The Newton-Raphson method for Non-Linear Equation

Theory

Let (x_0, y_0) be an initial approximation to the root of the system. If (xo+h, yo+k) is the root of the system, then we must have,

$$f(x_0+h, y_0+k) = 0, \quad g(x_0+h, y_0+k) = 0 \ldots\ldots\ldots\ldots (1)$$

Assuming that f and g are sufficiently differentiable , by expanding the equation no. 1 by Taylor series , it is obtained that,

$$f_0 + h\frac{\partial f}{x_0} + k\frac{\partial f}{y_0} + \cdots = 0$$

$$g_0 + h\frac{\partial g}{x_0} + k\frac{\partial g}{y_0} + \cdots = 0$$

where, $\frac{\partial f}{\partial x_0} = \left[\frac{\partial f}{\partial x_0}\right]_{x=x_0}$ $\qquad f_0 = f(x_0,y_0)$ etc.

Neglecting the second and higher order terms , the following system of linear equation is obtained,

$$h\frac{\partial f}{x_0} + k\frac{\partial f}{y_0} = -f_0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$h\frac{\partial g}{x_0} + k\frac{\partial g}{y_0} = -g_0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

if the Jacobian,

$$J(f,g) = \begin{vmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{vmatrix}$$

Does not vanish then the linear equation (2) and (3) possess a unique solution solving out the h and k. The new approximations are given by ,

$$x_1 = x_0 + h \qquad y_1 = y_0 + k$$

This process is repeated until the desired accuracy value is reached.

Code

```
#include<bits/stdc++.h>
using namespace std;

double F(double x,double y)
{
   return ( x*x - y*y -4);
}

double G(double x,double y)
{
   return ( x*x + y*y -16 );
}

double PDFx(double x)
{
   return 2*x;
}

double PDFy(double y)
{
   return -(2*y);
}

double PDGx(double x)
{
   return 2*x;
}

double PDGy(double y)
{
   return 2*y;
}

int Check(double a,double b,double  c,double d)
{
   if((a*d - b*c)!=0)
      return 1;
   else
      return 0;
}

double SolveH(double f0,double g0,double a,double b,double c,double d)
{
   return ((g0*b - f0*d) / (a*d - b*c));
}
```

```
double SolveK(double f0,double g0,double a,double b,double c,double d)
{
   return ((f0*c - g0*a) / (a*d - b*c));
}

void NewtonRaphson(double x0,double y0)
{
   double a,b,c,d,h,k,m,n,f0,g0,xn,yn;
   do
   {
      a = PDFx(x0);
      b = PDFy(y0);
      c = PDGx(x0);
      d = PDGy(y0);
      f0 = F(x0,y0);
      g0 = G(x0,y0);

      if(Check(a,b,c,d)!=0)
      {
         h = SolveH(f0,g0,a,b,c,d);
         k = SolveK(f0,g0,a,b,c,d);
         m = x0;
         n = y0;
         xn = x0 + h;
         yn = y0 + k;
         x0 = xn;
         y0 = yn;
      }
      else
      {
         cout<<"Wrong Assumption"<<endl;
         return;
      }

   cout<<"X: "<<xn<<"\t";
   cout<<"Y: "<<yn<<"\t";
   cout<<"Error rate (x): "<<fabs(xn-m)<<"\t"<<"Error Rate (y): "<<fabs(yn-n)<<endl;

   }while(fabs(xn-m)>=0.00001 && fabs(yn-n)>=0.00001);

   cout<<"\nFinal X: "<<xn<<endl;
   cout<<"Final Y: "<<yn<<endl;
   cout<<"Error rate of x : "<<fabs(xn-m)<<endl;
   cout<<"Error rate of y: "<<fabs(yn-n)<<endl;
}
```

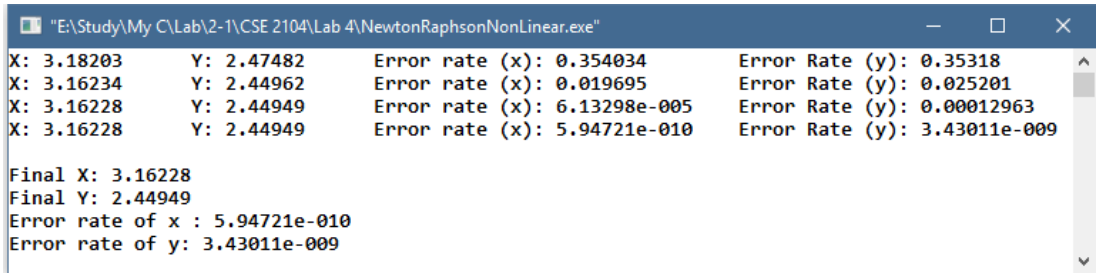
```
int main()
{
   double x0 = 2.828, y0 = 2.828;
   double a,b,c,d;

   a = PDFx(x0);
   b = PDFy(y0);
   c = PDGx(x0);
   d = PDGy(y0);

   if(Check(a,b,c,d)==0)
      cout<<" Wrong Initialization..."<<endl;
   else
      NewtonRaphson(x0,y0);

   return 0;
}
```

Output



Discussion

In the above code, Newton Raphson method for finding out the roots of non linear equation was performed. Here all the required functions was declared and defined which is necessary to calculate and solve the above mentioned equations in theory section. A do while loop was executed until the required amount of accuracy is gained. Thus the value of x and y of the given equations was obtained by the Newton-Raphson method.