1) BISECTION METHOD

In[1]:=
```
(*bisection method*)
f[x_] := Cos[x]
x0 = 0.0;
x1 = 2.0;
n = 14;

If[f[x0] * f[x1] > 0,
Print["These values dont fit the IVT. Please change the values"],
For[i = 1, i ≤ n, i++, a = (x0 + x1) / 2;
Print[i, "th iteration value is ", a];
If[f[x0] * f[a] < 0, x1 = a, x0 = a];];];
```

```
1th iteration value is 1.
2th iteration value is 1.5
3th iteration value is 1.75
4th iteration value is 1.625
5th iteration value is 1.5625
6th iteration value is 1.59375
7th iteration value is 1.57813
8th iteration value is 1.57031
9th iteration value is 1.57422
10th iteration value is 1.57227
11th iteration value is 1.57129
12th iteration value is 1.5708
13th iteration value is 1.57056
14th iteration value is 1.57068
```

In[21]:=

```
f[x_] := x^5 + 2*x - 1
x0 = 0.0;
x1 = 1.0;
n = 14;

If[f[x0]*f[x1] > 0,
Print["These values dont fit the IVT. Please change the values"],
For[i = 1, i ≤ n, i++, a = (x0 + x1)/2;
Print[i, "th iteration value is ", a];
If[f[x0]*f[a] < 0, x1 = a, x0 = a];];];
```

```
1th iteration value is 0.5
2th iteration value is 0.25
3th iteration value is 0.375
4th iteration value is 0.4375
5th iteration value is 0.46875
6th iteration value is 0.484375
7th iteration value is 0.492188
8th iteration value is 0.488281
9th iteration value is 0.486328
10th iteration value is 0.487305
11th iteration value is 0.486816
12th iteration value is 0.486572
13th iteration value is 0.48645
14th iteration value is 0.486389
```

2) REGULA FALSI AND SECANT METHOD

In[21]:=
```
(*regular falsi method*)
(*q1*)
f[x_] := Cos[x]
x0 = 1.0;
x1 = 3.0;
n = 5;

If[f[x0] * f[x1] > 0,
Print["These values dont fit the IVT. Please change the values"],
For[i = 1, i ≤ n, i++, p = x1 - ((x1 - x0) / (f[x1] - f[x0])) * f[x1];
Print[i, "th iteration value is ", p];
If[f[p] * f[x1] < 0, x0 = p, x1 = p];];];
```

```
1th iteration value is 1.70614

2th iteration value is 1.56503

3th iteration value is 1.57081

4th iteration value is 1.5708

5th iteration value is 1.5708
```

In[16]:=
```
(*q2*)
f[x_] := x^3 - 5 * x + 1
x0 = 0.0;
x1 = 1.0;
n = 5;

If[f[x0] * f[x1] > 0,
Print["These values dont fit the IVT. Please change the values"],
For[i = 1, i ≤ n, i++, p = x1 - ((x1 - x0) / (f[x1] - f[x0])) * f[x1];
Print[i, "th iteration value is ", p];
If[f[p] * f[x1] < 0, x0 = p, x1 = p];];];
```

```
1th iteration value is 0.25

2th iteration value is 0.202532

3th iteration value is 0.201654

4th iteration value is 0.20164

5th iteration value is 0.20164
```

In[26]:=
```
(*secant method*)
(*q1*)
f[x_] := Cos[x];
x0 = 1.0;
x1 = 3.0;
n = 5;

For[i = 1, i ≤ n , i++,
x = x1 - (((x1 - x0) * f[x1]) / (f[x1] - f[x0]));
x0 = x1;
x1 = x;
Print[i, "th iteration value is ", x]];
```

```
1th iteration value is 1.70614

2th iteration value is 1.50196

3th iteration value is 1.5709

4th iteration value is 1.5708

5th iteration value is 1.5708
```

In[31]:=
```
(*q2*)
f[x_] := x^3 - 5 * x + 1;
x0 = 0.0;
x1 = 1.0;
n = 5;

For[i = 1, i ≤ n , i++,
x = x1 - (((x1 - x0) * f[x1]) / (f[x1] - f[x0]));
x0 = x1;
x1 = x;
Print[i, "th iteration value is ", x]];
```

```
1th iteration value is 0.25
2th iteration value is 0.186441
3th iteration value is 0.201736
4th iteration value is 0.20164
5th iteration value is 0.20164
```

3)NEWTON RHAPSON METHOD

In[4]:=

```
(*Newtonraphson method*)
(*que1*)
Newtonraphson[x0_, max_] :=
Module[{p0 = N[x0]},
p1 = p0 - f[p0]/f'[p0];
k = 0;
While[k < max,
p1 = p0 - f[p0]/f'[p0];
p0 = p1;
k = k+1;
Print["value at ", k, "th iteration is = ", NumberForm[p1, 16]];
];
];
Newtonraphson[1, 10];
f[x_] := x^3 + 4*x^2 - 10;
```

```
value at 1th iteration is = 1.454545454545455
value at 2th iteration is = 1.368900401069519
value at 3th iteration is = 1.365236600202116
value at 4th iteration is = 1.365230013435367
value at 5th iteration is = 1.365230013414097
value at 6th iteration is = 1.365230013414097
value at 7th iteration is = 1.365230013414097
value at 8th iteration is = 1.365230013414097
value at 9th iteration is = 1.365230013414097
value at 10th iteration is = 1.365230013414097
```

In[9]:=
```
(*que2*)
Newtonraphson[x0_, max_] :=
Module[{p0 = N[x0]},
p1 = p0 - f[p0]/f'[p0];
k = 0;
While[k < max,
p1 = p0 - f[p0]/f'[p0];
p0 = p1;
k = k+1;
Print["value at ", k, "th iteration is = ", NumberForm[p1, 16]];
];
];
Newtonraphson[1.1, 10];
f[x_] := x^2 - 2;
```

```
value at 1th iteration is = 1.459090909090909

value at 2th iteration is = 1.414903709997168

value at 3th iteration is = 1.414213730689758

value at 4th iteration is = 1.414213562373105

value at 5th iteration is = 1.414213562373095

value at 6th iteration is = 1.414213562373095

value at 7th iteration is = 1.414213562373095

value at 8th iteration is = 1.414213562373095

value at 9th iteration is = 1.414213562373095

value at 10th iteration is = 1.414213562373095
```

In[27]:=
```
(*que3*)
Newtonraphson[x0_, max_] :=
Module[{p0 = N[x0]},
p1 = p0 - f[p0]/f'[p0];
k = 0;
While[k < max,
p1 = p0 - f[p0]/f'[p0];
p0 = p1;
k = k+1;
Print["value at ", k, "th iteration is = ", NumberForm[p1, 16]];
];
];
Newtonraphson[1, 10];
f[x_] := x^3 - 2*x + 3;
```

```
value at 1th iteration is = -1.
value at 2th iteration is = -5.
value at 3th iteration is = -3.465753424657534
value at 4th iteration is = -2.534423313343561
value at 5th iteration is = -2.058999811253937
value at 6th iteration is = -1.908689760957958
value at 7th iteration is = -1.893440881549332
value at 8th iteration is = -1.893289211231559
value at 9th iteration is = -1.893289196304498
value at 10th iteration is = -1.893289196304498
```

4)GAUSS ELIMINATION MATRIX METHOD

QUESTION 1

In[82]:=
```
A = {{1, 1, 1}, {2, -3, 1}, {-1, 2, -1}};
```

In[49]:=
```
A // MatrixForm
```

Out[49]//MatrixForm=

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

In[57]:=
```
x = {x1, x2, x3};
MatrixForm[x]
```

Out[58]//MatrixForm=

$$\begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix}$$

In[60]:=
```
b = {{4}, {2}, {-1}};
```

In[61]:=
```
b // MatrixForm
```

Out[61]//MatrixForm=

$$\begin{pmatrix} 4 \\ 2 \\ -1 \end{pmatrix}$$

In[70]:=
```
aug = ArrayFlatten[{{A, b}}];
```

In[71]:=
```
aug // MatrixForm
```

Out[71]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 4 \\ 2 & -3 & 1 & 2 \\ -1 & 2 & -1 & -1 \end{pmatrix}$$

In[72]:=
```
aug[[2]] = aug[[2]] - 2 * aug[[1]];
```

In[73]:=
```
aug[[3]] = aug[[3]] + aug[[1]];
```

In[74]:=
```
aug // MatrixForm
```

Out[74]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 4 \\ 0 & -5 & -1 & -6 \\ 0 & 3 & 0 & 3 \end{pmatrix}$$

In[75]:=
```
aug[[3]] = aug[[3]] + ((3 * aug[[2]]) / 5);
```

In[76]:=
```
aug // MatrixForm
```

Out[76]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 4 \\ 0 & -5 & -1 & -6 \\ 0 & 0 & -\frac{3}{5} & -\frac{3}{5} \end{pmatrix}$$

In[77]:=
```
upper = Take[aug, 3, 3];
```

In[78]:=
```
upper // MatrixForm
```

Out[78]//MatrixForm=

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

In[79]:=
```
c = Take[aug, 3, -1];
```

In[80]:=
```
c // MatrixForm
```

Out[80]//MatrixForm=

$$\begin{pmatrix} 4 \\ -6 \\ -\frac{3}{5} \end{pmatrix}$$

In[81]:=
```
Solve[upper.x == c]
```

Out[81]=

$\{\{x1 \to 2, x2 \to 1, x3 \to 1\}\}$

QUESTION 2

In[83]:=
```
A2 = {{6, -1, 1}, {1, 1, 1}, {10, 1, -1}};
```

In[84]:=
```
A2 // MatrixForm
```

Out[84]//MatrixForm=

$$\begin{pmatrix} 6 & -1 & 1 \\ 1 & 1 & 1 \\ 10 & 1 & -1 \end{pmatrix}$$

In[85]:=
```
x = {x1, x2, x3};
```

In[86]:=
```
MatrixForm[x]
```

Out[86]//MatrixForm=

$$\begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix}$$

In[87]:=
```
b = {{13}, {9}, {19}};
```

In[88]:=

```
b // MatrixForm
```

Out[88]//MatrixForm=

$$\begin{pmatrix} 13 \\ 9 \\ 19 \end{pmatrix}$$

In[95]:=

```
aug = ArrayFlatten[{{A2, b}}];
```

In[96]:=

```
aug // MatrixForm
```

Out[96]//MatrixForm=

$$\begin{pmatrix} 6 & -1 & 1 & 13 \\ 1 & 1 & 1 & 9 \\ 10 & 1 & -1 & 19 \end{pmatrix}$$

In[97]:=

```
aug[[2]] = aug[[2]] - (aug[[1]] / 6);
```

In[98]:=

```
aug[[3]] = aug[[3]] - 10 * (aug[[1]] / 6);
```

In[99]:=

```
aug // MatrixForm
```

Out[99]//MatrixForm=

$$\begin{pmatrix} 6 & -1 & 1 & 13 \\ 0 & \frac{7}{6} & \frac{5}{6} & \frac{41}{6} \\ 0 & \frac{8}{3} & -\frac{8}{3} & -\frac{8}{3} \end{pmatrix}$$

In[100]:=

```
aug[[3]] = aug[[3]] - (8 / 3) * (aug[[2]] * 6 / 7);
```

In[101]:=

```
aug // MatrixForm
```

Out[101]//MatrixForm=

$$\begin{pmatrix} 6 & -1 & 1 & 13 \\ 0 & \frac{7}{6} & \frac{5}{6} & \frac{41}{6} \\ 0 & 0 & -\frac{32}{7} & -\frac{128}{7} \end{pmatrix}$$

In[102]:=

```
upper = Take[aug, 3, 3];
```

In[103]:=

```
upper // MatrixForm
```

Out[103]//MatrixForm=

$$\begin{pmatrix} 6 & -1 & 1 \\ 0 & \frac{7}{6} & \frac{5}{6} \\ 0 & 0 & -\frac{32}{7} \end{pmatrix}$$

In[104]:=

```
c = Take[aug, 3, -1];
```

In[105]:=

```
c // MatrixForm
```

Out[105]//MatrixForm=

$$\begin{pmatrix} 13 \\ \frac{41}{6} \\ -\frac{128}{7} \end{pmatrix}$$

In[106]:=

```
Solve[upper.x == c]
```

Out[106]=

$\{\{x1 \to 2, x2 \to 3, x3 \to 4\}\}$

4)GAUSS JORDAN MATRIX METHOD

QUESTION 1

In[110]:=

```
A = {{1, 1, 1, 4}, {2, -3, 1, 2}, {-1, 2, -1, -1}}
```

Out[110]=

```
{{1, 1, 1, 4}, {2, -3, 1, 2}, {-1, 2, -1, -1}}
```

In[109]:=

```
RowReduce[A]
```

Out[109]=

```
{{1, 0, 0, 2}, {0, 1, 0, 1}, {0, 0, 1, 1}}
```

In[116]:=

```
Solve[{x1 == 2, x2 == 1, x3 == 1}, {x1, x2, x3}]
```

Out[116]=

```
{{x1 → 2, x2 → 1, x3 → 1}}
```

QUESTION 2

In[117]:=

```
B = {{6, -1, 1, 13}, {1, 1, 1, 9}, {10, 1, -1, 19}};
```

In[118]:=

```
RowReduce[B]
```

Out[118]=

```
{{1, 0, 0, 2}, {0, 1, 0, 3}, {0, 0, 1, 4}}
```

In[119]:=

```
Solve[{x1 == 2, x2 == 3, x3 == 4}, {x1, x2, x3}]
```

Out[119]=

```
{{x1 → 2, x2 → 3, x3 → 4}}
```

5) GAUSS JACOBI

```
GaussJacobi[A0_, B0_, X0_, maxiter_] := Module[
{A = N[A0], b = N[B0], xk = X0, xk1, i, j, k = 0, n, m, OutputDetails},
Size = Dimensions[A];
n = Size[[1]];
m = Size[[2]];
If[n ≠ m,
    Print["not a square matrix, can not Proceed with Gauss Jacobi Method "];
    Return[]];
OutputDetails = {xk};
xk1 = Table[0, {n}];
While[k < maxiter,
For[i = 1, i ≤ n, i++,
xk1[[i]] = (1 / A[[i, i]]) *
        (b[[i]] - Sum[A[[i, j]] * xk[[j]], {j, 1, i - 1}] - Sum[A[[i, j]] * xk[[j]], {j, i + 1, n}]);
];
k++;
OutputDetails = Append[OutputDetails, xk1];
xk = xk1;
];
colHeading = Table[X[k], {k, 1, n}];
Print[
    NumberForm[TableForm[OutputDetails, TableHeadings → {None, colHeading}], 6]];
Print["Number of iteration formed : ", maxiter];
];
```

```
In[7]:= A = {{5, 1, 2}, {-3, 9, 4}, {1, 2, -7}};

In[8]:= b = {10, -14, -33};

In[9]:= X0 = {0, 0, 0};

In[10]:= GaussJacobi[A, b, X0, 15];
```

X[1]	X[2]	X[3]
0	0	0
2.	-1.55556	4.71429
0.425397	-2.98413	4.55556
0.774603	-3.43845	3.92245
1.11871	-3.04067	3.84253
1.07112	-2.89044	4.00534
0.975953	-2.97867	4.04146
0.979148	-3.02644	4.00266
1.00422	-3.00813	3.98947
1.00584	-2.99391	3.99828
0.99947	-2.99729	4.00257
0.998428	-3.00132	4.0007
0.999985	-3.00083	3.9994
1.00041	-2.99974	3.99976
1.00004	-2.99976	4.00013
0.999898	-3.00004	4.00008

Number of iteration formed : 15

In[11]:=
```
A2 = {{6, -1, 1}, {-2, 6, 2}, {1, 4, -8}};
```

In[12]:=
```
b2 = {10, 5, -1};
```

In[13]:=
```
X0 = {0, 0, 0};
```

In[14]:=
```
GaussJacobi[A2, b2, X0, 15];
```

X[1]	X[2]	X[3]
0	0	0
1.66667	0.833333	0.125
1.78472	1.34722	0.75
1.7662	1.17824	1.0217
1.69276	1.0815	0.934896
1.6911	1.08595	0.877345
1.70143	1.10459	0.879364
1.7042	1.10736	0.889972
1.7029	1.10474	0.891704
1.70217	1.10373	0.890234
1.70225	1.10398	0.889637
1.70239	1.1042	0.889771
1.70241	1.10421	0.889901
1.70238	1.10417	0.889904
1.70238	1.10416	0.889882
1.70238	1.10417	0.889877

Number of iteration formed : 15

5) GAUSS SEIDAL

In[43]:=
```
GaussSeidal[A0_, B0_, X0_, maxiter_] := Module[
{A = N[A0], b = N[B0], xk = X0, xk1, i, j, k = 0, n, m, OutputDetails},
Size = Dimensions[A];
n = Size[[1]];
m = Size[[2]];
If[n ≠ m,
    Print["not a square matrix, can not Proceed with Gauss Seidal Method "];
    Return[]];
OutputDetails = {xk};
xk1 = Table[0, {n}];
While[k < maxiter,
For[i = 1, i ≤ n, i++,
xk1[[i]] = (1 / A[[i, i]]) *
        (b[[i]] - Sum[A[[i, j]] * xk1[[j]], {j, 1, i - 1}] - Sum[A[[i, j]] * xk[[j]], {j, i + 1, n}]);
];
k++;
OutputDetails = Append[OutputDetails, xk1];
xk = xk1;
];
colHeading = Table[X[k], {k, 1, n}];
Print[
    NumberForm[TableForm[OutputDetails, TableHeadings → {None, colHeading}], 6]];
Print["Number of iteration formed : ", maxiter];
];
```

In[48]:=
```
A = {{5, 1, 2}, {-3, 9, 4}, {1, 2, -7}};
```

In[49]:=
```
b = {10, -14, -33};
```

In[50]:=
```
X0 = {0, 0, 0};
```

In[51]:=
```
GaussSeidal[A, b, X0, 12];
```

X[1]	X[2]	X[3]
0	0	0
2.	−0.888889	4.74603
0.279365	−3.57178	3.73369
1.22088	−2.80801	4.08641
0.927039	−3.06272	3.97166
1.02388	−2.97944	4.00929
0.992174	−3.00674	3.99696
1.00256	−2.99779	4.001
0.99916	−3.00072	3.99967
1.00028	−2.99976	4.00011
0.99991	−3.00008	3.99996
1.00003	−2.99997	4.00001
0.99999	−3.00001	4.

```
Number of iteration formed : 12
```

6) LAGRANGE INTERPOLATION

In[226]:=

```
ClearAll;
LagrangePolynomial[x0_, f0_] := Module[{xi = x0 , fi = f0, n, m, Polynomial},
n = Length[xi];
m = Length[fi];
If[n ≠ m, Print["List of points and function value are not of the same size"];
    Return[];];
For[i = 1, i ≤ n, i++,
L[i, x_] = Product[(x - xi[[j]]) / (xi[[i]] - xi[[j]]), {j, 1, i - 1}] *
        Product[(x - xi[[j]]) / (xi[[i]] - xi[[j]]), {j, i + 1, n}];
];
Polynomial[x_] = Sum[L[k, x] * fi[[k]], {k, 1, n}];
Return[Polynomial[x]];
];
```

In[231]:=

```
nodes = {0, 1, 3};
```

In[232]:=

```
value = {1, 3, 55};
```

In[235]:=

```
LagrangePolynomial[nodes, value]
```

Out[235]=

$$\frac{1}{3}(1-x)(3-x)+\frac{3}{2}(3-x)x+\frac{55}{6}(-1+x)x$$

In[259]:=

```
nodes = {0, 1, 2};
value = {2, -1, 4};
LagrangePolynomial[nodes, value]
```

Out[261]=

$$(1-x)(2-x)-(2-x)x+2(-1+x)x$$

In[265]:=

```
nodes = {-1, 0, 1, 2};
value = {3, -1, -3, 1};
LagrangePolynomial[nodes, value]
```

Out[267]=

$$-\frac{1}{2}(1-x)(2-x)x-\frac{1}{2}(1-x)(2-x)(1+x)-\frac{3}{2}(2-x)x(1+x)+\frac{1}{6}(-1+x)x(1+x)$$

6) NEWTON INTERPOLATION

In[204]:=

```
ClearAll;
NthDividedDiff[x0_, f0_, startindex_, endindex_] :=
Module[{x = x0, f = f0 , i = startindex , j = endindex , answer},
If[i == j, Return[f[[i]]],
answer =
      (NthDividedDiff[x, f, i + 1, j] - NthDividedDiff[x, f, i, j - 1]) / (x[[j]] - x[[i]]);
Return[answer]];];
NewtonDDPoly[x0_, f0_] :=
Module[{x1 = x0 , f = f0 , n , NewtonPolynomial , k , j},
n = Length[x1];
NewtonPolynomial[y_] = 0;
For[i = 1 , i ≤ n , i++,
Prod[y_] = 1;
For[k = 1 , k ≤ i - 1 , k++ , Prod[y_] = Prod[y] * (y - x1[[k]])];
NewtonPolynomial[y_] = NewtonPolynomial[y] + NthDividedDiff[x1, f, 1, i] * Prod[y]];
Return[NewtonPolynomial[y]];
];
nodes = {0, 1, 3};
value = {1, 3, 55};
NewtonPoly[y_] = NewtonDDPoly[nodes, value]
NewtonPoly[y_] = Simplify[NewtonPoly[y]]
NewtonPoly[2]
```

Out[209]=

$1 + 2\,y + 8\,(-1 + y)\,y$

Out[210]=

$1 - 6\,y + 8\,y^2$

Out[211]=

21

In[212]:=

```
nodes = {0, 1, 2};
```

In[213]:=

```
value = {2, -1, 4};
```

In[214]:=

```
NewtonPoly[y_] = NewtonDDPoly[nodes, value]
NewtonPoly[y_] = Simplify[NewtonPoly[y]]
NewtonPoly[2]
```

Out[214]=

$$2 - 3\,y + 4\,(-1 + y)\,y$$

Out[215]=

$$2 - 7\,y + 4\,y^2$$

Out[216]=

$$4$$

In[217]:=

```
nodes = {-1, 0, 1, 2};
```

In[218]:=

```
value = {3, -1, -3, 1};
```

In[219]:=

```
NewtonPoly[y_] = NewtonDDPoly[nodes, value]
NewtonPoly[y_] = Simplify[NewtonPoly[y]]
NewtonPoly[2]
```

Out[219]=

$$3 - 4\,(1 + y) + y\,(1 + y) + \frac{2}{3}\,(-1 + y)\,y\,(1 + y)$$

Out[220]=

$$-1 - \frac{11\,y}{3} + y^2 + \frac{2\,y^3}{3}$$

Out[221]=

$$1$$

7) SIMPSON'S RULE

In[8]:=
```
SR1[a0_, b0_] := Module[{},
a = a0;
b = b0;
SI = (((b - a) / 6) * (f[a] + (4 * f[(a + b) / 2]) + f[b]));
Print["Integration by simpson Rule is : " N[SI]];
DI = Integrate[f[x], {x, a, b}];
Print["Integration by Direct : ", N[DI]];
Print["Error : ", N[SI - DI]];
]
```

In[9]:=
```
f[x_] := x ^ 5 + 2 * x ^ 4 + x + 1;
```

In[10]:=
```
SR1[1, 2];
```

```
25.4792 Integration by simpson Rule is :

Integration by Direct : 25.4

Error : 0.0791667
```

7) TRAPEZOIDAL RULE

In[11]:=
```
SR1[a0_, b0_] := Module[{},
a = a0;
b = b0;
SI = (((b - a) / 2) * (f[a] + f[b]));
Print["Integration by trapezoidal RUle is : " N[SI]];
DI = Integrate[f[x], {x, a, b}];
Print["Integration by Direct : ", N[DI]];
Print["Error : ", N[SI - DI]];
]
```

In[12]:=
```
f[x_] := 1 / (1 + x ^ 2);
```

In[13]:=
```
SR1[0, 1];
```

```
0.75 Integration by trapezoidal RUle is :

Integration by Direct : 0.785398

Error : -0.0353982
```

8) EULER METHOD FOR SOLVING 1st ORDER ODE

In[3]:=
```
Euler[a0_, b0_, h0_, f_, alpha_] := Module[{a = N[a0], b = N[b0], h = N[h0], n, x},
n = (b - a) / h;
y[0] = alpha;
For[i = 0, i ≤ n, i++, x[i] = a + h * i;
y[i + 1] = y[i] + h * f[x[i], y[i]];
Print["Value at x[", i, "]=", x[i], " is ", y[i]];
];
];
```

In[4]:=
```
f[x_, y_] := x ^ 2 + y ^ 2;
Euler[0, 2, 0.5, f, 1]
```

```
Value at x[0]=0. is 1
Value at x[1]=1. is 2.
Value at x[2]=2. is 7.
Value at x[3]=3. is 60.
```