Experiment - 1

Write a program to implement Method of Successive Substitution.

CODE

```
function [] = SuccessiveSubstituion()
disp("----------------------")
disp("Experiment - 1")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")

syms x x0 x1 f(x) c

f(x) = x^3 - 2*x - 8;
fprintf("\n Initial Equation f(x) is : ")
disp(f(x))
g(1) = (x^3 - 8)/2;
g(2) = (2*x + 8)^(1/3);
g(3) = x^3 - x - 8;

x0 = 2.5;
n = 100;

for i=1:3
    D = diff(g(i), x);
    D = subs(D, x, x0);
    if (D<1 && D>-1)
        break;
    end
end
```

```

fprintf("\n Equation phi(x) is : ");
disp(g(i));
eq(1)=x0;
for j=1:n
    eq(j+1) = subs(g(i), x, eq(j));
    fprintf("\n Value of x%d is %f ", j, eq(j+1));
    if(eq(j)-eq(j+1) < 0.001)
        break;
    end
end

fprintf("\n Root is %f \n", eq(j+1));

```

OUTPUT

```
>> SuccessiveSubstituion
----------------------
Experiment - 1
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

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


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


 Value of x1 is 2.351335
 Value of x2 is 2.333270
 Value of x3 is 2.331056
 Value of x4 is 2.330784
 Root is 2.330784
fx >>
```

Experiment – 2

Write a program to implement following methods:
1. Bisection Method
2. Newton Raphson Method
3. Secant Method
4. Regula Falsi Method

CODE (#Bisection Method)

```
function [] = Bisection_Method()
disp("----------------------")
disp("Experiment - 2(A)")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
syms x x0 x1 f(x) a b c

f(x) = x^2-6*x*exp(-x);
fprintf("\n Initial Equation f(x) is : ")
disp(f(x))

a = 1.4;
b = 1.5;
n = 15;

for i=1:n
    temp_1 = eval(f(a));
    temp_2 = eval(f(b));

    temp_avg = (a + b)/2;
    f_avg = eval(f(temp_avg));

    if f_avg*temp_1 < 0
        b = temp_avg;
        continue
    elseif f_avg*temp_1 > 0
        a = temp_avg;
        continue
    end

end

disp('Calculated Value of root of the given equation: ')
disp(temp_avg)

disp('Calculated Value of the given Function ')
disp(eval(f(temp_avg)))
```

OUTPUT

```
>> Bisection_Method
----------------------
Experiment - 2(A)
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

 Initial Equation f(x) is : x^2 - 6*x*exp(-x)

Calculated Value of root of the given equation:
    1.4324

Calculated Value of the given Function
   6.4120e-06

fx >>
```

CODE (#Newton Raphson Method)

```
function [] = Newton_Raphson()
disp("----------------------")
disp("Experiment - 2(B)")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
syms x x0 x1 f(x) a b c

f(x) = x^2-6*x*exp(-x);
fprintf("\n Initial Equation f(x) is : ")
disp(f(x))

a = 1.4;
b = 1.5;
n = 5;


for i=1:n
    temp_1 = eval(f(a));
    temp_2 = eval(f(b));

    temp_avg = (a + b)/2; % value of x_0

    f_avg = eval(f(temp_avg)); % value of f(x_0)
    temp = diff(f,x);
    f2_avg = eval(temp(temp_avg));

    x_new = temp_avg - (f_avg/f2_avg);
    f_xn = eval(f(x_new));

```

```
31          if f_xn*temp_1 < 0
32              b = x_new;
33              continue
34          elseif f_xn*temp_1 > 0
35              a = x_new;
36              continue
37          end
38
39      end
40
41
42
43      disp('Calculated Value of root of the given equation: ')
44      disp(x_new)
45
46      disp('Calculated Value of the given Function ')
47      disp(f_xn)
```

OUTPUT

```
>> Newton_Raphson
----------------------
Experiment - 2(B)
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

 Initial Equation f(x) is : x^2 - 6*x*exp(-x)

Calculated Value of root of the given equation:
    1.4325

Calculated Value of the given Function
   3.7491e-04

fx >>
```

CODE (#Secant Method)

```
function [] = Secant_Method()
disp("----------------------")
disp("Experiment - 2(C)")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
syms x x0 x1 f(x) a b c

f(x) = x^2-6*x*exp(-x);
fprintf("\n Initial Equation f(x) is : ")
disp(f(x))

a = 1.4; % x0
b = 1.5; % x1
n = 5;


for i=1:n

    f_x0 = eval(f(a)); % f(x0)
    f_x1 = eval(f(b)); % f(x1)

    x_new = b - ((f_x1*(b-a))/(f_x1 - f_x0));

    a = b;
    b = x_new;

end


disp('Calculated Value of root of the given equation: ')
disp(x_new)

disp('Calculated Value of the given Function using Secant Method')
disp(eval(f(x_new)))

end
```

OUTPUT

```
>> Secant_Method
----------------------
Experiment - 2(C)
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

 Initial Equation f(x) is : x^2 - 6*x*exp(-x)

Calculated Value of root of the given equation:
    1.4324

Calculated Value of the given Function using Secant Method
     0

fx >>
```

CODE (#Regula Falsi Method)

```
function [] = Regula_Falsi_Method()
disp("----------------------")
disp("Experiment - 2(D)")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
syms x x0 x1 f(x) a b c

f(x) = x^2-6*x*exp(-x);
fprintf("\n Initial Equation f(x) is : ")
disp(f(x))

a = 1.4;
b = 1.5;
n = 30;


for i=1:n
    temp_1 = eval(f(a));
    temp_2 = eval(f(b));

    x_new = (a*temp_1 - b*temp_2)/(temp_1 - temp_2);

    f_xn = eval(f(x_new));

```

```
26          if f_xn*temp_1 < 0
27              b = x_new;
28              continue
29          elseif f_xn*temp_1 > 0
30              a = x_new;
31              continue
32          end
33
34      end
35
36
37      disp('Calculated Value of root of the given equation: ')
38      disp(x_new)
39
40      disp('Calculated Value of the given Function using Regula Falsi Method')
41      disp(f_xn)
```

OUTPUT

```
>> Regula_Falsi_Method
----------------------
Experiment - 2(D)
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

 Initial Equation f(x) is : x^2 - 6*x*exp(-x)

Calculated Value of root of the given equation:
    1.4324

Calculated Value of the given Function using Regula Falsi Method
  -7.7139e-06

fx >>
```

Experiment - 3

Write a program to implement Gauss Elimination Method WITH and WITHOUT Partial Pivoting.

CODE (# WITH Partial Pivoting)

```
disp("----------------------")
disp("Experiment - 3(A)")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")

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

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

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

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

end

Aug

temp_A = Aug(:,1:N);
temp_B = Aug(:,N+1);

linsolve(temp_A, temp_B)
```

OUTPUT

```
>> temp_gauss_1
----------------------
Experiment - 3(A)
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

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


ans =

     1
     2
     3
     4

fx >>
```

CODE (# WITHOUT Partial Pivoting)

```
function [] = GaussElimination_2()
disp("----------------------")
disp("Experiment - 3(B)")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
A = [1 2 -1 1;-1 1 2 -1;2 -1 2 2;1 1 -1 2];
b = [6;3;14;8];

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

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
Aug
temp_A = Aug(:,1:N);
temp_B = Aug(:,N+1);
linsolve(temp_A, temp_B)
end

```

OUTPUT

```
>> GaussElimination_2
----------------------
Experiment - 3(B)
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

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


ans =

     1
     2
     3
     4

fx >>
```

Experiment - 4

Write a program to implement Gauss Seidel and Jacobi Iterative Methods

CODE (Gauss Seidal Method)

```
% Gauss Seidal Method
disp("----------------------")
disp("Experiment - 4(A)")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
syms x1 x2 x3 x4
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;
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 abs(X-Xold) < err
        break
    end
    Xold=X;
end
disp('Values of x1 x2 x3 x4')
fprintf('\n')
disp(X)
fprintf("Total Steps -> %d \n",steps)
```

OUTPUT

```
>> Gauss_Seidal_Method
----------------------
Experiment - 4(A)
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------
Values of x1 x2 x3 x4

    0.9999
    1.9999
    3.0000
   -0.0000

Total Steps -> 6
```

CODE (Gauss Jacobi Method)

```
disp("----------------------")
disp("Experiment - 4(B)")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
syms x1 x2 x3 x4
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;
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 abs(X-Xold) < err
            break
        end
    Xold=X;
    end
disp('Values of x1 x2 x3 x4')
fprintf('\n')
disp(X)
fprintf("Total Steps -> %d \n",steps)
```

OUTPUT

```
>> Gauss_Jacobian_Method
----------------------
Experiment - 4(B)
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------
Values of x1 x2 x3 x4

    0.9996
    1.9996
    2.9996
   -0.0004

Total Steps -> 9
```

Experiments - 5

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

CODE

```
% Power Method
disp("----------------------")
disp("Experiment - 5")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
A = [2,-1,0;-1,2,-1;0,-1,2];
X = [1;0;0];
EigValold = 0;
err = 0.001;
count = 0;
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;
    count = count + 1;

end
MaxEigValue = EigValnew
MaxEigVect = X/EigValnew
fprintf("Counts = %.1f \n", count);
```

OUTPUT

```
>> Power_Method
----------------------
Experiment - 5
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

MaxEigValue =

    3.4143


MaxEigVect =

    0.7406
   -1.0000
    0.6736

Counts = 6.0
```

Experiment - 6

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

CODE

```
% Forward and Backward Difference Table
disp("----------------------")
disp("Experiment - 6")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
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

```
>> Forward_Backward_Table
----------------------
Experiment - 6
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

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
```

Experiment - 7

Write a program to implement Lagrange's Method of Interpolation

CODE

```
% Lagrange Method Of Interpolation
disp("----------------------")
disp("Experiment - 7")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
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_Interpolation
----------------------
Experiment - 7
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

Value using Lagrange Interpolation is -6.00

Value using polynomial is -6.00

Error is 0.00
```

Experiment - 8

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

CODE

```
% Trapezoid and Simpson's 1/3rd Rule for Integration
disp("----------------------")
disp("Experiment - 8")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
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

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

Value of Integral is 1.410799

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

Value of Integral is 1.366173

fx >>
```

Experiment - 9

Write a program to implement Runge Kutta Method

CODE

```
% Runge Kutta Method
disp("----------------------")
disp("Experiment - 9")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
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

```
>> Runge_Kutta_Method
----------------------
Experiment - 9
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

k1 =

0.2


k2 =

0.24


k3 =

0.244


k4 =

0.2888


k =

0.2428


Approximate value of y(0.2) is 1.242800

>>
```

Experiment - 10

Write a program to implement Picard's Method

CODE

```
% Picard Method
disp("----------------------")
disp("Experiment - 10")
disp("Name -> M K Lino Roshaan")
disp("Roll No -> 2K22/MC/87")
disp("----------------------")
syms x y ysol
diff = x + y*y;
ysol(1) = 0;
x0 = 0;
xfind = 0.3;
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

```
>> picard_method
----------------------
Experiment - 10
Name -> M K Lino Roshaan
Roll No -> 2K22/MC/87
----------------------

Iteration_equation =

x^2/2

Value at 1 iteration is 0.045000


Iteration_equation =

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

Value at 2 iteration is 0.045122


Iteration_equation =

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

Value at 3 iteration is 0.045122


Iteration_equation =

x^2/2 + x^5/20 + x^8/160 + (7*x^11)/8800 + (3*x^14)/49280 + (87*x^17)/23936000 + x^20/7040000 + x^23/445280000

Value at 4 iteration is 0.045122
```