MA 322: Lab Assignment #4

Due on Sunday, August 30, 2015

Jiten Chandra Kalita

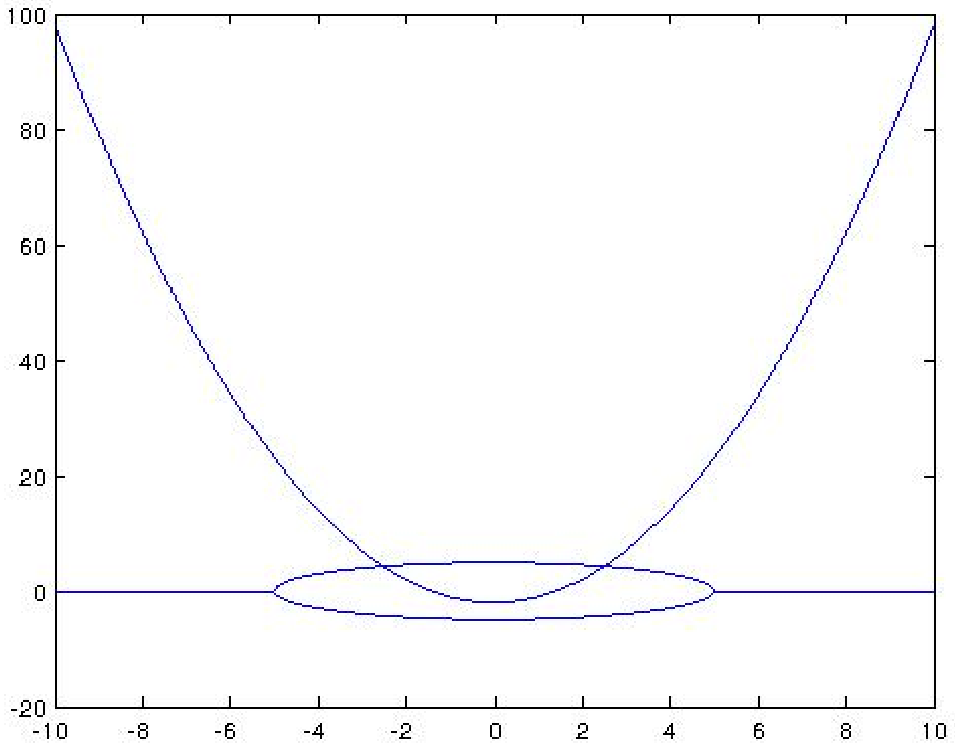
Silvi Pandey-130123045

Contents

PROBLEM 1

```
x=[-10:0.01:10];
z=25-x.*x;
y=sqrt(z);
y1=-sqrt(z);
y2=x.*x-2;
plot(x,y);
hold on;
plot(x,y1);
plot(x,y2);
hold off;
```

OUTPUT PLOT



EXPLANATION

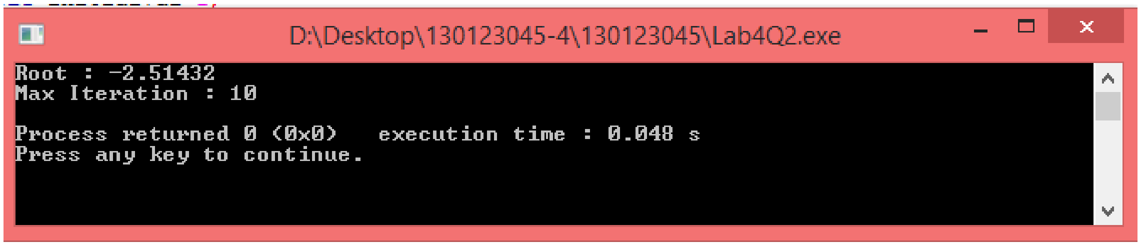
(a)According to the plot,the approximate roots are 2.5 and -2.5.
(b)By eliminating y form both equations,we get a 4 degree polynomial in x given by $x^4 - 3x^2 - 21$
(c)Two roots are complex conjugates(Not visible in the plot).

PROBLEM 2(PART 1)

```cpp
#include<iostream>
#include<math.h>
#include<stdlib.h>
using namespace std;

double func(double val);
void newtonMethod(double initialVal);
double derivativeFunction(double val);
int main()
{
     double initialVal=1;
     newtonMethod(initialVal);
}
double function(double val)
{
     return pow(val,4)-3*pow(val,2)-21;
}
double derivativeFunction(double val)
{
     return 4*pow(val,3)-6*val;
}
void newtonMethod(double initialVal)
{
     double preVal=initialVal;
     double currVal;
     int Iter=0;
     while(1)
     {
          currVal=preVal-function(preVal)/derivativeFunction(preVal);
          Iter++;
          if(fabs(function(currVal))<=0.000001)
               break;
          preVal=currVal;
     }
     cout<<"Root : "<<currVal<<endl;
     cout<<"Max Iteration : "<<Iter<<endl;
}
```

OUTPUT



```
Root : -2.51432
Max Iteration : 10

Process returned 0 (0x0)   execution time : 0.048 s
Press any key to continue.
```

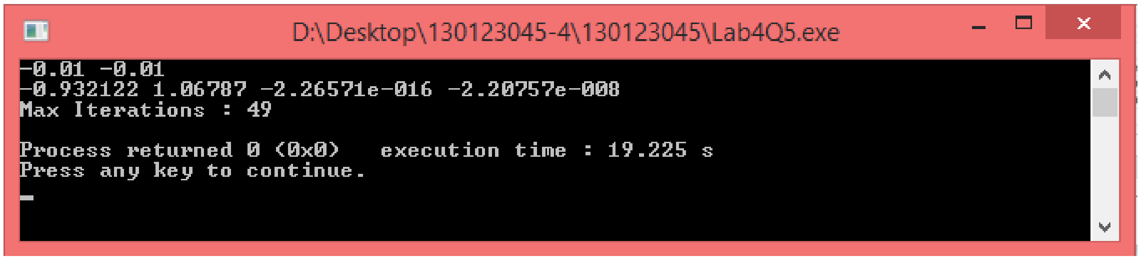
PROBLEM 2(PART 2)

```
#include<iostream>
#include<math.h>
#include<stdlib.h>
using namespace std;

double gFunction(double x,double y);
double hFunction(double x,double y);
double gxderivative(double x,double y);
double hxderivative(double x,double y);
double gyderivative(double x,double y);
double hyderivative(double x,double y);
void newtonMethod(double initialx,double initialy);

int main()
{
    double initialx,initialy;
    cin>>initialx>>initialy;
    newtonMethod(initialx,initialy);
}
double gFunction(double x,double y)
{
  return sin(x+y)-exp(x-y);
}
double hFunction(double x,double y)
{
  return cos(x+y)-x*x*y*y;
}
double gxderivative(double x,double y)
{
   return cos(x+y)-exp(x-y);
}
double gyderivative(double x,double y)
{
   return cos(x+y)+exp(x-y);
}
double hxderivative(double x,double y)
{
  return -sin(x+6)-2*x*y*y;
}
double hyderivative(double x,double y)
{
   return -2*x*x*y;
}
void newtonMethod(double initialx,double initialy)
{
    double preValx,preValy,currValx,currValy;
    preValx=initialx;
    preValy=initialy;
    int Iter=0;
    while(1)
    {
        currValx=preValx-(gFunction(preValx,preValy)*hyderivative(preValx,preValy)-
    hFunction(preValx,preValy)*gyderivative(preValx,preValy))
                                    /(gxderivative(preValx,preValy)
                                    *hyderivative(preValx,preValy)-gyderivative(preValx,preValy)
*hxderivative(preValx,preValy));
        currValy=preValy-(hFunction(preValx,preValy)*gxderivative(preValx,preValy)
        -gFunction(preValx,preValy)*hxderivative(preValx,preValy))
                                   /(gxderivative(preValx,preValy)
                                   *hyderivative(preValx,preValy)-gyderivative(preValx,preValy)
                                   *hxderivative(preValx,preValy));
        Iter++;
        if(fabs(currValx-preValx)/fabs(preValx)<0.0000001
        &&fabs(currValy-preValy)/fabs(preValy)<0.0000001)
            break;
        preValx=currValx;
        preValy=currValy;
    }
    cout<<currValx<<" "<<currValy<<endl;
    cout<<gFunction(currValx,currValy)<<" "<<hFunction(currValx,currValy)<<endl;
    cout<<"Max Iterations : "<<Iter<<endl;
}
```

OUTPUT



```
-0.01 -0.01
-0.932122 1.06787 -2.26571e-016 -2.20757e-008
Max Iterations : 49

Process returned 0 (0x0)   execution time : 19.225 s
Press any key to continue.
```

PROBLEM 3

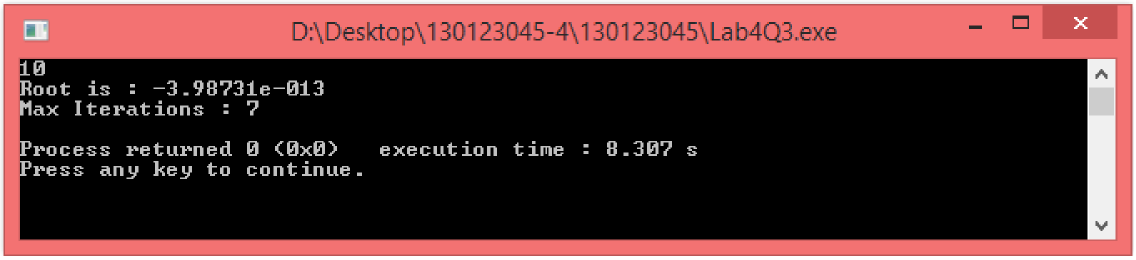
```
#include<iostream>
#include<stdlib.h>
#include<math.h>
using namespace std;

double function(double val);
double derivativeFunction(double val);
double secondDerivativeFunction(double val);
double newFunction(double val);
double newderivativeFunction(double val);
void modifiedNewtonMethod(double initialVal);

int main()
{
     double initialVal;
     cin>>initialVal;
     modifiedNewtonMethod(initialVal);
}
```

```
double function(double val)
{
    return exp(val)-val-1;
}
double derivativeFunction(double val)
{
    return exp(val)-1;
}
double secondDerivativeFunction(double val)
{
    return exp(val);
}
double newFunction(double val)
{
    return function(val)/derivativeFunction(val);
}
double newderivativeFunction(double val)
{
    return 1-function(val)*secondDerivativeFunction(val)/pow(derivativeFunction(val),2);
}
void modifiedNewtonMethod(double initialVal)
{
    double preVal=initialVal;
    double currVal;
    int Iter=0;
    while(1)
    {
        currVal=preVal-newFunction(preVal)/newderivativeFunction(preVal);
        Iter++;
        if(fabs(preVal-currVal)/fabs(preVal)<0.0000001)
            break;
        preVal=currVal;
    }
    cout<<"Root is : "<<currVal<<endl;
    cout<<"Max Iterations : "<<Iter<<endl;
}
```

OUTPUT



```
10
Root is : -3.98731e-013
Max Iterations : 7

Process returned 0 (0x0)   execution time : 8.307 s
Press any key to continue.
```

PROBLEM 4

f'(z) $= g_x + ih_x$
f'(z) $= h_x - ig_y$

(a) Hence, $g_x = h_y$ and $h_x = -g_y$

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

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
$x_{n+1} = x_n - \frac{gh_y-hg_y}{g_xh_y-h_xg_y}$
$y_{n+1} = y_n - \frac{hg_x-gh_x}{g_xh_y-h_xg_y}$

Hence proved

PROBLEM 5

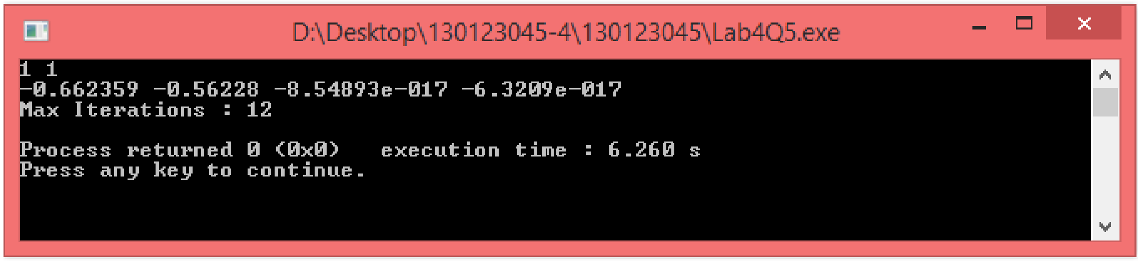
```
#include<iostream>
#include<math.h>
#include<stdlib.h>
using namespace std;

double gFunction(double x,double y);
double hFunction(double x,double y);
double gxderivative(double x,double y);
double hxderivative(double x,double y);
double gyderivative(double x,double y);
double hyderivative(double x,double y);
void newtonMethod(double initialx,double initialy);

int main()
{
    double initialx,initialy;
    cin>>initialx>>initialy;
    newtonMethod(initialx,initialy);
}
double gFunction(double x,double y)
{
  return x*x*x-x-1-3*x*y*y;
}
double hFunction(double x,double y)
{
  return 3*x*x*y-y*y*y-y;
}
double gxderivative(double x,double y)
{
   return 3*x*x-3*y*y-1;
}
double gyderivative(double x,double y)
{
   return -6*x*y;
}
double hxderivative(double x,double y)
{
     return 6*x*y;
}
double hyderivative(double x,double y)
{
    return 3*x*x-3*y*y-1;
}
void newtonMethod(double initialx,double initialy)
{
    double preValx,preValy,currValx,currValy;
    preValx=initialx;
    preValy=initialy;
    int Iter=0;
    while(1)
    {
        currValx=preValx-(gFunction(preValx,preValy)*hyderivative(preValx,preValy)-
    hFunction(preValx,preValy)*gyderivative(preValx,preValy))
                            /(gxderivative(preValx,preValy)
                            *hyderivative(preValx,preValy)-gyderivative(preValx,preValy)
*hxderivative(preValx,preValy));
        currValy=preValy-(hFunction(preValx,preValy)*gxderivative(preValx,preValy)
        -gFunction(preValx,preValy)*hxderivative(preValx,preValy))
                           /(gxderivative(preValx,preValy)
                           *hyderivative(preValx,preValy)-gyderivative(preValx,preValy)
                           *hxderivative(preValx,preValy));
        Iter++;
        if(fabs(currValx-preValx)/fabs(preValx)<0.0000001
        &&fabs(currValy-preValy)/fabs(preValy)<0.0000001)
            break;
        preValx=currValx;
        preValy=currValy;
    }
    cout<<currValx<<" "<<currValy<<endl;
    cout<<gFunction(currValx,currValy)<<" "<<hFunction(currValx,currValy)<<endl;
    cout<<"Max Iterations : "<<Iter<<endl;
}
```

OUTPUT

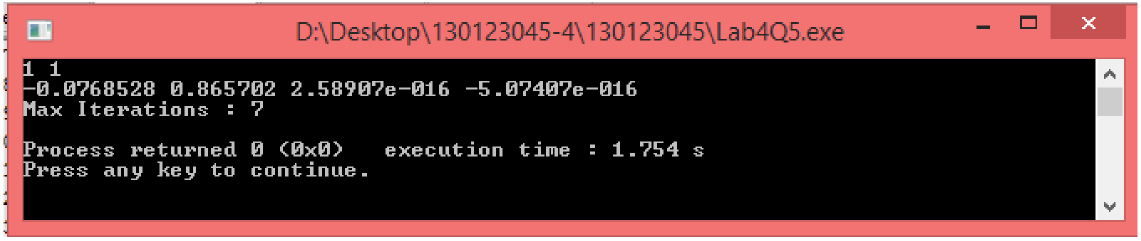
(a)



```
1 1
-0.662359 -0.56228 -8.54893e-017 -6.3209e-017
Max Iterations : 12

Process returned 0 (0x0)   execution time : 6.260 s
Press any key to continue.
```

(b)



```
1 1
-0.0768528 0.865702 2.58907e-016 -5.07407e-016
Max Iterations : 7

Process returned 0 (0x0)   execution time : 1.754 s
Press any key to continue.
```

EXPLANATION/RESULT

(a) Complex root : -0.662359-0.56228i

(b) Complex root : -0.076852+0.86570i