5) = %.4G \n',p2_5)
```

Scilab code Exa 9.7 Newton Divided Difference Interpolation

```
//Example No. 9_07
//Newton Divided Difference Interpolation
//Pg No. 291
clear ; close ; clc ;

i = 0:4
X = 1:5
Fx = [ 0 7 26 63 124];
x = poly(0,'x');
A = [ i' X' Fx']
for i = 4:7
     for j = 1:8-i
        A(j,i) = ( A(j+1,i-1)-A(j,i-1) )/(X(j+i-3)-X(j)) ;
    end
end
disp(A)
p = A(1,3);
```

```
p1_5(1) = p ;
for i = 4:7
    p = p +A(1,i)* prod(x*ones(1,i-3) - X(1:i-3));
    p1_5(i-2) = horner(p,1.5);
end
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));
disp(p1_5(5),'The function value at x = 1.5 is')
```

Scilab code Exa 9.8 Newton Gregory Forward Difference Formula

```
//Example No. 9_08
//Newton-Gregory forward difference formula
//Pg No. 297
clear ; close ; clc ;

X = [ 10 20 30 40 50]
Fx = [ 0.1736 0.3420 0.5000 0.6428 0.7660]
x = poly(0,'x');
A = [X' Fx'];
for i = 3:6
     A(1:7-i,i) = diff(A(1:8-i,i-1))
end
disp(A)
x0 = X(1);
h = X(2) - X(1) ;
x1 = 25
s = (x1 - x0)/h ;
p(1) = Fx(1);
for j = 1:4
    p(j+1) = p(j) + prod(s*ones(1,j)-[0:j-1])*A(1,j
       +2)/factorial(j)
end
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))
printf(' Thus sin(%d) = %.4G \n ',x1,p(5))
```

Scilab code Exa 9.9 Newton Backward Difference Formula

```
//Example No. 9_09
//Newton Backward difference formula
//Pg No. 299
clear ;close ;clc ;

X = [ 10 20 30 40 50]
Fx = [ 0.1736 0.3420 0.5000 0.6428 0.7660]
x = poly(0,'x');
A = [X' Fx'];
for i = 3:6
     A(i-1:5,i) = diff(A(i-2:5,i-1))
end
disp(A);
xn = X(5);
h = 10 ;
xuk = 25;
s = (xuk - xn)/h ;
disp(s,'s = ');
p(1) = Fx(5)
for j = 1:4
    p(j+1) = p(j) + prod(s*ones(1,j)-[0:j-1])*A(5,j
       +2)/factorial(j)
end
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))
printf(' Thus sin(%d) = %.4G \n ',xuk,p(5))
```

Scilab code Exa 9.10 Splines

```
//Example No. 9_10
//Splines
//Pg No. 301
clear ; close ; clc ;

x = poly(0,'x');
function isitspline(f1,f2,f3,x0,x1,x2,x3)
    n1 = degree(f1),n2 = degree(f2),n3 = degree(f3)
    n = max(n1,n2,n3)
    f1_x1 = horner(f1,x1)
    f2_x1 = horner(f2,x1)
    f2_x2 = horner(f2,x2)
    f3_x2 = horner(f3,x2)
    if n ==1 & f1_x1 == f2_x1 & f2_x2 == f3_x2 then
        printf('The piecewise polynomials are continuous and f(x) is a linear spline')
    elseif f1_x1 == f2_x1 & f2_x2 == f3_x2
        for i = 1:n-1
            df1 = derivat(f1)
            df2 = derivat(f2)
            df3 = derivat(f3)
            df1_x1 = horner(df1,x1)
            df2_x1 = horner(df2,x1)
            df2_x2 = horner(df2,x2)
            df3_x2 = horner(df3,x2)
            f1 = df1, f2 = df2, f3 = df3
            if df1_x1 ~= df2_x1 | df2_x2 ~= df3_x2 then
                printf('The %ith derivative of polynomial is not continuours',i)
                break
            end
        end
        if i == n-1 & df1_x1 == df2_x1 & df2_x2 == df3_x2 then
            printf('The polynomial is continuous and its derivatives from 1 to %i are continuous, f(x) is a %ith degree
```

```
            polynomial',i,i+1)
        end
    else
            printf('The polynomial is not continuous
                ')
    end

endfunction
n = 4 , x0 = -1 , x1 = 0, x2 = 1 , x3 = 2
f1 = x+1 ;
f2 = 2*x + 1 ;
f3 = 4 - x ;
disp('case 1')
isitspline(f1,f2,f3,x0,x1,x2,x3)
n = 4, x0 = 0 , x1= 1 , x2 = 2 , x3 = 3
f1 = x^2 + 1 ;
f2 = 2*x^2 ;
f3 = 5*x - 2 ;
disp('case 2')
isitspline(f1,f2,f3,x0,x1,x2,x3)
n = 4, x0 = 0, x1 = 1, x2 = 2, x3 = 3
f1 = x,
f2 = x^2 - x + 1,
f3 = 3*x - 3
disp('case 3')
isitspline(f1,f2,f3,x0,x1,x2,x3)
```

Scilab code Exa 9.11 Cubic Spline Interpolation

```
//Example No. 9_11
//Cubic Spline Interpolation
//Pg No. 306
clear ; close ; clc ;

X = [ 4 9 16]
```

```
Fx = [ 2 3 4]
n = length(X)
h = diff(X)
disp(h)
x = poly(0,'x');
A(1) = 0;
A(n) = 0;

//Since we do not know only a1 = A(2) we just have one equation which can be solved directly without solving tridiagonal matrix
A(2) = 6*( ( Fx(3) - Fx(2) )/h(2) - ( Fx(2) - Fx(1) )/h(1) )/( 2*( h(1) + h(2) ) );
disp(A(2),'a1 = ');
xuk = 7;
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(2)*( u(i)^3 - ( h(i)^2 )*u(i))/6*h(i)  ) + ( Fx(i+1)*u(i)- Fx(i)*u(i+1) )/h(i);
disp(s,'s(x) = ');
s_7 = horner(s,xuk);
disp(s_7,'s(7)')
```

Scilab code Exa 9.12 Cubic Spline Interpolation

```
//Example No. 9_12
//Cubic Spline Interpolation
//Pg No. 313
clear ; close ;clc ;

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

```
//Example No. 10_01
//Fitting a Straight Line
//Pg No. 326
clear ;close ;clc ;

x = poly(0,'x')
X = 1:5
Y = [ 3 4 5 6 8 ];
n = length(X);
b = ( n*sum(X.*Y) - sum(X)*sum(Y) )/( n*sum(X.*X) -
    (sum(X))^2 )
a = sum(Y)/n - b*sum(X)/n
disp(b,'b = ')
disp(a,'a = ')
y = a + b*x
```

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

```
//Example No. 10_02
//Fitting a Power-Function model to given data
//Pg No. 331
clear ;close ;clc ;

x = poly(0,'x');
X = 1:5
Y = [ 0.5 2 4.5 8 12.5 ]
Xnew = log(X)
Ynew = log(Y)
n = length(Xnew)
b = ( n*sum(Xnew.*Ynew) - sum(Xnew)*sum(Ynew) )/( n*sum(Xnew.*Xnew) - ( sum(Xnew) )^2 )
lna = sum(Ynew)/n - b*sum(Xnew)/n
a = exp(lna)
disp(b,'b = ')
disp(lna,'lna = ')
disp(a,'a = ')
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

```
//Example No. 10_03
//Pg No. 332
clear ;close ;clc ;

time = 1:4
T = [ 70 83 100 124 ]
t = 6
Fx = exp(time/4)
n = length(Fx)
Y = T ;
b = ( n*sum(Fx.*Y) - sum(Fx)*sum(Y) )/( n*sum(Fx.*Fx) - (sum(Fx))^2 )
```

```
a = sum(Y)/n - b*sum(Fx)/n
disp(b,'b = ')
disp(a,'a = ')
printf('The relationship between T and t is \n T = %
    .4G*e^(t/4) + %.4G \n',b,a)
deff('T = T(t)','T = b*exp(t/4) + a ')
T_6 = T(6)

disp(T_6,'The temperature at t = 6 is')
```

Scilab code Exa 10.4 Curve Fitting

```
//Example No. 10_04
//Curve Fitting
//Pg NO. 335
clear ; close ; clc ;

x = 1:4 ;
y = [6 11 18 27 ];
n = length(x) //Number of data points
m = 2+1       //Number of unknowns
disp('Using CA = B form , we get')
for j = 1:m
    for k = 1:m
        C(j,k) = sum(x.^(j+k-2))
    end
    B(j) = sum( y.*( x.^( j-1 ) ) )
end
disp(B,'B = ',C,'C = ')
A = inv(C)*B
disp(A,'A = ')
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

```
//Example No. 10_05
//Plane Fitting
//Pg No. 342
clear ; close ; clc ;

x = 1:4
z = 0:3
y = 12:6:30
n = length(x) //Number of data points
m = 3         //Number of unknowns
G = [ ones(1,n) ; x ; z]
H = G'
C = G*H
B = y*H
D = C\B'
disp(C,B)
disp(D)
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

```
//Example No. 11_01
//First order forward difference
//Pg No. 348
clear ;close ;clc ;

x = poly(0,"x");
deff('F = f(x)','F = x^2');
deff('DF = df(x,h)','DF = (f(x+h)-f(x))/h');
dfactual = derivat(f(x));
h = [0.2 ; 0.1 ; 0.05 ; 0.01 ]
for i = 1:4
    y(i) = df(1,h(i));
    err(i) = y(i) - horner(dfactual,1)
end
table = [h y err];
disp(table)
```

Scilab code Exa 11.2 Three Point Formula

```
//Example No. 11_02
//Three-Point Formula
//Pg No. 350
clear ;close ;clc ;

x = poly(0,"x");
deff('F = f(x)','F = x^2');
deff('DF = df(x,h)','DF = (f(x+h)-f(x-h))/(2*h)');
dfactual = derivat(f(x));
h = [0.2 ; 0.1 ; 0.05 ; 0.01 ]
for i = 1:4
    y(i) = df(1,h(i));
    err(i) = y(i) - horner(dfactual,1)
end
table = [h y err];
disp(table)
```

Scilab code Exa 11.3 Error Analysis

```
//Example No. 11_03
//Pg No. 353
close ;clear ;clc ;

x = 0.45;
deff('F = f(x)','F = sin(x)');
deff('DF = df(x,h)','DF = (f(x+h) - f(x))/h');
dfactual =  cos(x);
h = 0.01:0.005:0.04;
n = length(h);
for i = 1:n
    y(i) = df(x,h(i))
    err(i) = y(i) - dfactual ;
end
table = [ h' y err];
disp(table)
```

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

Scilab code Exa 11.4 Approximate Second Derivative

```
//Example No. 11_04
//Approximate second derivative
//Pg No. 354
clear ;close ;clc ;

x = 0.75;
h = 0.01;
deff('F = f(x)','F = cos(x)');
deff('D2F = d2f(x,h)','D2F = ( f(x+h) - 2*f(x) + f(x-h) )/h^2 ');
y = d2f(0.75,0.01);
d2fexact = -cos(0.75)
err = d2fexact - y ;
disp(err,'err = ',d2fexact,'d2fexact = ',y,'y = ')
```

Scilab code Exa 11.5 Differentiation of Tabulated Data

```
//Example No. 11_05
//Differentiation of tabulated data
//Pg No. 358
clear ;close ;clc ;

T = 5:9 ;
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

```
//Example No. 9_01
//Richardson's Extrapolation Technique
//Pg No. 362
clear ;close ;clc ;

x = -0.5:0.25:1.5
h = 0.5 ;
r = 1/2 ;

deff('F = f(x)','F = exp(x)');
deff('D = D(x,h)' ,'D = [ f(x + h) - f(x-h) ]/(2*h)' );
deff('df = df(x,h,r)','df = [D(x,r*h) - r^2*D(x,h)]/(1-r^2)');

df_05 = df(0.5,0.5,1/2);
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) = ')
dfexact = derivative(f,0.5)
disp(dfexact,'Exact df(0.5) = ')
disp('The result by richardsons technique is much better than other results')

//r = 2
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

```
//Example No. 12_01
//Trapezoidal Rule
//Pg No. 373
clear ;close ;clc ;

x = poly(0,"x");
deff('F = f(x)','F = x^3 + 1');

//case(a)
a = 1;
b = 2 ;
h = b - a ;
It = (b-a)*(f(a)+f(b))/2
d2f = derivat(derivat( f(x) ))
Ett = h^3*horner(d2f,2)/12
Iexact = intg(1,2,f)
Trueerror = It - Iexact
disp(Trueerror,'True error = ',Iexact,'Iexact = ',
    Ett,'Ett = ',It,'It = ','case(a)')
disp('Here Error bound is an overestimate of true
    error')
```

```
//case(b)
a = 1;
b = 1.5 ;
h = b - a ;
It = (b-a)*(f(a)+f(b))/2
Ett = h^3*horner(d2f,1.5)/12
Iexact = intg(1,1.5,f)
Trueerror = It - Iexact
disp(Trueerror,'True error = ',Iexact,'Iexact = ',
   Ett,'Ett = ',It,'It = ','case(b)')
```

Scilab code Exa 12.2 Trapezoidal Rule

```
//Example No. 12_02
//Tapezoidal rule
//Pg No. 376
clear ;close ;clc ;

deff('F = f(x)','F = exp(x)');
a = -1 ;
b = 1 ;

//case(a)
n = 2
h = (b-a)/n
I = 0
for i = 1:n
    I = I + f(a+(i-1)*h)+f(a+i*h);
end
I = h*I/2 ;
disp(I,'intergral for case(a),Ia = ')

//case(b)
n = 4
```

```
h = (b-a)/n
I = 0
for i = 1:n
    I = I + f(a+(i-1)*h)+f(a+i*h);
end
I = h*I/2 ;
Iexact = 2.35040
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

```
//Example No. 12_03
//Simpon's 1/3 rule
//Pg No. 381
clear ;close ;clc ;

funcprot(0) //To avoid warning message for defining
   function f(x) twice
//case(a)
deff('F = f(x)','F = exp(x)');
a = -1;
b = 1;
h = (b-a)/2
x1 = a+h
Is1 = h*( f(a) + f(b) + 4*f(x1) )/3
disp(Is1,'Integral for case(a) , Is1 = ',h,'h = ')

//case(b)
deff('F = f(x)','F = sqrt(sin(x))');
a = 0
b = %pi/2
h = (b-a)/2
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

```
//Example No. 12_05
//Simpson's 3/8 rule
//Pg No. 386
clear ;close ;clc ;

funcprot(0)
//case(a)
deff('F = f(x)','F = x^3 + 1');
a = 1 ;
b = 2 ;
h = (b-a)/3
x1 = a + h
x2 = a + 2*h
Is2 = 3*h*( f(a) + 3*f(x1) + 3*f(x2) + f(b) )/8 ;
disp(Is2,'Integral of x^3 +1 from 1 to 2 is')
//case(b)
deff('F = f(x)','F = sqrt( sin(x) )');
a = 0 ;
b = %pi/2 ;
h = (b-a)/3
x1 = a + h
x2 = a + 2*h
Is2 = 3*h*( f(a) + 3*f(x1) + 3*f(x2) + f(b) )/8 ;
disp(Is2,'Integral of sqrt(sin(x)) from 0 to %pi/2
    is')
```

Scilab code Exa 12.6 Booles Five Point Formula

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

```
//Booles's Five-Point formula
//Pg No. 387
clear ;close ;clc ;

deff('F = f(x)','F = sqrt( sin(x) )')
x0 = 0;
xb = %pi/2 ;
n = 4 ;
h = (xb - x0)/n
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;
disp(Ib,'Ib = ')
```

Scilab code Exa 12.7 Romberg Integration Formula

```
//Example No. 12_07
//Romberg Integration formula
//Pg No. 391
clear ;close ;clc ;

deff('F = f(x)','F = 1/x');
//since we can't have (0,0) element in matrix we
   start with (1,1)
a = 1 ;
b = 2 ;
h = b-a ;
R(1,1) = h*(f(a)+f(b))/2
disp(R(1,1),'R(0,0) = ')
for i = 2:3
    h(i) = (b-a)/2^(i-1)
    s = 0
    for k = 1:2^(i-2)
        s = s + f(a + (2*k - 1)*h(i));
    end
    R(i,1) = R(i-1,1)/2 + h(i)*s;
```

```
    printf('\nR(%i,0) = %f \n',i-1,R(i,1))
end
for j = 2:3
    for i = j:3
        R(i,j) = (4^(j-1)*R(i,j-1) - R(i-1,j-1))/(4^(j-1)-1);
        printf('\nR(%i,%i) = %f \n',i-1,j-1,R(i,j))
    end
end
```

Scilab code Exa 12.8 Two Point Gauss Legefre Formula

```
//Example No. 12_08
//Two Point Gauss -Legedre formula
//Pg No. 397
clear ;close ;clc ;

deff('F = f(x)','F = exp(x)');
x1 = -1/sqrt(3)
x2 = 1/sqrt(3)
I = f(x1) + f(x2)
disp(I,'I = ',x2,'x2 = ',x1,'x1 = ')
```

Scilab code Exa 12.9 Gaussian Two Point Formula

```
//Example No. 12_09
//Gaussian two point formula
//Pg No. 398
clear ;close ;clc ;

a = -2 ;
b = 2 ;
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

```
//Example No. 13_01
//Taylor method
//Pg No. 414
clear ; close ; clc ;

deff('F = f(x,y)','F = x^2 + y^2')
deff('D2Y = d2y(x,y)','D2Y = 2*x + 2*y*f(x,y)');
deff('D3Y = d3y(x,y)','D3Y = 2 + 2*y*d2y(x,y) + 2*f(x,y)^2');
deff('Y = y(x)','Y = 1 + f(0,1)*x + d2y(0,1)*x^2/2 + d3y(0,1)*x^3/6');
disp(y(0.25),'y(0.25) = ')
disp(y(0.5),'y(0.5) = ')
```

Scilab code Exa 13.2 Recursive Taylor Method

```
//Example No. 13_02
//Recursive Taylor Method
//Pg No. 415
clear ; close ; clc ;

deff('F = f(x,y)','F = x^2 + y^2')
deff('D2Y = d2y(x,y)','D2Y = 2*x + 2*y*f(x,y)');
deff('D3Y = d3y(x,y)','D3Y = 2 + 2*y*d2y(x,y) + 2*f(x,y)^2');
deff('D4Y = d4y(x,y)','D4Y = 6*f(x,y)*d2y(x,y) + 2*y*d3y(x,y) ');
h = 0.2 ;
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)');
x0 = 0;
y0 = 0 ;
for i = 1:2
    y_(i) = y(x0,y0)
   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))
    x0 = x0 + i*h
    y0 = y_(i)
   printf('y(0) = %f\n\n',y_(i))
end
```

Scilab code Exa 13.3 Picards Method

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

```
//y(1) = x^3/3
//y(2) = 0 + integral(xY2 + y1^2,x0,x)
//     = integral(x^2 + x^6/9,0,x) = x^3/3 + x^7/63
//therefore y(x) = x^3 /3 + x^7/63
deff('Y = y(x)','Y = x^3/3 + x^7/63 ')
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 ')

//y'(x) = x*e^y, y(0) = 0
//y0 = 0 , x0 = 0
//Y(1) = 0 + integral(x*e^0,0,x) = x^2/2
//y(2) = 0 + integral( x*e^( x^2/2 ) ,0,x) = e^(x^2/2)-1
//therefore y(x) = e^(x^2/2) - 1
deff('Y = y(x)','Y = exp(x^2/2) - 1 ')
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

```
//Example No. 13_04
//Euler's Method
//Pg No. 417
clear ; close ; clc ;

deff('DY = dy(x)','DY = 3*x^2 + 1')
x0 = 1
y(1) = 2 ;
//h = 0.5
h = 0.5
mprintf('for h = %f\n',h)
for i = 2 : 3
    y(i) = y(i-1) + h*dy(x0+(i-2)*h)
    mprintf('y(%f) = %f\n',x0+(i-1)*h,y(i))
```

```
end
//h = 0.25
h = 0.25
mprintf('\nfor h = %f\n',h)
for i = 2 : 5
    y(i) = y(i-1) + h*dy(x0+(i-2)*h)
    mprintf('y(%f) = %f\n',x0+(i-1)*h,y(i))
end
```

Scilab code Exa 13.5 Error Estimation in Eulers Method

```
//Example No. 13_05
//Error estimation in Euler's Method
//Pg No. 422
clear ; close ; clc ;

deff('DY = dy(x)','DY = 3*x^2 + 1')
deff('D2Y = d2y(x)','D2Y = 6*x')
deff('D3Y = d3y(x)','D3Y = 6')
deff('exacty = exacty(x)','exacty = x^3 + x')
x0 = 1
y(1) = 2
h = 0.5
for i = 2 : 3
    x(i-1) = x0 + (i-1)*h
    y(i) = y(i-1) + h*dy(x0+(i-2)*h)
    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))
    Et(i-1) = d2y(x0+(i-2)*h)*h^2/2 +  d3y(x0+(i-2)*
       h)*h^3/6
    mprintf('\n Et(%i) = %f\n',i-1,Et(i-1))
    truey(i-1) = exacty(x0+(i-1)*h)
    gerr(i-1) = truey(i-1) - y(i)
end

```

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

Scilab code Exa 13.6 Heuns Method

```
//Example No. 13_06
//Heun's Method
//Pg No. 427
clear ; close ;clc ;

deff('F = f(x,y)','F = 2*y/x ')
deff('exacty = exacty(x)','exacty = 2*x^2')
x(1) = 1 ;
y(1) = 2 ;
h = 0.25 ;
//Euler's Method
disp('EULERS METHOD')
for i = 2:5
    x(i) = x(i-1) + h ;
    y(i) = y(i-1) + h*f(x(i-1),y(i-1));
    mprintf('y(%f) = %f \n ',x(i),y(i))
end
eulery = y
//Heun's Method
disp('HEUNS METHOD')
for i = 2:5
    m1 = f(x(i-1),y(i-1)) ;
    ye(i) = y(i-1) + h*f(x(i-1),y(i-1));
    m2 = f(x(i),ye(i)) ;
    y(i) = y(i-1) + h*(m1 + m2)/2
   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))
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 ;
```

```
for i = 1:2
    m1 = f(  x(i)   ,  y(i)   ) ;
    m2 = f(  x(i) + h/2   ,  y(i) + m1*h/2   ) ;
    m3 = f(  x(i) + h/2   ,  y(i) + m2*h/2   ) ;
    m4 = f(  x(i) + h   ,  y(i) + m3*h   ) ;
    x(i+1) = x(i) + h ;
    y(i+1) = y(i) +  (m1 + 2*m2 + 2*m3 + m4)*h/6 ;

    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))
end
```

Scilab code Exa 13.9 Optimum Step size

```
//Example No. 13_09
//Optimum step size
//Pg No. 444
clear ; close ; clc ;

x = 0.8 ;
h1 = 0.05 ;
y1 = 5.8410870 ;
h2 = 0.025 ;
y2 = 5.8479637 ;

//d = 4
h = ((h1^4 - h2^4)*10^(-4)/(2*(y2 - y1)))^(1/4)
disp(h,'h = ','for four decimal places')

//d = 6
h = ((h1^4 - h2^4)*10^(-6)/(2*(y2 - y1)))^(1/4)
disp(h,'h = ','for six decimal places')
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

```
//Example No. 13_10
//Milne-Simpson Predictor-Corrector method
//Pg NO. 446
clear;close;clc;

deff('F = f(x,y)','F = 2*y/x')
x0 = 1 ;
y0 = 2 ;
h = 0.25 ;
//Assuming y1 ,y2 and y3(required for milne-simpson
    formula) are estimated using Fourth- order Runge
    kutta method
x1 = x0 + h
y1 = 3.13 ;
x2 = x1 + h
y2 = 4.5 ;
x3 = x2 + h
y3 = 6.13 ;
//Milne Predictor formula
yp4 = y0 + 4*h*(2*f(x1,y1) - f(x2,y2) + 2*f(x3,y3))
    /3
x4 = x3 + h
fp4 = f(x4,yp4) ;
disp(fp4,'fp4 = ',yp4,'yp4 = ')
//Simpson Corrector formula
yc4 = y2 + h*( f(x2,y2) + 4*f(x3,y3) + fp4)/3
f4 = f(x4,yc4)
disp(f4,'f4 = ',yc4,'yc4 = ')

yc4 = y2 + h*( f(x2,y2) + 4*f(x3,y3) + f4)/3
disp(yc4 ,'yc4 = ')
```

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

Scilab code Exa 13.11 Adams Bashforth Moulton Method

```
//Example No. 13_11
//Adams-Bashforth-Moulton Method
//Pg NO. 446
clear;close;clc;

deff('F = f(x,y)','F = 2*y/x')
x0 = 1 ;
y0 = 2 ;
h = 0.25 ;
x1 = x0 + h
y1 = 3.13 ;
x2 = x1 + h
y2 = 4.5 ;
x3 = x2 + h
y3 = 6.13 ;
//Adams Predictor formula
yp4 = y3 + h*(55*f(x3,y3) - 59*f(x2,y2) + 37*f(x1,y1
   ) - 9*f(x0,y0))/24
x4 = x3 + h
fp4 = f(x4,yp4)
disp(fp4,'fp4 = ',yp4,'yp4 = ','Adams Predictor
   formula')
//Adams Corrector formula
yc4 = y3 + h*( f(x1,y1) - 5*f(x2,y2) + 19*f(x3,y3) +
    9*fp4 )/24
f4 = f(x4,yc4)
disp(f4,'f4 = ',yc4,'yc4 = ','Adams Corrector
   formula')

yc4 = y3 + h*( f(x1,y1) - 5*f(x2,y2) + 19*f(x3,y3) +
    9*f4 )/24
```

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

Scilab code Exa 13.12 Milne Simpson Method Using Modifier

```
//Example No. 13_12
//Milne-Simpson Method using modifier
//Pg No. 453
clear ; close ; clc ;

deff('F = f(y)','F = -y^2 ')
x = [ 1 ; 1.2 ; 1.4 ; 1.6 ] ;
y = [ 1 ; 0.8333333 ; 0.7142857 ; 0.625 ] ;
h = 0.2 ;

for i = 1:2
    yp = y(i) + 4*h*( 2*f( y(i+1) ) - f( y(i+2) ) + 2*f( y(i+3) ) )/3
    fp = f(yp) ;
    yc = y( i+2) + h*(f( y(i+2) ) + 4*f( y(i+3) ) + fp )/3 ;
    Etc = -(yc - yp)/29
    y(i+4) = yc + Etc
    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))
end
exactanswer = 0.5 ;
err = exactanswer - y(6) ;
disp(err,'error = ')
```

Scilab code Exa 13.13 System of Differential Equations

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

```
//System of differetial Equations
//Pg No. 455
clear ; close ; clc ;

deff('F1 = f1(x,y1,y2)','F1 = x + y1 + y2 ')
deff('F2 = f2(x,y1,y2)','F2 = 1 + y1 + y2 ')

x0 = 0 ;
y10 = 1 ;
y20 = -1 ;
h = 0.1 ;
m1(1) = f1( x0 , y10 , y20 )
m1(2) = f2( x0 , y10 , y20 )
m2(1) = f1( x0+h , y10 + h*m1(1) , y20 + h*m1(2) )
m2(2) = f2( x0+h , y10 + h*m1(1) , y20 + h*m1(2) )
m(1) = (m1(1) + m2(1))/2
m(2) = (m1(2) + m2(2))/2

y1_0_1 = y10 + h*m(1)
y2_0_1 = 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_1,y2_0_1)
```

Scilab code Exa 13.14 Higher Order Differential Equations

```
//Example No. 13_14
//Higher Order Differential Equations
//Pg No. 457
clear ; close ; clc ;

x0 = 0
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

```
//Example No. 14_01
//Shooting Method
//Pg No. 467
clear ; close ; clc ;

function [B,y] = heun(f,x0,y0,z0,h,xf)
    x(1) = x0 ;
    y(1) = y0 ;
    z(1) = z0 ;
    n = (xf - x0)/h
    for i = 1:n
        m1(1) = z(i) ;
        m1(2) = f(x(i),y(i))
        m2(1) = z(i) + h*m1(2)
        m2(2) = f(x(i)+h,y(i)+h*m1(1))
        m(1) = (m1(1) + m2(1))/2
        m(2) = ( m1(2) + m2(2) )/2
        x(i+1) = x(i) + h
        y(i+1) = y(i) + h*m(1)
```

```
        z(i+1) = z(i) + h*m(2)
    end
    B = y(n+1)
endfunction

deff('F = f(x,y)','F = 6*x')
x0 = 1 ;
y0 = 2 ;
h = 0.5 ;
z0 = 2
M1 = z0
xf = 2
B = 9
[B1,y] = heun(f,x0,y0,z0,h,xf)
disp(B1,'B1 = ')
if B1 ~= B then
    disp('Since B1 is less than B , let z(1) = y(1)
       = 4*(M2)')
    z0 = 4
    M2 = z0
    [B2,y] = heun(f,x0,y0,z0,h,xf)
    disp(B2,'B2 = ')
    if B2 ~= B then
        disp('Since B2 is larger than B ,let us have
            third estimate of z(1) = M3 ')
        M3 = M2 - (B2 - B)*(M2 - M1)/(B2 - B1)
        z0 = M3
        [B3,y] = heun(f,x0,y0,z0,h,xf)
        disp(y,'The solution is ',B3,'B3 = ')
    end
end
```

Scilab code Exa 14.2 Finite Difference Method

```
//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)
```

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

Scilab code Exa 14.4 Fadeev Leverrier Method

```
//Example No. 14_04
//Fadeev - Leverrier method
//Pg No. 474
clear ; close ; clc ;

A = [ -1 0 0 ; 1 -2 3 ; 0 2 -3 ]
[r,c] = size(A)
A1 = A
p(1) = trace(A1)
for i = 2:r
    A1 = A*( A1 - p(i-1)*eye())
    p(i) = trace(A1)/i
    mprintf('\nA%i = ',i)
    disp(A1)
    mprintf('\np%i = %f\n',i,p(i))
end
x = poly(0,'x');
p = p($:-1:1)
polynomial = poly([-p ; 1],'x','coeff')
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

```
//Example No. 15_01
//Elliptic Equations
//Pg No. 488
clear ; close ; clc ;

l = 15
h = 5
n = 1 + 15/5
f(1,1:4) = 100 ;
f(1:4,1) = 100 ;
f(4,1:4) = 0 ;
f(1:4,4) = 0 ;

//At point 1 :  f2 + f3 - 4f1 + 100 + 100 = 0
//At point 2 :  f1 + f4 - 4f2 + 100 +   0 = 0
//At point 3 :  f1 + f4 - 4f3 + 100 +   0 = 0
//At point 4 :  f2 + f3 - 4f4 +   0 +   0 = 0
//
//Final Equations are
```

```
//          −4f1 + f2 + f3 + 0 = −200
//            f1 − 4f2 + 0 + f4 = −100
//            f1 + 0 − 4f3 + f4 = −100
//            0  + f2 + f3 − 4f4 = 0
A = [ -4 1 1 0 ; 1 -4 0 1 ; 1 0 -4 1 ; 0 1 1 -4 ]
B = [-200 ; -100 ; -100 ; 0]
C = A\B
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

```
//Example No. 15_02
//Liebmann's Iterative method
//Pg No. 489
clear ; close ; clc ;

f(1,1:4) = 100 ;
f(1:4,1) = 100 ;
f(4,1:4) = 0 ;
f(1:4,4) = 0 ;
f(3,3) = 0
for n = 1:5
    for i = 2:3
        for j = 2:3
            if n == 1 & i == 2 & j == 2 then
                f(i,j) = ( f(i+1,j+1) + f(i-1,j-1) +
                    f(i-1,j+1) + f(i+1,j-1) )/4
            else
                f(i,j) = ( f(i+1,j) + f(i-1,j) + f(i
                   ,j+1) + f(i,j-1) )/4
            end
        end
    end
```

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

Scilab code Exa 15.3 Poissons Equation

```
//Example No. 15_03
//Poisson's Equation
//Pg No. 490
clear ; close ; clc ;

//D2f = 2*x^2 * y^2
// f = 0
// h = 1
//Point 1 : 0 + 0 + f2 + f3 - 4f1 = 2(1)^2 * 2^2
//          f2 + f3 - 4f1 = 8
//Point 2 : 0 + 0 + f1 + f4 -4f2 = 2*(2)^2*2^2
//           f1 - 4f2 = f4 = 32
//Point 3 : 0 + 0 + f1 + f4 - 4f4 = 2*(1^2)*1^2
//           f1 -4f3 + f4 = 2
//Point 4 : 0 + 0 + f2 + f3 - 4f4 = 2* 2^2 * 1^2
//            f2 + f3 - 4f4 = 8
//Rearranging the equations
//            -4f1 + f2 + f3 = 8
//             f1 - 4f2 + f4 = 32
//             f1 - 4f3 + f4 = 2
//              f2 + f3 - 4f4 = 8
A = [ -4 1 1 0 ; 1 -4 0 1 ; 1 0 -4 1 ; 0 1 1 -4]
B = [ 8 ; 32 ; 2 ; 8 ]
C = A\B ;
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

```
//Example No. 15_04
//Gauss-Seidel Iteration
//Pg No. 491
clear ; close ; clc ;

f2 = 0
f3 = 0
for i = 1:4
    f1 = (f2 + f3 - 8)/4
    f4 = f1
    f2 = (f1 + f4 -32)/4
    f3 = (f1 + f4 - 2)/4
    mprintf('\nIteration %i\n f1 = %f,     f2 = %f,
        f3 = %f,     f4 = %f\n',i,f1,f2,f3,f4)
end
```

Scilab code Exa 15.5 Initial Value Problems

```
//Example No. 15_05
//Initial Value Problems
//Pg No. 494
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(1:7,1) = 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

```
//Example No. 15_07
//Hyperbolic Equations
//Pg No. 500
clear ; close ;clc ;

h = 1
Tbyp = 4
tau = sqrt(h^2 /4)
r = 1+(2.5 - 0)/tau
c = 1+(5 - 0)/h
for i = 2:c-1
    f(1,i) = (i-1)*(5 - (i-1) )
end
f(1:r,1) = 0
f(1:r,c) = 0
for i = 2:c-1
    g(i) = 0
    f(2,i) = (f(1,i+1) + f(1,i-1))/2 + tau*g(i)
end
for j = 2:r-1
    for i = 2:c-1
        f(j+1,i) = -f(j-1,i) + f(j,i+1) + f(j,i-1)
    end
end
disp(f,'The values estimated are ')
```