SCIENTIFIC COMPUTING LAB MC204

ANEESH PANCHAL
2K20/MC/21

DEPARTMENT OF
APPLIED MATHEMATICS

—

Submitted to

—

Dr. CP Singh & Ms. Vinita Khatri

ACKNOWLEDGEMENT

I would sincerely like to thank Prof. Dr. C.P Singh who encouraged me in carrying out the experiments and has a guiding spirit behind completion of this course.

I am very thankful to him for putting tremendous efforts from his side to assist me as much as possible.

CONTENTS

PROGRAM-1 (20/01/2022)

Write a program to implement Method of Successive Substitution.

Code I:

```
syms x

%Initial Equation
fx = x^3-2*x-8;
fprintf("\n Initial equation f(x) is: ");
disp(fx);

%Different equations x=phi(x)
eq(1)= x^3-x-8;
eq(2)= (x^3-8)/2;
eq(3)= (2*x+8)^(1/3);

%Initial Approxiamations
x0=2.5;
fprintf("\n Initial Value x0: %f\n\n", x0);

%Finding perfect fit equation for the required problem
for i=1:length(eq)
   d=diff(eq(i),x);
   d=subs(d,x,x0);
   if d<1 && d>-1
      break;
   end
end

%Printing satisfied equation
fprintf("\n Satisfying phi(x) is: ");
disp(eq(i));

%Finding Solution and saving them in an array
xn(1)=x0;
a=1;
while a>0
   xn(a+1)=subs(eq(i),x,xn(a));
   if (xn(a)-xn(a+1)<0.001)
      break;
   end
   a=a+1;
end
```

```
%Printing solution of each step
for i=1:a+1
   fprintf("\n Iteration %d value of x is: %f\n",i,xn(i));
end

%Printing Final Solution
fprintf("\n Root is: %f\n\n", xn(a+1));
```

Output:

```
>> SuccSubs

 Initial equation f(x) is: x^3 - 2*x - 8


 Initial Value x0: 2.500000


 Satisfying phi(x) is: (2*x + 8)^(1/3)


 Iteration 1 value of x is: 2.500000

 Iteration 2 value of x is: 2.351335

 Iteration 3 value of x is: 2.333270

 Iteration 4 value of x is: 2.331056

 Iteration 5 value of x is: 2.330784

 Root is: 2.330784

>>
```

Code II:

```
syms x

%Initial Equation
fx = sin(x)+x^2-1;
fprintf("\n Initial equation f(x) is: ");
disp(fx);
```

```
%Different equations x=phi(x)
eq(1)= (1-sin(x))^(1/2);
eq(2)= asin(1-x^2);

%Initial Approxiamation
x0=0.5;
fprintf("\n Initial Value x0: %f\n\n", x0);

%Finding perfect fit equation for the required problem
for i=1:length(eq)
   d=diff(eq(i),x);
   d=subs(d,x,x0);
   if d<1 && d>-1
      break;
   end
end

%Printing satisfied equation
fprintf("\n Satisfying phi(x) is: ");
disp(eq(i));

%Finding Solution and saving them in an array
xn(1)=x0;
a=1;
while a>0
   xn(a+1)=subs(eq(i),x,xn(a));
   if (xn(a)-xn(a+1)<0.001 && xn(a+1)-xn(a)<0.001)
      break;
   end
   a=a+1;
end

%Printing solution of each step
for i=1:a+1
   fprintf("\n Iteration %d value of x is: %f\n",i,xn(i));
end

%Printing Final Solution
fprintf("\n Root is: %f\n\n", xn(a+1));
```

Output:

```
>> SuccSubs2

 Initial equation f(x) is: sin(x) + x^2 - 1
```

```
Initial Value x0: 0.500000

Satisfying phi(x) is: (1 - sin(x))^(1/2)

Iteration 1 value of x is: 0.500000

Iteration 2 value of x is: 0.721508

Iteration 3 value of x is: 0.582651

Iteration 4 value of x is: 0.670642

Iteration 5 value of x is: 0.615232

Iteration 6 value of x is: 0.650270

Iteration 7 value of x is: 0.628171

Iteration 8 value of x is: 0.642133

Iteration 9 value of x is: 0.633321

Iteration 10 value of x is: 0.638886

Iteration 11 value of x is: 0.635373

Iteration 12 value of x is: 0.637591

Iteration 13 value of x is: 0.636191

Iteration 14 value of x is: 0.637075

Root is: 0.637075

>>
```

PROGRAM-2 (27/01/2022)

Write a program to implement following methods:

1. Bisection Method
2. Regula Falsi Method
3. Secant Method
4. Newton Raphson Method

Code:

```
syms x f(x)

f(x) = input('\nInput equation: ');
cs = input('\n1. Bisection Method\n2. Regula Falsi Method\n3. Secant Method\n4. Newton Raphson Method\nEnter a num: ');
a = input('\nInput value of a: ');
b = input('Input value of b: ');
fprintf('\n');
root = 1/0;
n=2;
x(n)=b;
x(n-1)=a;

switch cs
    case 1
        %Bisection Method
        while true
            x0 = (a+b)/2;
            if (root-x0<0.001 && x0-root<0.001)
                fprintf('x0 = %f\tf(x0) = %f\n',x0,f(x0));
                root = x0;
                break;
            end
            root = x0;
            if f(x0)*f(a)<0
                fprintf('x0 = %f\tf(x0) = %f\n',x0,f(x0));
                b=x0;
            elseif f(x0)*f(b)<0
                fprintf('x0 = %f\tf(x0) = %f\n',x0,f(x0));
                a=x0;
            else
                if f(a)==0
                    root = a;
                elseif f(b)==0
                    root = b;
                else
                    root = x0;
```

```
        end
      end
    end

  case 2
    %Regula Falsi Method
    while true
      x0 = (a*f(b)-b*f(a))/(f(b)-f(a));
      if (root-x0<0.001 && x0-root<0.001)
        fprintf('x0 = %f\tf(x0) = %f\n',x0,f(x0));
        root = x0;
        break;
      end
      root = x0;
      if f(x0)*f(a)<0
        fprintf('x0 = %f\tf(x0) = %f\n',x0,f(x0));
        b=x0;
      elseif f(x0)*f(b)<0
        fprintf('x0 = %f\tf(x0) = %f\n',x0,f(x0));
        a=x0;
      else
        if f(a)==0
          root = a;
        elseif f(b)==0
          root = b;
        else
          root = x0;
        end
      end
    end

  case 3
    %Secant Method
    while true
      x0 = (x(n-1)*f(x(n))-x(n)*f(x(n-1)))/(f(x(n))-f(x(n-1)));
      if (root-x0<0.001 && x0-root<0.001)
        fprintf('x0 = %f\tf(x0) = %f\n',x0,f(x0));
        root = x0;
        break;
      end
      fprintf('x0 = %f\tf(x0) = %f\n',x0,f(x0));
      root = x0;
      x(n-1)=x(n);
      x(n)=x0;
    end

  case 4
    %Newton Raphson Method
```

```
    while true
      fd = diff(f);
      x(n) = x(n-1) - (f(x(n-1))/fd(x(n-1)));
      if (root-x(n)<0.001 && x(n)-root<0.001)
        fprintf('xn = %f\tf(xn) = %f\n',x(n),f(x(n)));
        root = x(n);
        break;
      end
      fprintf('xn = %f\tf(xn) = %f\n',x(n),f(x(n)));
      root = x(n);
      x(n-1)=x(n);
    end

end
fprintf('\nSolution is: %f\n\n',root);
```

Output I:

```
>> Bisection_Regula_Secant_NewtonR

Input equation: x^2 - 6*x*exp(-x)

1. Bisection Method
2. Regula Falsi Method
3. Secant Method
4. Newton Raphson Method
Enter a num: 1

Input value of a: 1.4
Input value of b: 1.5

x0 = 1.450000      f(x0) = 0.061738
x0 = 1.425000      f(x0) = -0.025722
x0 = 1.437500      f(x0) = 0.017789
x0 = 1.431250      f(x0) = -0.004022
x0 = 1.434375      f(x0) = 0.006870
x0 = 1.432812      f(x0) = 0.001421
x0 = 1.432031      f(x0) = -0.001301

Solution is: 1.432031

>> Bisection_Regula_Secant_NewtonR

Input equation: x^2 - 6*x*exp(-x)

1. Bisection Method
2. Regula Falsi Method
3. Secant Method
```

```
4. Newton Raphson Method
Enter a num: 2

Input value of a: 1.4
Input value of b: 1.5

x0 = 1.431540        f(x0) = -0.003010
x0 = 1.432382        f(x0) = -0.000079

Solution is: 1.432382

>> Bisection_Regula_Secant_NewtonR

Input equation: x^2 - 6*x*exp(-x)

1. Bisection Method
2. Regula Falsi Method
3. Secant Method
4. Newton Raphson Method
Enter a num: 3

Input value of a: 1.4
Input value of b: 1.5

x0 = 1.431540        f(x0) = -0.003010
x0 = 1.432382        f(x0) = -0.000079

Solution is: 1.432382

>> Bisection_Regula_Secant_NewtonR

Input equation: x^2 - 6*x*exp(-x)

1. Bisection Method
2. Regula Falsi Method
3. Secant Method
4. Newton Raphson Method
Enter a num: 4

Input value of a: 1.4
Input value of b: 1.5

xn = 1.432848        f(xn) = 0.001544
xn = 1.432405        f(xn) = 0.000000

Solution is: 1.432405

>>
```

Output II:

```
>> Bisection_Regula_Secant_NewtonR

Input equation: x^3 - x -1

1. Bisection Method
2. Regula Falsi Method
3. Secant Method
4. Newton Raphson Method
Enter a num: 1

Input value of a: 1
Input value of b: 2

x0 = 1.500000      f(x0) = 0.875000
x0 = 1.250000      f(x0) = -0.296875
x0 = 1.375000      f(x0) = 0.224609
x0 = 1.312500      f(x0) = -0.051514
x0 = 1.343750      f(x0) = 0.082611
x0 = 1.328125      f(x0) = 0.014576
x0 = 1.320313      f(x0) = -0.018711
x0 = 1.324219      f(x0) = -0.002128
x0 = 1.326172      f(x0) = 0.006209
x0 = 1.325195      f(x0) = 0.002037

Solution is: 1.325195

>> Bisection_Regula_Secant_NewtonR

Input equation: x^3 - x -1

1. Bisection Method
2. Regula Falsi Method
3. Secant Method
4. Newton Raphson Method
Enter a num: 2

Input value of a: 1
Input value of b: 2

x0 = 1.166667      f(x0) = -0.578704
x0 = 1.253112      f(x0) = -0.285363
x0 = 1.293437      f(x0) = -0.129542
x0 = 1.311281      f(x0) = -0.056588
x0 = 1.318989      f(x0) = -0.024304
x0 = 1.322283      f(x0) = -0.010362
```

x0 = 1.323684 f(x0) = -0.004404
x0 = 1.324279 f(x0) = -0.001869

Solution is: 1.324279

>> Bisection_Regula_Secant_NewtonR

Input equation: x^3 - x -1

1. Bisection Method
2. Regula Falsi Method
3. Secant Method
4. Newton Raphson Method
Enter a num: 3

Input value of a: 1
Input value of b: 2

x0 = 1.166667 f(x0) = -0.578704
x0 = 1.253112 f(x0) = -0.285363
x0 = 1.337206 f(x0) = 0.053881
x0 = 1.323850 f(x0) = -0.003698
x0 = 1.324708 f(x0) = -0.000043

Solution is: 1.324708

>> Bisection_Regula_Secant_NewtonR

Input equation: x^3 - x -1

1. Bisection Method
2. Regula Falsi Method
3. Secant Method
4. Newton Raphson Method
Enter a num: 4

Input value of a: 1
Input value of b: 2

xn = 1.500000 f(xn) = 0.875000
xn = 1.347826 f(xn) = 0.100682
xn = 1.325200 f(xn) = 0.002058
xn = 1.324718 f(xn) = 0.000001

Solution is: 1.324718

PROGRAM-3 (03/02/2022)

Write a program to implement Gauss Elimination Method with and without Partial Pivoting.

Code I:

```
A = [1 2 -1 1;-1 1 2 -1;2 -1 2 2;1 1 -1 2];
b = [6;3;14;8];

action = input('Whether you want to perform GEM:\n1. without Partial Pivoting\n2. with Partial
Pivoting\nEnter a num: ');

Aug = A;
N = max(size(Aug));
Aug(:,N+1) = b

%Without Partial Pivoting
if action == 1
   for j=2:N
      for i=j:N
         m = Aug(i,j-1)/Aug(j-1,j-1);
         Aug(i,:) = Aug(i,:) - Aug(j-1,:)*m;
      end
   end

   %Display Augmented Matrix after ELementary Row Transf
   Aug

   x = zeros(N,1);
   x(N) = Aug(N,N+1)/Aug(N,N);
   %Backward subs
   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

%With Partial Pivoting
else
   %Performing Partial Pivoting
   for i =1:N-1
      maxel = max(Aug(:,i));
      for t =1:N
         if Aug(t,i)==maxel
            extra = Aug(t,:);
            Aug(t,:)=Aug(i,:);
            Aug(i,:)=extra;
            break;
```

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

    %Display Augmented Matrix after ELementary Row Transf
    Aug

    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
  end

%Display Value of x after GEM
x
```

Output:

```
>> >> GEM
Whether you want to perform GEM:
1. without Partial Pivoting
2. with Partial Pivoting
Enter a num: 1

Aug =

     1     2    -1     1     6
    -1     1     2    -1     3
     2    -1     2     2    14
     1     1    -1     2     8


Aug =

    1.0000    2.0000   -1.0000    1.0000    6.0000
         0    3.0000    1.0000         0    9.0000
         0         0    5.6667         0   17.0000
         0         0         0    1.0000    4.0000
```

```
x =

     1
     2
     3
     4

>> GEM
Whether you want to perform GEM:
1. without Partial Pivoting
2. with Partial Pivoting
Enter a num: 2

Aug =

     1     2    -1     1     6
    -1     1     2    -1     3
     2    -1     2     2    14
     1     1    -1     2     8


Aug =

     2    -1     2     2    14
    -1     1     2    -1     3
     1     2    -1     1     6
     1     1    -1     2     8


Aug =

    2.0000   -1.0000    2.0000    2.0000   14.0000
         0    2.5000   -2.0000         0   -1.0000
         0    0.5000    3.0000         0   10.0000
         0    1.5000   -2.0000    1.0000    1.0000


Aug =

    2.0000   -1.0000    2.0000    2.0000   14.0000
         0    2.5000   -2.0000         0   -1.0000
         0         0    3.4000         0   10.2000
         0         0   -0.8000    1.0000    1.6000


Aug =
```

```
    2.0000   -1.0000    2.0000    2.0000   14.0000
         0    2.5000   -2.0000         0   -1.0000
         0         0    3.4000         0   10.2000
         0         0         0    1.0000    4.0000


x =

     1
     2
     3
     4

>>
```

Code II:

```
A = [2 1 -1 2;4 5 -3 6;-2 5 -2 6;4 11 -4 8];
b = [5;9;4;2];

action = input('Whether you want to perform GEM:\n1. without Partial Pivoting\n2. with Partial Pivoting\nEnter a num: ');

Aug = A;
N = max(size(Aug));
Aug(:,N+1) = b

%Without Partial Pivoting
if action == 1
   for j=2:N
      for i=j:N
         m = Aug(i,j-1)/Aug(j-1,j-1);
         Aug(i,:) = Aug(i,:) - Aug(j-1,:)*m;
      end
   end

   %Display Augmented Matrix after ELementary Row Transf
   Aug

   x = zeros(N,1);
   x(N) = Aug(N,N+1)/Aug(N,N);
   %Backward subs
   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

%With Partial Pivoting
else
    %Performing Partial Pivoting
    for i =1:N-1
        maxel = max(Aug(:,i));
        for t =1:N
            if Aug(t,i)==maxel
                extra = Aug(t,:);
                Aug(t,:)=Aug(i,:);
                Aug(i,:)=extra;
                break;
            end
        end
        Aug
        for j=i+1:N
            m = Aug(j,i)/Aug(i,i);
            Aug(j,:) = Aug(j,:) - Aug(i,:)*m;
        end
    end

    %Display Augmented Matrix after ELementary Row Transf
    Aug

    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
end

%Display Value of x after GEM
x
```

Output:

```
>> GEM2
Whether you want to perform GEM:
1. without Partial Pivoting
2. with Partial Pivoting
Enter a num: 1

Aug =
```

```
    2    1   -1    2    5
    4    5   -3    6    9
   -2    5   -2    6    4
    4   11   -4    8    2


Aug =

    2    1   -1    2    5
    0    3   -1    2   -1
    0    0   -1    4   11
    0    0    0    2    6


x =

    1
   -2
    1
    3


>> GEM2
Whether you want to perform GEM:
1. without Partial Pivoting
2. with Partial Pivoting
Enter a num: 2

Aug =

    2    1   -1    2    5
    4    5   -3    6    9
   -2    5   -2    6    4
    4   11   -4    8    2


Aug =

    4    5   -3    6    9
    2    1   -1    2    5
   -2    5   -2    6    4
    4   11   -4    8    2


Aug =
```

```
    4.0000    5.0000   -3.0000    6.0000    9.0000
         0    7.5000   -3.5000    9.0000    8.5000
         0   -1.5000    0.5000   -1.0000    0.5000
         0    6.0000   -1.0000    2.0000   -7.0000


Aug =

    4.0000    5.0000   -3.0000    6.0000    9.0000
         0    7.5000   -3.5000    9.0000    8.5000
         0         0    1.8000   -5.2000  -13.8000
         0         0   -0.2000    0.8000    2.2000


Aug =

    4.0000    5.0000   -3.0000    6.0000    9.0000
         0    7.5000   -3.5000    9.0000    8.5000
         0         0    1.8000   -5.2000  -13.8000
         0         0         0    0.2222    0.6667


x =

    1.0000
   -2.0000
    1.0000
    3.0000

>>
```

PROGRAM-4 (17/02/2022)

Write a program to implement Gauss Seidel and Jacobi Iterative Methods.

Code I:

```
syms x1 x2 x3 x4

opr = input('1. Gauss Seidel Method\n2. Jacobi Method\nChoice: ');
eqn1 = 10*x1-2*x2-x3-x4==3;
eqn2 = -2*x1+10*x2-x3-x4==15;
eqn3 = -x1-x2+10*x3-2*x4==27;
eqn4 = -x1-x2-2*x3+10*x4==-9;

n1 = solve(eqn1,x1)
n2 = solve(eqn2,x2)
n3 = solve(eqn3,x3)
n4 = solve(eqn4,x4)

Xold = [0;0;0;0];
X = [0;0;0;0];
err(1:4,1) = 0.001;

switch opr
   case 1
      %Gauss Seidal method
      steps=0;
      while true
         X(1) = subs(subs(subs(n1,x2,X(2)),x3,X(3)),x4,X(4));
         X(2) = subs(subs(subs(n2,x1,X(1)),x3,X(3)),x4,X(4));
         X(3) = subs(subs(subs(n3,x1,X(1)),x2,X(2)),x4,X(4));
         X(4) = subs(subs(subs(n4,x1,X(1)),x3,X(3)),x2,X(2));
         steps = steps+1;
         if X-Xold<err
            if Xold-X<err
               break
            end
         end
         Xold=X;
      end

   case 2
      %Jacobi method
      steps=0;
      while true
         X(1) = subs(subs(subs(n1,x2,Xold(2)),x3,Xold(3)),x4,Xold(4));
         X(2) = subs(subs(subs(n2,x1,Xold(1)),x3,Xold(3)),x4,Xold(4));
         X(3) = subs(subs(subs(n3,x1,Xold(1)),x2,Xold(2)),x4,Xold(4));
         X(4) = subs(subs(subs(n4,x1,Xold(1)),x3,Xold(3)),x2,Xold(2));
         steps = steps+1;
         if X-Xold<err
```

```
        if Xold-X<err
            break
        end
    end
    Xold=X;
  end
end
X
steps
```

Output:

```
>> >> GSJ
1. Gauss Seidel Method
2. Jacobi Method
Choice: 1

n1 =

x2/5 + x3/10 + x4/10 + 3/10


n2 =

x1/5 + x3/10 + x4/10 + 3/2


n3 =

x1/10 + x2/10 + x4/5 + 27/10


n4 =

x1/10 + x2/10 + x3/5 - 9/10


X =

    0.9999
    1.9999
    3.0000
   -0.0000


steps =

     6

>> GSJ
1. Gauss Seidel Method
2. Jacobi Method
```

```
Choice: 2

n1 =

x2/5 + x3/10 + x4/10 + 3/10


n2 =

x1/5 + x3/10 + x4/10 + 3/2


n3 =

x1/10 + x2/10 + x4/5 + 27/10


n4 =

x1/10 + x2/10 + x3/5 - 9/10


X =

    0.9996
    1.9996
    2.9996
   -0.0004


steps =

     9

>>
```

Code II:

```
syms x1 x2 x3 x4

opr = input('1. Gauss Seidel Method\n2. Jacobi Method\nChoice: ');
eqn1 = 13*x1 + 5*x2  - 3*x3 + x4 == 18;
eqn2 = 2*x1 + 12*x2 + x3  - 4*x4 == 13;
eqn3 = x1 - 4*x2 + 10*x3 + x4 == 29;
eqn4 = 2*x1 + x2 - 3*x3 + 9*x4 == 31;

n1 = solve(eqn1,x1)
n2 = solve(eqn2,x2)
n3 = solve(eqn3,x3)
n4 = solve(eqn4,x4)
```

```
Xold = [0;0;0;0];
X = [0;0;0;0];
err(1:4,1) = 0.001;

switch opr
   case 1
      %Gauss Seidal method
      steps=0;
      while true
         X(1) = subs(subs(subs(n1,x2,X(2)),x3,X(3)),x4,X(4));
         X(2) = subs(subs(subs(n2,x1,X(1)),x3,X(3)),x4,X(4));
         X(3) = subs(subs(subs(n3,x1,X(1)),x2,X(2)),x4,X(4));
         X(4) = subs(subs(subs(n4,x1,X(1)),x3,X(3)),x2,X(2));
         steps = steps+1;
         if X-Xold<err
            if Xold-X<err
               break
            end
         end
         Xold=X;
      end

   case 2
      %Jacobi method
      steps=0;
      while true
         X(1) = subs(subs(subs(n1,x2,Xold(2)),x3,Xold(3)),x4,Xold(4));
         X(2) = subs(subs(subs(n2,x1,Xold(1)),x3,Xold(3)),x4,Xold(4));
         X(3) = subs(subs(subs(n3,x1,Xold(1)),x2,Xold(2)),x4,Xold(4));
         X(4) = subs(subs(subs(n4,x1,Xold(1)),x3,Xold(3)),x2,Xold(2));
         steps = steps+1;
         if X-Xold<err
            if Xold-X<err
               break
            end
         end
         Xold=X;
      end
end
X
steps
```

Output:

```
>> GSJ2
1. Gauss Seidel Method
2. Jacobi Method
Choice: 1

n1 =
```

```
(3*x3)/13 - (5*x2)/13 - x4/13 + 18/13


n2 =

x4/3 - x3/12 - x1/6 + 13/12


n3 =

(2*x2)/5 - x1/10 - x4/10 + 29/10


n4 =

x3/3 - x2/9 - (2*x1)/9 + 31/9


X =

    1.0414
    1.9954
    3.1886
    4.0542

steps =

     6

>> GSJ2
1. Gauss Seidel Method
2. Jacobi Method
Choice: 2

n1 =

(3*x3)/13 - (5*x2)/13 - x4/13 + 18/13


n2 =

x4/3 - x3/12 - x1/6 + 13/12


n3 =

(2*x2)/5 - x1/10 - x4/10 + 29/10


n4 =

x3/3 - x2/9 - (2*x1)/9 + 31/9
```

```
X =

    1.0412
    1.9955
    3.1882
    4.0539


steps =

     9

>>
```

PROGRAM-5 (26/02/2022)

Write a program to implement Power Method for finding maximum eigen value and corresponding eigen vector.

Code I:

```
A = [2,-1,0;-1,2,-1;0,-1,2];
X = [1;0;0];
EigValold = 0;
err = 0.001;

while true
  X = A*X;
  EigValnew = max(abs(X));
  if EigValnew-EigValold<err
     if EigValold-EigValnew<err
        break
     end
  end
  EigValold = EigValnew;
  X = X/EigValnew;
end

MaxEigValue = EigValnew
MaxEigVect = X/EigValnew
```

Output:

```
>> PowerMethod

MaxEigValue =

   3.4143


MaxEigVect =

   0.7406
  -1.0000
   0.6736

>>
```

Code II:

```
A = [15,-4,-3;-10,12,-6;-20,4,-2];
X = [1;1;1];
EigValold = 0;
err = 0.001;
```

```
while true
  X = A*X;
  EigValnew = max(abs(X));
  if EigValnew-EigValold<err
      if EigValold-EigValnew<err
          break
      end
  end
  EigValold = EigValnew;
  X = X/EigValnew;
end

MaxEigValue = EigValnew
MaxEigVect = X/EigValnew
```

Output:

```
>> PowerMethod2

MaxEigValue =

  19.9994


MaxEigVect =

   1.0000
  -0.4999
  -1.0000

>>
```

PROGRAM-6 (03/03/2022)

Write a program to create Forward and Backward Difference Table from given data.

Code I:

```
x = [1;2;3;4];
y = [1.1;2;4.4;7.9];
FDT = zeros(length(x),length(x)+1);
FDT(:,1) = x;
FDT(:,2) = y;

BDT = zeros(length(x),length(x)+1);
BDT(:,1) = x;
BDT(:,2) = y;

i = length(x)-1;
j = 3;
while j<=length(x)+1
   for i = 1:length(x)+2-j
      FDT(i,j) = FDT(i+1,j-1)-FDT(i,j-1);
      BDT(length(x)+1-i,j) = BDT(length(x)+1-i,j-1)-BDT(length(x)-i,j-1);
   end
   j=j+1;
end

fprintf('\nForward Difference Table:\n');
fprintf('x\t   f(x)\t\tDI\t\tDII\t\tDIII\n');
for i=1:length(x)
   fprintf('%.2f\t%.2f\t%.2f\t%.2f\t%.2f\n',FDT(i,1),FDT(i,2),FDT(i,3),FDT(i,4),FDT(i,5));
end

fprintf('\n\nBackward Difference Table:\n');
fprintf('x\t   f(x)\t\tDI\t\tDII\t\tDIII\n');
for i=1:length(x)
   fprintf('%.2f\t%.2f\t%.2f\t%.2f\t%.2f\n',BDT(i,1),BDT(i,2),BDT(i,3),BDT(i,4),BDT(i,5));
end
fprintf('\n\n')
```

Output:

```
>> DiffTable

Forward Difference Table:
x       f(x)    DI      DII     DIII
1.00    1.10    0.90    1.50    -0.40
2.00    2.00    2.40    1.10    0.00
3.00    4.40    3.50    0.00    0.00
4.00    7.90    0.00    0.00    0.00
```

Backward Difference Table:

x	f(x)	DI	DII	DIII
1.00	1.10	0.00	0.00	0.00
2.00	2.00	0.90	0.00	0.00
3.00	4.40	2.40	1.50	0.00
4.00	7.90	3.50	1.10	-0.40

>>

Code II:

```
x = [0;1;2;3;4];
y = [1;1.5;2.2;3.1;4.6];
FDT = zeros(length(x),length(x)+1);
FDT(:,1) = x;
FDT(:,2) = y;

BDT = zeros(length(x),length(x)+1);
BDT(:,1) = x;
BDT(:,2) = y;

i = length(x)-1;
j = 3;
while j<=length(x)+1
   for i = 1:length(x)+2-j
      FDT(i,j) = FDT(i+1,j-1)-FDT(i,j-1);
      BDT(length(x)+1-i,j) = BDT(length(x)+1-i,j-1)-BDT(length(x)-i,j-1);
   end
   j=j+1;
end

fprintf('\nForward Difference Table:\n');
fprintf('x\t   f(x)\t\tDI\t\tDII\t\tDIII\tDIV\n');
for i=1:length(x)
   fprintf('%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\n',FDT(i,1),FDT(i,2),FDT(i,3),FDT(i,4),FDT(i,5),FDT(i,6));
end

fprintf('\n\nBackward Difference Table:\n');
fprintf('x\t   f(x)\t\tDI\t\tDII\t\tDIII\tDIV\n');
for i=1:length(x)
   fprintf('%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\n',BDT(i,1),BDT(i,2),BDT(i,3),BDT(i,4),BDT(i,5),BDT(i,6));
end
fprintf('\n\n')
```

Output:

```
>> DiffTable2

Forward Difference Table:
x       f(x)    DI      DII     DIII    DIV
0.00    1.00    0.50    0.20    -0.00   0.40
1.00    1.50    0.70    0.20    0.40    0.00
2.00    2.20    0.90    0.60    0.00    0.00
3.00    3.10    1.50    0.00    0.00    0.00
4.00    4.60    0.00    0.00    0.00    0.00


Backward Difference Table:
x       f(x)    DI      DII     DIII    DIV
0.00    1.00    0.00    0.00    0.00    0.00
1.00    1.50    0.50    0.00    0.00    0.00
2.00    2.20    0.70    0.20    0.00    0.00
3.00    3.10    0.90    0.20    -0.00   0.00
4.00    4.60    1.50    0.60    0.40    0.40


>>
```

PROGRAM-7 (10/03/2022)

Write a program to implement Lagrange's Method of Interpolation.

Code I:

```
X = [0;1;3;6];
Y = [18;10;-18;90];
xfind = 2;
yfinal = 0;
for i=1:4
   yint =1;
   for j=1:4
      if j==i
         continue
      end
      yint = yint*(xfind-X(j))/(X(i)-X(j));
   end
   yfinal = yfinal + yint*Y(i);
end
fprintf('\nValue using Lagrange Interpolation is %.2f\n\n',yfinal);

p = [2 -10 0 18];
fprintf('Value using polynomial is %.2f\n\n',polyval(p,xfind));

fprintf('Error is %.2f\n\n',polyval(p,xfind)-yfinal);
```

Output:

```
>> Lagrange

Value using Lagrange Interpolation is -6.00

Value using polynomial is -6.00

Error is 0.00

>>
```

Code II:

```
X = [0;1;2;5];
Y = [2;3;12;147];
xfind = 3;
yfinal = 0;
for i=1:4
   yint =1;
   for j=1:4
      if j==i
         continue
      end
```

```
        yint = yint*(xfind-X(j))/(X(i)-X(j));
    end
    yfinal = yfinal + yint*Y(i);
end
fprintf('\nValue using Lagrange Interpolation is %.2f\n\n',yfinal);

p = [1 1 -1 2];
fprintf('Value using polynomial is %.2f\n\n',polyval(p,xfind));

fprintf('Error is %.2f\n\n',polyval(p,xfind)-yfinal);
```

Output:

```
>> Lagrange2

Value using Lagrange Interpolation is 35.00

Value using polynomial is 35.00

Error is 0.00

>>
```

PROGRAM-8 (24/03/2022)

Write a program to implement Trapezoid and Simpson's 1/3rd Rule for Integration.

Code I:

```
syms x f(x)
opt = input('1. Trapezoidal Rule\n2. Simpsons 1/3 Rule\nChoice:');
f(x) = 1/(1+x^2);
a = 0;
b = 6;
n = 6;
h = (b-a)/n;

i = 0;
sum = 0;
switch opt
   case 1
      while i<=n
         if i==0 || i==n
            sum = sum + f(a+i*h);
         else
            sum = sum + 2*f(a+i*h);
         end
         i = i+1;
      end
      int = h*sum/2;

   case 2
      while i<=n
         if i==0 || i==n
            sum = sum + f(a+i*h);
         elseif rem(i,2)==1
            sum = sum + 4*f(a+i*h);
         else
            sum = sum + 2*f(a+i*h);
         end
         i = i+1;
      end
      int = h*sum/3;
end

fprintf('\nValue of Integral is %f\n\n',int);
```

Output:

```
>> Trap_Simp
1. Trapezoidal Rule
2. Simpsons 1/3 Rule
Choice:1
```

```
Value of Integral is 1.410799

>> Trap_Simp
1. Trapezoidal Rule
2. Simpsons 1/3 Rule
Choice:2

Value of Integral is 1.366173

>>
```

Code II:

```
syms x f(x)
opt = input('1. Trapezoidal Rule\n2. Simpsons 1/3 Rule\nChoice:');
f(x) = exp(x^2);
a = 0;
b = 2;
n = 10;
h = (b-a)/n;

i = 0;
sum = 0;
switch opt
   case 1
      while i<=n
         if i==0 || i==n
            sum = sum + f(a+i*h);
         else
            sum = sum + 2*f(a+i*h);
         end
         i = i+1;
      end
      int = h*sum/2;

   case 2
      while i<=n
         if i==0 || i==n
            sum = sum + f(a+i*h);
         elseif rem(i,2)==1
            sum = sum + 4*f(a+i*h);
         else
            sum = sum + 2*f(a+i*h);
         end
         i = i+1;
      end
      int = h*sum/3;
end

fprintf('\nValue of Integral is %f\n\n',int);
```

Output:

```
>> Trap_Simp_2
1. Trapezoidal Rule
2. Simpsons 1/3 Rule
Choice:1

Value of Integral is 17.170210

>> Trap_Simp_2
1. Trapezoidal Rule
2. Simpsons 1/3 Rule
Choice:2

Value of Integral is 16.490203

>>
```

PROGRAM-9 (31/03/2022)

Write a program to implement Runge Kutta Method for solving ODE.

Code I:

```
syms f(x,y)
f(x,y)=x+y;

% x0=0 y0=1 h=0.2
% y(0.2)
x0=0;
y0=1;
h=0.2;
k1=vpa(h*f(x0,y0))
k2=vpa(h*f(x0+h/2,y0+k1/2))
k3=vpa(h*f(x0+h/2,y0+k2/2))
k4=vpa(h*f(x0+h,y0+k3))
k=(1/6)*(k1+2*k2+2*k3+k4)
y1=y0+k;
fprintf("\nApproximate value of y(0.2) is %f\n\n",y1);
```

Output:

```
>> RungeKutta1

k1 =

0.2


k2 =

0.24


k3 =

0.244


k4 =

0.2888


k =

0.2428

Approximate value of y(0.2) is 1.242800
```

Code II:

```
syms f(x,y)
f(x,y)=(y^2-x^2)/(y^2+x^2);

% x1=0 y1=1 h=0.2
% y(0.2)
fprintf("\n\nFor y(0.2)")
x0=0;
y0=1;
h=0.2;
k1=vpa(h*f(x0,y0))
k2=vpa(h*f(x0+h/2,y0+k1/2))
k3=vpa(h*f(x0+h/2,y0+k2/2))
k4=vpa(h*f(x0+h,y0+k3))
k=(1/6)*(k1+2*k2+2*k3+k4)
y1=y0+k;
fprintf("Approximate value of y(0.2) is %f",y1);

% x1=0.2 y1=1.196 h=0.2
% y(0.4)
fprintf("\n\nFor y(0.4)")
x1=0.2;
k1=vpa(h*f(x1,y1))
k2=vpa(h*f(x1+h/2,y1+k1/2))
k3=vpa(h*f(x1+h/2,y1+k2/2))
k4=vpa(h*f(x1+h,y1+k3))
k=(1/6)*(k1+2*k2+2*k3+k4)
y2=y1+k;
fprintf("\n\nApproximate value of y(0.4) is %f\n\n",y2);
```

Output:

```
>> RungeKutta2


For y(0.2)
k1 =

0.2


k2 =

0.19672131147540983606557377049 18


k3 =

0.19671159756175696664538322349162
```

k4 =

0.18913131083246846753432215720218

k =

0.19599952148446701215937269086 15

Approximate value of y(0.2) is 1.196000

For y(0.4)
k1 =

0.18911871711487530339652960215282

k2 =

0.17949351514801297540411576659793

k3 =

0.17934765547323766449106905310299

k4 =

0.16880452896997015036086331591101

k =

0.17926759788789112225796042624428

Approximate value of y(0.4) is 1.375267

>>

PROGRAM-10 (21/04/2022)

Write a program to implement Picard's Method for solving ODE.

Code I:

```
syms x y ysol

diff = x + y*y;
ysol(1) = 0;
x0 = 0;
xfind = 0.3;

for i=1:3
   ysol(i+1) = ysol(1) + int(subs(diff,y,ysol(i)),x0,x);
   Iteration_equation = simplify(ysol(i+1))
   fprintf("Value at %d iteration is %f\n\n",i,subs(ysol(i+1),x,xfind));
end
```

Output:

```
>> Picard

Iteration_equation =

x^2/2

Value at 1 iteration is 0.045000


Iteration_equation =

(x^2*(x^3 + 10))/20

Value at 2 iteration is 0.045122


Iteration_equation =

(x^2*(2*x^9 + 55*x^6 + 440*x^3 + 4400))/8800

Value at 3 iteration is 0.045122

>>
```

Code II:

```
syms x y ysol

diff = y + exp(x);
ysol(1) = 0;
```

```
x0 = 0;
xfind = 1;

for i=1:4
   ysol(i+1) = ysol(1) + int(subs(diff,y,ysol(i)),x0,x);
   Iteration_equation = simplify(ysol(i+1))
   fprintf("Value at %d iteration is %f\n\n",i,subs(ysol(i+1),x,xfind));
end
```

Output:

```
>> Picard2

Iteration_equation =

exp(x) - 1

Value at 1 iteration is 1.718282


Iteration_equation =

2*exp(x) - x - 2

Value at 2 iteration is 2.436564


Iteration_equation =

3*exp(x) - 2*x - x^2/2 - 3

Value at 3 iteration is 2.654845


Iteration_equation =

4*exp(x) - 3*x - x^2 - x^3/6 - 4

Value at 4 iteration is 2.706461

>>
```