Bisection Method

```
% BISECTION'S METHOD
clc
clear all
a = 1;
b = 2;
tol = 0.001;
x = 0:10;
c=(a+b)/2;
f = @(x) x^3+4*x^2-10;
if(f(a)*f(b)>0)
disp('wrong a and b value')
else
    while abs(f(c)>tol)
        if f(a)*f(c)<0
            b=c;
        else
            a=c;
        end
        c=(a+b)/2;
    end
end
fprintf('Root = %f',c)
```

Fixed Point Iteration Method

```
% FIXED POINT ITERATION
clc
clear all
i=0;
N=10;
x0=1;
tol=0.001;

x=1:0.1:2;
g=@(x) sqrt(10/(4+x));

while i<=N
    x1=g(x0);
    if abs(x1-x0)<=tol
        fprintf('Solution of the given eqn. is %f',x1)
        break
    end
    x0=x1;
    i=i+1;
end

if i==N
    fprintf('Solution not found')
end
```

Newton Method

```
%Newton Method
clc
clear all
syms x;
f=@(x) x^2-17;
x0=1;
tol=10^(-5);
i=1;
n=10;
df=inline(diff(f(x)));
while(i<=n)
    x1=x0-(f(x0)/df(x0));
    if abs(x0-x1)<tol
        fprintf('The root is %f',x1)
        break;
    else
        x0=x1;
    end
i=i+1;
end
```

Secant Method

```
%Secant Method
clc
clear all
f=@(x) x^2-17;
x0=0;
x1=1;
tol=10^(-5);
i=1;
n=10;
while(i<=n)
    x2=x1-((x1-x0)/(f(x1)-f(x0))*f(x1));
    if abs((x2-x1)/x2)<tol
        fprintf('The root is %f',x2)
        break;
    else
        x0=x1;
        x1=x2;
    end
i=i+1;
end
```

LU Method

```
%LU
clc
clear all
A=[2,-1,1;3,3,9;3,3,5];
m=size(A,1);
n=size(A,2);
l=eye(m);
for j=1:n-1
    for i=j+1:m
        l(i,j)=A(i,j)/A(j,j);
        A(i,:)=A(i,:)-l(i,j)*A(j,:);
    end
end
l
A
```

Gauss Seidel

```
%Gauss Seidel
clc
clear all
A=[4.63,-1.21,3.22;-3.07,5.43,2.11;1.26,3.11,4.57];
b=[2.22,-3.17,5.11];
x0=[0,0,0];
x=[0,0,0];
tol=0.001;
n=3;
N=100;
k=1;
while k<=N
    for i=1:n
        sum1=0;
        sum2=0;
        for j=1:i-1
            sum1=sum1+ A(i,j)*x(j);
        end
        for j=i+1:n
        sum2=sum2+ A(i,j)*x0(j);
        end

        x(i)=(1/A(i,i))*(b(i)-sum1-sum2);

        if norm(x-x0)<tol
            break
        end
    end
k=k+1;
x0=x;
end
disp(x)
A*(x')
```

Gauss Elimination Method

```
%Gauss Elimination
A= [10 8 -3 1 16; 2 10 1 -4 9; 3 -4 10 1 10; 2 2 -3 10 11];
n= size(A,1);
l= eye(n);
for j=1:n+1
    for i=j+1:n
        l(i,j)=A(i,j)./A(j,j);
        for k=1:n+1
            A(i,k)=A(i,k)-l(i,j)*A(j,k);
        end
    end
end
A
l
x(n)=A(n,n+1)/A(n,n);
for i=n-1:-1:1
    sum= 0;
    for j=i+1:n
        sum= sum+A(i,j)*x(j);
    end
    x(i)=(A(i,n+1)-sum)/A(i,j);
end
x
```

SOR Method

```
%SOR Method
clc;
clear all;
w=1.2;
A=[4.63 -1.21 3.22;
    -3.07 5.48 2.11;
    1.26 3.11 4.57];
b=[2.22 -3.17 5.11];
x=[0;0;0];
tol= 10 ^(-3);
err=1;
n=3;
while(norm(err,inf)>tol)
    xold=x;
    for i=1:n
        s=0;
        for j=1:i-1
            s=s+(A(i,j)*x(j));
        end
        for j=i+1:n
            s=s+A(i,j)*xold(j);
        end
        x(i)=(1-w)*x(i)+w*((b(i)-s)/A(i,i));
        err=x-xold;
    end
end
x
```

Power Method

```
%Power Method
clc;
clear all;
A=[4 1 0; 1 20 1; 0 1 4];
x1=[1; 1; 1];
tol=10^(-3);
n=50;
x=x1;
K(1)=0;
for i=2:n
    y=A*x;
    K(i)=norm(y,inf);
    x=(1/K(i))*y;
    if abs(K(i)-K(i-1))<tol
        disp(K(i));
        break
    else
        x1=x
        i=i+1;
    end
end
```

Lagrange Interpolation

```
%Langrange Interpolation
clc;
clear all;
n=input("Enter number of data points: ");
for i=1:n
    x(i)=input("Enter value of x(i): ");
    y(i)=input("Enter value of y(i): ");
end
b=input("Enter point of interpolation: ");
sum=0;
for i=1:n
    L(i)=1;
    for j=1:n
        if(i~=j)
            L(i)=L(i)*((b-x(j))/(x(i)-x(j)));
        end
    end
    sum=sum+(L(i)*y(i));
end
sum
```

Newton Divided Difference

```
%Newton Divided Difference
clc;
clear all;
n=input('Enter number of data points: ');
for i=1:n
    x(i)=input("Enter value of x(i): ");
    f(i)=input("Enter value of f(i): ");
end
b=input("Enter point of interpolation b: ");
for i=1:n
    d(1,i)=f(i);
end
for i=2:n
    for j=1:n-i+1
        d(i,j)=(d(i-1,j+1)-d(i-1,j))/(x(i+j-1)-x(j));
    end
end
sum=d(1,1);
product=1;
for i=2:n
    product=product*(b-x(i-1));
    sum=sum+product*(d(i,1));
end
fprintf("value of f(%f) is: %f",b,sum);
```

Trapezoidal Rule

```
%Trapezoidal Rule
clc;
clear all;
f= @(x) 1/(1+(x^2));
h=input("Enter step size: ");
x0=input("Enter value of a: ");
xn=input("Enter value of b: ");
a(1)=x0;
b(1)=f(x0);
n=6;
b(n)=f(xn);
sum=0;
for i=1:n
    a(i+1)=a(i)+h;
    b(i+1)=f(a(i+1));
    if i~=n
        sum=sum+b(i+1);
    end
end
sum = h*(sum+((b(1)+b(n))/2))
```

Simpsons Method

```
%Simpsons
clc;
clear all;
f=@(x)(exp(-x^2))*cos(x);
a=-1;
b=1;
N=6;
h=abs((b-a))/N;
sum=0;

for(i=1:N-1)
    x=a+(i*h);
    if rem(i,2)==0
        sum=sum+2*f(x);
    else
        sum=sum+4*f(x);
    end
end
I=h/3*(f(a)+sum+f(b))
```

Euler Method

```
%Euler's
clc
clear all
f=@(t,y)-y+2*cos(t);
y(1)=1;
a=0;
b=1;
h=0.2;
t=a;
n=abs(b-a)/h;
for i=1:n
    t(i+1)=t(i)+h ;
    k1=h*f(t(i),y(i));
    k2=h*f(t(i)+h,y(i)+k1);
    y(i+1)=y(i)+(k1+k2)/2;
end
y
t
```

Range Kutta Method

```
%Kutta
clc
clear all
f=@(t,y)-y+2*cos(t);
y(1)=1;
a=0;
b=1;
h=0.2;
t=a;
n=abs(b-a)/h;
for i=1:n
    t(i+1)=t(i)+h ;
    k1=h*f(t(i),y(i));
    k2=h*f(t(i)+(h/2),y(i)+(k1/2));
    k3=h*f(t(i)+(h/2),y(i)+(k2/2));
    k4=h*f(t(i)+h,y(i)+k3);
    k=(k1+2*(k2+k3)+k4)/6;
    y(i+1)=y(i)+k;
end
y
k
```