Rajshahi University of Engineering & Technology

CSE 2104: Sessional Based on CSE 2103

Lab Report 03

Dated: 28.02.18

Submitted to

Shyla Afroge

Assistant Professor

Dept. of Computer Science & Engineering

&

Instructor, CSE 2104

Submitted by

Fuad Al Abir

Roll: 1603021

Section: A

Dept. of Computer Science & Engineering

Problem#01: Determine the root of the equation $x^3-2x-5=0$ by Newton-Raphson Method

THEORY: The main formula to determining the root using Newton Raphson method is -

$$X_{n+1} = X_n - f(X_n)/f'(X_n)$$

Where, $f(X_n)$ is the given function, $f'(X_n)$ is the first derivative and n = 0, 1, 2, 3… This calculation is continued autonomously till the tolerated error level.

```
#include <iostream>
#include <cmath>
#include <iomanip>

using namespace std;

double f(double x)
{
    return x*x*x - 2*x - 5;
}

double fd(double x)
{
    return 3*x*x - 2;
}

double newRaph(double x)
{
    return x - f(x)/fd(x);
}

int main()
{
    double xo = 1, x1, x2;
    int n = 0;
    while(1)
    {
        x1 = newRaph(xo);
        x2 = newRaph(x1);
        cout << n << "\t" << fixed << setprecision(10) << xo << "\t "
<< x1 << "\t " << x2 << endl;
        if(fabs(x1 - x2) < .000001)
        {
            cout << "\nRoot: " << x2 << endl;
            break;
        }
        xo = x1;
        x1 = x2;
        n++;
    }
}
```

OUTPUT:

```
0       1.0000000000     7.0000000000    4.7655172414
1       7.0000000000     4.7655172414    3.3487027595
2       4.7655172414     3.3487027595    2.5315996410
3       3.3487027595     2.5315996410    2.1739158849
4       2.5315996410     2.1739158849    2.0978836864
5       2.1739158849     2.0978836864    2.0945577159
6       2.0978836864     2.0945577159    2.0945514816
7       2.0945577159     2.0945514816    2.0945514815

Root: 2.0945514815
```

DISCUSSION: Though the tollerence level is 0.000001, using this method, determining the root is faster than the previous methods as it takes only 7 iterations.

Problem#02: Determine the root of the equation $x^3-6x^2+11x-6=0$ by Ramanujan's Method

THEORY: All the co-efficient of a polynomial equation (powered upto 3) is given as input and the lowest root to the initial guess is determined by Ramanujan'n Method. The algorithm of the method is to rearrange the equation so that the first element of the equation must be 1 and the followings are noted as a1, a2, a3 etc. Then, b1, b2, b3 and so on is declared to be the ratio of the a's and those were taken as root of the equation as if the final one satisfies the tolerance level of error.

```
#include <iostream>
#include <cmath>
#include <iomanip>

using namespace std;

double arr1[100];
double arr2[100];
double DIV[100];

double find_B(int n)
{
    double sum=0.0;
    for(int i=1;i<n;i++)
    {
        sum = sum + (1.0*arr1[i]*arr2[n-i]);
    }
    return sum;
}

void div()
{
    cout<<"\nn\t DIV\n"<<endl;
    for(int i=1;i<10;i++)
    {
        DIV[i]=(arr2[i]/(1.0*arr2[i+1]));
    }
    cout<<"\n";
}

int main()
{
    double x,a,b,c,d;
    double fx;
    cout<<"Equation's Co-efficients: ";

    cin>> a >> b >>  c >> d;
```

```
    arr1[1]= c/(-d);
    arr1[2]= b/(-d);
    arr1[3]= a/(-d);

    arr2[1]=1;

    for(int i=2;i<12;i++)
    {
        arr2[i]=find_B(i);
    }

    div();
    int i;
    for(i=1;i<=10;i++)
    {
        if(fabs(DIV[i]-DIV[i+1]) > 0.0001 && DIV[i+1])
        {
            cout<<endl;
            cout<<i<<"\t "<<DIV[i]<<endl;
        }
        else
            break;
    }
    cout << "\n\nRoot is: ";
    cout << fixed << setprecision(6)<<DIV[i]<<endl;
    return 0;

}
```

TERMINAL:

```
Equation's Co-efficients: 1 -6 11 -6

n       DIV

1        0.545455
2        0.776471
3        0.886957
4        0.942365
5        0.970616
6        0.985063
7        0.992434
8        0.996181

Root is: 0.998077
```

DISCUSSION: The is some ambiguity determining the b's and some arrays are taken as to calculate them separately. The algorithm used in the program is not an effitient one though it works properly.