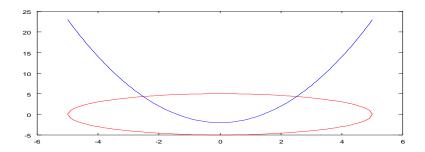
SCIENTIFIC COMPUTING

ASSIGNMENT 4

TUSHAR SIRCAR 130123038

1. SOURCE CODE:

```
function v = func1(x)
 2
3
     v = (ones(size(x))*25) - (x.^2);
 4
     v = sqrt(v);
 5
 6
    plot(x,v,"color","r");
     plot(x,-v, "color", "r");
8
9
10
     end
1
     function v = func2(x)
     v = (x.^2) - (2*ones(size(x)));
3
     plot(x,v);
4
     end
   function v = makePlot()
2
3
    x = [-5:0.1:5];
    hold on;
5
    func1(x);
6
    func2(x);
    hold off;
7
8
9
     end
```



From graph, solution to the two equations is -2.5 and 2.5. Two root are imaginary!

2.

(a) SOURCE CODE

```
1
       #include<iostream>
 2
       #include<cmath>
 3
       using namespace std;
 4
 5
       double getF(double x)
 6
     □ {
 7
           return (x*x*x*x) - (3*x*x) - 21;
 8
 9
10
       double getFDASH(double x)
     □ {
11
12
           return (4*x*x*x) - (6*x);
13
14
15
       int main()
16
     □ {
           double prevVal;
17
           double currentVal;
18
19
           double firstInitial;
20
           double secondInitial;
21
           int n;
22
           double epsilon = 0.0001;
23
           bool rootFound = false;
24
25
           cout<<"Enter initial value: ";
26
           cin>>firstInitial;
27
           prevVal = firstInitial;
28
29
           cout<<"Enter number of iterations: ";</pre>
30
           cin>>n;
31
```

```
32
                cout << "EVALUATING USING NEWTONS METHOD: \n";
 33
                for(int i=0; i<n; i++)</pre>
 34
 35
                      currentVal = prevVal - (getF(prevVal)/getFDASH(prevVal));
 36
                      if(abs(currentVal - prevVal) < epsilon)
 37
 38
                            cout<<"Root found at: "<<currentVal;
39
                            cout<<"\nNumber of iterations: "<<i+1;
                            rootFound = true;
 40
 41
                            break;
 42
                      prevVal = currentVal;
 43
 44
                    // cout<<pre>cont<<pre>//
 45
 46
 47
                if(!rootFound)
 48
                      cout << "Failed to converge!";
 49
 50
 51
 52
 53
                 return 0;
 54
C:\Users\tushar\Desktop\Semesters\Semester5\ScientificComputing\assignmen...
Enter initial value: 2.5
Enter number of iterations: 200
EUALUATING USING NEWIONS METHOD:
Root found at: 2.51432
Number of iterations: 3
Process returned 0 (0x0) execu
                                        execution time : 2.377 s
Press any key to continue.
■ C:\Users\tushar\Desktop\Semesters\Semester5\ScientificComputing\assignmen... - □
Enter initial value: -2.5
Enter number of iterations: 200
EUALUATING USING NEWTONS METHOD:
Root found at: -2.51432
Number of iterations: 3
Process returned 0 (0x0) executors
Press any key to continue.
                                        execution time : 3.485 s
```

NEWTONS METHOD

Initial Value: 2.5 Root Found: 2.51432

Number Of Iterations: 3

NEWTONS METHOD

Initial Values: -2.5

Root Found: -2.51432 Number Of Iterations: 3

(b). SOURCE CODE

```
#include<iostream>
 2
       #include<cmath>
3
       using namespace std;
 4
 5
      double getG(double x, double y)
 6
           return sin(x+y) - exp(x-y);
8
9
       double getGx(double x, double y)
10
11
           return cos(x+y) - exp(x-y);
12
13
       double getGy(double x, double y)
14
15
           return cos(x+y) + exp(x-y);
16
17
       double getH(double x, double y)
18
19
           return cos(x+6) - (x*x*y*y);
20
21
       double getHx(double x, double y)
22
23
           return -sin(x+6) - (2*x*y*y);
24
25
       double getHy(double x, double y)
26
27
           return (-2*x*x*y);
28
29
     int main()
30
```

```
1
      #include<iostream>
 2
       #include<cmath>
 3
      using namespace std;
 4
 5
      double getF(double x)
 6
 7
           return log((x*x)+1) - (exp(0.4*x)*cos(3.14*x));
 8
9
10
       int signum(double x)
11
12
           if(x < 0)
13
               return -1;
           if(x > 0)
14
15
               return 1:
16
           return 0;
17
18
19
       int main()
20
21
           double low, high;
22
           int n;
23
           double epsilon = 0.000001;
           bool rootFound = false;
24
           double prevValue;
25
26
           cout<<"Enter interval to find ONLY NEGATIVE ZERO: ";</pre>
27
28
           cin>>low>>high;
29
30
           cout<<"Enter number of iterations: ";</pre>
31
           cin>>n;
```

```
30
            int main()
 31
 32
                      double prevValX, prevValY;
 33
                      double currentValX, currentValY;
 34
                      double initialX, initialY;
 35
                      int n;
 36
                      double epsilon = 0.0000001;
 37
                      bool rootFound = false;
 38
 39
                      cout<<"Enter initial value for x: ";
 40
                      cin>>initialX;
 41
                      cout<<"Enter initial value for y: ";
 42
                      cin>>initialY;
 43
                      prevValX = initialX;
 44
                      prevValY = initialY;
4.5
45
46
            cout < "Enter number of iterations: ";
47
            cin>>n;
48
49
           cout << "EVALUATING USING NEWTONS METHOD: \n";
50
51
           for(int i=0; i<n; i++)
52
53
                double x = prevValX;
54
                double y = prevValY;
55
                \texttt{currentValX} = \texttt{x} - \{(\texttt{getG}(\texttt{x},\texttt{y}) * \texttt{getHy}(\texttt{x},\texttt{y}) - \texttt{getH}(\texttt{x},\texttt{y}) * \texttt{getGy}(\texttt{x},\texttt{y})\}/\{(\texttt{getGx}(\texttt{x},\texttt{y}) * \texttt{getHy}(\texttt{x},\texttt{y}) - \texttt{getGy}(\texttt{x},\texttt{y}) * \texttt{getHx}(\texttt{x},\texttt{y})\}\}
56
                 \text{currentValY} = y - \{(\text{getH}(\mathbf{x}, \mathbf{y}) * \text{getG}(\mathbf{x}, \mathbf{y}) - \text{getG}(\mathbf{x}, \mathbf{y}) * \text{getHx}(\mathbf{x}, \mathbf{y})\} / \{(\text{getGx}(\mathbf{x}, \mathbf{y}) * \text{getHy}(\mathbf{x}, \mathbf{y}) - \text{getG}(\mathbf{x}, \mathbf{y}) * \text{getHx}(\mathbf{x}, \mathbf{y}))\} \} 
57
                if abs currentValX - prevValX) < epsilon 66 abs currentValX - prevValX) < epsilon
58
59
                    cout<<"Root found at: ("<<currentValX<<" "<<currentValY<<")";
60
                    cout << "\nNumber of iterations: "<<i+1;
61
                    rootFound = true;
62
                    cout<<endl<<"Yalue of functions: "<<getG(currentValX, currentValY) << " "<<getH(currentValX, currentValY);
63
                     break:
64
65
                prevValX = currentValX;
66
                prevValY = currentValY;
67
               // pout <<pre>prevlat<<end);</pre>
68
69
   70
                         if(!rootFound)
   71
                                cout<<"Failed to converge!";
   72
   73
   74
   75
   76
                         return 0;
   77
78
```

Using Cauchy Rimeanns equation and using Newtons Method for complex numbers as given in ques 4, we get the following result:

Root Found At: (-0.756509,0.940737) that is -0.756509 + 0.940737i Number Of Iterations: 22

3. SOURCE CODE

```
1
       #include<iostream>
 2
       #include<cmath>
 3
       using namespace std;
 4
 5
       double getF(double x)
     □{
 6
 7
           return 1 - (x/(exp(x)-1));
 8
 9
10
       double getFDASH(double x)
     □ {
11
12
           return (\exp(x)*(x-1) + 1)/((\exp(x)-1)*(\exp(x)-1));
13
14
       int main()
15
16
     □ {
17
           double prevVal;
18
           double currentVal;
19
           double firstInitial;
20
           double secondInitial;
21
           int n;
22
           double epsilon = 0.0001;
           bool rootFound = false;
23
24
25
           cout<<"Enter initial value: ";
26
           cin>>firstInitial;
27
           prevVal = firstInitial;
28
29
           cout<<"Enter number of iterations: ";
30
           cin>>n;
```

```
31
32
           cout << "EVALUATING USING NEWTONS METHOD: \n";
33
           for(int i=0; i<n; i++)
34
35
                currentVal = prevVal - (getF(prevVal)/getFDASH(prevVal));
36
               if(abs(currentVal - prevVal) < epsilon)
37
38
                    cout<<"Root found at: "<<currentVal;</pre>
39
                    cout<<"\nNumber of iterations: "<<i+1;
40
                    rootFound = true;
41
                    break;
42
43
               prevVal = currentVal;
44
               cout<<pre>cout<<endl;</pre>
45
46
47
           if(!rootFound)
48
               cout<<"Failed to converge!";
49
50
51
52
           return 0;
53
54
```

```
C:\Users\tushar\Desktop\Semesters\Semester5\ScientificComputing\assignmen... 

Enter initial value: 2
Enter number of iterations: 200
EUALUATING USING NEWTONS METHOD:
-1.34268
-0.193492
-0.00585127
-5.6951e-006
Root found at: -5.40437e-012
Number of iterations: 5
Process returned 0 (0x0) execution time: 2.297 s
Press any key to continue.
```

Root Found At: -5.40437*exp(-12)

Number Of Iterations: 5

4.

$$\begin{split} z_{n+1} &= z_n - \frac{f(z)}{f'(z)} \\ x_{n+1} + i y_{n+1} &= x_n + i y_n - \frac{g + i h}{g_x + i h_x} \end{split}$$

Rationalizing the fractional part of the equation

$$x_{n+1} + iy_{n+1} = x_n + iy_n - \frac{(g+ih)(g_x+ih_x)}{(g_x+ih_x)(g_x+ih_x)}$$

Separating the real and imaginary parts

$$x_{n+1} = x_n - \frac{gg_x + hh_x}{g_x^2 + h_x^2}$$
$$y_{n+1} = y_n - \frac{-gh_x + hg_x}{g_x^2 + h_x^2}$$

Using (a) in the above equation

$$x_{n+1} = x_n - \frac{gh_y - hg_y}{g_x h_y - h_x g_y}$$
$$y_{n+1} = y_n - \frac{hg_x - gh_x}{g_x h_y - h_x g_y}$$

Hence proved

5. (a)

SOURCE CODE

```
#include<iostream>
 2
       #include<cmath>
 3
       using namespace std;
 4
       double getG(double x, double y)
     - {
 6
 7
           return (x*x*x) - (3*x*y*y) - x - 1;
 8
 9
       double getGx(double x, double y)
10
11
           return (3*x*x) - (3*y*y) - 1;
12
13
       double getGy(double x, double y)
14
15
           return -6*x*y;
16
       double getH(double x, double y)
17
     □ {
18
19
           return (-y*y*y) + (3*x*x*y) - y;
20
21
       double getHx(double x,double y)
     ₽ {
22
23
           return (6*x*y);
24
25
       double getHy(double x, double y)
26
           return (-3*y*y) + (3*x*x) - 1;
27
28
```

```
30 int main()
                   3
 31
 32
                                         double prevValX, prevValY;
 33
                                         double currentValX, currentValY;
 34
                                         double initialX, initialY;
 35
 36
                                         double epsilon = 0.0001;
 37
                                        bool rootFound = false;
 38
                                        cout << "Enter initial value for x: ";
 39
  40
                                        cin>>initialX;
                                        cout<<"Enter initial value for Y: ";
  41
  42
                                         cin>>initialY;
  43
                                         prevValX = initialX;
                                         prevValY = initialY;
  44
  45
  46
                                         cout<<"Enter number of iterations: ";
  47
                                         cin>>n:
  48
                                         cout < "EVALUATING USING NEWTONS METHOD: \n";
  49
 50
 51
                                         for(int i=0; i<n; i++)
 52
 53
                                                        double x = prevValX;
 54
                                                        double y = prevValY;
 55
                                                        \texttt{currentValX} = \texttt{x} - (\{\texttt{getG}(\texttt{x},\texttt{y}) * \texttt{getHy}(\texttt{x},\texttt{y}) - \texttt{getH}(\texttt{x},\texttt{y}) * \texttt{getGy}(\texttt{x},\texttt{y})) / \{\texttt{getGx}(\texttt{x},\texttt{y}) * \texttt{getHy}(\texttt{x},\texttt{y}) - \texttt{getGy}(\texttt{x},\texttt{y}) * \texttt{getHy}(\texttt{x},\texttt{y}) \} 
  56
                                                         \texttt{currentValY} = \texttt{y} - (\{\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getGx}(\texttt{x},\texttt{y}) * \texttt{getGx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) \}) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBy}(\texttt{x},\texttt{y}) * \texttt{getBy}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) \}) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBy}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) \} \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBy}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBy}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) ) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) \\ (\texttt{getB}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) ) \\ (\texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) ) \\ (\texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) ) \\ (\texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) ) ) \\ (\texttt{getBx}(\texttt{x},\texttt{y}) * \texttt{getBx}(\texttt{x},\texttt{y}) ) ) \\
  57
                                                        if (abs |currentValX - prevValX) < epsilon 66 abs (currentValX - prevValX) < epsilon)
    57
                                                                                   if(abs(currentVaiX - prevVaiX) < epsilon && abs(currentVaiX - prevVaiX) < eps</pre>
    58
                                                                                                        cout<<"Root found at: ("<<currentValX<<" "<<currentValY<<")";</pre>
    59
                                                                                                       cout<<"\nNumber of iterations: "<<i+1;</pre>
    60
                                                                                                       rootFound = true;
    61
    62
                                                                                                      break;
    63
    64
                                                                                  prevValX = currentValX;
    65
                                                                                  prevValY = currentValY;
    66
                                                                              // cout<<pre>cont<<pre>// cout<<<pre>condl;
    67
    68
    69
                                                              if(!rootFound)
    70
                                                                                   cout<<"Failed to converge!";
    71
    72
    73
    74
    75
                                                              return 0:
    76
   77
```

```
C:\Users\tushar\Desktop\Semesters\Semester5\ScientificComputing\assignmen... 

Enter initial value for x: 1
Enter initial value for y: 1
Enter number of iterations: 200
EUALUATING USING NEWTONS METHOD:
Root found at: <-0.662359 -0.56228>
Number of iterations: 11
Process returned 0 (0x0) execution time: 2.672 s
Press any key to continue.
```

Root Found At: (-0.662359,-0.56228) that is -0.662359 -0.56228i

Number Of Iterations: 11

(b) SOURCE CODE

```
#include<iostream>
          2
                                           #include<cmath>
          3
                                          using namespace std;
          4
          5
                                          double getG(double x, double y)
          6
          7
                                                                return (2*x*x*x) - (6*x*y*y) - (6*x) + (6*y) - 6;
         8
        9
                                           double getGx(double x,double y)
    10
                                                               return (6*x*x) - (6*y*y) - 6;
    11
    12
    13
                                          double getGy(double x,double y)
    14
                                                                return (-12*x*y) + 6;
    15
    16
    17
                                          double getH(double x, double y)
    18
    19
                                                                return (6*x*x*y) - (2*y*y*y) - (6*x) - (6*y) + 6;
    20
    21
                                          double getHx(double x,double y)
    22
    23
                                                               return (12*x*y) - 6;
    24
    25
                                          double getHy(double x,double y)
    26
    27
                                                              return (6*x*x) - (6*y*y) - 6;
    28
  20
38
                   int main()
31
32
                             double prevValX, prevValY;
33
                             double currentValX, currentValY;
34
                              double initialX, initialY;
35
                             int n:
36
                              double epsilon = 0.0001;
37
                             bool rootFound = false;
38
39
                               cout<<"Enter initial value for x: ";
40
                                cin>>initialX;
                               cout<<"Enter initial value for y: ";
41
42
                              cinyminitialy:
                              prevValX = initialX;
43
44
                              prevValY = initialY;
45
46
                              cout << "Enter number of iterations: ":
47
                             cin>>n;
48
49
                             cout << "EVALUATING USING NEWTONS METHOD: \n";
50
51
                              for(int i=0; i<n; i++)
52
53
                                          double x = prevValX;
54
                                          double y = prevValY;
55
                                         \texttt{currentValX} = \mathbf{x} - \{(\texttt{getG}(\mathbf{x}, \mathbf{y}) * \texttt{getB}(\mathbf{x}, \mathbf{y}) - \texttt{getB}(\mathbf{x}, \mathbf{y}) * \texttt{getG}(\mathbf{x}, \mathbf{y}) * \texttt{getB}(\mathbf{x}, \mathbf{y}) * \texttt{getH}(\mathbf{x}, \mathbf{
56
                                         \texttt{currentValY} = y - \{(\texttt{getH}(\mathbf{x}, y) \cdot \texttt{getGx}(\mathbf{x}, y) - \texttt{getG}(\mathbf{x}, y) \cdot \texttt{getHx}(\mathbf{x}, y)\}/(\texttt{getGx}(\mathbf{x}, y) \cdot \texttt{getHy}(\mathbf{x}, y) - \texttt{getGy}(\mathbf{x}, y) \cdot \texttt{getHx}(\mathbf{x}, y)\}\}
                                                                                                                                                                                         .....
```

```
\texttt{currentValY} = y - ([\texttt{getH}(\mathbf{x}, \mathbf{y}) * \texttt{getGx}(\mathbf{x}, \mathbf{y}) - \texttt{getG}(\mathbf{x}, \mathbf{y}) * \texttt{getHx}(\mathbf{x}, \mathbf{y})) / (\texttt{getGx}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y}) - \texttt{getGy}(\mathbf{x}, \mathbf{y}) * \texttt{getHx}(\mathbf{x}, \mathbf{y})) / (\texttt{getGx}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y}) - \texttt{getGy}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y})) / (\texttt{getGx}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y}) - \texttt{getGy}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y})) / (\texttt{getGx}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y}) - \texttt{getGy}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y})) / (\texttt{getGx}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y}) - \texttt{getGy}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y})) / (\texttt{getGx}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y}) + \texttt{getGy}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y})) / (\texttt{getGx}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y}) + \texttt{getHy}(\mathbf{x}, \mathbf{y})) / (\texttt{getGx}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y}) + \texttt{getHy}(\mathbf{x}, \mathbf{y})) / (\texttt{getGx}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y}) + \texttt{getHy}(\mathbf{x}, \mathbf{y})) / (\texttt{getGx}(\mathbf{x}, \mathbf{y}) * \texttt{getHy}(\mathbf{x}, \mathbf{y})) / (
                                                                                                                                if(abs(currentValX - prevValX) < epsilon & abs(currentValX - prevValX) < epsilon)
59
                                                                                                                                                                      cout<< "Root found at: ("<<currentValX<<" "<<currentValY<<")";
60
                                                                                                                                                                    coutce"\nMumber of iterations: "<<i+1;
 61
                                                                                                                                                                    rootFound = true:
 62
                                                                                                                                                               break:
 63
                                                                                                                            prevValX = currentValX;
prevValY = currentValY;
 64
 65
 66
                                                                                                                      // cour < nrayVal < andl:
 68
 69
                                                                                       if (!rootFound)
 70
                                                                                                                                cout<<"Failed to converge!":
71
72
 73
                                                                                          return 0;
77
```

```
C:\Users\tushar\Desktop\Semesters\Semester5\ScientificComputing\assignmen... - \ \ Enter initial value for x: 1
Enter initial value for y: 1
Enter number of iterations: 200
EVALUATING USING NEWTONS METHOD:
Root found at: (-0.0768527 0.865702)
Number of iterations: 5
Process returned 0 (0x0) execution time: 3.516 s
Press any key to continue.
```

Root Found At: (-0.0768527,0.865702) that is -0.0768

Number Of Iterations: 5