DELHI TECHNOLOGICAL UNIVERSITY

(Department of Applied Mathematics)

Scientific Computing (Lab File)

Submitted to:

Prof. L. N. Das Sir

Submitted by:

Tanay Singhania
2k17/MC/107

DELHI TECHNOLOGICAL UNIVERSITY

Vision

" To be a world class university through education , innovation and research for the service of humanity "

Mission

1. To establish centres of excellence in emerging areas of science, engineering, technology, management and allied areas.
2. To foster an ecosystem for incubation, product development, transfer of technology and entrepreneurship.
3. To create environment of collaboration, experimentation, imagination and creativity.
4. To develop human potential with analytical abilities, ethics and integrity.
5. To provide environment friendly, reasonable and sustainable solutions for local & global needs.

Department of Applied Mathematics

Vision

To emerge as a center of excellence and eminence by imparting futuristic technical education with solid mathematical background in keeping with global standards, making our students technologically and mathematically competent and ethically strong so that they can readily contribute to the rapid advancement of society and mankind.

Mission

1. To achieve academic excellence through innovative teaching and learning practices.
2. To improve the research competence to address social needs.
3. To inculcate a culture that supports and reinforces ethical, professional behaviors for a harmonious and prosperous society.
4. Strive to make students to understand, appreciate and gain mathematical skills and develop logic, so that they are able to contribute intelligently in decision making which characterizes our scientific and technological age.

INDEX

EXPERIMENT -1

AIM : To obtain solution of a polynomial equation using bisection method

CODE :

```
function p = bisection(f,a,b)

if f(a)*f(b)>0
    disp('Wrong choice')
else
    p = (a + b)/2;
    err = abs(f(p));
    while err > 1e-7
        if f(a)*f(p)<0
           b = p;
        else
           a = p;
        end
        p = (a + b)/2;
        err = abs(f(p));
    end
end
```

OUTPUT :

```
>> eqn = @(x) (x^2) -5*x +6

eqn =

  function_handle with value:

    @(x)(x^2)-5*x+6
```

```
>> bisection(eqn,0,2.5)

ans =

    2.0000
```

```
>> bisection(eqn,2.1,4)

ans =

    3.0000
```

EXPERIMENT -2

AIM : To find the root of a polynomial using Newton-Rhapson method.

CODE :

```
function [solution] = newtonrMethod(n,xn)
    syms F(x)
    F = x*log10(x)-1.2;
    while(n>0)
        val = eval(subs(F,x,xn));
        if(val==0)
            break;
        end
        diffval = eval(subs(diff(F),x,xn));
        if(diffval==0)
            fprintf('Solution not found');
            break;
        end
        xn = xn-(val/diffval);
        n = n-1;
    end
    solution = xn;
end
```

OUTPUT :

```
>> newtonrMethod(100, 1)

ans =

    2.7406
```

EXPERIMENT -3

AIM : To find the root of a polynomial using Regula Falsi method.

CODE :

```
function[] = regular_falsi()
clc
itr=0;
x0= input('Enter the value of x0 :');
x1 = input('Enter the value of x1 :');
aerr = input('Enter the allowed error :');
maxitr = input('Enter the maximum number of iterations:');
[x2,itr] = regula(x0,x1,f(x0),f(x1),itr);
while(itr<maxitr)
    if (f(x0)*f(x1)<0)
        x1=x2;
    else
        x0 = x2;
    end
    [x3,itr] = regula(x0,x1,f(x0),f(x1),itr);
    if(abs(x3-x2)<aerr)
        fprintf('After %d iterations, roots %f \n',itr, x3);
        return
    end
    x2=x3;
end
fprintf('Iterations not sufficient, solution does not
converge\n');
end
function[x,itr_r] = regula(x0,x1,a,b,itr)
x =  x0 - ((x1-x0)/(b-a)) * a;
itr_r=itr+1;
fprintf('iteration no. %d * %f \n',itr, x);
end
function[y] = f(x)
y = (cos(x)-x*exp(x));
end
```

OUTPUT:

```
Enter the value of x0 :0
Enter the value of x1 :3
Enter the allowed error :0.001
Enter the maximum number of iterations:20
iteration no. 0 * 0.048195
iteration no. 1 * 0.931560
iteration no. 2 * 0.356566
iteration no. 3 * 0.468578
iteration no. 4 * 0.503344
iteration no. 5 * 0.513585
iteration no. 6 * 0.516554
iteration no. 7 * 0.517411
After 8 iterations, roots 0.517411
```

EXPERIMENT -4

AIM : to find the solution of a system of linear equations using Gauss Elimination method

CODE :

```
function [] = gauss_elimnation_method()
clc
N=3;
a = input('Enter the element of matrix');
for i=1:N-1
    for j=i+1:N
        t = a(i,i)/a(j,j);
        for k=1:N
            a(i,k) = a(i,k) - a(j,k) * t;
        end
    end
end
for i=1:N
    for l=1:N
        fprintf('%8.4f',a(i,l));
    end
    fprintf('\n');
end
for i=N:-1:1
    s=0;
    for o=i+1:N-1
        s = s + a(i,o+1).*x(i);
    end
    x(i) = (a(i,N)-s)/a(i,i);
end
end
```

OUTPUT:

```
Enter the element of matrix[1 2 1; 4 7 9; 3 9 6;]
  0.2143  0.3571 -0.7143
  0.5000 -3.5000  2.0000
  3.0000  9.0000  6.0000
```

EXPERIMENT -5

AIM : to find the solution of a system of linear equations using Gauss Jordan method.

CODE :

```
function[] = gauss_jordan_method()
N=4;
a = input('Enter the element of matrix :-\n');
for j=1:N
    for i=1:N
        if(i ~=j)
            t =  a(i,j)/a(i,i);
            for k = 1:N+1
                a(i,k) = a(i,k)-a(j,k).*t;
            end
        end
    end
end
fprintf('\n THe diagonal Matrix is :-\n');
disp(a);
fprintf('\n THe solution is :-\n');
for l=1:N
    fprintf('\n X[%d]=%f\n',l,a(l,N+1)./a(l,l));
end
end
```

OUTPUT:

```
>> gauss_jordan_method
Enter the element of matrix :-
[2 4 6; 8 9 3; 7 6 4;]

 THe diagonal Matrix is :-
   -7.8476   -8.7656    4.3950
    7.3120    4.6569   -4.9652
    2.5427   -1.8376   -6.1410


 THe solution is :-

 X[1]=-0.560041

 X[2]=-1.066203

 X[3]=1.000000
```

EXPERIMENT -6

AIM : to find the solution of a system of linear equations using Gauss Seidal method.

CODE :

```
function x = gauss_seidal(coeff_mat, b, ini)

lS = tril(coeff_mat);
u = triu(coeff_mat);
lSInv = inv(lS);
t = -lSInv*u;
c = lSInv*b;
sz = size(coeff_mat);
x = ini;
xtemp = zeros(sz(1), 1);
while(1 == 1)
    xtemp = t*x+c;
    if(abs(x-xtemp) < 0.00001)
        break;
    end
    x = xtemp;
end

end
```

OUTPUT:

```
>> coeffmat = [16 3; 7 -11];
>> b = [11; 13];
>> x0 = [0; 0];
>> gauss_seidal(coeffmat,b,x0)
X =
   0.6875
  -0.744
```

EXPERIMENT -7

AIM : to find the value in given dataset using Newton's forward difference interpolation

CODE :

```
function yi = Newton_FD(x, y, xi)
n=length(y)-1;
d=y;
for i=1:n
    for j=n+1:-1:i+1
        d(j)=(d(j-1)-d(j))/(x(j-1)-x(j));
    end
    yi=d(i+1);
    for j=i:-1:1
        yi=d(j)+(xi-x(j))*yi;
    end
    fprintf('Degree %d:\n',i)
    fprintf('\n')
    fprintf('Coefficients=')
    fprintf('\n\n')
    Coefficients=d(1:i+1);
    disp(Coefficients)
    fprintf('Value=')
    fprintf('\n\n')
    Value=yi;
    disp(Value)
end
```

OUTPUT:

```
>> x = [1 2 4 5]

x =

     1     2     4     5

>> y = [2 4 8 10]

y =

     2     4     8    10
```

```
>> Newton_FD(x,y,3)
Degree 1:

Coefficients=

     2     2

Value=

     6

Degree 2:

Coefficients=

     2     2     0
```

```
Value=

     6

Degree 3:

Coefficients=

     2     2     0     0

Value=

     6


ans =

     6
```

EXPERIMENT -8

AIM : to find the value in a given dataset using Lagrange interpolation

CODE :

```
function yi = Langrange(x, y, xi)
f = 0;
n=length(y);
for i=1:n
    nr = 1;
    dr = 1;
    for j=1:n
        if (j~=i)
            nr = nr*(xi-x(j));
            dr = dr*(x(i)-x(j));
        end
    end
    f = f + (nr/dr)*y(i);
end
disp(f);
yi = f;
end
```

OUTPUT:

```
x =

     1     2     4     8    10

>> y = [ 2 4 8 16 20]

y =

     2     4     8    16    20

>> Langrange(x,y,3)
     6


ans =

     6
```

EXPERIMENT -9

AIM : to find the integral of a function using trapezoidal rule.

CODE :

```
function[] = TrapezoidalRule()

clc
y = inline('1/(1+x.*x)');
x0 = input('Enter x0 : ');
xn = input('Enter xn : ');
n = input('Enter number of sub intervals');
h = (xn-x0)/n;
s = y(x0)+y(xn);
for i=1:n-1
    s = s + 2.*y(x0+i*4);
end
fprintf('The number of integral is :
%f\n',h/2.*s);
end
```

OUTPUT:

```
Enter x0 : 0
Enter xn : 2
Enter number of sub intervals8
The number of integral is : 0.172624
```

EXPERIMENT -10

AIM : To find the integral of a function using Simpson rule

CODE :

```
function[] = SimpsonRule()

clc
y = inline('1/(1+x.*x)');
x0 = input('Enter x0 : ');
xn = input('Enter xn : ');
n = input('Enter number of sub intervals');
h = (xn-x0)/n;
s = y(x0)+y(xn);
for i=1:n-1
    s = s + 2.*y(x0+i*2) + 4.*y(x0 + (2*i-1));
end
fprintf('The number of integral is :
%f\n',h/2.*s);
end
```

OUTPUT:

```
Enter x0 : 0
Enter xn : 1
Enter number of sub intervals4
The number of integral is : 0.578193
```