Numerical Analysis

LAB FILE

MSMA 110

2022-2023

Submitted To:

Dr. Vivek Kumar Aggarwal
Dr. Dinesh Udar
Ms. Kirti
Ms. Anu
Department of Applied Mathematics

Submitted By:

RITIKA GUPTA
2K22/MSCMAT/54

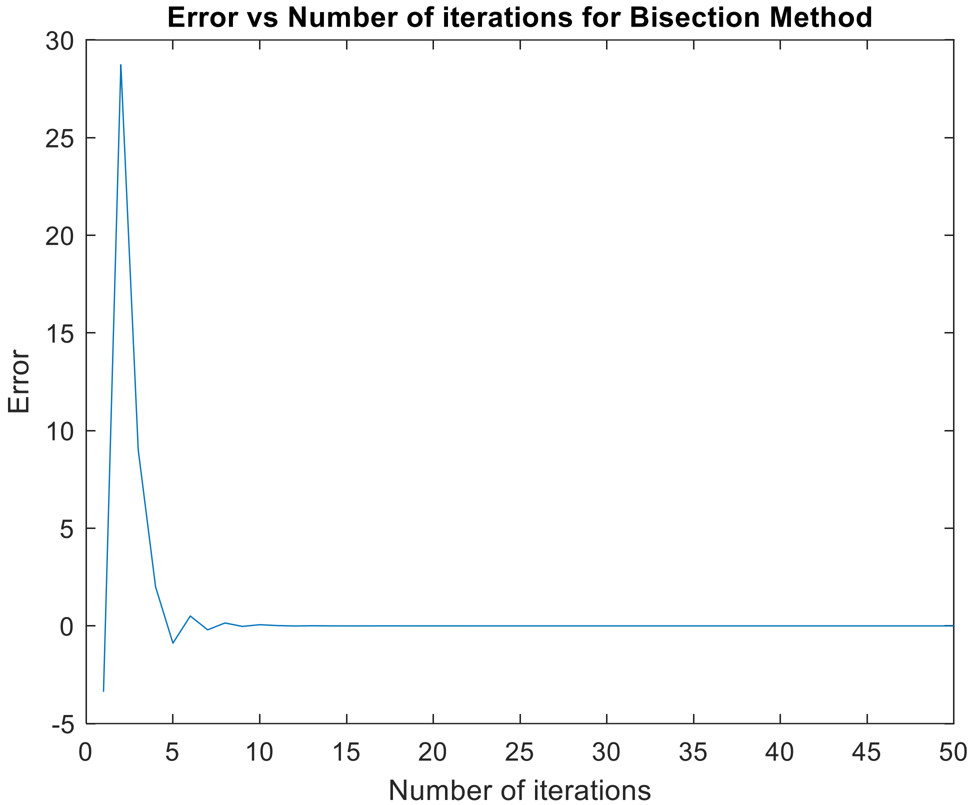
INDEX

S. NO.	PRACTICAL	DATE
1	Bisection method	04-01-23
2	Newton Raphson’s method and its convergence	11-01-23
3(a)	Gauss Elimination method	18-01-23
3(b)	LU Decomposition method	25-01-23
4	Gauss Seidel, Gauss Jacobi Method	08-02-23
5	Finding eigenvalues using power method	22-03-23
6	Newton Divided Difference	29-03-23
7	Lagrange Method for Interpolation	05-04-23
8	Spline’s Method for Interpolation	12-04-23
9	Simpson’s Rule	19-04-23
10	Trapezoidal Rule	19-04-23

```matlab
% PRACTICAL 1
% Bisection Method
% RITIKA GUPTA MSCMAT54

clear all;
f = input('Enter f(x) = ');
a = input('Enter value for a: ');
b = input('Enter value for b: ');

n = input('Enter maximum number of iterations: ');

if sign(f(a)) == sign(f(b))
    error('Intermediate value theorem not satisfied.')
end

p1=a; p2=b;
for i=1:200
    c=(a+b)/2;
    if f(c)>0
        b=c;
    else
        a=c;
    end
end
a=p1; b=p2; r=c;
for i=1:n
    c=(a+b)/2;
    er(i)=f(c)-f(r);
    if f(c)>0
        b=c;
    else
        a=c;
    end
end

fprintf('Approximate root of the given equation after %d iterations is %f',n,c);

plot(er);
xlabel('Number of iterations');
ylabel('Error');
title('Error vs Number of iterations for Bisection Method');
```

Command Window

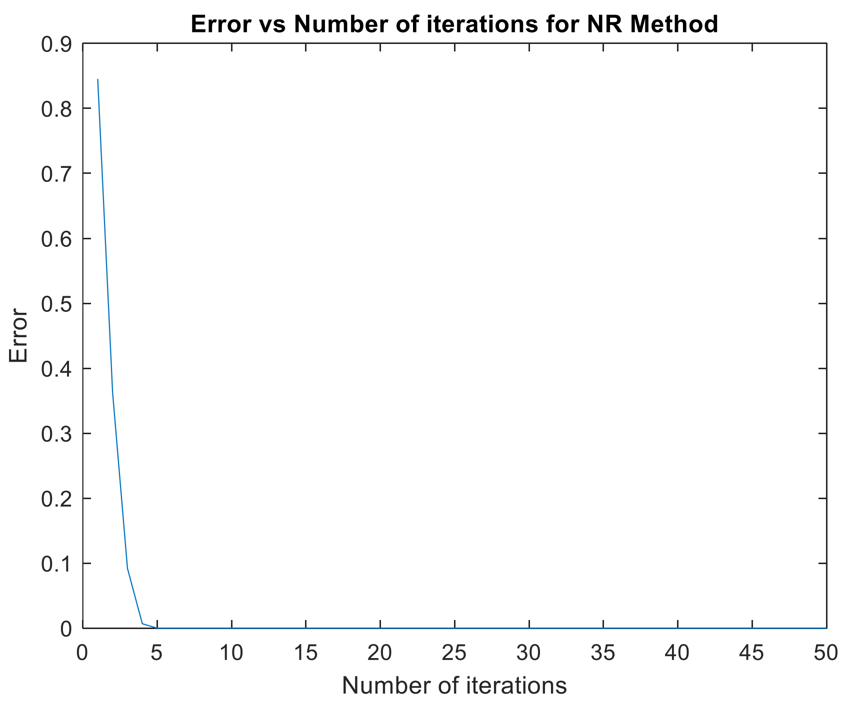
```
>> prac1_mscmat54_bisection
Enter f(x) = @(x) x^3-4*x-9
Enter value for a: 0
Enter value for b: 5
Enter maximum number of iterations: 50
fx Approximate root of the given equation after 50 iterations is 2.706528>>
```


Error vs Number of iterations for Bisection Method
Error
Number of iterations
30
25
20
15
10
5
0
-5
0
5
10
15
20
25
30
35
40
45
50

```
% PRACTICAL 2
% Newton Raphson Method
% RITIKA GUPTA MSCMAT54

clear all;
f = @(x) 2*x^3 + 2*x - 1;
df=@(x) 6*x^2 + 2;
x = input('Enter value for x0: ');
n = input('Enter maximum number of iterations: ');

x0=x;
for i=1:200
    x1=x- f(x)/df(x);
    x=x1;
end
root=x; x=x0;
for i=1:n
    x1=x- f(x)/df(x);
    x=x1;
    er(i)=x1-root;
end

fprintf('Approximate root of the given equation after %d iterations is %f',n,root);

plot(er);
xlabel('Number of iterations');
ylabel('Error');
title('Error vs Number of iterations for NR Method');
```

Command Window

```
>> prac2_mscmat54_newtonraphson
Enter value for x0: 2
Enter maximum number of iterations: 50
Approximate root of the given equation after 50 iterations is 0.423854>>
```



```
% PRACTICAL 3(a)
% Gauss Elimination Method
% RITIKA GUPTA MSCMAT54

clear all;
a=input('Enter coefficient matrix: ');
b=input('Enter column vector: ');
aug=[a b];
Augmented_form=aug;
n=length(b);

%reducing to echelon form
for j=1:n-1
    for i=j+1:n
        if aug(j,j)==0  %partial pivoting
            t=aug(j,:);aug(j,:)=aug(i,:);
            aug(i,:)=t;
        else
            m=aug(i,j)/aug(j,j);
            aug(i,:)=aug(i,:)-m*aug(j,:);
        end
    end
end

x=zeros(n,1);

%backward substitution
for i=n:-1:1
    x(i)=( aug(i,n+1) - aug(i,i+1:n)*x(i+1:n) )/aug(i,i);
end

Augmented_form
Echelon_form=aug
disp('By Gauss Elimination method, '); x
```

Command Window

```
>> prac3_mscmat54_Gauss_elimination
Enter coefficient matrix: [1 1 1 ; 4 3 -1 ; 3 5 3]
Enter column vector: [1;6;4]

Augmented_form =

     1     1     1     1
     4     3    -1     6
     3     5     3     4


Echelon_form =

     1     1     1     1
     0    -1    -5     2
     0     0   -10     5

By Gauss Elimination method,

x =

    1.0000
    0.5000
   -0.5000
```

```
% PRACTICAL 3(b)
% LU Decomposition
% Crout's Method
% RITIKA GUPTA MSCMAT54

clear all;
a=input('Enter coefficient matrix: ');
b=input('Enter column vector: ');
n=length(b);
l=zeros(n);
u=eye(n);
z=zeros(n,1);
x=zeros(n,1);

for i=1:n
    for j=1:n
        if i>=j
            l(i,j)=a(i,j)-l(i,1:j-1)*u(1:j-1,j);
        else
            u(i,j)=(a(i,j)-l(i,1:j-1)*u(1:j-1,j))/l(i,i);
        end
    end
end

%forward substitution for LZ=B
for i=1:n
    z(i)=( b(i) - l(i,1:i-1)*z(1:i-1) )/l(i,i);
end

%backward substitution for UX=Z
for i=n:-1:1
    x(i)=( z(i) - u(i,i+1:n)*x(i+1:n) )/u(i,i);
end

A=a
disp("Using Crout's method for LU Decomposition:");
L=l
U=u
X=x
```

Command Window

```
>> prac3_mscmat54_LU_decomposition
Enter coefficient matrix: [1 1 1 ; 4 3 -1 ; 3 5 3]
Enter column vector: [1;6;4]

A =

     1     1     1
     4     3    -1
     3     5     3

Using Crout's method for LU Decomposition:

L =

     1     0     0
     4    -1     0
     3     2   -10


U =

     1     1     1
     0     1     5
     0     0     1


X =

    1.0000
    0.5000
   -0.5000
fx
```

```
% PRACTICAL 4
% GAUSS JACOBI, GAUSS SEIDEL METHOD
% RITIKA GUPTA MSCMAT54

clear all;
a=input('Enter coefficient matrix: ');
b=input('Enter column vector: ');
num=input('Enter number of iterations to perform: ');
n=length(b);
x=zeros(n,1);
p=input('Enter initial approximation of X: ');
p0=p;

Augmented_form = [a b]

%Gauss Jacobi
disp('BY GAUSS JACOBI METHOD');
for j=1:num
    for i=1:n
        x(i)=b(i)/a(i,i)- (a(i,[1:i-1,i+1:n])*p([1:i-1,i+1:n]))/a(i,i);
    end
    fprintf('%dth iteration:',j); x'
    p=x;
end

%Gauss Seidel
clear x,p;
p=p0;x=zeros(n,1);
disp('BY GAUSS SEIDEL METHOD');
for j=1:num
    for i=1:n
        x(i)=b(i)/a(i,i)- (a(i,[1:i-1,i+1:n])*p([1:i-1,i+1:n]))/a(i,i);
        p(i)=x(i);
    end
    fprintf('%dth iteration:',j); x'
end
```

Command Window

```
>> prac4_mscmat54_gauss_jacobi_seidel
Enter coefficient matrix: [4 1 1 ; 1 5 2 ; 1 2 3]
Enter column vector: [2 ; -6 ; -4]
Enter number of iterations to perform: 3
Enter initial approximation of X: [0.5;-0.5;-0.5]

Augmented_form =

     4     1     1     2
     1     5     2    -6
     1     2     3    -4

BY GAUSS JACOBI METHOD
1th iteration:
ans =

    0.7500   -1.1000   -1.1667

2th iteration:
ans =

    1.0667   -0.8833   -0.8500

3th iteration:
ans =

    0.9333   -1.0733   -1.1000

BY GAUSS SEIDEL METHOD
1th iteration:
ans =

    0.7500   -1.1500   -0.8167

2th iteration:
ans =

    0.9917   -1.0717   -0.9494

3th iteration:
ans =

    1.0053   -1.0213   -0.9876
```

```matlab
% PRACTICAL  5
% Power method to determine eigenvalue
% RITIKA GUPTA MSCMAT54

clear all;
A=input('Enter matrix A: ');
v=input('Enter the initial vector: ');
tol=input('Enter the eroor tolerance: ');
n=input('Enter maximum number of iterations: ');
v0=v;

for i=1:n
    v=A*v0;
    m=max(abs(v));
    if m==max(v)
        m=m;
    else
        m=-m;
    end
    v=v/m;
    if abs(v-v0)<tol
        fprintf('\nAfter %d iterations,\n',i-1);
        break
    else
        v0=v;
    end
end

fprintf('By Power method,\n');
Dominant_Eigenvalue=m
Corresponding_Eigenvector=v
```

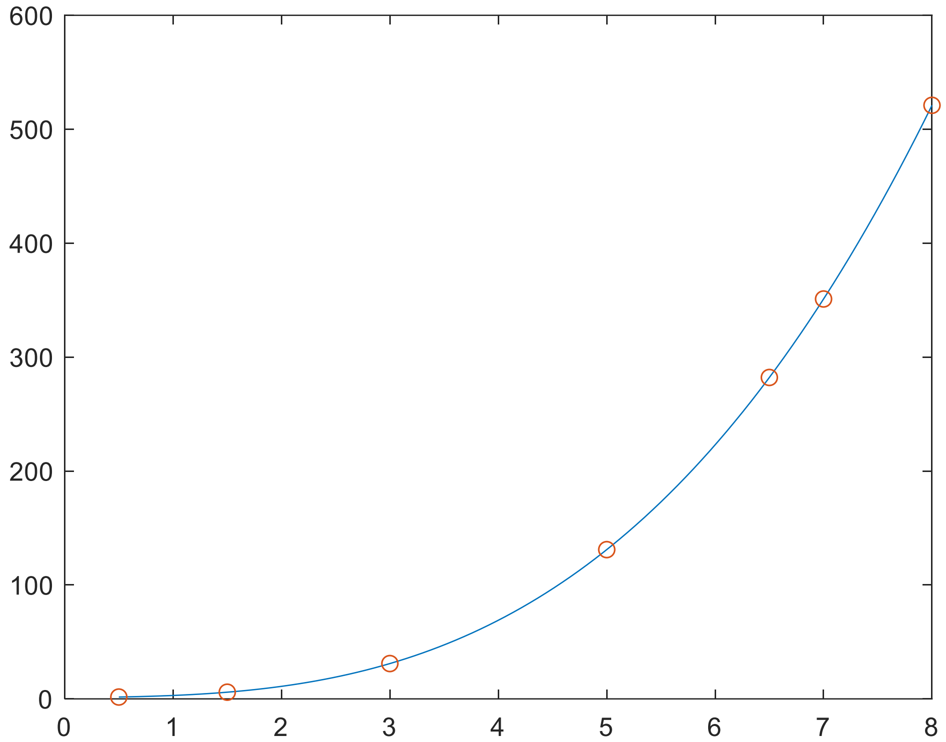
Command Window

```
>> prac5_mscmat54_power_method
Enter matrix A: [2 -12 ; 1 -5]
Enter the initial vector: [1;1]
Enter the eroor tolerance: 0.000001
Enter maximum number of iterations: 100

After 16 iterations,
By Power method,

Dominant_Eigenvalue =

   -2.0000


Corresponding_Eigenvector =

    1.0000
    0.3333
```

```
% PRACTICAL 6
% Newton Divided Difference Method
% RITIKA GUPTA MSCMAT54

x=input('Enter values of x = ');
y=input('Enter corresponding f(x) = ');
p0=input('Enter point of approximation: ');
n=length(x);
D=zeros(n); %Divided difference table
D(:,1)=y; %First column of DD table set as f(x)

for j=2:n
    for i=n:-1:j
        D(i,j) = (D(i,j-1) - D(i-1,j-1))/(x(i)-x(i-j+1));
    end
end
Divided_Difference_Table = [x D]

P=D(n,n);
for k=n-1:-1:1
    P=conv(P,poly(x(k)));   %poly(a) generates polynomial (x-a)
    m=length(P);
    P(m) = P(m) + D(k,k);
end
disp('Coefficients of the interpolated polynomial are:'); P
%conv(u,v) is convolution of u,v (polynomial multiplication)

%approximating polynomials at a point
fp=polyval(P,p0);
fprintf('f(%f) = %f',p0,fp);

%plotting the interpolated polynomial
X=linspace(x(1),x(n),100);
Y=polyval(P,X);
plot(X,Y,[x;p0],[y;fp],'o')
```

Command Window

```
>> prac6_mscmat54_Newton_divided_diff
Enter values of x = [0.5 ; 1.5 ; 3 ; 5 ; 6.5 ; 8]
Enter corresponding f(x) = [1.625 ; 5.875 ; 31 ; 131 ; 282.125 ; 521]
Enter point of approximation: 7

Divided_Difference_Table =

    0.5000    1.6250         0         0         0         0         0
    1.5000    5.8750    4.2500         0         0         0         0
    3.0000   31.0000   16.7500    5.0000         0         0         0
    5.0000  131.0000   50.0000    9.5000    1.0000         0         0
    6.5000  282.1250  100.7500   14.5000    1.0000         0         0
    8.0000  521.0000  159.2500   19.5000    1.0000         0         0

Coefficients of the interpolated polynomial are:

P =

     0     0     1     0     1     1

fx f(7.000000) = 351.000000>>
```

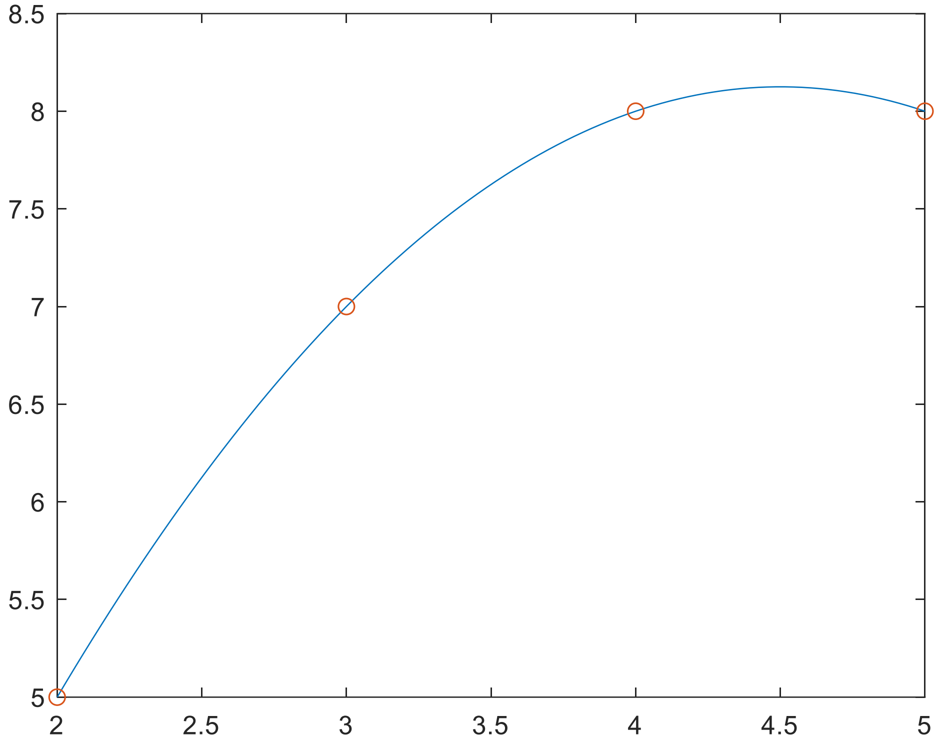


600
500
400
300
200
100
0
0
1
2
3
4
5
6
7
8

```
% PRACTICAL 7
% Lagrange method for interpolation
% RITIKA GUPTA MSCMAT54

clear all;
x=input('Enter values of x: ');
y=input('Enter corresponding f(x): ');
p0=input('Enter point of approximation: ');
n=length(x);
L=zeros(n);

for i=1:n
    v=1;
    for j=1:n
        if i~=j
            v=conv(v,poly(x(j)))/(x(i)-x(j));
        end
    end
    L(i,:)=v*y(i);
end
L  %row i represents L_i*y(i)
P=sum(L);
for i=1:n
    if i==n
        fprintf('(%.2f)',P(i));
    else
        fprintf('(%.2fx^%d)+',P(i),n-i);   % %.2f prints 2 decimal places
    end
end

%approximating polynomials at a point
fp=polyval(P,p0);
fprintf('\nf(%f) = %f',p0,fp);

%plotting the interpolated polynomial
X=linspace(x(1),x(n),100);
Y=polyval(P,X);
plot(X,Y,[x;p0],[y;fp],'o')
```

Command Window

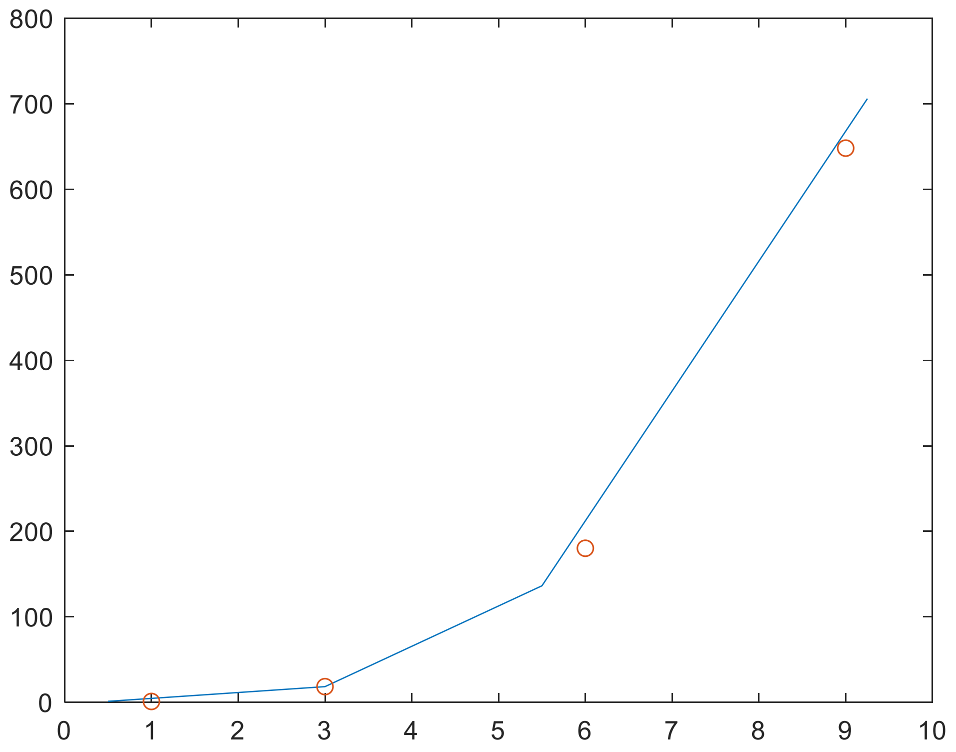
```
>> prac7_mscmat54_lagrange_interpolation
Enter values of x: [2;3;5]
Enter corresponding f(x): [5;7;8]
Enter point of approximation: 4

L =

    1.6667  -13.3333   25.0000
   -3.5000   24.5000  -35.0000
    1.3333   -6.6667    8.0000

(-0.50x^2)+(4.50x^1)+(-2.00)
f(4.000000) = 8.000000>>
```


8.5
8
7.5
7
6.5
6
5.5
5
2
2.5
3
3.5
4
4.5
5

```
% PRACTICAL 8
% Splines method for interpolation
% cubic spline method
% RITIKA GUPTA MSCMAT54

clear all;
x=input('Enter values of x: ');
y=input('Enter y=f(x)= ');
n=length(x);
query=input('Enter values to find interpolated polynomial for: ');
ycurve=spline(x,y,query);
ycurve
plot(query,ycurve,x,y,'o')
```

Command Window

```
>> prac8_mscmat54_spline_interpolation
Enter values of x: [1 3 6 9]
Enter y=f(x)= (x.^3) - (x.^2) + (x.^-1)
Enter values to find interpolated polynomial for: [0.5 3 5.5 9.25]

ycurve =

    1.1535   18.3333  136.2847  705.9708
```



```matlab
% PRACTICAL 9
% Simpson's rule for Numerical Integration
% using simpson's 1/3 rule
% RITIKA GUPTA MSCMAT54

clear all;
f=input('Enter f(x): ');
a=input('Enter lower limit: ');
b=input('Enter upper limit: ');
n=input('Enter number of sub intervals (even): ');
if rem(n,2)==0
    h=(b-a)/n;
    s=f(a)+f(b);
    for i=1:2:n-1
        s=s+ 4*f(a+i*h);
    end
    for i=2:2:n-2
        s=s+ 2*f(a+i*h);
    end
    Integral =(h/3)*s
else
    disp('n not an even number.');
end
```

Command Window

```
>> prac9_mscmat54_simpson
Enter f(x): @(x) 1/(1+x)
Enter lower limit: 0
Enter upper limit: 4
Enter number of sub intervals (even): 20

Integral =

    1.6095
```

```
% PRACTICAL 10
% Trapezoidal rule
% RITIKA GUPTA MSCMAT54

clear all;
f=input('Enter f(x): ');
a=input('Enter lower limit: ');
b=input('Enter upper limit: ');
n=input('Enter number of sub intervals: ');
h=(b-a)/n;
I=(f(a)+f(b))/2;
for i=1:n-1
    I=I+ f(a+i*h);
end
Integral = h*I
```

Command Window

```
>> prac10_mscmat54_trapezoidal
Enter f(x): @(x) x*sin(x)
Enter lower limit: 0
Enter upper limit: pi/2
Enter number of sub intervals: 55

Integral =

    1.0001
```