Scilab Textbook Companion for Numerical Methods For Scientists And Engineers by K. S. Rao[1]

Created by
Viswanath Pasumarthi
Numerical Methods
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 For Scientists And Engineers

Author: K. S. Rao

Publisher: PHI Learning Pvt. Ltd., New Delhi

Edition: 2

Year: 2004

ISBN: 81-203-2395-5

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

Basics in Computing

Scilab code Exa 1.1 Conversion of Decimal to Binary

```
//Example 1.1
clc
clear

dec_N = 47;
bin_N = dec2bin(dec_N)
disp(bin_N)
```

Scilab code Exa 1.2 Conversion of Binary to Decimal

```
//Example 1.2
clc
clear

dec = 0.7625;
iter = 1;
while(1)
    dec = 2 * dec;
```

```
    p(iter) = int(dec);
    dec = dec - int(dec);
    if iter == 8 then
        break
    end
    iter = iter + 1;
end
a = strcat(string(p));
bin = strcat(['0.',a])
disp(bin)
```

Scilab code Exa 1.3 Conversion of Decimal to Binary and Octal

```
//Example 1.3
clc
clear

dec_N = 59;
bin_N = dec2bin(dec_N)
oct_N = dec2oct(dec_N)
disp(bin_N,"Binary:")
disp(oct_N,"Octal:")
```

Chapter 2

Solution of Algebraic and Transcendental Equations

Scilab code Exa 2.1 `Root using Bisection Method`

```
//Example 2.1
clc
clear

function [root] = Bisection(fun,x,tol,maxit)
// Bisection: Computes roots of the function in the
   given range using Bisection Method
//// Input: Bisection(fun,x,tol,maxit)
// fun = function handle
// x = range in between sign change is evident
// tol = Maximum error between iterations that can
   be tolerated
// maxit = Maximum number of iterations
//// Output: [root]
// Root: Root of the given function in defined range

if fun(x(1)) > 0 then
    xu = x(1);    xl = x(2);
else
```

```
    xu = x(2);    xl = x(1);
end

Ea = 1;
iter = 1;

while(1)
    xr(iter) = (xl(iter) + xu(iter)) / 2;
    if fun(xr(iter)) > 0 then
        xu(iter+1) = xr(iter);
        xl(iter+1) = xl(iter);
    elseif fun(xr(iter)) < 0 then
        xl(iter+1) = xr(iter);
        xu(iter+1) = xu(iter);
    else
        break
    end

    if iter>1 then
        Ea(iter) = 100 * abs((xr(iter) - xr(iter-1))
            / xr(iter));
    end

    if Ea(iter) < tol | iter == maxit then
        break
    end
    iter = iter + 1;
end
root = xr(iter);
endfunction

function f = fun1(x)
    f = x.^3 -9*x + 1;
endfunction

x = [2 4];
tol = 1e-4;
maxit = 5;
```

```
root = Bisection(fun1,x,tol,maxit);
disp(root,"root = ")
```

Scilab code Exa 2.2 Root using Regula Falsi Method

```
//Example 2.2
clc
clear

function [root] = FalsePosition(fun,x,tol,maxit)
// FalsePosition: Computes roots of the function in
    the given range using False Position Method
//// Input: FalsePosition(fun,x,tol,maxit)
// fun = function handle
// x = range in between sign change is evident
// tol = Maximum error between iterations that can
    be tolerated
// maxit = Maximum number of iterations
//// Output: [root]
// Root: Root of the given function in defined range

if fun(x(1)) > 0 then
    xu = x(1);    xl = x(2);
else
    xu = x(2);    xl = x(1);
end

Ea = 1;
iter = 1;

while(1)
    xr(iter) = xl(iter) - ((xu(iter)-xl(iter)) / (
        fun(xu(iter))-fun(xl(iter))) * fun(xl(iter)))
        ;
    if fun(xr(iter)) > 0 then
```

```
        xu(iter+1) = xr(iter);
        xl(iter+1) = xl(iter);
    elseif fun(xr(iter)) < 0 then
        xl(iter+1) = xr(iter);
        xu(iter+1) = xu(iter);
    else
        break
    end

    if iter>1 then
        Ea(iter) = 100 * abs((xr(iter) - xr(iter-1)) / xr(iter));
    end

    if Ea(iter) < tol | iter == maxit then
        break
    end
    iter = iter + 1;
end
root = xr(iter);
endfunction

function f = fun1(x)
    f = x.^3 -9*x + 1;
endfunction

x = [2 4; 2 3];
tol = 1e-4;
maxit = 3;
for i = 1:2
    root = FalsePosition(fun1,x(i,:),tol,maxit);
    root = round(root*10^5)/10^5;
    disp(strcat(["root(",string(i),") = ",string(root)]))
end
```

Scilab code Exa 2.3 Root using Regula Falsi Method

```
//Example 2.3
clc
clear

function [root] = FalsePosition(fun,x,tol,maxit)
// FalsePosition: Computes roots of the function in the given range using False Position Method
//// Input: FalsePosition(fun,x,tol,maxit)
// fun = function handle
// x = range in between sign change is evident
// tol = Maximum error between iterations that can be tolerated
// maxit = Maximum number of iterations
//// Output: [root]
// Root: Root of the given function in defined range

if fun(x(1)) > 0 then
    xu = x(1);    xl = x(2);
else
    xu = x(2);    xl = x(1);
end

Ea = 1;
iter = 1;

while(1)
    xr(iter) = xl(iter) - ((xu(iter)-xl(iter)) / (fun(xu(iter))-fun(xl(iter))) * fun(xl(iter)));
    if fun(xr(iter)) > 0 then
        xu(iter+1) = xr(iter);
        xl(iter+1) = xl(iter);
```

```
    elseif fun(xr(iter)) < 0 then
        xl(iter+1) = xr(iter);
        xu(iter+1) = xu(iter);
    else
        break
    end

    if iter>1 then
        Ea(iter) = 100 * abs((xr(iter) - xr(iter-1))
            / xr(iter));
    end

    if Ea(iter) < tol | iter == maxit then
        break
    end
    iter = iter + 1;
end
root = xr(iter);
endfunction

function f = fun3(x)
    f = log(x) - cos(x);
endfunction

x = [1 2];
tol = 1e-4;
maxit = 5;
root = FalsePosition(fun3,x,tol,maxit);
disp(round(root*10^4)/10^4,"root = ")
```

Scilab code Exa 2.4 Root using Regula Falsi Method

```
//Example 2.4
clc
clear
```

```
function [root] = FalsePosition(fun,x,tol,maxit)
// FalsePosition: Computes roots of the function in the given range using False Position Method
//// Input: FalsePosition(fun,x,tol,maxit)
// fun = function handle
// x = range in between sign change is evident
// tol = Maximum error between iterations that can be tolerated
// maxit = Maximum number of iterations
//// Output: [root]
// Root: Root of the given function in defined range

if fun(x(1)) > 0 then
    xu = x(1);    xl = x(2);
else
    xu = x(2);    xl = x(1);
end

Ea = 1;
iter = 1;

while(1)
    xr(iter) = xl(iter) - ((xu(iter)-xl(iter)) / (fun(xu(iter))-fun(xl(iter))) * fun(xl(iter)));
    if fun(xr(iter)) > 0 then
        xu(iter+1) = xr(iter);
        xl(iter+1) = xl(iter);
    elseif fun(xr(iter)) < 0 then
        xl(iter+1) = xr(iter);
        xu(iter+1) = xu(iter);
    else
        break
    end

    if iter>1 then
        Ea(iter) = 100 * abs((xr(iter) - xr(iter-1))
```

```
              / xr(iter));
    end

    if Ea(iter) < tol | iter == maxit then
        break
    end
    iter = iter + 1;
end
root = xr(iter);
endfunction

function f = fun4(x)
    f = x.*log10(x) - 1.2;
endfunction

clc
x = [2 3];
tol = 1e-4;
maxit = 2;
root = FalsePosition(fun4,x,tol,maxit);
disp(round(root*10^4)/10^4,"root = ")
```

Scilab code Exa 2.5 Root using Method of Iteration

```
//Example 2.5
clc
clear

function f = fun5(x)
    f = exp(-x)/10;
endfunction

clc
tol = 1e-4;
maxit = 4;
```

```
xold = 0;
iter = 1;
while(1)
    xnew = fun5(xold);
    EA = abs(xnew - xold);
    if EA < tol | iter > maxit then
        break
    end
    xold = xnew;
    iter = iter + 1;
end
root = round(xnew*10^4) / 10^4;      //rounded to 4
   decimal places
disp(root,"root = ")
```

Scilab code Exa 2.6 Root using Method of Iteration

```
//Example 2.6
clc
clear

function f = fun6(x)
    f = 1./ sqrt(x+1);
endfunction

tol = 1e-4;
maxit = 6;
xold = 1;
iter = 1;
while(1)
    xnew = fun6(xold);
    EA = abs(xnew - xold);
    if EA < tol | iter > maxit then
        break
    end
```

```
    xold = xnew;
    iter = iter + 1;
end
root = round(xnew*10^4) / 10^4;      //rounded to 4
   decimal places
disp(root,"root = ")
```

Scilab code Exa 2.7 `Root using Newton Raphson Method`

```
//Example 2.7
clc
clear

function [f,df] = fun7(x)
    f  = x.*exp(x) - 2;
    df = x.*exp(x) + exp(x);
endfunction

xold = 1;
maxit = 2;
iter = 1;

while (1)
    [fx,dfx] = fun7(xold);
    xnew = xold - fx/dfx;
    if iter == maxit then
        break
    end
    xold = xnew;
    iter = iter + 1;
end
root = round(xnew*10^3) / 10^3;
disp(root,"root = ")
```

Scilab code Exa 2.8 Root using Newton Raphson Method

```
//Example 2.8
clc
clear

function [f,df] = fun8(x)
    f  = x.^3 - x - 1;
    df = 3*x.^2 - 1;
endfunction

xold = 1;
maxit = 5;
iter = 1;

while (1)
    [fx,dfx] = fun8(xold);
    xnew = xold - fx/dfx;
    if iter == maxit then
        break
    end
    xold = xnew;
    iter = iter + 1;
end
root = round(xnew*10^4) / 10^4;
disp(root,"root = ")
```

Scilab code Exa 2.9 Newton Scheme of Iteration

```
// Example 2.9
// This is an analytical problem and need not be
   coded.
```

Scilab code Exa 2.10 Newton Formula

```
//Example 2.10
clc
clear

N = 12;
xold = 3.5;
iter = 1;
maxit = 3;

while (1)
    xnew = (xold + N/xold) / 2;
    if iter == maxit then
        break
    end
    xold = xnew;
    iter = iter + 1;
end
root = round(xnew*10^4) / 10^4;
disp(root,"root = ")
```

Scilab code Exa 2.11 Newton Raphson Extended Formula

```
// Example 2.11
// This is an analytical problem and need not be
   coded.
```

Scilab code Exa 2.12 Root using Muller Method

```
//Example 2.12
clc
clear

function [f] = fun12(x)
    f  = x.^3 - x - 1;
endfunction

x = [0 1 2];
h = [x(2)-x(1) x(3)-x(2)];
lamdai = h(2)/h(1);
deli = 1 + lamdai;
f = fun12(x);

g = f(1)*lamdai^2 - f(2)*deli^2 + f(3)*(lamdai +
   deli);
lamda = -2*f(3)*deli / (g + sqrt(g^2 - 4*f(3)*deli*(
   f(1)*lamdai - f(2)*deli + f(3))));
xnew = x(3) + lamda*h(2);
xnew = round(xnew*10^5) / 10^5;
disp(xnew,"root = ")
```

Scilab code Exa 2.13 Graeffe Root Squaring Method

```
//Example 2.13
clc
clear

a = [-6 11 -6 1];
maxit = 3;
for iter = 1:maxit
    a = [a(4)^2 -(a(3)^2 -2*a(2)*a(4)) (a(2)^2 - 2*a
       (1)*a(3)) -a(1)^2];
    root = abs([a(4)/a(3) a(3)/a(2) a(2)/a(1)])
       ^(1/(2^iter));
```

```
end
root = round(root*10^5) / 10^5;
disp(root,"Estimated roots for the polynomial are: "
    )
```

Chapter 3

Solution of Linear System of Equations and Matrix Inversion

Scilab code Exa 3.1 Gauss Elimination Method

```
//Example 3.1
clc
clear

A = [2 3 -1; 4 4 -3; -2 3 -1];  //Coefficient Matrix
B = [5; 3; 1];  //Constant Matrix

n = length(B);
Aug = [A,B];

// Forward Elimination
for j = 1:n-1
    for i = j+1:n
        Aug(i,j:n+1) = Aug(i,j:n+1) - Aug(i,j) / Aug
           (j,j) * Aug(j,j:n+1);
    end
end

// Backward Substitution
```

```
x = zeros(n,1);
x(n) = Aug(n,n+1) / Aug(n,n);
for i = n-1:-1:1
    x(i) = (Aug(i,n+1)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,
        i);
end
disp(strcat(["x = ",string(x(1))]))
disp(strcat(["y = ",string(x(2))]))
disp(strcat(["z = ",string(x(3))]))
```

Scilab code Exa 3.2 Gauss Elimination Method with Partial Pivoting

```
//Example 3.2
clc
clear

A = [1 1 1; 3 3 4; 2 1 3];  //Coefficient Matrix
B = [7; 24; 16];  //Constant Matrix

n = length(B);
Aug = [A,B];

// Forward Elimination
for j = 1:n-1
    // Partial Pivoting
    [dummy,t] = max(abs(Aug(j:n,j)));
    lrow = t(1)+j-1;
    Aug([j,lrow],:) = Aug([lrow,j],:);

    for i = j+1:n
        Aug(i,j:n+1) = Aug(i,j:n+1) - Aug(i,j) / Aug
            (j,j) * Aug(j,j:n+1);
    end
end

```

```
// Backward Substitution
x = zeros(n,1);
x(n) = Aug(n,n+1) / Aug(n,n);
for i = n-1:-1:1
    x(i) = (Aug(i,n+1)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,
        i);
end
disp(strcat(["x = ",string(x(1))]))
disp(strcat(["y = ",string(x(2))]))
disp(strcat(["z = ",string(x(3))]))
```

Scilab code Exa 3.3 Gauss Elimination Method with Partial Pivoting

```
//Example 3.3
clc
clear

A = [0 4 2 8; 4 10 5 4; 4 5 6.5 2; 9 4 4 0];   //
    Coefficient Matrix
B = [24; 32; 26; 21];   //Constant Matrix

n = length(B);
Aug = [A,B];

// Forward Elimination
for j = 1:n-1
    // Partial Pivoting
    [dummy,t] = max(abs(Aug(j:n,j)));
    lrow = t(1)+j-1;
    Aug([j,lrow],:) = Aug([lrow,j],:);

    for i = j+1:n
        Aug(i,j:n+1) = Aug(i,j:n+1) - Aug(i,j) / Aug
            (j,j) * Aug(j,j:n+1);
    end
```

```
end

// Backward Substitution
x = zeros(n,1);
x(n) = Aug(n,n+1) / Aug(n,n);
for i = n-1:-1:1
    x(i) = (Aug(i,n+1)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,
       i);
end

disp(strcat(["x1 = ",string(x(1))]))
disp(strcat(["x2 = ",string(x(2))]))
disp(strcat(["x3 = ",string(x(3))]))
disp(strcat(["x4 = ",string(x(4))]))
```

Scilab code Exa 3.4 Gauss Jordan Method

```
//Example 3.4
clc
clear

A = [1 2 1; 2 3 4; 4 3 2];
B = [8; 20; 16];
n = length (B);
Aug = [A,B];

// Forward Elimination
for j = 1:n-1
    for i = j+1:n
        Aug(i,j:n+1) = Aug(i,j:n+1) - Aug(i,j) / Aug
           (j,j) * Aug(j,j:n+1);
    end
end

// Backward Elimination
```

```
for j = n:-1:2
    Aug(1:j-1,:) = Aug(1:j-1,:) - Aug(1:j-1,j) / Aug
       (j,j) * Aug(j,:);
end

// Diagonal Normalization
for j=1:n
    Aug(j,:) = Aug(j,:) / Aug(j,j);
end
x = Aug(:,n+1);
disp(strcat(["x = ",string(x(1))]))
disp(strcat(["y = ",string(x(2))]))
disp(strcat(["z = ",string(x(3))]))
```

Scilab code Exa 3.5 Crout Reduction Method

```
//Example 3.5
clc
clear

A = [5 -2 1; 7 1 -5; 3 7 4];
B = [4; 8; 10];

n = length (B);
L = zeros(n,n);          // L = Lower Triangular
   Matrix Initiation
U = eye(n,n);            // U = Upper Triangular
   Matrix Initiation

// LU Decomposition
for i = 1:n
    sum1 = zeros(n-i+1,1);
    for k = 1:i-1
        sum1 = sum1 + L(i:n,k) * U(k,i);
    end
```

```
    L(i:n,i) = A(i:n,i) - sum1;

    sum2 = zeros(1,n-i);
    for k = 1:i-1
        sum2 = sum2 + L(i,k) * U(k,i+1:n);
    end
    U(i,i+1:n) = (A(i,i+1:n) - sum2) / L(i,i);
end

// Forward Substitution
D = ones(n,1);
for i = 1:n
    sum3 = 0;
    for k = 1:i-1
        sum3  = sum3 + L(i,k) * D(k);
    end
    D(i) = (B(i) - sum3) / L(i,i);
end

// Back Substitution
x = ones(n,1);
for i = n:-1:1
    sum4 = 0;
    for k = i+1:n
        sum4 = sum4 + U(i,k) * x(k);
    end
    x(i) = D(i) - sum4;
end

disp(strcat(["x1 = ",string(x(1))]))
disp(strcat(["x2 = ",string(x(2))]))
disp(strcat(["x3 = ",string(x(3))]))
```

Scilab code Exa 3.6 Jacobi Iterative Method

```
//Example 3.6
clc
clear

A = [83 11 -4; 7 52 13; 3 8 29];
B = [95; 104; 71];

n = length (B);
tol = 1e-4;
iter = 1;
maxit = 5;

x = zeros(n,1);              //Intial guess
E = ones(n,1);               //Assuming to avoid variable size error
S = diag(diag(A));


while (1)
    x(:,iter+1) = S\(B + (S-A)*(x(:,iter)));
    E(:,iter+1) = (x(:,iter+1)-x(:,iter))./x(:,iter+1)*100;
    if x(:,iter) == 0
        Error = 1;
    else
        Error = sqrt((sum((E(:,iter+1)).^2))/n);
    end

    if Error <= tol | iter == maxit
        break
    end
    iter = iter+1;
end
xact = x(:,iter);
x = round(x*10^4)/10^4;
x(:,1) = [];
mprintf('%s %3s %9s %9s','Iter No.','x','y','z');
disp([(1:iter)' x']);
```

Scilab code Exa 3.7 Gauss Seidel Method

```
//Example 3.7
clc
clear

A = [1 -1/4 -1/4 0; -1/4 1 0 -1/4; -1/4 0 1 -1/4; 0
    -1/4 -1/4 1];
B = [1/2; 1/2; 1/4; 1/4];

n = length (B);
tol = 1e-4;
iter = 1;
maxit = 5;

x = zeros(n,1);              //Intial guess
E = ones(n,1);              //Assuming to avoid
    variable size error
S = diag(diag(A));
T = S-A;
xold = x;

while (1)
    for i = 1:n
        x(i,iter+1) = (B(i) + T(i,:) * xold) / A(i,i
           );
        E(i,iter+1) = (x(i,iter+1)-xold(i))/x(i,iter
           +1)*100;
        xold(i) = x(i,iter+1);
    end

    if x(:,iter) == 0
        E = 1;
    else
```

```
        E = sqrt((sum((E(:,iter+1)).^2))/n);
    end

    if E <= tol | iter == maxit
        break
    end
    iter = iter + 1;
end
X = x(:,iter);
x = round(x*10^5)/10^5;
x(:,1) = [];
mprintf('%s %3s %11s %10s %10s','Iter No.','x1','x2'
   ,'x3','x4');
disp([(1:iter)' x']);
```

Scilab code Exa 3.8 Relaxation Method

```
//Example 3.8
clc
clear

A = [6 -3 1;  2 1 -8;1 -7 1];
b = [11; -15; 10];

n = length (b);
tol = 1e-4;
iter = 1;
maxit = 9;

x = zeros(1,n);            //Intial guess
absA = abs(A);
[dummy,index] = max(absA(1,:),absA(2,:),absA(3,:));
if length(unique(index)) == n
    nu_T = diag(diag(A(index,:))) - A(index,:);
    if abs(diag(A(index,:))) - (sum(abs(nu_T),2)) >
```

```
        0
        A(index,:) = A;
        b(index,:) = b;
    end
end

for iter = 1:maxit
    R(iter,:) = b' - x(iter,:) * A';
    [mx,i] = max(abs(R(iter,:)));
    Rmax(iter) = R(iter,i);
    dx(iter) = Rmax(iter) ./ A(i,i);
    x(iter+1,:) = x(iter,:);
    x(iter+1,i) = x(iter,i) + dx(iter);
end
R = round(R*10^4)/10^4;
Rmax = round(Rmax*10^4)/10^4;
dx = round(dx*10^4)/10^4;
x = round(x*10^4)/10^4;
mprintf('%s %3s %9s %9s %12s %10s %6s %9s %9s','Iter
    No.','R1','R2','R3','Max Ri','Diff dxi','x1','x2
    ','x3');
disp([(1:maxit)' R Rmax dx x(1:maxit,:)])
```

Scilab code Exa 3.9 Matrix Inverse using Gauss Elimination Method

```
//Example 3.9
clc
clear

A = [1 1 1; 4 3 -1; 3 5 3];
n = length (A(1,:));
Aug = [A,eye(n,n)];

// Forward Elimination
for j = 1:n-1
```

```
    for i = j+1:n
        Aug(i,j:2*n) = Aug(i,j:2*n) - Aug(i,j) / Aug
           (j,j) * Aug(j,j:2*n);
    end
end

// Backward Elimination
for j = n:-1:2
    Aug(1:j-1,:) = Aug(1:j-1,:) - Aug(1:j-1,j) / Aug
       (j,j) * Aug(j,:);
end

// Diagonal Normalization
for j=1:n
    Aug(j,:) = Aug(j,:) / Aug(j,j);
end
Inv_A = Aug(:,n+1:2*n);
disp(Inv_A,"Inverse of A (A-1) = ")
```

Scilab code Exa 3.10 `Matrix Inverse using Gauss Jordan Method`

```
//Example 3.10
clc
clear

A = [1 1 1; 4 3 -1; 3 5 3];
n = length (A(1,:));
Aug = [A,eye(n,n)];

N = 1:n;
for i = 1:n
    dummy1 = N;
    dummy1(i) = [];
    index(i,:) = dummy1;
end
```

```

// Forward Elimination
for j = 1:n
    [dummy2,t] = max(abs(Aug(j:n,j)));
    lrow = t+j-1;
    Aug([j,lrow],:) = Aug([lrow,j],:);
    Aug(j,:) = Aug(j,:) / Aug(j,j);
    for i = index(j,:)
        Aug(i,:) = Aug(i,:) - Aug(i,j) / Aug(j,j) * Aug(j,:);
    end
end
Inv_A = Aug(:,n+1:2*n);
disp(Inv_A,"Inverse of A (A-1) = ")
```

Chapter 4

Eigenvalue Problems

Scilab code Exa 4.1 Eigenvalues and Eigenvectors

```
//Example 4.1
clc
clear

A = [2 3 2; 4 3 5; 3 2 9];
v = [1; 1; 1];
iter = 1;
maxit = 5;

while(1)
    u(:,iter) = A * v(:,iter);
    q(iter) = max(u(:,iter));
    v(:,iter+1) = u(:,iter) / q(iter);
    if iter == maxit then
        break
    end
    iter = iter + 1;
end
X = round(v(:,iter)*10^2) / 10^2;
disp(X,"Eigen Vector:")
```

Scilab code Exa 4.2 Eigenvalues and Eigenvectors using Jacobi Method

```
//Example 4.2
clc
clear

rt2 = sqrt(2);
A = [1 rt2 2; rt2 3 rt2; 2 rt2 1];
[n,n] = size(A);
iter = 1;
maxit = 3;
D = A;
S = 1;

while(1)
    D_offdiag = D - diag(diag(D));
    [mx,index1] = max(abs(D_offdiag));
    i = index1(1);
    j = index1(2);
    if (D(i,i)-D(j,j)) == 0 then
        theta = %pi/4;
    else
        theta = atan(2*D(i,j)/(D(i,i)-D(j,j))) / 2;
    end
    S1 = eye(n,n);
    S1(i,i) = cos(theta);
    S1(i,j) = -sin(theta);
    S1(j,i) = sin(theta);
    S1(j,j) = cos(theta);

    D1 = inv(S1) * D * S1;
    for j = 1:n
        for i = 1:n
            if abs(D1(i,j)) < 1D-10 then
```

```
                D1(i,j) = 0;
            end
        end
    end
    S = S * S1;

    if D1 - diag(diag(D1)) == zeros(n,n) | iter == maxit then
        eigval = diag(D1);
        disp('Eigen Values:')
        disp(eigval)

        disp('Eigen Vectors:')
        disp(S(:,1))
        disp(S(:,2))
        disp(S(:,3))
        break
    end

    iter = iter + 1;
    D = D1;
end
```

Scilab code Exa 4.3 Eigenvalues using Jacobi Method

```
//Example 4.3
clc
clear

A = [2 -1 0; -1 2 -1; 0 -1 2];
[n,n] = size(A);
iter = 1;
maxit = 3;
//Note: Diagonal form may be achieved at iter = 9. Modify maxit to greater than 9 for exact result.
```

```
D = A;
S = 1;

while(1)
    D_offdiag = D - diag(diag(D));
    [mx,index1] = max(abs(D_offdiag));
    i = index1(1);
    j = index1(2);
    if (D(i,i)-D(j,j)) == 0 then
        theta = %pi/4;
    else
        theta = atan(2*D(i,j)/(D(i,i)-D(j,j))) / 2;
    end
    S1 = eye(n,n);
    S1(i,i) = cos(theta);
    S1(i,j) = -sin(theta);
    S1(j,i) = sin(theta);
    S1(j,j) = cos(theta);

    D1 = inv(S1) * D * S1;
    for j = 1:n
        for i = 1:n
            if abs(D1(i,j)) < 1D-10 then
                D1(i,j) = 0;
            end
        end
    end
    S = S * S1;

    if D1 - diag(diag(D1)) == zeros(n,n) | iter == maxit then
        eigval = diag(D1);
        eigval = round(eigval*10^3)/10^3;
        disp('Eigen Values:')
        disp(eigval)
        break
    end
```

```

    iter = iter + 1;
    D = D1;
end
```

Scilab code Exa 4.4 Eigenvalues and Eigenvectors using Jacobi Method

```
//Example 4.4
clc
clear

A = [5 0 1; 0 -2 0; 1 0 5];
[n,n] = size(A);
iter = 1;
maxit = 3;
D = A;
S = 1;

while(1)
    D_offdiag = D - diag(diag(D));
    [mx,index1] = max(abs(D_offdiag));
    i = index1(1);
    j = index1(2);
    if (D(i,i)-D(j,j)) == 0 then
        theta = %pi/4;
    else
        theta = atan(2*D(i,j)/(D(i,i)-D(j,j))) / 2;
    end
    S1 = eye(n,n);
    S1(i,i) = cos(theta);
    S1(i,j) = -sin(theta);
    S1(j,i) = sin(theta);
    S1(j,j) = cos(theta);

    D1 = inv(S1) * D * S1;
```

```
    for j = 1:n
        for i = 1:n
            if abs(D1(i,j)) < 1D-10 then
                D1(i,j) = 0;
            end
        end
    end
    S = S * S1;

    if D1 - diag(diag(D1)) == zeros(n,n) | iter == maxit then
        eigval = diag(D1);
        disp('Eigen Values:')
        disp(eigval)

        disp('Eigen Vectors:')
        disp(S(:,1))
        disp(S(:,2))
        disp(S(:,3))
        break
    end

    iter = iter + 1;
    D = D1;
end
```

Chapter 5

Curve Fitting

Scilab code Exa 5.1 Method of Group Averages

```
//Example 5.1
clc
clear

x = 10:10:80;
y = [1.06 1.33 1.52 1.68 1.81 1.91 2.01 2.11];

X = log(x);
Y = log(y);

n = length(Y);
M1 = [sum(Y); sum(X.*Y)];
M2 = [n sum(X); sum(X) sum(X.^2)];

A = M2\M1;

m = exp(A(1));
n = A(2);

disp(round(m*10^4)/10^4, "m =")
disp(round(n*10^4)/10^4, "n =")
```

Scilab code Exa 5.2 Method of Group Averages

```
//Example 5.2
clc
clear

x = [20 30 35 40 45 50];
y = [10 11 11.8 12.4 13.5 14.4];

X = x.^2;
Y = y;

n = length(Y);
M1 = [sum(Y); sum(X.*Y)];
M2 = [n sum(X); sum(X) sum(X.^2)];

A = M2\M1;

a = A(1);
b = A(2);

disp(round(a*10^4)/10^4, "a =")
disp(round(b*10^4)/10^4, "b =")
```

Scilab code Exa 5.3 Method of Group Averages

```
//Example 5.3

clc
clear

```

```
x = [8 10 15 20 30 40];
y = [13 14 15.4 16.3 17.2 17.8];

X = 1 ./x;
Y = 1 ./y;

n = length(Y);
M1 = [sum(Y); sum(X.*Y)];
M2 = [n sum(X); sum(X) sum(X.^2)];

A = M2\M1;

b = A(1);
a = A(2);

disp(round(a*10^4)/10^4, "a =")
disp(round(b*10^4)/10^4, "b =")
```

Scilab code Exa 5.4 Method of Least Squares

```
//Example 5.4

clc
clear

X = 0.5:0.5:3;
Y = [15 17 19 14 10 7];

n = length(Y);
M1 = [sum(Y); sum(X.*Y)];
M2 = [n sum(X); sum(X) sum(X.^2)];

A = M2\M1;

b = A(1);
```

```
a = A(2);

disp(round(a*10^4)/10^4, "a =")
disp(round(b*10^4)/10^4, "b =")
```

Scilab code Exa 5.5 Method of Least Squares

```
//Example 5.5

clc
clear

x = 1:6;
y = [2.6 5.4 8.7 12.1 16 20.2];

X = x;
Y = y ./x;

n = length(Y);
M1 = [sum(Y); sum(X.*Y)];
M2 = [n sum(X); sum(X) sum(X.^2)];

A = M2\M1;

a = A(1);
b = A(2);

disp(round(a*10^5)/10^5, "a =")
disp(round(b*10^5)/10^5, "b =")
```

Scilab code Exa 5.6 Method of Least Squares

```
//Example 5.6
```

```

clc
clear

X = 1:0.2:2;
Y = [0.98 1.4 1.86 2.55 2.28 3.2];

n = length(Y);
M1 = [sum(X.^4) sum(X.^3) sum(X.^2); sum(X.^3) sum(X
    .^2) sum(X); sum(X.^2) sum(X) n];
M2 = [sum(X.^2 .* Y); sum(X.*Y); sum(Y)];
A = M1\M2;

a = A(1);
b = A(2);
c = A(3);

disp(round(a*10^4)/10^4, "a =")
disp(round(b*10^4)/10^4, "b =")
disp(round(c*10^4)/10^4, "c =")
```

Scilab code Exa 5.7 Method of Least Squares

```
//Example 5.7

clc
clear

x = 2:5;
y = [27.8 62.1 110 161];

X = log(x);
Y = log(y);

n = length(Y);
```

```
M1 = [sum(X.^2) sum(X); sum(X) n];
M2 = [sum(X.*Y); sum(Y)];
M = M1\M2;

b = M(1);
A = M(2);
a = exp(A);

disp(round(a*10^4)/10^4, "a =")
disp(round(b*10^4)/10^4, "b =")
```

Scilab code Exa 5.8 Principle of Least Squares

```
//Example 5.8

clc
clear

x = 1:4;
y = [1.65 2.7 4.5 7.35];

X = x;
Y = log10(y);

n = length(Y);
M1 = [sum(X.^2) sum(X); sum(X) n];
M2 = [sum(X.*Y); sum(Y)];
M = M1\M2;

B = M(1);
A = M(2);
a = 10^A;
b = B/log10(%e);

disp(round(a), "a =")
```

```
disp(round(b*10^4)/10^4, "b =")
```

Scilab code Exa 5.9 Method of Moments

```
//Example 5.9

clc
clear

x = 2:5;
y = [27 40 55 68];

delx = x(2) - x(1);
mu1 = delx * sum(y);
mu2 = delx * sum(x.*y);

n = length(y);
l = x(1) - delx/2;
u = x(n) + delx/2;

M1 = [integrate("x",'x',l,u) u-l; integrate("x^2",'x
    ',l,u) integrate("x",'x',l,u)];
M2 = [mu1; mu2];
M = M1\M2;

a = M(1);
b = M(2);

disp(round(a*10^4)/10^4, "a =")
disp(round(b*10^4)/10^4, "b =")
```

Scilab code Exa 5.10 Method of Moments

```
//Example 5.10

clc
clear

x = 3:7;
y = [31.9 34.6 33.8 27 31.6];

delx = x(2) - x(1);
mu1 = delx * sum(y);
mu2 = delx * sum(x.*y);
mu3 = delx * sum(x^2 .*y);

n = length(y);
l = x(1) - delx/2;
u = x(n) + delx/2;

t0 = u-l;
t1 = integrate("x",'x',l,u);
t2 = integrate("x^2",'x',l,u);
t3 = integrate("x^3",'x',l,u);
t4 = integrate("x^4",'x',l,u);

M1 = [t2 t1 t0; t3 t2 t1; t4 t3 t2];
M2 = [mu1; mu2; mu3];
M1 = round(M1*10^2)/10^2;
M = M1\M2;

c = M(1);
b = M(2);
a = M(3);

disp(round(a*10^4)/10^4, "a =")
disp(round(b*10^4)/10^4, "b =")
disp(round(c*10^4)/10^4, "c =")
```

Chapter 6

Interpolation

Scilab code Exa 6.1 Forward Difference Table

```
//Example 6.1

clc
clear

x = 0.1:0.2:1.3;
y = [0.003 0.067 0.148 0.248 0.37 0.518 0.697];

n = length(x);
del = %nan*ones(n,6);
del(:,1) = y';
for j = 2:6
    for i = 1:n-j+1
        del(i,j) = del(i+1,j-1) - del(i,j-1);
    end
end
del = [x' del];
del = round(del*10^3)/10^3;
mprintf("%5s %7s %8s %9s %8s %8s %8s",'x','y','dy','d2y','d3y','d4y','d5y')
disp(del)
```

Scilab code Exa 6.2 Expression for Finite Difference Elements

```
// Example 6.2
// This is an analytical problem and need not be
   coded.
```

Scilab code Exa 6.3 Expression for Finite Difference Elements

```
// Example 6.3
// This is an analytical problem and need not be
   coded.
```

Scilab code Exa 6.4 Expression for Finite Difference Elements

```
// Example 6.4
// This is an analytical problem and need not be
   coded.
```

Scilab code Exa 6.5 Proof of Relation

```
// Example 6.5
// This is an analytical problem and need not be
   coded.
```

Scilab code Exa 6.6 Proofs of given Relations

```
// Example 6.6
// This is an analytical problem and need not be
   coded.
```

Scilab code Exa 6.7 Proof for Commutation of given Operations

```
// Example 6.7
// This is an analytical problem and need not be
   coded.
```

Scilab code Exa 6.8 Newton Forward Difference Interpolation Formula

```
//Example 6.8

clc
clear

x = 10:10:50;
y = [46 66 81 93 101];

n = length(x);
del = %nan*ones(n,5);
del(:,1) = y';
for j = 2:5
    for i = 1:n-j+1
        del(i,j) = del(i+1,j-1) - del(i,j-1);
    end
end
del(:,1) = [];

X = 15; //input
```

```
for i = 1:n
    if X>x(i) then
        h = x(i+1) - x(i);
        p = (X-x(i)) / h;
        x0 = x(i);
        y0 = y(i);
        dely0 = del(i,:);
        dely0(isnan(y0)) = [];
    end
end

Y = y0;

for i = 1:length(dely0)
    t = 1;
    for j = 1:i
        t = t * (p-j+1);
    end
    Y = Y + t*dely0(i)/factorial(i);
end
Y = round(Y*10^4)/10^4;
disp(Y,"f(15) = ")
```

Scilab code Exa 6.9 Newton Forward Difference Interpolation Formula

```
//Example 6.9

clc
clear

x = 0.1:0.1:0.5;
y = [1.4 1.56 1.76 2 2.28];

n = length(x);
del = %nan*ones(n,5);
```

```
del(:,1) = y';
for j = 2:5
    for i = 1:n-j+1
        del(i,j) = del(i+1,j-1) - del(i,j-1);
    end
end
del(:,1) = [];

X = poly(0, "X");
h = x(2) - x(1);
p = (X-x(1)) / h;
x0 = x(1);
y0 = y(1);
dely0 = del(1,:);

Y = y0;

for i = 1:length(dely0)
    t = 1;
    for j = 1:i
        t = t * (p-j+1);
    end
    Y = Y + t*dely0(i)/factorial(i);
end
Y = round(Y*10^2)/10^2;
disp(Y,"Required Newton''s Interpolating Polynomial:
   ")
```

Scilab code Exa 6.10 Newton Forward Difference Interpolation Formula

```
//Example 6.10

clc
clear

```

```
x = 1:5;
Y = poly(0, "Y");
y = [2 5 7 Y 32];

n = length(x);
del = %nan*ones(n,5);
del(:,1) = y';
for j = 2:5
    for i = 1:n-j+1
        del(i,j) = del(i+1,j-1) - del(i,j-1);
    end
end
del(:,1) = [];

// del4 = 0

y0 = del(:,4);
y0(isnan(y0)) = [];
Y = roots(y0)
disp(Y,"Missing value f(x3) = ")
```

Scilab code Exa 6.11 Newton Forward Difference Interpolation Formula

```
//Example 6.11

clc
clear

x = 0:5;
y = [-3 3 11 27 57 107];

n = length(x);
del = %nan*ones(n,4);
del(:,1) = y';
for j = 2:4
```

```
    for i = 1:n-j+1
        del(i,j) = del(i+1,j-1) - del(i,j-1);
    end
end
del(:,1) = [];

X = poly(0, "x");
h = x(2) - x(1);
p = (X-x(1)) / h;
x0 = x(1);
y0 = y(1);
dely0 = del(1,:);

Y = y0;

for i = 1:length(dely0)
    t = 1;
    for j = 1:i
        t = t * (p-j+1);
    end
    Y = Y + t*dely0(i)/factorial(i);
end
disp(Y,"Required cubic polynomial:")
```

Scilab code Exa 6.12 Newton Backward Difference Interpolation Formula

```
//Example 6.12

clc
clear

x = 1:8;
y = x^3;

n = length(x);
```

```
del = %nan*ones(n,4);
del(:,1) = y';
for j = 2:4
    for i = 1:n-j+1
        del(i+j-1,j) = del(i+j-1,j-1) - del(i+j-2,j-1);
    end
end

X = 7.5;
h = x(2) - x(1);
p = (X-x(n)) / h;
xn = x(n);
yn = y(n);
delyn = del(n,:);

Y = 0;

for i = 0:length(delyn)-1
    t = 1;
    for j = 1:i
        t = t * (p+j-1);
    end
    Y = Y + t*delyn(i+1)/factorial(i);
end
disp(Y,"y(7.5) = ")
```

Scilab code Exa 6.13 `Newton Backward Difference Interpolation Formula`

```
//Example 6.13

clc
clear

x = 1974:2:1982;
```

```
y = [40 43 48 52 57];

n = length(x);
del = %nan*ones(n,5);
del(:,1) = y';
for j = 2:5
    for i = 1:n-j+1
        del(i+j-1,j) = del(i+j-1,j-1) - del(i+j-2,j
           -1);
    end
end

X = 1979;
h = x(2) - x(1);
p = (X-x(n)) / h;
xn = x(n);
yn = y(n);
delyn = del(n,:);

Y = 0;

for i = 0:length(delyn)-1
    t = 1;
    for j = 1:i
        t = t * (p+j-1);
    end
    Y = Y + t*delyn(i+1)/factorial(i);
end
Y = round(Y*10^4)/10^4;
disp(Y,"Estimated sales for the year 1979: ")
```

Scilab code Exa 6.14 Lagrange Interpolation Formula

```
//Example 6.14

```

```
clc
clear

x = [1 3 4 6];
y = [-3 0 30 132];

n = length(x);
Y = 0;
X = poly(0, "X");
//X = 5;
for i = 1:n
    t = x;
    t(i) = [];
    p = 1;
    for j = 1:length(t)
        p = p * (X-t(j))/(x(i)-t(j));
    end
    Y = Y + p*y(i);
end
Y5 = horner(Y,5);
disp(Y5,"y(5) = ")
```

Scilab code Exa 6.15 Lagrange Interpolation Formula

```
//Example 6.15

clc
clear

x = [1 2 5];
y = [1 4 10];

n = length(x);
Y = 0;
X = poly(0, "X");
```

```
//X = 5;
for i = 1:n
    t = x;
    t(i) = [];
    p = 1;
    for j = 1:length(t)
        p = p * (X-t(j))/(x(i)-t(j));
    end
    Y = Y + p*y(i);
end
Y5 = horner(Y,3);
disp(Y5,"f(3) = ")
```

Scilab code Exa 6.16 Lagrange and Newton Divided Difference Interpolation Formulae

```
//Example 6.16

clc
clear

x = [0 1 2 4];
y = [1 1 2 5];

n = length(x);
del = %nan*ones(n,4);
del(:,1) = y';
for j = 2:4
    for i = 1:n-j+1
        del(i,j) = (del(i+1,j-1) - del(i,j-1)) / (x(
           i+j-1) - x(i));
    end
end
del(:,1) = [];

Y = 0;
```

```
X = poly(0, "X");
for i = 1:n
    t = x;
    t(i) = [];
    p = 1;
    for j = 1:length(t)
        p = p * (X-t(j))/(x(i)-t(j));
    end
    Y = Y + p*y(i);
end
disp(round(Y*10^4)/10^4,"Interpolating polynomial:")
```

Scilab code Exa 6.17 Newton Divided Difference Interpolation Formulae

```
//Example 6.17

clc
clear

x = [0 1 4];
y = [2 1 4];

n = length(x);
del = %nan*ones(n,3);
del(:,1) = y';
for j = 2:3
    for i = 1:n-j+1
        del(i,j) = (del(i+1,j-1) - del(i,j-1)) / (x(
           i+j-1) - x(i));
    end
end
del(:,1) = [];

Y = 0;
X = 2;
```

```
for i = 1:n
    t = x;
    t(i) = [];
    p = 1;
    for j = 1:length(t)
        p = p * (X-t(j))/(x(i)-t(j));
    end
    Y = Y + p*y(i);
end
disp(Y,"y(2) = ")
```

Scilab code Exa 6.18 Identity Proof for Newton and Lagrange Interpolation Formulae

```
// Example 6.18
// This is an analytical problem and need not be coded.
```

Scilab code Exa 6.19 Interpolation in Two Dimensions

```
//Example 6.19

clc
clear

x = 0:4;
n = length(x);
f = "X^2 + Y^2 - Y";
tab = %nan*ones(n,5);

for j = 0:4
    fj = strsubst(f,'Y','j');
    for i = 1:n
        tab(i,j+1) = eval(strsubst(fj,'X','x(i)'));
```

```
    end
end
//tab(:,1) = [];
mprintf("%4s %6s %6s %6s %6s %6s",'x','y=0','y=1','y
   =2','y=3','y=4')
disp([(0:4)' tab])
tab2 = tab(2:4,2:4)';
n1 = length(tab2(:,1));
y = 2:4;

del1 = %nan*ones(n1,3);
del1(:,1) = tab2(:,1);
for j = 2:4
    for i = 1:n1-j+1
        del1(i,j) = del1(i+1,j-1) - del1(i,j-1);
    end
end

del2 = %nan*ones(n1,3);
del2(:,1) = tab2(:,2);
for j = 2:4
    for i = 1:n1-j+1
        del2(i,j) = del2(i+1,j-1) - del2(i,j-1);
    end
end

del3 = %nan*ones(n1,3);
del3(:,1) = tab2(:,3);
for j = 2:4
    for i = 1:n1-j+1
        del3(i,j) = del3(i+1,j-1) - del3(i,j-1);
    end
end

y0 = y(1);
Y = 3.5;
hy = y(2) - y(1);
py = (Y-y0)/hy;
```

```
f1y = 0;
del1y0 = del1(1,:);
for i = 0:length(del1y0)-1
    t = 1;
    for j = 1:i
        t = t * (py-j+1);
    end
    f1y = f1y + t*del1y0(i+1)/factorial(i);
end

f2y = 0;
del2y0 = del2(1,:);
for i = 0:length(del2y0)-1
    t = 1;
    for j = 1:i
        t = t * (py-j+1);
    end
    f2y = f2y + t*del2y0(i+1)/factorial(i);
end

f3y = 0;
del3y0 = del3(1,:);
for i = 0:length(del3y0)-1
    t = 1;
    for j = 1:i
        t = t * (py-j+1);
    end
    f3y = f3y + t*del3y0(i+1)/factorial(i);
end

del = %nan*ones(n1,3);
del(:,1) = [f1y; f2y; f3y];
for j = 2:4
    for i = 1:n1-j+1
        del(i,j) = del(i+1,j-1) - del(i,j-1);
    end
end
```

```

f = 0;
X = 2.5;
x0 = x(2);
hx = x(2) - x(1);
px = (X-x0)/hx;
del0 = del(1,:)
for i = 0:length(del0)-1
    t = 1;
    for j = 1:i
        t = t * (px-j+1);
    end
    f = f + t*del0(i+1)/factorial(i);
end
disp(f,"f(2.5,3.5) = ")
```

Scilab code Exa 6.20 Cubic Spline Curve

```
//Example 6.20

clc
clear

function [p] = cubicsplin(x,y)
// Fits point data to cubic spline fit

n = length(x);
a = y(1:n-1);    // Spline Initials

M1 = zeros(3*(n-1));
M2 = zeros(3*(n-1),1);
// Point Substitutions
for i = 1:n-1
    M1(i,i) = x(i+1) - x(i);
    M1(i,i+n-1) = (x(i+1) - x(i))^2;
```

```
    M1(i,i+2*(n-1)) = (x(i+1) - x(i))^3;
    M2(i) = y(i+1) - y(i);
end

// Knot equations
for i = 1:n-2
    // Derivative (S') continuity
    M1(i+n-1,i) = 1;
    M1(i+n-1,i+1) = -1;
    M1(i+n-1,i+n-1) = 2*(x(i+1)-x(i));
    M1(i+n-1,i+2*(n-1)) = 3*(x(i+1)-x(i))^2;
    // S'' continuity
    M1(i+2*n-3,i+n-1) = 2;
    M1(i+2*n-3,i+n) = -2;
    M1(i+2*n-3,i+2*(n-1)) = 6*(x(i+1)-x(i));
end
// Given BC
M1(3*n-4,n) = 1;
M1(3*n-3,2*n-2) = 1;
M1(3*n-3,3*n-3) = 3*(3-2);

var = M1\M2;
var = round(var);
b = var(1:n-1);
c = var(n:2*(n-1));
d = var(2*(n-1)+1:3*(n-1));
p = [d c b a(:)];
endfunction

x = 0:3;
y = [1 4 0 -2];
p = cubicsplin(x,y);
for i = 1:length(p(:,1))
    disp(strcat(["S",string(i-1),"(x) ="]))
    disp(poly(p(i,:),"X",["coeff"]))
end
```

Scilab code Exa 6.21 Cubic Spline Curve

```
//Example 6.21

clc
clear

function [p] = cubicsplin(x,y)
// Fits point data to cubic spline fit

n = length(x);
a = y(1:n-1);    // Spline Initials

M1 = zeros(3*(n-1));
M2 = zeros(3*(n-1),1);
// Point Substitutions
for i = 1:n-1
    M1(i,i) = x(i+1) - x(i);
    M1(i,i+n-1) = (x(i+1) - x(i))^2;
    M1(i,i+2*(n-1)) = (x(i+1) - x(i))^3;
    M2(i) = y(i+1) - y(i);
end

// Knot equations
for i = 1:n-2
    // Derivative (S') continuity
    M1(i+n-1,i) = 1;
    M1(i+n-1,i+1) = -1;
    M1(i+n-1,i+n-1) = 2*(x(i+1)-x(i));
    M1(i+n-1,i+2*(n-1)) = 3*(x(i+1)-x(i))^2;
    // S'' continuity
    M1(i+2*n-3,i+n-1) = 2;
    M1(i+2*n-3,i+n) = -2;
    M1(i+2*n-3,i+2*(n-1)) = 6*(x(i+1)-x(i));
```

```
end
// Given BC
M1(3*n-4,1) = 1;
M1(3*n-3,n-1) = 1;
M1(3*n-3,2*n-2) = 2*(3-2);
M1(3*n-3,3*n-3) = 3*(3-2)^2;
M2(3*n-4) = 2;
M2(3*n-3) = 2;

var = M1\M2;
var = round(var);
b = var(1:n-1);
c = var(n:2*(n-1));
d = var(2*(n-1)+1:3*(n-1));

p = [a(:) b c d];
endfunction

x = 0:3;
y = [1 4 0 -2];
p = cubicsplin(x,y);
for i=1:length(p(:,1))
    disp(strcat(["S",string(i-1),"(x) = "]))
    disp(poly(p(i,:),"x",["coeff"]))
end
```

Scilab code Exa 6.22 Minima of a Tabulated Function

```
//Example 6.22

clc
clear

x = 3:8;
y = [0.205 0.24 0.259 0.262 0.25 0.224];
```

```
n = length(x);
del = %nan*ones(n,5);
del(:,1) = y';
for j = 2:5
    for i = 1:n-j+1
        del(i,j) = del(i+1,j-1) - del(i,j-1);
    end
end

X = poly(0, "X");
x0 = x(1);
y0 = y(1);
h = x(2) - x(1);
p = (X-x0)/h;
del0 = del(1,:);
del0 = round(del0*10^4)/10^4;
del0 = del0(1:find(del0==0)-1);

Y = 0;
for i = 0:length(del0)-1
    t = 1;
    for j = 1:i
        t = t * (p-j+1);
    end
    Y = Y + t*del0(i+1)/factorial(i);
end
disp(Y,"y = ")

dydx = derivat(Y);
minx = roots(dydx);
miny = round(horner(Y,minx)*10^5)/10^5;
disp(minx,"min_x = ")
disp(miny,"min_y = ")
//min_y value is incorrectly displayed in textbook as 0.25425 instead of 0.26278
```

Scilab code Exa 6.23 Maxima of a Tabulated Function

```
//Example 6.23

clc
clear

x = [-1 1 2 3];
y = [-21 15 12 3];

n = length(x);
X = poly(0, "X");
Y = 0;
for i = 1:n
    t = x;
    t(i) = [];
    p = 1;
    for j = 1:length(t)
        p = p * (X-t(j))/(x(i)-t(j));
    end
    Y = Y + p*y(i);
end

dydx = derivat(Y);
extx = real(roots(dydx));
extx = round(extx*10^4)/10^4;
d2ydx = derivat(dydx);

if  horner(d2ydx,extx(1)) < 0 then
    maxx = extx(1);
    maxy = horner(Y,maxx);
else
    maxx = extx(2);
    maxy = horner(Y,maxx);
```

```
end
maxy = round(maxy*10^4)/10^4;
disp(maxx,"max_x = ")
disp(maxy,"max_y = ")
```

Scilab code Exa 6.24 Determination of Function Value

```
//Example 6.24

clc
clear

x = 1:3:10;
F = [500426 329240 175212 40365];

n = length(x);
del = %nan*ones(n,4);
del(:,1) = F';
for j = 2:4
    for i = 1:n-j+1
        del(i,j) = del(i+1,j-1) - del(i,j-1);
    end
end

del0 = del(1,:);
X = 2;
x0 = x(1);
h = x(2) - x(1);
p = (X-x0) / h;
F2 = 0;
for i = 0:length(del0)-1
    t = 1;
    for j = 1:i
        t = t * (p-j+1);
    end
```

```
    F2 = F2 + t*del0(i+1)/factorial(i);
end

f2 = F(1) - F2;
disp(f2,"f(2) = ")
```

Chapter 7

Numerical Differentiation and Integration

Scilab code Exa 7.1 Determination of Differential Function Value

```
//Example 7.1

clc
clear

x = 0:0.2:1;
y = [1 1.16 3.56 13.96 41.96 101];

n = length(x);
del = %nan*ones(n,6);
del(:,1) = y';
for j = 2:6
    for i = 1:n-j+1
        del(i,j) = del(i+1,j-1) - del(i,j-1);
    end
end
del = round(del*10^2)/10^2;
mprintf("%5s %6s %9s %8s %8s %8s %7s",'x','y','dy','d2y','d3y','d4y','d5y')
```

```
disp([x' del])

h = x(2) - x(1);
del0 = del(1,:);
del1 = del(2,:);

df1 = (del1(2) - del1(3)/2 + del1(4)/3 - del1(5)/4) / h;
d2f0 = (del0(2) - del0(3) + del0(4)*11/12 - del0(5)*5/6) / h^2;
disp(round(d2f0*10^1)/10^1,"f''''(0) = ")
disp(round(df1*10)/10,"f''(0.2) = ")
```

Scilab code Exa 7.2 Determination of Differential Function Value

```
//Example 7.2

clc
clear

x = 1.4:0.2:2.2;
y = [4.0552 4.953 6.0496 7.3891 9.025];

n = length(x);
del = %nan*ones(n,5);
del(:,1) = y';
for j = 2:5
    for i = 1:n-j+1
        del(i+j-1,j) = del(i+j-1,j-1) - del(i+j-2,j-1);
    end
end
mprintf("%5s %6s %10s %10s %9s %9s",'x','y','dy','d2y','d3y','d4y')
disp([x' del])
```

```

h = x(2) - x(1);
deln = del(5,:);

dfn = (deln(2) + deln(3)/2 + deln(4)/3 + deln(5)/4)
   / h;
d2fn = (deln(3) + deln(4) + deln(5)*11/12) / h^2;
dfn = round(dfn*10^4)/10^4;
d2fn = round(d2fn*10^4)/10^4;
disp(dfn,"y''(2.2) = ")
disp(d2fn,"y''''(2.2) = ")
```

Scilab code Exa 7.3 Determination of Differential Function Value

```
//Example 7.3

clc
clear

x = 0:4;
y = [6.9897 7.4036 7.7815 8.1281 8.451];

n = length(x);
del = %nan*ones(n,5);
del(:,1) = y';
for j = 2:6
    for i = 1:n-j+1
        del(i,j) = del(i+1,j-1) - del(i,j-1);
    end
end
del(:,1) = [];
n0 = length(del(1,:));

X = 2;
i = find(x==X);
```

```
dowy = 0;

for j = 1:n0
    if j==2*int(j/2) then
        add = del(i,j);
    else
        add = (del(i-1,j) + del(i,j))/2;
        i = i-1;
        if i==0 then
            break
        end
    end

    if add == %nan then
        break
    else
        dowy(j) = add;
    end
end
mprintf("%5s %6s %10s %9s %9s %9s",'x','y','dy','d2y
    ','d3y','d4y')
disp([x' y' del])

mu = 1;
h = x(2) - x(1);
dy2 = mu/h*(dowy(1) - 1/6*dowy(3));
d2y2 = mu/h^2*(dowy(2)-1/12*dowy(4));
dy2 = round(dy2*10^4)/10^4;
d2y2 = round(d2y2*10^4)/10^4;

disp(dy2,"y''(2) = ")
disp(d2y2,"y''''(2) = ")
```

Scilab code Exa 7.4 Determination of Differential Function Value

```
//Example 7.4

clc
clear

x = [0.15 0.21 0.23 0.27 0.32 0.35];
y = [0.1761 0.3222 0.3617 0.4314 0.5051 0.5441];

n = length(x);
del = %nan*ones(n,6);
del(:,1) = y';
for j = 2:6
    for i = 1:n-j+1
        del(i,j) = (del(i+1,j-1) - del(i,j-1)) / (x(
           i+j-1)-x(i));
    end
end
del(:,1) = [];
del = round(del*10^3)/10^3;
mprintf("%5s %6s %10s %10s %8s %9s %9s",'x','y','dy'
   ,'d2y','d3y','d4y','d5y')
disp([x' y' del])
X = poly(0, "X");
del0 = del(1,:);
y0 = y(1);
Y = y0;
for i = 1:length(del0)
    p = 1;
    for j = 1:i
        p = p*(X-x(j));
    end
    Y = Y + p*del0(i);
end

dydx = derivat(Y);
d2ydx = derivat(dydx);

XX = 0.25;
```

```
dy = horner(dydx,XX);
d2y = horner(d2ydx,XX);

disp(round(dy*10^4)/10^4,"y''(0.25) = ")
disp(d2ydx,"y''''(x) = ")
disp(d2y,"y''''(0.25) = ")
//The constant term in y''(x) is incorrectly
   computed to -91.7 instead of -97.42 in the text.
```

Scilab code Exa 7.5 Richardson Extrapolation Limit

```
//Example 7.5

clc
clear

function [f] = y(x)
    f = -1/x;
endfunction

H = [0.0128 0.0064 0.0032];
n = length(H);
x = 0.05;
h = H(1);
Fh = (y(x+h) - y(x-h)) / (2*h);
Fh2 = (y(x+h/2) - y(x-h/2)) / (h);
Fh4 = (y(x+h/4) - y(x-h/4)) / (h/2);

F1h2 = (4*Fh2 - Fh) / (4-1);
F1h4 = (4*Fh4 - Fh2) / (4-1);
F2h4 = (4^2*F1h4 - F1h2) / (4^2-1);
del = %nan*ones(n,3);
del(:,1) = [Fh Fh2 Fh4]';
del(1:2,2) = [F1h2 F1h4]';
del(1,3) = F2h4;
```

```

disp(del(1,n),"y''(0.05) = ")
Exact = 1/x^2;
disp(Exact,"Exact Value:")
```

Scilab code Exa 7.6 Integral using Trapezoidal and Simpson One Third Rule

```
//Example 7.6

clc
clear

function [I] = trap (fun,a,b,n)
// Integrate the function over the interval using Trapezoidal Formula
// trap (fun,a,b,n)
// fun - function to be integrated
// a - lower limit of integration
// b - upper limit of integration
// n - No. of times trapezoidal rule needs to be performed

N = n + 1;  // N - total no. of points
h = (b-a) / (N-1);
x = linspace(a,b,N);
y = fun(x);

sum1 = y(1) + 2 * sum(y(2:N-1)) + y(N);
I = h * sum1 / 2;                   // Trapezoidal Integral Value
endfunction

function [I] = simp13 (fun,a,b,n)
// Integrate the function over the interval using Simpson's 1/3rd rule
```

```
// simp13 (fun,a,b,n)
// fun - function to be integrated
// a - lower limit of integration
// b - upper limit of integration
// n - No. of times simpson's 1/3rd rule needs to be performed

N = 2 * n + 1;       // N - total no. of points
h = (b-a) / (N-1);
x = linspace(a,b,N);
y = fun(x);

sum1 = y(1) + 4 * sum(y(2:2:N-1)) + 2 * sum(y(3:2:N-2)) + y(N);
I = h* sum1 / 3;                    // Simpson's 1/3rd Integral Value
endfunction

n = 6;
ntrap = n;
ns13 = n/2;
I = [trap(sin,0,%pi,ntrap); simp13(sin,0,%pi,ns13)];
I = round(I*10^4)/10^4;
true = integrate('sin(x)','x',0,%pi);
err = abs(true - I) / true*100;
err = round(err*100)/100;

disp(I(1),"y_trap = ")
disp(I(2),"y_simp13 = ")
disp(err(1),"error_trap = ")
disp(err(2),"error_simp13 = ")
```

Scilab code Exa 7.7 Integral using Simpson One Third Rule

```
//Example 7.7
```

```
clc
clear

function [I] = simp13 (fun,a,b,n)
// Integrate the function over the interval using Simpson's 1/3rd rule
// simp13 (fun,a,b,n)
// fun - function to be integrated
// a - lower limit of integration
// b - upper limit of integration
// n - No. of times simpson's 1/3rd rule needs to be performed

N = 2 * n + 1;        // N - total no. of points
h = (b-a) / (N-1);
x = linspace(a,b,N);
y = fun(x);

sum1 = y(1) + 4 * sum(y(2:2:N-1)) + 2 * sum(y(3:2:N-2)) + y(N);
I = h* sum1 / 3;                    // Simpson's 1/3rd Integral Value
endfunction

n = 8;
ns13 = n/2;
I = simp13(log,1,5,ns13);
I = round(I*10^4)/10^4;
deff('[y] = true(x)',['y = x * log(x) - x']);
trueVal = true(5) - true(1);
err = abs(trueVal - I) / trueVal*100;
err = round(err*100)/100;

disp(I,"y_simp13 = ")
disp(trueVal,"Actual Integral = ")
disp(err,"error_simp13 = ")
```

Scilab code Exa 7.8 `Integral using Trapezoidal and Simpson One Third Rule`

```
//Example 7.8

clc
clear

function [I] = trap (fun,a,b,n)
// Integrate the function over the interval using
    Trapezoidal Formula
// trap (fun,a,b,n)
// fun - function to be integrated
// a - lower limit of integration
// b - upper limit of integration
// n - No. of times trapezoidal rule needs to be
    performed

N = n + 1;  // N - total no. of points
h = (b-a) / (N-1);
x = linspace(a,b,N);
y = fun(x);

sum1 = y(1) + 2 * sum(y(2:N-1)) + y(N);
I = h * sum1 / 2;                      // Trapezoidal
    Integral Value
endfunction

function [I] = simp13 (fun,a,b,n)
// Integrate the function over the interval using
    Simpson's 1/3rd rule
// simp13 (fun,a,b,n)
// fun - function to be integrated
// a - lower limit of integration
// b - upper limit of integration
```

```
// n - No. of times simpson's 1/3rd rule needs to be performed

N = 2 * n + 1;        // N - total no. of points
h = (b-a) / (N-1);
x = linspace(a,b,N);
y = fun(x);

sum1 = y(1) + 4 * sum(y(2:2:N-1)) + 2 * sum(y(3:2:N-2)) + y(N);
I = h* sum1 / 3;                        // Simpson's 1/3rd Integral Value
endfunction

function [f] = fun1(x)
    f = 1 ./(1+x^2);
endfunction


n = 4;
ntrap = n;
ns13 = n/2;
I = [trap(fun1,0,1,ntrap); simp13(fun1,0,1,ns13)];
I = round(I*10^4)/10^4;
true = intg(0,1,fun1);

disp(I(1),"y_trap = ")
disp(I(2),"y_simp13 = ")
disp(I(2)*4,"Approx pi = ")
```

Scilab code Exa 7.9 Integral using Simpson One Third Rule

```
//Example 7.9

clc
```

```
clear

function [I] = simp13 (fun,a,b,n)
// Integrate the function over the interval using
    Simpson's 1/3rd rule
// simp13 (fun,a,b,n)
// fun - function to be integrated
// a - lower limit of integration
// b - upper limit of integration
// n - No. of times simpson's 1/3rd rule needs to be
     performed

N = 2 * n + 1;       // N - total no. of points
h = (b-a) / (N-1);
x = linspace(a,b,N);
y = fun(x);

sum1 = y(1) + 4 * sum(y(2:2:N-1)) + 2 * sum(y(3:2:N
    -2)) + y(N);
I = h* sum1 / 3;                        // Simpson's 1/3
    rd Integral Value
endfunction

function [f] = fun1(x)
    f = sqrt(2/%pi)*exp(-x^2/2);
endfunction

h = 0.125;
n = (1-0)/h;
ns13 = n/2;
I = simp13(fun1,0,1,ns13);
I = round(I*10^4)/10^4;

disp(I,"Integral value, I = ")
```

Scilab code Exa 7.10 `Integral using Simpson One Third Rule`

```
//Etample 7.10

clc
clear

t = 0:10:80;
a = [30 31.63 33.34 35.47 37.75 40.33 43.25 46.69
    50.67];

h = t(2) - t(1);
n = length(t);

Is13 = a(1);
for i = 2:n-1
    rem2 = i-fix(i./2).*2;
    if rem2 == 0 then
        Is13 = Is13 + 4*a(i);
    else
        Is13 = Is13 + 2*a(i);
    end
end
Is13 = (Is13 + a(n))/10^3;
Is13 = round(h/3*Is13*10^4)/10^4;
disp(strcat(["v = ",string(Is13)," km/s"]))
```

Scilab code Exa 7.11 `Romberg Integration Method`

```
//Example 7.11

clc
clear

x = 1:0.1:1.8;
```

```
x = round(x*10)/10;
y = [1.543 1.669 1.811 1.971 2.151 2.352 2.577 2.828 3.107];
n = length(x);
x0 = x(1);
xn = x(n);

N = [1 2 4 8]
for j = 1:length(N)
    h = (xn - x0)./N(j);
    I = y(1);
    for xx = x0+h:h:xn-h
        xx = round(xx*10)/10;
        I = I + 2*y(x==xx);
    end
    Itrap(j) = h/2*(I + y(n));
    IRomb(1) = Itrap(1);
    if j~=1 then
        IRomb(j) = (4^(j-1)*Itrap(j)-IRomb(j-1))/(4^(j-1)-1);
    end
end
IRomb = round(IRomb*10^5)/10^5;

disp(Itrap(length(N)),"Integral using Trapezoidal rule:")
disp(IRomb(length(N)),"Integral using Romberg''s formula:")
//In third step of computation of integral using Romberg's formula, author mistakenly took the 1.7672 instead of 1.7684 which resulted in a difference
```

Scilab code Exa 7.12 `Double Integral using Trapezoidal Rule`

```
//Example 7.12

clc
clear

function [f] = fun1(x,y)
    f = 1 / (x+y);
endfunction

x = 1:0.25:2;
y = x;

m = length(x);
n = length(y);

del = %nan*ones(m,n);
for j = 1:n
    for i = 1:m
        del(i,j) = fun1(x(i),y(j));
    end
end

hx = x(2) - x(1);
for i = 1:m
    I = del(i,1);
    for j = 2:n-1
        I = I + 2*del(i,j);
    end
    Itrap1(i) = hx/2 * (I+del(i,n));
end
Itrap1 = round(Itrap1*10^4)/10^4;

hy = y(2) - y(1);
Itrap2 = Itrap1(1)
for i = 2:n-1
    Itrap2 = Itrap2 + 2* Itrap1(i);
end
Itrap2 = round(hy/2*(Itrap2+Itrap1(m))*10^4)/10^4;
```

```
disp(Itrap2,"I = ")
```

Scilab code Exa 7.13 Double Integral using Trapezoidal Rule

```
//Example 7.13

clc
clear

function [f] = fun1(x,y)
    f = sqrt(sin(x+y));
endfunction

x = 0:%pi/8:%pi/2;
y = x;

m = length(x);
n = length(y);

del = %nan*ones(m,n);
for j = 1:n
    for i = 1:m
        del(i,j) = fun1(x(i),y(j));
    end
end

hx = x(2) - x(1);
for i = 1:m
    I = del(i,1);
    for j = 2:n-1
        I = I + 2*del(i,j);
    end
    Itrap1(i) = hx/2 * (I+del(i,n));
end
Itrap1 = round(Itrap1*10^4)/10^4;
```

```
hy = y(2) - y(1);
Itrap2 = Itrap1(1)
for i = 2:n-1
    Itrap2 = Itrap2 + 2* Itrap1(i);
end
Itrap2 = round(hy/2*(Itrap2+Itrap1(m))*10^4)/10^4;
disp(Itrap2,"I = ")
```

Scilab code Exa 7.14 One Point Gauss Legendre Quadrature Formula

```
//Example 7.14

clc
clear

n = 1;
if n==1 then
    M = [0 2];
elseif n==2
    M = [sqrt(1/3) 1; -sqrt(1/3) 1];
elseif n==3
    M = [0 8/9; -0.774597 5/9; 0.774597 5/9];
elseif n==4
    M = [-0.339981 0.652145; -0.861136 0.347855;
       0.339981 0.652145; 0.861136 0.347855];
elseif n==5
    M = [-0 0.568889; -0.538469 0.467914; -0.906180
       0.236927; 0 0.568889; 0.538469 0.467914;
       0.906180 0.236927];
elseif n==6
    M = [-0.238619 0.467914; -0.661209 0.360762;
       -0.932470 0.171325; 0.238619 0.467914;
       0.661209 0.360762; 0.932470 0.171325];
end
```

```

X = M(:,1);
W = M(:,2);

disp(X,"E1 = ")
disp(W,"W1 = ")
```

Scilab code Exa 7.15 Two Point Gauss Legendre Quadrature Formula

```
//Example 7.15

clc
clear

n = 2;
if n==1 then
    M = [0 2];
elseif n==2
    M = [sqrt(1/3) 1; -sqrt(1/3) 1];
elseif n==3
    M = [0 8/9; -0.774597 5/9; 0.774597 5/9];
elseif n==4
    M = [-0.339981 0.652145; -0.861136 0.347855;
       0.339981 0.652145; 0.861136 0.347855];
elseif n==5
    M = [-0 0.568889; -0.538469 0.467914; -0.906180
       0.236927; 0 0.568889; 0.538469 0.467914;
       0.906180 0.236927];
elseif n==6
    M = [-0.238619 0.467914; -0.661209 0.360762;
       -0.932470 0.171325; 0.238619 0.467914;
       0.661209 0.360762; 0.932470 0.171325];
end

X = M(:,1);
```

```
W = M(:,2);

disp(W(1),"W1 = ")
disp(W(2),"W2 = ")
disp(X(1),"E1 = ")
disp(X(2),"E2 = ")
```

Scilab code Exa 7.16 Four Point Gauss Legendre Quadrature Formula

```
//Example 7.16

clc
clear

function [f] = fun1(x)
    f = 3*x^2 + x^3;
endfunction

n = 4;
if n==1 then
    M = [0 2];
elseif n==2
    M = [sqrt(1/3) 1];
elseif n==3
    M = [0 8/9; -0.774597 5/9; 0.774597 5/9];
elseif n==4
    M = [-0.339981 0.652145; -0.861136 0.347855;
       0.339981 0.652145; 0.861136 0.347855];
elseif n==5
    M = [-0 0.568889; -0.538469 0.467914; -0.906180
       0.236927; 0 0.568889; 0.538469 0.467914;
       0.906180 0.236927];
elseif n==6
    M = [-0.238619 0.467914; -0.661209 0.360762;
       -0.932470 0.171325; 0.238619 0.467914;
```

```
        0.661209 0.360762; 0.932470 0.171325];
end

X = M(:,1);
W = M(:,2);
I = 0;
for i = 1:length(X)
    I = I + W(i)*fun1(X(i));
end
disp(I,"I = ")
```

Chapter 8

Ordinary Differential Equations

Scilab code Exa 8.1 Initial Value Problem using Taylor Series Method

```
//Example 8.1

clc
clear

function [f] = dydt(t,y)
    f = t+y;
endfunction

y0 = 0;
t0 = 1;
t = 1.2;
h = 0.1;

n = (t-t0)/h;
tt = t0;
y = y0;
den = [1 2 6 24 120];
for i = 1:n
    d2ydt = 1 + dydt(tt,y);
    d3ydt = d2ydt;
```

```
    d4ydt = d3ydt;
    d5ydt = d4ydt;
    dy = [dydt(tt,y) d2ydt d3ydt d4ydt d5ydt];
    tt = tt + h;
    for j = 1:length(dy)
        y = y + dy(j)*(tt-t0)^j/den(j);
    end
    t0 = tt;
end
disp(y,"y(1.2) = ")

function [f] = closed(t)
    f = -t -1 + 2*exp(t-1);
endfunction
yclosed = closed(1.2);
yclosed = round(yclosed*10^4)/10^4;
disp(yclosed,"y_closed form = ")
disp("Comparing the results obtained numerically and
    in closed form, we observe ")
disp("that they agree up to four decimals")
```

Scilab code Exa 8.2 Initial Value Problem using Euler Method

```
//Example 8.2

clc
clear

function [f] = dydt(t,y)
    f = (y-t) / (y+t);
endfunction

y0 = 1;
t0 = 0;
t = 0.1;
```

```
n = 5;
h = (t-t0)/n;

tt = t0;
y = y0;
for i = 1:n
    y = y +h*dydt(tt,y);
    y = round(y*10^4)/10^4;
    tt = tt + h;
end
disp(y,"y(t = 0.1) = ")
```

Scilab code Exa 8.3 Initial Value Problem using Modified Euler Method

```
//Example 8.3

clc
clear

function [f] = dydt(t,y)
    f = t + sqrt(y);
endfunction

y0 = 1;
t0 = 0;
h = 0.2;
t = 0.6;
n = (t-t0)/h;

tt = t0;

for i = 1:n
    y11 = y0 + h*dydt(tt,y0);
    t1 = tt + h;
    y1 = y0 + h/2*(dydt(tt,y0) + dydt(t1,y11));
```

```
    y1 = round(y1*10^4)/10^4;

    y(i) = y1;
    y0 = y1;
    tt = t1;
end
mprintf("%5s %8s",'t','y')
disp([(t0+h:h:t)' y])
```

Scilab code Exa 8.4 Initial Value Problem using Second Order Runge Kutta Method

```
//Example 8.4

clc
clear

function [f] = fun1(x,y)
    f = (y+x) / (y-x);
endfunction

function [f] = rk2(x,y)
    k1 = h*fun1(x,y);
    k2 = h*fun1(x+3/2*h,y+3/2*k1);
    f = y + 1/3*(2*k1+k2);
endfunction

x0 = 0;
y0 = 1;
h = 0.2;
x = 0.4;
n = (x-x0)/h;

for i = 1:n
    y = rk2(x0,y0);
    x0 = x0 + h;
```

```
    y0 = y;
    y = round(y*10^5)/10^5;
end

disp(y,"y(0.4) = ")
```

Scilab code Exa 8.5 Initial Value Problem using Fourth Order Runge Kutta Method

```
//Etample 8.5

clc
clear

function [f] = fun1(t,y)
    f = t+y;
endfunction

function [f] = rk4(t,y)
    k1 = h*fun1(t,y);
    k2 = h*fun1(t+1/2*h,y+1/2*k1);
    k3 = h*fun1(t+1/2*h,y+1/2*k2);
    k4 = h*fun1(t+h,y+k1);
    f = y + 1/6*(k1+2*k2+2*k3+k4);
endfunction

t0 = 0;
y0 = 1;
h = 0.1;
t = 0.4;
n = (t-t0)/h;

for i = 1:n
    y = rk4(t0,y0);
    t0 = t0 + h;
    y0 = y;
```

```
    y = round(y*10^5)/10^5;
end

disp(y,"y(0.4) = ")
```

Scilab code Exa 8.6 Van Der Pol Equation using Fourth Order Runge Kutta Equation

```
//Example 8.6

clc
clear

function [f] = f1(x,y,p)
    f = p;
endfunction

function [f] = f2(x,y,p)
    f = 0.1*(1-y^2)*p - y;
endfunction

x0 = 0;
y0 = 1;
p0 = 0;
h = 0.2;
x = 0.2;
n = (x-x0)/h;

for i = 1:n
    k1 = h*f1(x0,y0,p0);
    l1 = h*f2(x0,y0,p0);
    k2 = h*f1(x0+h/2,y0+k1/2,p0+l1/2);
    l2 = h*f2(x0+h/2,y0+k1/2,p0+l1/2);
    k3 = h*f1(x0+h/2,y0+k2/2,p0+l2/2);
    l3 = h*f2(x0+h/2,y0+k2/2,p0+l2/2);
    k4 = h*f1(x0+h,y0+k3,p0+l3);
```

```
    l4 = h*f2(x0+h,y0+k3,p0+l3);
    y = y0 + 1/6*(k1+2*(k2+k3)+k4);
    p = p0 + 1/6*(l1+2*(l2+l3)+l4);
    y = round(y*10^4)/10^4;
    p = round(p*10^4)/10^4;
end

disp(y,"y(0.2) = ")
disp(p,"y''(0.2) = ")
```

Scilab code Exa 8.7 `Milne Predictor Corrector Method`

```
//Example 8.7

clc
clear

function [f] = dy(t,y)
    f = 1/2*(t+y);
endfunction

tt = 0:0.5:1.5;
yy = [2 2.636 3.595 4.968];

t0 = tt(1);
y0 = yy(1);
t = 2;
h = tt(2) - tt(1);
n = (t-t0)/h;
for i = 1:n
    dydt(1) = dy(t0,yy(1));
    dydt(2) = dy(t0+h,yy(2));
    dydt(3) = dy(t0+2*h,yy(3));
    dydt(4) = dy(t0+3*h,yy(4));

```

```
    yP = yy(1) + 4*h/3*(2*dydt(2)-dydt(3)+2*dydt(4))
        ;
    dydt(5) = dy(t0+4*h,yP);
    yC = yy(3) + h/3*(dydt(3)+4*dydt(4)+dydt(5));
end
yC = round(yC*10^4)/10^4;
disp(yC,"y(2.0) = ")
```

Scilab code Exa 8.8 Milne Predictor Corrector Method

```
//Example 8.8

clc
clear

function [f] = dy(t,y)
    f = t+y;
endfunction


tt = 0:0.1:0.3;
yy = [1 1.1103 1.2428 1.3997];

t0 = tt(1);
y0 = yy(1);
t = 2;
h = tt(2) - tt(1);
n = (t-t0)/h;
for i = 1:n
    dydt(1) = dy(t0,yy(1));
    dydt(2) = dy(t0+h,yy(2));
    dydt(3) = dy(t0+2*h,yy(3));
    dydt(4) = dy(t0+3*h,yy(4));

    yP = yy(1) + 4*h/3*(2*dydt(2)-dydt(3)+2*dydt(4))
```

```
    ;
    dydt(5) = dy(t0+4*h,yP);
    yC = yy(3) + h/3*(dydt(3)+4*dydt(4)+dydt(5));
end
yC = round(yC*10^4)/10^4;
disp(yC,"y(0.4) = ")

t = [tt'; t0+4*h];
y = [yy'; yC];
mprintf("\n%6s %8s",'t','y')
disp([t y])
```

Scilab code Exa 8.9 Adam Moulton Predictor Corrector Method

```
//Example 8.9

clc
clear

function [f] = fun1(t,y)
    f = y - t^2;
endfunction

function [f] = rk4(t,y)
    k1 = h*fun1(t,y);
    k2 = h*fun1(t+1/2*h,y+1/2*k1);
    k3 = h*fun1(t+1/2*h,y+1/2*k2);
    k4 = h*fun1(t+h,y+k1);
    f = y + 1/6*(k1+2*k2+2*k3+k4);
endfunction

t0 = 0;
y0 = 1;
t = 1;
h = 0.2;
```

```
n = (t-t0)/h;
y = y0;

for i = 2:4
    y(i) = rk4(t0,y0);
    t0 = t0 + h;
    y0 = y(i);
end

t0 = 0;
dydt(1) = fun1(t0,y(1));
dydt(2) = fun1(t0+h,y(2));
dydt(3) = fun1(t0+2*h,y(3));
dydt(4) = fun1(t0+3*h,y(4));

for i = 1:n-3
    yP = y(4) + h/24*(55*dydt(4)-59*dydt(3)+37*dydt(2)-9*dydt(1));
    dydt(5) = fun1(t0+(3+i)*h,yP);
    yC = y(4) + h/24*(9*dydt(5)+19*dydt(4)-5*dydt(3)+dydt(2));
    y = [y(2:4); yC];
    dydt = [dydt(2:4); fun1(t0+(3+i)*h,yC)]
end
disp(yC,"Computed Solution: y(1.0) = ")

function [f] = true(t)
    f = t^2 + 2*t +2 - exp(t);
endfunction
ytrue = true(1.0);
ytrue = round(ytrue*10^4)/10^4;
disp(ytrue,"Analytical Solution: y(1.0) = ")
```

Chapter 9

Parabolic Partial Differential Equations

Scilab code Exa 9.1 `Taylor Series Expansion`

```
// Example 9.1
// This is an analytical problem and need not be
   coded.
```

Scilab code Exa 9.2 `Initial Boundary Value Problem using Explicit Finite Differenc`

```
//Example 9.2

clc
clear

delx = 0.1;
delt = 0.002;
xf = 1;
tf = 0.006;
x = 0:delx:xf;
```

```
t = 0:delt:tf;
m = length(x);
n = length(t);
lamda = delt/delx^2;

y = zeros(n,m);
y(1:n,1) = 0;
y(1:n,m) = 0;
y(1,1:m) = sin(%pi*x);
for k = 2:n
    M1 = zeros(m-2);
    M2 = zeros(m-2,1);
    for i = 1:m-2
        M1(i,i) = 1+2*lamda;
        if i==1
            M1(i,i+1) = -lamda;
            M2(i) = y(k-1,i+1) + lamda*y(k,i);
        elseif i==m-2
            M1(i,i-1) = -lamda;
            M2(i) = y(k-1,i+1) + lamda*y(k,i+2);
        else
            M1(i,i+1) = -lamda;
            M1(i,i-1) = -lamda;
            M2(i) = y(k-1,i+1);
        end
    end
    y(k,2:m-1) = (M1\M2)';
end
y = round(y*10^4)/10^4;
mprintf("%4s %7s %9s %8s %9s %9s %9s %9s %9s %9s %9s
    %9s %9s",'n','t','x = 0.0','x = 0.1','x = 0.2','
   x = 0.3','x = 0.4','x = 0.5','x = 0.6','x = 0.7',
   'x = 0.8','x = 0.9','x = 1.0');
disp([(0:n-1)' t' y])

disp("At t = 0.006:")
disp(y(n,1:m),"Computed Solution:")
Texact = exp(-%pi^2*tf)*sin(%pi*x);
```

```
Texact = round(Texact*10^4)/10^4;
disp(Texact,"Analytical Solution:")
```

Scilab code Exa 9.3 Initial Boundary Value Problem using Explicit Finite Differenc

```
//Example 9.3

clc
clear

delx = 0.1;
delt = 0.001;
xf = 0.5;
tf = 0.003;
x = 0:delx:xf;
t = 0:delt:tf;
m = length(x);
n = length(t);
r = delt/delx^2;

T = zeros(m,n);
T(1:m,1) = 0;
delTxi = 0;
delTxf = 1;

for j = 1:n
    M1 = zeros(m,m);
    M2 = zeros(m,1);
    for i = 1:m
        if i == 1 then
            M1(i,i) = 1;
            M1(i,i+1) = -1;
            M2(i) = -delx * delTxi;
        elseif  i == m then
            M1(i,i) = 1;
```

```
            M1(i,i-1) = -1;
            M2(i) = delx * delTxf;
        else
            M1(i,i) = 1;
            M2(i) = r*T(i+1,j) + (1-2*r) * T(i,j) +
               r*T(i-1,j);
        end
    end
    T(1:m,j+1) = (M1\M2);
end
T = T(:,2:n+1);
mprintf("%4s %7s %9s %5s %7s %9s %9s %9s",'n','t','x
    = 0.0','x=0.1','x = 0.2','x = 0.3','x = 0.4','x
   = 0.5');
disp([(0:n-1)' t' T'])
```

Scilab code Exa 9.4 Crank Nicolson Finite Difference Method

```
//Example 9.4

clc
clear

delx = 0.25;
delt = 1/32;
xf = 1;
tf = delt;
x = 0:delx:xf;
t = 0:delt:tf;
m = length(x);
n = length(t);
r = delt/delx^2;


T = zeros(m,n);
```

```
T(1:m,1) = 1;
T(1,1:n) = 0;
T(m,1:n) = 0;

for j = 1:n-1
    M1 = zeros(m-2,m-2);
    M2 = zeros(m-2,1);
    for i = 2:m-1
        if i == 2 then
            M1(i-1,i-1)   = -2*(1+r);
            M1(i-1,i) = r;
            M2(i-1)     = -(r*T(i+1,j) + 2*(1-r)*T(i,j) + r*T(i-1,j));
        elseif i == m-1
            M1(i-1,i-2) = r;
            M1(i-1,i-1)   = -2*(1+r);
            M2(i-1)     = -(r*T(i+1,j) + 2*(1-r)*T(i,j) + r*T(i-1,j));
        else
            M1(i-1,i-2) = r;
            M1(i-1,i-1)   = -2*(1+r);
            M1(i-1,i) = r;
            M2(i-1)     = -(r*T(i+1,j) + 2*(1-r)*T(i,j) + r*T(i-1,j));
        end
    end
    T(2:m-1,j+1) = (M1\M2);
end
T1 = M1\M2;
for i = 1:length(T1)
    disp(strcat(["T",string(i)," = ",string(T1(i))]));
end
```

Scilab code Exa 9.5 Crank Nicolson Finite Difference Method

```
//Example 9.5

clc
clear

delx = 1;
delt = 1.893;
alpha = 0.132;
xf = 4;
tf = delt;
x = 0:delx:xf;
t = 0:delt:tf;
m = length(x);
n = length(t);
r = alpha*delt/delx^2;
r = round(r*10^2)/10^2;

T = zeros(m,n);
T(1:m,1) = 1000;

for j = 1:n-1
    M1 = zeros(m,m);
    M2 = zeros(m,1);
    for i = 1:m
        if i == 1 then
            M1(i,i) = -(2+2.15*r);
            M1(i,i+1) = 2*r;
            M2(i) = -(2*r*T(2,j) + (2-2.15*r)*T(1,j) + 21*r);
    elseif  i == m then
        M1(i,i) = -(2+1.75*r);
        M1(i,i-1) = 2*r;
        M2(i) = -(2*r*T(m-1,j) + (2-1.75*r)*T(m,j) - 35*r);
        else
            M1(i,i-1) = r;
            M1(i,i)   = -2*(1+r);
            M1(i,i+1) = r;
```

```
                M2(i)      = -(r*T(i+1,j) + 2*(1-r)*T(i,j
                    ) + r*T(i-1,j));
            end
        end
        T(1:m,j+1) = (M1\M2);
end
disp(M1,"Coefficient Matrix:")
disp(M2,"Constant Matrix:")
T = round(T*10^4)/10^4;
disp(T',"Table:")
```

Scilab code Exa 9.6 Crank Nicolson Scheme for Diffusion Equation

```
// Example 9.6
// This is an analytical problem and need not be
   coded.
```

Scilab code Exa 9.7 Alternate Direction Implicit Method

```
//Example 9.7

clc
clear

h = 2;
delt = 4;
tf = 8;
xf = 8;
yf = 6;
x = 0:h:xf;
y = 0:h:yf;
t = 0:delt:tf;
m = length(x);
```

```
n = length(y);
p = length(t);
r = delt/h^2;
r = round(r*10^2)/10^2;

T = 50*ones(n,m);
T0 = T;
T(1,1:m) = 110:-10:70;
T(n,1:m) = 0:10:40;
T(2:n-1,1) = [65; 25];
T(2:n-1,m) = [60; 50];

u = (m-2)*(n-2);
index = [repmat(2:m-1,1,n-2); gsort(repmat(2:n-1,1,m
    -2))];

M1 = zeros(u,u);
M2 = zeros(u,1);
for j = 2:m-1
    for i = 2:n-1
        ind = find(index(1,:)== j& index(2,:) == i);
        if j == 2 then
            M1(ind,ind) = 1+2*r;
            M1(ind,ind+1) = -r;
            M2(ind) = r*T(i,j-1) + r*T0(i-1,j) +
                (1-2*r)*T0(i,j) + r*T0(i+1,j);
        elseif j == m-1 then
            M1(ind,ind-1) = -r;
            M1(ind,ind) = 1+2*r;
            M2(ind) = r*T(i,j+1) + r*T0(i-1,j) +
                (1-2*r)*T0(i,j) + r*T0(i+1,j);
        else
            M1(ind,ind-1) = -r;
            M1(ind,ind) = 1+2*r;
            M1(ind,ind+1) = -r;
            M2(ind) = r*T0(i-1,j) + (1-2*r)*T0(i,j)
                + r*T0(i+1,j);
        end
```

```
    end
end
value = M1\M2;
value = round(value*10^4)/10^4;
for i = 1:length(index(1,:))
    t = index(:,i);
    T(t(2),t(1)) = value(i);
end
disp(T,"At t = 4:")
T0 = T;

index = gsort(index,'lc','i');
M1 = zeros(u,u);
M2 = zeros(u,1);
for j = 2:m-1
    for i = 2:n-1
        ind = find(index(1,:)== j& index(2,:) == i);
        if i == 2 then
            M1(ind,ind) = 1+2*r;
            M1(ind,ind+1) = -r;
            M2(ind) = r*T(i-1,j) + r*T0(i,j-1) + (1-2*r)*T0(i,j) + r*T0(i,j+1);
        elseif i == n-1 then
            M1(ind,ind-1) = -r;
            M1(ind,ind) = 1+2*r;
            M2(ind) = r*T(i+1,j) + r*T0(i,j-1) + (1-2*r)*T0(i,j) + r*T0(i,j+1);
        else
            M1(ind,ind-1) = -r;
            M1(ind,ind) = 1+2*r;
            M1(ind,ind+1) = -r;
            M2(ind) = + r*T0(i,j-1) + (1-2*r)*T0(i,j) + r*T0(i,j+1);
        end
    end
end
value = M1\M2;
value = round(value*10^4)/10^4;
```

```
for i = 1:length(index(1,:))
    t = index(:,i);
    T(t(2),t(1)) = value(i);
end
disp(T,"At t = 8:")
```

Chapter 10

Elliptical Partial Differential Equations

Scilab code Exa 10.1 Laplace Equation using Five Point Formulae

```
//Example 10.1

clc
clear

h = 1/4;
xf = 1;
yf = 1;
x = 0:h:xf;
y = 0:h:yf;
m = length(y);
n = length(x);

u = zeros(m,n);
u(m,:) = 100*x;
u(:,n) = 100*y';
u0 = u;

I = ceil(m/2);
```

```
J = ceil(n/2);

u(J,I) = (u0(J-2,I-2) + u0(J-2,I+2) + u0(J+2,I-2) +
    u0(J+2,I+2)) / 4;

for j = [J-1 J+1]
    for i = [I-1 I+1]
        u(j,i) = (u(j-1,i-1) + u(j-1,i+1) + u(j+1,i
            -1) + u(j+1,i+1)) / 4;
    end
end

j1 = [J-1 J J J+1];
i1 = [I I-1 I+1 I];
for k = 1:4
    i = i1(k);
    j = j1(k);
    u(j,i) = (u(j,i-1) + u(j,i+1) + u(j-1,i) + u(j
        +1,i)) / 4;
end

disp(u,"u:")
```

Scilab code Exa 10.2 Temperature in Two Dimensional Geometry

```
// Example 10.2
// This is an analytical problem and need not be
    coded.
```

Scilab code Exa 10.3 Laplace Equation in Two Dimension using Five Point Formulae

```
//Example 10.3

```

```
clc
clear

m = 5;
n = 5;
u = zeros(m,n);
u(m,:) = [50 100 100 100 50];
u0 = u;
I = ceil(m/2);
J = ceil(n/2);

u(J,I) = (u0(J-2,I-2) + u0(J-2,I+2) + u0(J+2,I-2) +
    u0(J+2,I+2)) / 4;

for j = [J-1 J+1]
    for i = [I-1 I+1]
        u(j,i) = (u(j-1,i-1) + u(j-1,i+1) + u(j+1,i
            -1) + u(j+1,i+1)) / 4;
    end
end

j1 = [J-1 J J J+1];
i1 = [I I-1 I+1 I];
for k = 1:4
    i = i1(k);
    j = j1(k);
    u(j,i) = (u(j,i-1) + u(j,i+1) + u(j-1,i) + u(j
        +1,i)) / 4;
end

kf = 2;
tab = zeros(kf+1,(m-2)*(n-2));
row = [];
for j = 2:n-1
    row = [row u(j,2:m-1)];
end
tab(1,:) = row;
for k = 1:kf
```

```
    row = [];
    for j = 2:n-1
        for i = 2:m-1
            u(j,i) = (u(j,i-1) + u(j,i+1) + u(j-1,i) + u(j+1,i)) / 4;
        end
        row = [row u(j,2:m-1)];
    end
    row = round(row*10^4)/10^4;
    tab(k+1,:) = row;
end
mprintf("%4s %9s %9s %9s %9s %10s %10s %10s %10s %10s",'r','u11','u21','u31','u12','u22','u32','u13','u23','u33')
disp([(1:k+1)' tab])
```

Scilab code Exa 10.4 Poisson Equation using Liebmann Iterative Method

```
//Example 10.4

clc
clear

h = 1/3;
x = 0:h:1;
y = 0:h:1;
m = length(y);
n = length(x);
u = zeros(m,n);
u(m,2:n-1) = 1;

kf = 5;
tab = zeros(kf,(m-2)*(n-2));
for k = 1:kf
    row = [];
```

```
    for j = 2:n-1
        for i = 2:m-1
            constant = 10/9* (5 + 1/9*(i-1)^2 +
               1/9*(j-1)^2);
            u(j,i) = (u(j,i-1) + u(j,i+1) + u(j-1,i)
                + u(j+1,i) + constant) / 4;
        end
        row = [row u(j,2:m-1)];
    end
    row = round(row*10^4)/10^4;
    tab(k,:) = row;
end
mprintf("%4s %9s %9s %9s %9s",'r','u11','u21','u12',
   'u22')
disp([(1:k)' tab])
```

Scilab code Exa 10.5 Laplace Equation using Liebmann Over Relaxation Method

```
//Example 10.5

clc
clear

x = 0:4;
y = 0:4;
m = length(y);
n = length(x);
u = zeros(m,n);
u(m,:) = x.^3;
u(:,n) = 16*y';
u0 = u;

I = ceil(m/2);
J = ceil(n/2);

```

```
u(J,I) = (u0(J-2,I-2) + u0(J-2,I+2) + u0(J+2,I-2) +
    u0(J+2,I+2)) / 4;

for j = [J-1 J+1]
    for i = [I-1 I+1]
        u(j,i) = (u(j-1,i-1) + u(j-1,i+1) + u(j+1,i
            -1) + u(j+1,i+1)) / 4;
    end
end

j1 = [J-1 J J J+1];
i1 = [I I-1 I+1 I];
for k = 1:4
    i = i1(k);
    j = j1(k);
    u(j,i) = (u(j,i-1) + u(j,i+1) + u(j-1,i) + u(j
        +1,i)) / 4;
end
disp(u,"u:")

p = m-1;
q = n-1;
c = cos(%pi/p) + cos(%pi/q);
w = 4/(2+sqrt(4-c^2));
w = round(w*10^3)/10^3;

kf = 10;
tab = zeros(kf+1,(m-2)*(n-2));
row = [];
for j = 2:n-1
    row = [row u(j,2:m-1)];
end
tab(1,:) = row;
for k = 1:kf
    row = [];
    for j = 2:n-1
        for i = 2:m-1
            u(j,i) = (u(j,i-1) + u(j,i+1) + u(j-1,i)
```

```
                + u(j+1,i)) *w/4 + (1-w)*u(j,i);
        end
        row = [row u(j,2:m-1)];
    end
    row = round(row*10^4)/10^4;
    tab(k+1,:) = row;
end
mprintf("\n\n%8s %9s %10s %10s %9s %10s %10s %9s %9s
   ",'u11','u21','u31','u12','u22','u32','u13','u23'
   ,'u33')
disp(tab)
```

Chapter 11

Hyperbolic Partial Differential Equations

Scilab code Exa 11.1 `Initial Value Problem using Wave Equation`

```
//Example 11.1

clc
clear

delx = 1/8;
delt = 1/8;
x = 0:delx:1;
t = 0:delt:1;
m = length(x);
n = length(t);
u =zeros(n,m);
u(1,:) = sin(%pi*x);
N = 1/delx;
r = delt/delx;

for j = 2:n
    for i = 2:m-1
        if j == 2 then
```

```
            u(j,i) = (2*(1-r^2)*u(j-1,i) + r^2*(u(j
               -1,i-1) + u(j-1,i+1))) /2;
        else
            u(j,i) = 2*(1-r^2)*u(j-1,i) + r^2*(u(j
               -1,i-1) + u(j-1,i+1)) - u(j-2,i);
        end
    end
end
u = round(u*10^4)/10^4;
mprintf("\n\n%6s %9s %9s %8s %s %8s %10s %10s %9s
   %7s %s",'t','x = 0.0','x = 0.125','x = 0.25','x =
    0.375','x = 0.5','x=0.625','x = 0.75','x=0.875',
   'x = 1.0','n');
disp([(0:1/8:1)' u (0:n-1)']);
mprintf("\n\n");
t = [1/2 1];
for i = 1:length(t)
    Ex(i,:) = sin(%pi*x) * cos(%pi*t(i));
end
Ex = round(Ex*10^4)/10^4;
disp("At t = 1/2:")
disp(u(find(x==1/2),:),"Computed Solution:")
disp(Ex(1,:),"Actual Solution:")

disp("At t = 1:")
disp(u(find(x==1),:),"Computed Solution:")
disp(Ex(2,:),"Actual Solution:")
```

Scilab code Exa 11.2 `Initial Value Problem using Wave Equation`

```
//Example 11.2

clc
clear

```

```
delx = 0.2;
delt = 0.2;
x = 0:delx:1;
t = 0:delt:0.8;
m = length(x);
n = length(t);
u =zeros(n,m);
u(1,:) = x^2;
u(:,m) = 1+t';
N = 1/delx;
r = delt/delx;

for j = 2:n
    for i = 2:m-1
        if j == 2 then
            u(j,i) = (2*(1-r^2)*u(j-1,i) + r^2*(u(j-1,i-1) + u(j-1,i+1)) + 2*delt) /2;
        else
            u(j,i) = 2*(1-r^2)*u(j-1,i) + r^2*(u(j-1,i-1) + u(j-1,i+1)) - u(j-2,i);
        end
    end
end
u = round(u*10^4)/10^4;
mprintf("\n%5s %9s %7s %7s %s %6s %6s",'t','x = 0.0','x = 0.2','x = 0.4','x = 0.6','x = 0.8','x = 1.0');
disp([t' u])
```