ACHARYA NARENDRA DEV COLLEGE

B.Sc. (H) Computer Science
Semester – IV 2022-23

GE 4 :Numerical Methods Practical

Submitted By: -

Avishkaar Pawar

Roll No: - 21001570021

Submitted To: -

Dr. Sada Nand Prasad

Dr. Deo Datta Arya

INDEX

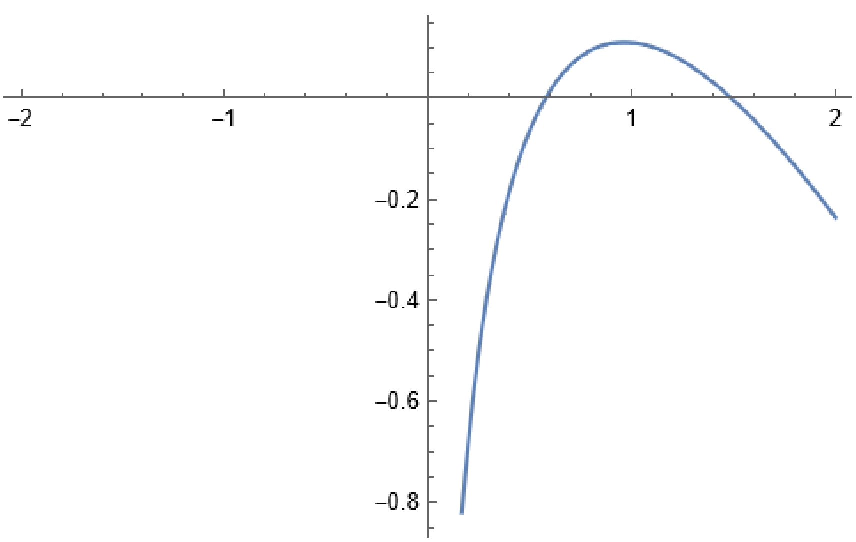
PRACTICAL NO. 1

BISECTION METHOD

QUESTION . The function f (x) = 1.15 - 1.04 * x + Log[x] has a zero on the interval (1,3). Perform five iterations and use the bisection method to approximate the root.

Mathematical Command:

```
bisection[f_, ao_, bo_, n_] := Module[{},
   a = N[ao];
   b = N[bo];
   If[f[a] * f[b] > 0, Print["Bisection method can not applied"];
    Return[]];
   p = (a + b) / 2;
   i = 1;
   While[i ≤ n,
    If[f[a] * f[b] < 0, b = p, a = p];
    Print[i, "   ", a, "   ", b];
    i++;
    p = (a + b) / 2];
   Print["Roots=", p]];
f[x_] := 1.15 - 1.04 * x + Log[x];
Plot[f[x], {x, -2, 2}]
```

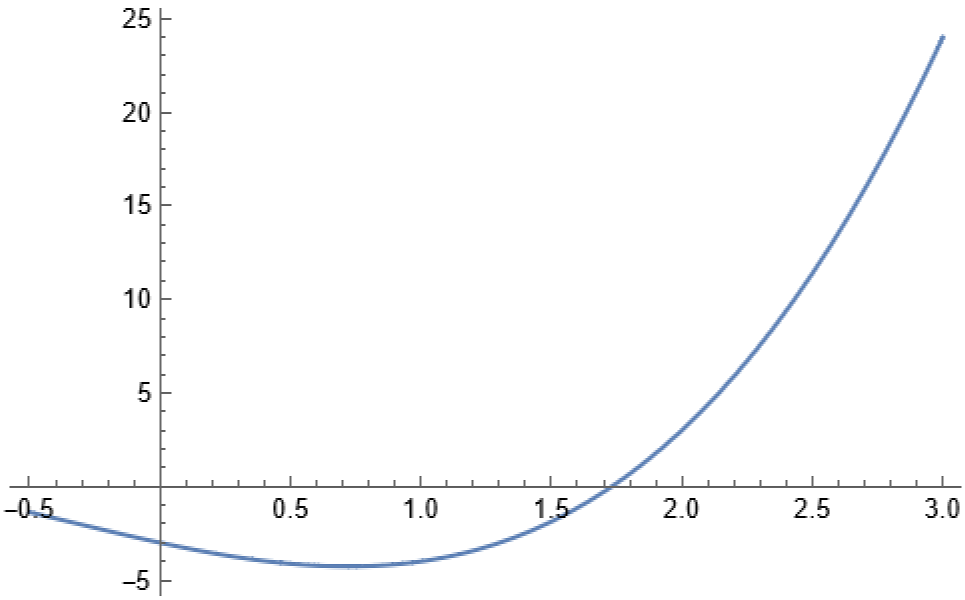


In[ ]:=
```
bisection[f, 1, 3, 5]
```

```
1  1.   2.
2  1.   1.5
3  1.   1.25
4  1.125  1.25
5  1.1875  1.25
Roots=1.21875
```

Question 1 -

In[ ]:=
```
f[x_] := x^3 + x^2 - 3 * x - 3;
Plot[f[x], {x, -0.5, 3}]
```

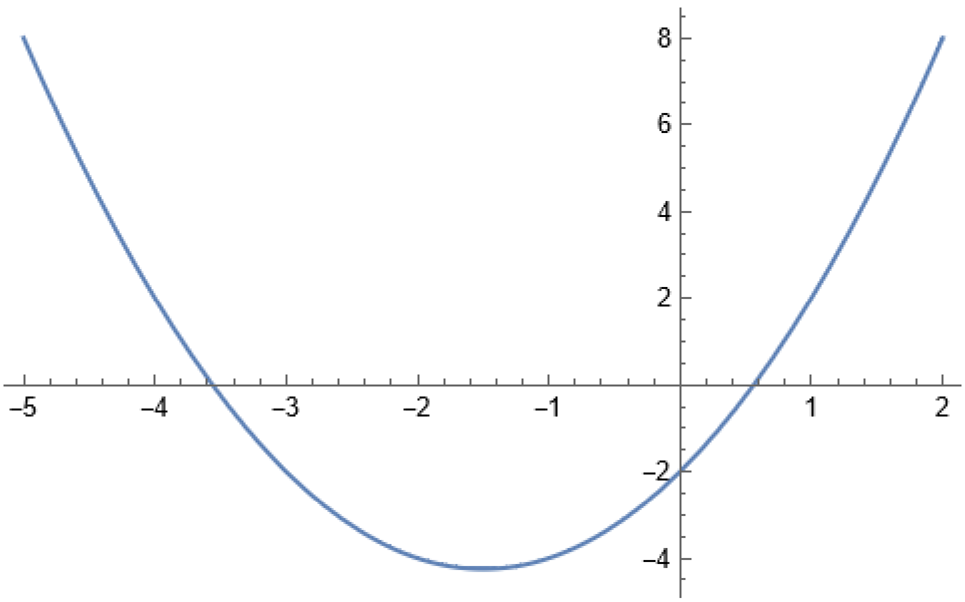


Question 2-

```
In[∘]:= bisection[f, 0, 2, 5]
```

```
1  0.   1.
2  0.5   1.
3  0.75   1.
4  0.875   1.
5  0.9375   1.
Roots=0.96875
```

```
In[∘]:= g[x_] := x^2 + 3 * x - 2;
Plot[g[x], {x, -5, 2}]
```

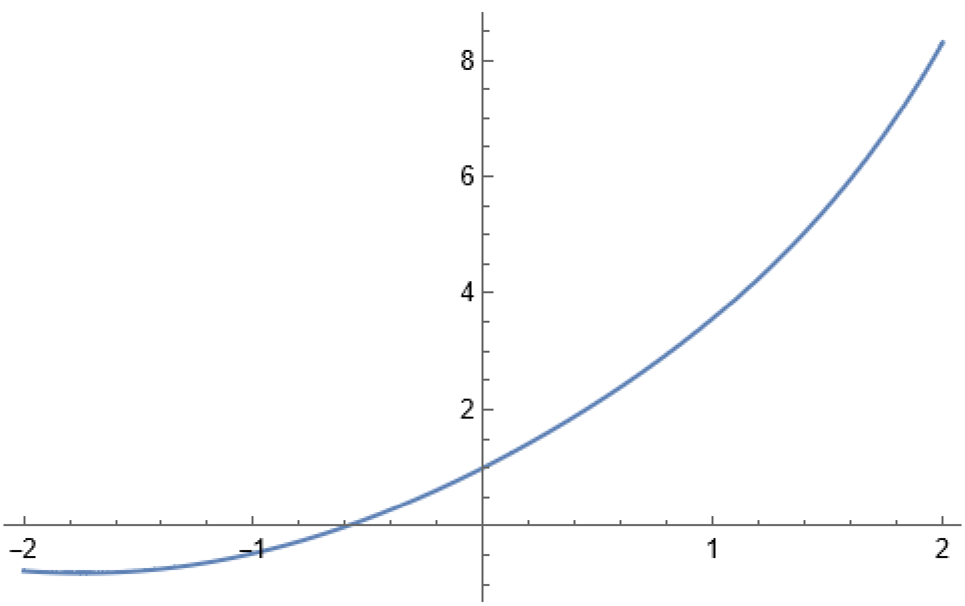


```
In[•]:= bisection[g, -1, 1, 5];
1  -1.  0.
2  -0.5  0.
3  -0.25  0.
4  -0.125  0.
5  -0.0625  0.
Roots=-0.03125
```

```
In[•]:= bisection[g, 1, 2, 4];
Bisection method can not applied
```

Question 3-

```
In[•]:= f[x_] := Sin[x] + E^x;
Plot[f[x], {x, -2, 2}]
```



PRACTICAL NO. 2

SECANT METHOD

QUESTION . The function f (x) = 1.15 - 1.04 * x + Log[x] has a zero on the interval (1, 3). Perform five iterations and use the bisection method to approximate the root.

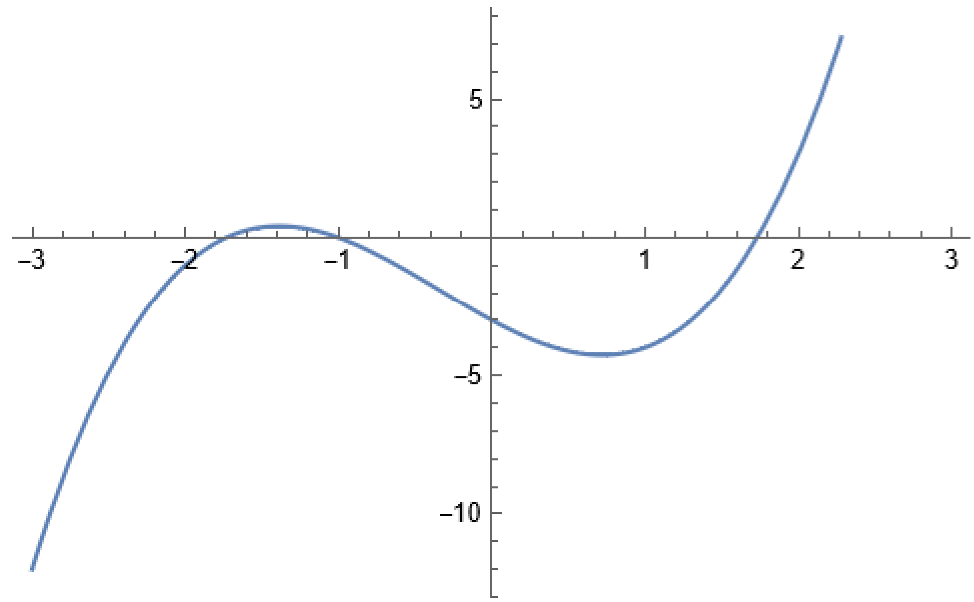
Mathematical Command :

```
In[ ]:= secant[f_, ao_, bo_, n_] := Module[{}, p0 = N[ao]; p1 = N[bo];
   If[f[p0] * f[p1] > 0, Print["Secant method cannot applied"];
    Return[]];
   i = 1;
   While[i ≤ n, p2 = N[p1 - ((p1 - p0) * f[p1] / (f[p1] - f[p0]))];
    Print[i, "   ", p0, "   ", p1];
    i++;
    p0 = p1;
    p1 = p2];
   Print["Roots=", p2]]
  f[x_] := 1.15 - 1.04 * x + Log[x];
  Plot[f[x], {x, -2, 2}]
```

```
1  1.  3.
2  3.  1.22417
3  1.22417  1.372
4  1.372  1.51854
5  1.51854  1.48535
Roots=1.48772
```

Question 1 :-

```
In[ ]:= f[x_] := x^3 + x^2 - 3 * x - 3;
  Plot[f[x], {x, -3, 3}]
```

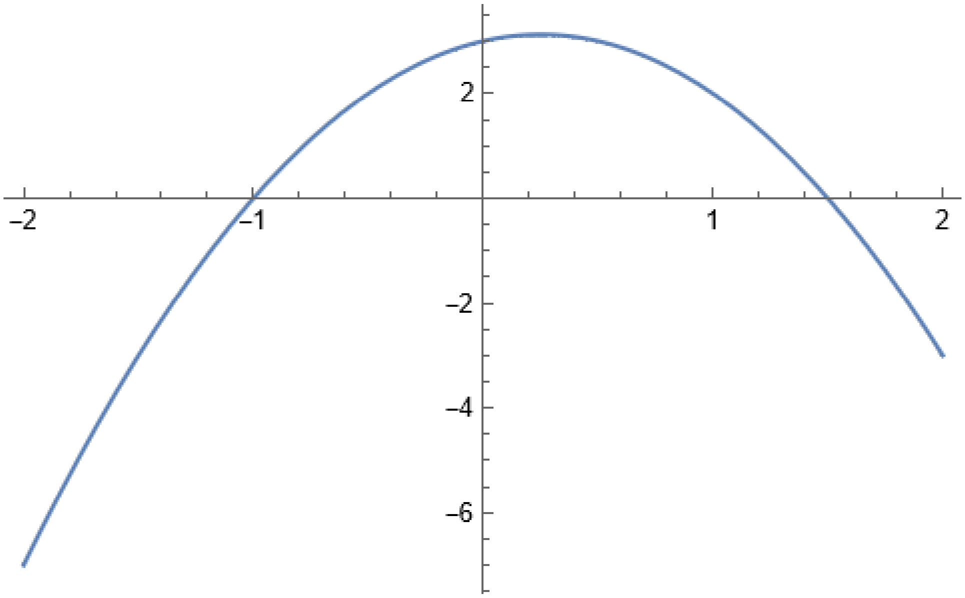



```
In[ ]:= secant[f, 1, 2, 5]
1  1.  2.
2  2.  1.57143
3  1.57143  1.70541
4  1.70541  1.73514
5  1.73514  1.732
Roots=1.73205
```

```
In[ ]:= secant[f, 0, 1, 4];
Secant method cannot applied
```

Question 2:

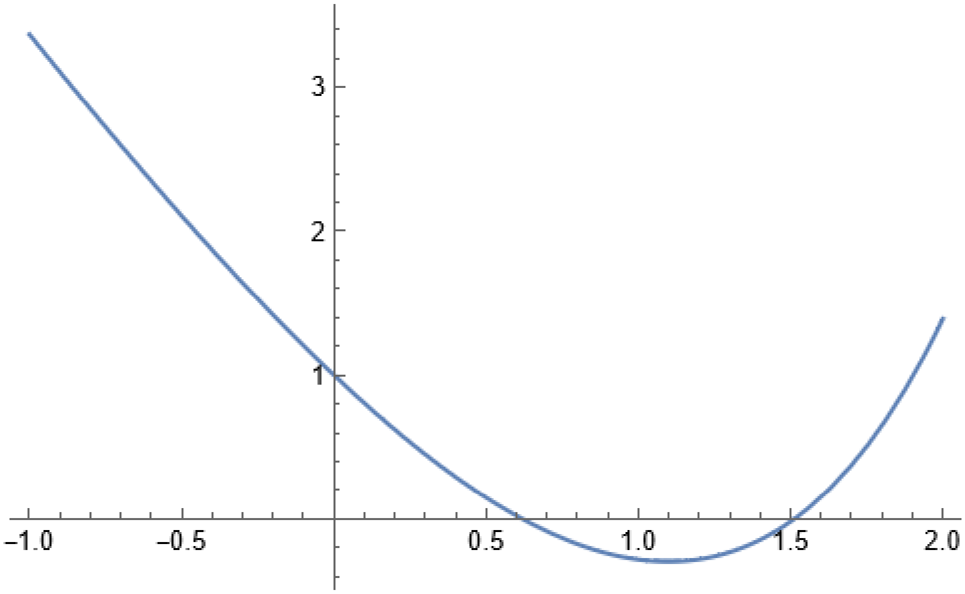
```
In[ ]:= f[x_] := -2*x^2 + x + 3;
Plot[f[x], {x, -2, 2}]
```




```
In[•]:= secant[f, -0, 2, 5]
1  0.  2.
2  2.  1.
3  1.  1.4
4  1.4  1.52632
5  1.52632  1.49892
Roots=1.49999
```

Question 3 :-

```
In[•]:= h[x_] := E^x - 3 * x;
Plot[h[x], {x, -1, 2}]
```



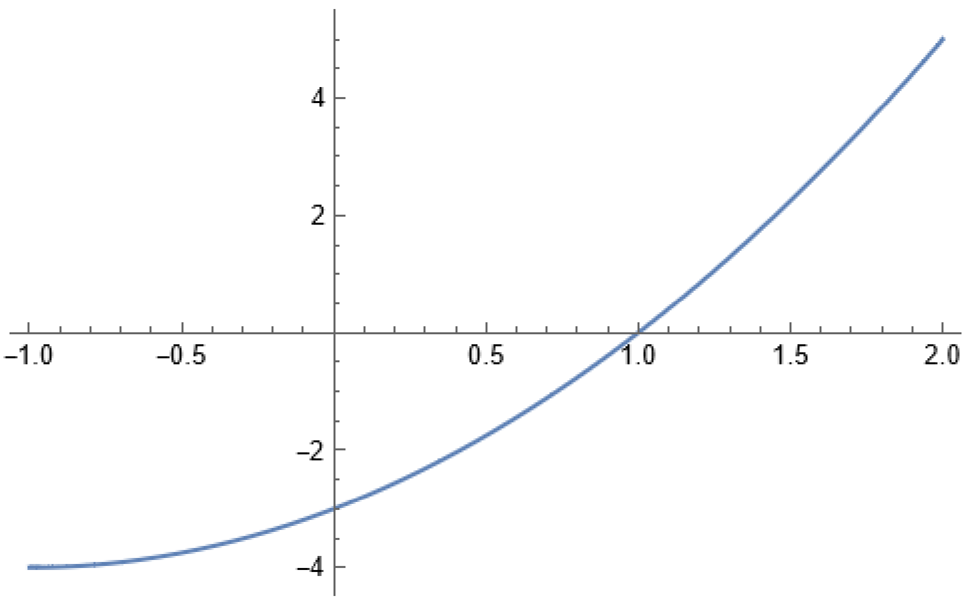

```
In[]:= secant[h, 0, 1, 5]
1  0.  1.
2  1.  0.780203
3  0.780203  0.496679
4  0.496679  0.635952
5  0.635952  0.62056
Roots=0.61904
In[]:= secant[h, -1, 0, 5]

Secant method cannot applied "
```

REGULA FALSI METHOD

Example 1:

```
In[]:= RegulaFalsi[ao_, bo_, f_, n_] := Module[{}, s = N[ao];
         t = N[bo];
         u = (s*f[t] - t*f[s]) / (f[t] - f[s]);
         k = N[n];
         While[k < 1, If[Sign[f[t]] == Sign[f[u]], u = t, s = u];
          u = (s*f[t] - t*f[s]) / (f[t] - f[s]);
          k = k + 1];
         Print["u =", NumberForm[u, 16]];
         Print["f[u] =" NumberForm[f[u], 16]];]
```

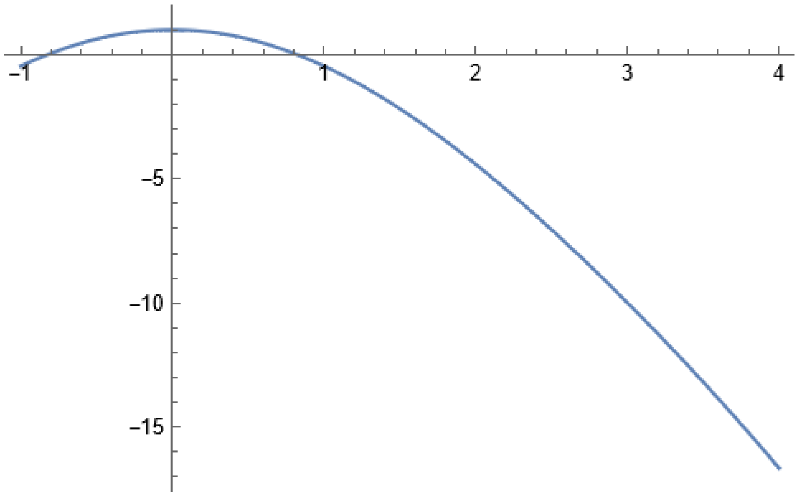
```
In[]:= f[x_] := x^2 + 2*x - 3;
       Plot[f[x], {x, -1, 2}]
```



Example 2:

```
In[]:= RegulaFalsi[0, 1, g, 7]
       u =0.6850733573260452
       f[u] = 0.305047128889926
```

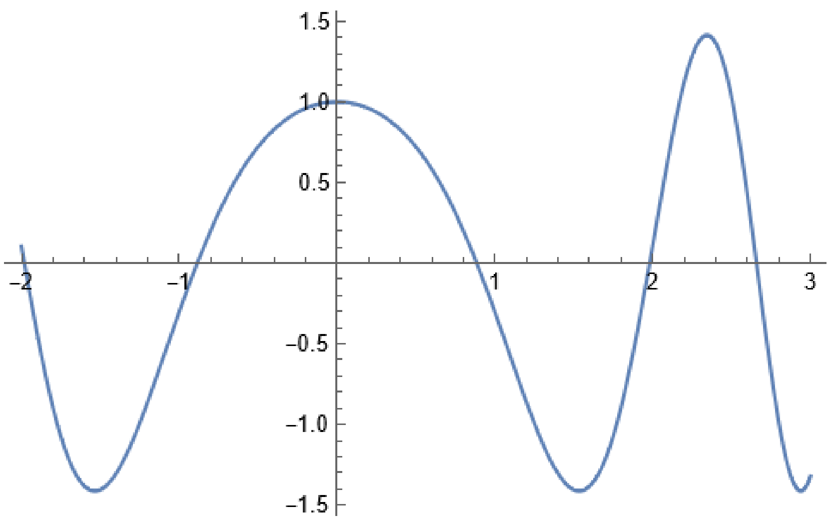
```
In[]:= h[x_] := Cos[x^2] - Sin[x^2];
       Plot[h[x], {x, -2, 3}]
```



Example 3 :

```
In[•]:= RegulaFalsi[0, 1, g, 7]
        u =0.6850733573260452
        f[u] = 0.305047128889926
```

```
In[•]:= h[x_] := Cos[x^2] - Sin[x^2];
        Plot[h[x], {x, -2, 3}]
```




```
In[•]:= RegulaFalsi[1, 2.5, h, 7]
        u =1.338696902269548
        f[u] = -1.195120913099001
```

PRACTICAL 3

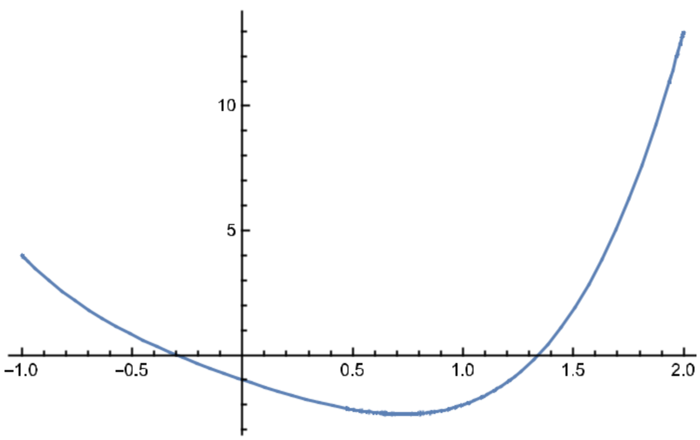
NEWTON-RAPHSON METHOD

In[ ]:=
```
newtonraphson[f_, p0_, eps_] := Module[{}, pold = N[p0];
  i = 1;
  pnew = 0;
  df[x_] = D[f[x], x];
  While[i ≤ 50 && Abs[N[f[pold]]] > eps ,
   pnew = N[pold - N[f[pold]] / N[df[pold]]];
   Print[i, " ", pnew];
   i++;
   pold = pnew;];
  Print["Root =", pnew];]
```

Example 1:-

In[ ]:=
```
f[x_] = x^4 + x^2 - 3*x - 1;
Plot[f[x], {x, -1, 2}]
newtonraphson[f, 1, .0000001]
```

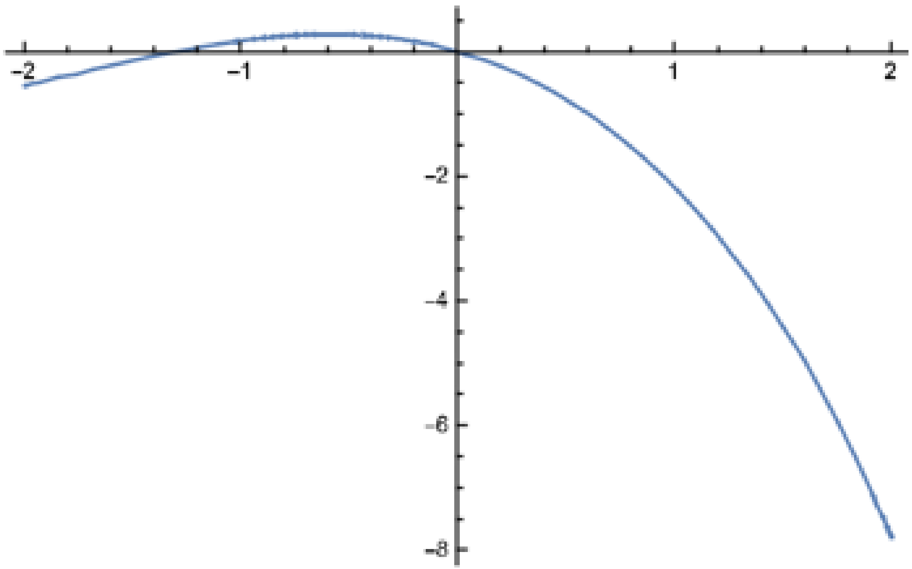
Out[ ]=




```
1 1.66667
2 1.42829
3 1.34865
4 1.34012
5 1.34002
6 1.34002
Root =1.34002
```

Example 2:-

```
f[x_] = Cos[x] - E^x;
Plot[f[x], {x, -2, 2}]
newtonraphson[f, -1.1, .00000001]
```



1 -1.31622

2 -1.29291

3 1.2927

4 -1.2927

Root --1.2927

PRACTICAL 4

Gaussian Elimination Method

Example 1:

```
In[19]:= A = {{1, 1, 1}, {2, 1, -3}, {4, 3, 5}};
         b = {1, 2, 3};
         aug = Transpose[Append[Transpose[A], b]]
         {row, col} = Dimensions[aug]
         m = RowReduce[aug]
         m // MatrixForm
         Take[m, {1, row}, {col, col}]
         % // MatrixForm // N
```

Out[21]= {{1, 1, 1, 1}, {2, 1, -3, 2}, {4, 3, 5, 3}}

Out[22]= {3, 4}

Out[23]= $\left\{\left\{1, 0, 0, \frac{1}{3}\right\}, \left\{0, 1, 0, \frac{5}{6}\right\}, \left\{0, 0, 1, -\frac{1}{6}\right\}\right\}$

Out[24]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & \frac{1}{3} \\ 0 & 1 & 0 & \frac{5}{6} \\ 0 & 0 & 1 & -\frac{1}{6} \end{pmatrix}$$

Out[25]= $\left\{\left\{\frac{1}{3}\right\}, \left\{\frac{5}{6}\right\}, \left\{-\frac{1}{6}\right\}\right\}$

Out[26]//MatrixForm=

$$\begin{pmatrix} 0.333333 \\ 0.833333 \\ -0.166667 \end{pmatrix}$$

Example 2 :

```
In[ ]:= A = {{10, -1, 2, 5}, {1, 10, -1, 6}, {2, 3, 20, 9}, {5, 8, 9, -9}};
b = {4, 3, 7, 8};
aug = Transpose[Append[Transpose[A], b]];
{row, col} = Dimensions[aug];
For[i = 1, i ≤ (row - 1), i++,
  If[aug[[i, i]] == 0, Print["Gaussian elimination method can not apply directly, it requires row replacement"],
   For[k = i + 1, k ≤ row, k++, aug[[k]] = aug[[k]] - (aug[[k, i]] / aug[[i, i]]) * aug[[i]];
    ];
   ];
 ];
Print["Required transformed matrix is ", aug // MatrixForm];
(*back Substitution method*)
A1 = Take[aug, {1, row}, {1, row}];
b1 = Take[aug, {1, row}, {col, col}];
x = Array[p, {row, 1}];
Do[summ = Sum[A1[[i, j]] * x[[j]], {j, i + 1, row}];
  x[[i]] = (b1[[i]] - summ) / A1[[i, i]], {i, row, 1, -1}];
x // N
```

Required transformed matrix is $\begin{pmatrix} 10 & -1 & 2 & 5 & 4 \\ 0 & \frac{101}{10} & -\frac{6}{5} & \frac{11}{2} & \frac{13}{5} \\ 0 & 0 & \frac{2018}{101} & \frac{632}{101} & \frac{543}{101} \\ 0 & 0 & 0 & -\frac{19121}{1009} & \frac{1400}{1009} \end{pmatrix}$

```
Out[ ]= {{0.411406}, {0.331991}, {0.292009}, {-0.0732179}}
```

```
In[7]:= ClearAll[nmax, xnew1, xnew2, xnew3, xold, x1, x2, x3]
x1 = 0;
x2 = 0;
x3 = 0;
nmax = 10;
For[n = 1, n ≤ nmax, n++, xold = {x1, x2, x3};
 xnew1 = 1/5 (10 - x2 - 2 x3);
 xnew2 = 1/5 (-14 - 3 x1 - 4 x3);
 xnew3 = -1/7 (-33 - x1 - 2 x2);
 xnew = {x1 = xnew1, x2 = x2 = xnew2, x3 = xnew3};
 Print["Solution after ", n, "iteration is : ", N[xnew, 7]];]

Solution after 1iteration is : {2.000000, -2.800000, 4.714286}
Solution after 2iteration is : {0.6742857, -7.771429, 4.200000}
Solution after 3iteration is : {1.874286, -6.564571, 2.590204}
Solution after 4iteration is : {2.276833, -5.996735, 3.106449}
Solution after 5iteration is : {1.956767, -6.651259, 3.326195}
Solution after 6iteration is : {1.999774, -6.635016, 3.093464}
Solution after 7iteration is : {2.089618, -6.474636, 3.104249}
Solution after 8iteration is : {2.053228, -6.537170, 3.162907}
```

Gauss Jordan Method

Example 1:

```
In[]:= A = {{1, 1, 1}, {1, 2, 3}, {1, 3, 2}};
b = {3, 0, 3};
A // MatrixForm
b // MatrixForm
aug = Transpose[Append[Transpose[A], b]];
alpha = aug[[2, 1]] / aug[[1, 1]];
aug[[2]] = aug[[2]] - alpha * aug[[1]];
alpha = aug[[3, 1]] / aug[[1, 1]];
aug[[3]] = aug[[3]] - alpha * aug[[1]];
alpha = aug[[3, 2]] / aug[[2, 2]];
aug[[3]] = aug[[3]] - alpha * aug[[2]];
x = ConstantArray[0, 3];
x[[3]] = aug[[3, 4]] / aug[[3, 3]];
x[[2]] = (1 / aug[[2, 2]]) * (aug[[2, 4]] - aug[[2, 3]] * x[[3]]);
x[[1]] = (1 / aug[[1, 1]]) * (aug[[1, 4]] - aug[[1, 2]] * x[[2]] - aug[[1, 3]] * x[[3]]);
sol = x;
x // MatrixForm
```

//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$$

//MatrixForm=

$$\begin{pmatrix} 3 \\ 0 \\ 3 \end{pmatrix}$$

//MatrixForm=

$$\begin{pmatrix} \\ \\ -2 \end{pmatrix}$$

PRACTICAL 5

Jacobi Method

Example 1:

```
In[ ]:= ClearAll[nmax, xnew1, xnew2, xnew3, xold, x1, x2, x3]
x1 = 0;
x2 = 0;
x3 = 0;
nmax = 10;
For[n = 1, n ≤ nmax, n++, xold = {x1, x2, x3};
 xnew1 = 1/5 (10 - x2 - 2 x3);
 xnew2 = 1/5 (-14 - 3 x1 - 4 x3);
 xnew3 = -1/7 (-33 - x1 - 2 x2);
 xnew = {x1 = xnew1, x2 = x2 = xnew2, x3 = xnew3};
 Print["Solution after ", n, "iteration is : ", N[xnew, 7]];]
```

```
Solution after 1iteration is : {2.000000, -2.800000, 4.714286}
Solution after 2iteration is : {0.6742857, -7.771429, 4.200000}
Solution after 3iteration is : {1.874286, -6.564571, 2.590204}
Solution after 4iteration is : {2.276833, -5.996735, 3.106449}
Solution after 5iteration is : {1.956767, -6.651259, 3.326195}
Solution after 6iteration is : {1.999774, -6.635016, 3.093464}
Solution after 7iteration is : {2.089618, -6.474636, 3.104249}
Solution after 8iteration is : {2.053228, -6.537170, 3.162907}
Solution after 9iteration is : {2.042271, -6.562262, 3.139841}
Solution after 10iteration is : {2.056516, -6.537236, 3.131107}
```

Example 2:

```
In[ ]:= ClearAll[nmax, xnew1, xnew2, xnew3, xold, x1, x2, x3]
x1 = 0;
x2 = 0;
x3 = 0;
nmax = 10;
For[n = 1, n ≤ nmax, n++, xold = {x1, x2, x3};
 xnew1 = 1/5 (10 - x2 - 2 x3);
 xnew2 = 1/5 (-14 - 3 x1 - 4 x3);
 xnew3 = -1/7 (-33 - x1 - 2 x2);
 xnew = {x1 = xnew1, x2 = x2 = xnew2, x3 = xnew3};
 Print["Solution after ", n, "iteration is : ", N[xnew, 7]];]
```

```
Solution after 1iteration is : {2.000000, -2.800000, 4.714286}
Solution after 2iteration is : {0.6742857, -7.771429, 4.200000}
Solution after 3iteration is : {1.874286, -6.564571, 2.590204}
Solution after 4iteration is : {2.276833, -5.996735, 3.106449}
Solution after 5iteration is : {1.956767, -6.651259, 3.326195}
Solution after 6iteration is : {1.999774, -6.635016, 3.093464}
Solution after 7iteration is : {2.089618, -6.474636, 3.104249}
Solution after 8iteration is : {2.053228, -6.537170, 3.162907}
Solution after 9iteration is : {2.042271, -6.562262, 3.139841}
Solution after 10iteration is : {2.056516, -6.537236, 3.131107}
```

Gauss-Seidel Method

Example 1:

```
In[ ]:= gaussSeidalMethodN[A0_, b0_, X0_, maxIterations_] := Module[
   {A, b, xk, i, j, k, n, m, outputDetails},
   A = N[A0];
   b = N[b0];
   xk = X0;
   dimA = Dimensions[A];
   n = dimA[[1]];
   m = dimA[[2]];
   If[n ≠ m,
    Print["Gauss Seidal Method can not be applied since A is not a square matrix"];
    Return[]];
   outputDetails = {xk};
   For[k = 0, k ≤ maxIterations, k++,
    For[i = 1, i ≤ n, i++,
     xk[[i]] = 1 / A[[i, i]] ×
       (b[[i]] + A[[i, i]] * xk[[i]] - Sum[A[[i, j]] * xk[[j]], {j, 1, n}]);
     ];
    outputDetails = Append[outputDetails, xk];
    ];
   columnHeading = Table[X[p], {p, 1, n}];
   Print[NumberForm[TableForm[outputDetails,
      TableHeadings → {None, columnHeading}], 8]];
   Print["Number of iterations performed =", maxIterations];
   ];
```

```
A = {{4, 1, 1}, {1, 5, 2}, {1, 2, 3}};
b = {2, -6, -4};
X0 = {0.5, -0.5, -0.5};
gaussSeidalMethodN[A, b, X0, 10]
```

X[1]	X[2]	X[3]
0.5	-0.5	-0.5
0.75	-1.15	-0.81666667
0.99166667	-1.0716667	-0.94944444
1.0052778	-1.0212778	-0.98757407
1.002213	-1.005413	-0.99712901
1.0006355	-1.0012755	-0.9993615
1.0001592	-1.0002872	-0.99986158
1.0000372	-1.0000628	-0.99997053
1.0000083	-1.0000135	-0.99999381
1.0000018	-1.0000028	-0.99999871
1.0000004	-1.0000006	-0.99999973
1.0000001	-1.0000001	-0.99999995

```
Number of iterations performed =10
```

Example 2:

```
In[ ]:= gaussSeidalMethodET[A0_, b0_, X0_, errorTolerance_] := Module[
    {A, b, xk, xk1, i, j, k, n, m, outputDetails, maxNorm},
    A = N[A0];
    b = N[b0];
    xk = X0;
    dimA = Dimensions[A];
    n = dimA[[1]];
    m = dimA[[2]];
    If[n ≠ m,
     Print["Gauss Seidal Method can not be applied since A is not a square matrix"];
     Return[]];
    outputDetails = {xk};
    maxNorm = 1 000 000;
    xk1 = xk;
    For[k = 0, maxNorm > errorTolerance, k++,
     For[i = 1, i ≤ n, i++,
      xk1[[i]] = 1 / A[[i, i]] ×
        (b[[i]] + A[[i, i]] * xk1[[i]] - Sum[A[[i, j]] * xk1[[j]], {j, 1, n}]);
     ];
     maxNorm = Max[Abs[xk1 - xk]];
     xk = xk1;
     outputDetails = Append[outputDetails, xk];
    ];
```

```
    columnHeading = Table[X[p], {p, 1, n}];
    Print[NumberForm[TableForm[outputDetails,
       TableHeadings → {None, columnHeading}], 8]];
    Print["Number of iterations performed to achieve desired accuracy=", k];
    Print["Maximum Norm at ", k, "th iteration =", maxNorm];
  ];
A = {{4, 1, 1}, {1, 5, 2}, {1, 2, 3}};
b = {2, -6, -4};
X0 = {0.5, -0.5, -0.5};
gaussSeidalMethodET[A, b, X0, 0.0001]
```

X[1]	X[2]	X[3]
0.5	-0.5	-0.5
0.75	-1.15	-0.81666667
0.99166667	-1.0716667	-0.94944444
1.0052778	-1.0212778	-0.98757407
1.002213	-1.005413	-0.99712901
1.0006355	-1.0012755	-0.9993615
1.0001592	-1.0002872	-0.99986158
1.0000372	-1.0000628	-0.99997053
1.0000083	-1.0000135	-0.99999381

Number of iterations performed to achieve desired accuracy=8

Maximum Norm at 8th iteration =0.0000493535

PRACTICAL 6

Lagrange Interpolation

Example 1:

```
In[•]:= No = 3; sum = 0;
lagrange[No_, n_] := Product[If[Equal[k, n], 1, (x - x[k]) / (x[n] - x[k])], {k, 1, No}];
For[i - 1, i ≤ No, i++, sum += (f[x[i]] * lagrange[No, i])];
Print[sum]
(x - x[362.6]) (x - x[423.3]) {0.055389, 0.047485, 0.040914}[x[308.6]] / ((x[308.6] - x[362.6]) (x[308.6] - x[423.3]) □) + (x - x[308.6]) (x - x[423.3]) {0.055389, 0.047485, 0.040914}[x[362.6]] / ((-x[308.6] + x[362.6]) (x[362.6] - x[423.3]) □) +
(x - x[308.6]) (x - x[362.6]) {0.055389, 0.047485, 0.040914}[x[423.3]] / ((-x[308.6] + x[423.3]) (-x[362.6] + x[423.3]) □);
sum = 0;
points = {{308.6, 0.055389}, {362.6, 0.047485}, {423.3, 0.040914}};
No = Length[points]
y = points[[All, 1]]
f = points[[All, 2]]
lagrange[No_, n_] := Product[If[Equal[k, n], 1, (x - y[[k]]) / (y[[n]] - y[[k]])], {k, 1, No}]
For[i = 1, i ≤ No, i++, sum += (f[[i]] * lagrange[No, i])]
Expand[sum]
sum /. x → 500

0

Out[•]= 3

Out[•]= {308.6, 362.6, 423.3}

Out[•]= {0.055389, 0.047485, 0.040914}

Out[•]= 0.137745 - 0.000369421 x + 3.32316 × 10^-7 x^2

Out[•]= 0.0361131
```

Newton Interpolation

Example 1:

```
ClearAll[n, y, f, g, points];
points = {{80, 25}, {90, 30}, {100, 42}, {110, 50}};
n = Length[points];
y = points[[All, 1]];
f = points[[All, 2]];
dd[k_] := Sum[(f[[i]] / Product[If[Equal[j, i], 1, (y[[i]] - y[[j]])], {j, 1, k}]), {i, 1, k}];
g[x_] = Sum[(dd[i] * Product[If[i ≤ j, 1, x - y[[j]]], {j, 1, i - 1}]), {i, 1, n}];
Simplify[g[x]]
Evaluate[g[2.5]]
```

$$1557 - \frac{2989\,x}{60} + \frac{53\,x^2}{100} - \frac{11\,x^3}{6000}$$

```
1435.74
```

Example 2:

```
points = {{1, 2}, {2, 3}, {3, 5}, {4, 9}, {5, 17}};
n = Length[points];
y = points[[All, 1]];
f = points[[All, 2]];
dd[k_] :=
 Sum[(f[[i]] / Product[If[Equal[j, i], 1, (y[[i]] - y[[j]])], {j, 1, k}]), {i, 1, k}]
p[x_] = Sum[(dd[i] * Product[If[i ≤ j, 1, x - y[[j]]], {j, 1, i - 1}]), {i, 1, n}];
Simplify[p[x]]
Evaluate[p[2.5]]
```

$$\frac{1}{24}\left(48 - 18\,x + 23\,x^2 - 6\,x^3 + x^4\right)$$

```
3.83594
```

PRACTICAL 7

Trapezoidal Rule

Example 1:

```
In[•]:= trapezoidalRule1[a_, b_, f_] := (b - a) ((f[a] + f[b]) / 2);

In[•]:= f1[x_] := 1 / (1 + x);
        trapezoidalRule1[0, 2, f1] // N

Out[•]= 1.33333

In[•]:= trapezoidalRule2[a_, b_, f_] := Module[{approxIntegral},
          h = b - a;
          approxIntegral = h ((f[a] + f[b]) / 2);
          Return[approxIntegral];
         ];

In[•]:= trapezoidalRule2[0, 2, f1] // N

Out[•]= 1.33333
```

Simpson’s Rule

Example 1 :

```
In[•]:= simpsonRule[a_, b_, f_] := (b - a) / 3 (f[a] + f[b] + 2 f[(a + b) / 2]);

In[•]:= f1[x_] := 1 / (1 + x^2);
        simpsonRule[0, 1, f1]
```

Out[•]= $\frac{31}{30}$

PRACTICAL 8

Euler Methods for Solving first order initial value problems of ODE's

Example 1:

```
In[ ]:= a = 0;
b = 0.8;
n = 5;
f[t_, x_] := t / x
p = 1;
h = (b - a) / (n - 1);
t = Range[a, b, h]; m = Length[t]; sol = {p};
For[i = 1, i < m, i++,
 newsol = sol[[i]] + h * f[t[[i]], sol[[i]]];
 sol = Append[sol, newsol];]
Print["Solution is : "]
Grid[{Prepend[N[t], "t"], Prepend[N[sol], "x"]}, Frame → All]

Solution is :
```

Out[ ]=

t	0.	0.2	0.4	0.6	0.8
x	1.	1.	1.04	1.11692	1.22436