

The Newton Raphson Method

Theory

This method is generally used to obtain the improved result than the Bisection, False position or Iteration method. The method starts with a function f defined over the real numbers x , the function's derivative f' , and an initial guess x_0 for a root of the function f . If the function satisfies the assumptions made in the derivation of the formula and the initial guess is close, then a better approximation x_n is

$$X_n = X_{n-1} - \frac{F(X_{n-1})}{F'(X_{n-1})}$$

The process is repeated as until a sufficiently accurate value is reached.

Code

```
#include<bits/stdc++.h>
using namespace std;
#define error 0.000001

double f(double x)
{
    return (3*x-1-cos(x));
}

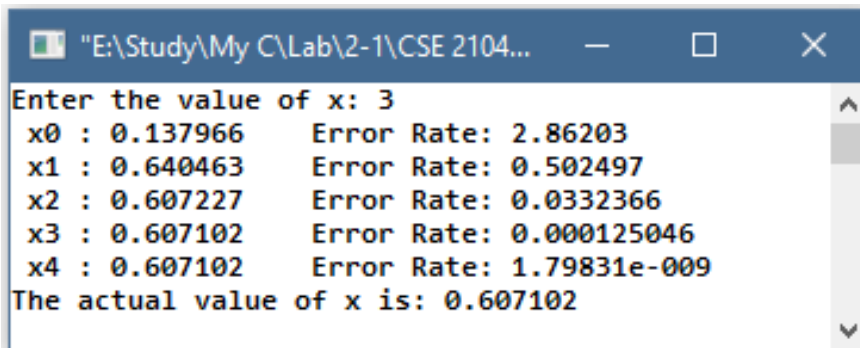
double fdiff(double x)
{
    return (3+sin(x));
}

void NewtonRaphson(double x)
{
    int i=0;
    double m,xn;
    do
    {
        m = x;
        xn = x - ( (f(x))/fdiff(x) );
        x = xn;
        cout<<" x"<<i<<" : "<<xn<<"   Error Rate: "<<fabs(xn-m)<<endl;
        i++;
    }while(fabs(xn-m)>=error);
    cout<<"The actual value of x is: "<<xn<<endl;
}
```

```
int main()
{
    double x;
    cout<<"Enter the value of x: ";
    cin>>x;
    NewtonRaphson(x);

    return 0;
}
```

Output



```
"E:\Study\My C\Lab\2-1\CSE 2104..."
Enter the value of x: 3
x0 : 0.137966      Error Rate: 2.86203
x1 : 0.640463      Error Rate: 0.502497
x2 : 0.607227      Error Rate: 0.0332366
x3 : 0.607102      Error Rate: 0.000125046
x4 : 0.607102      Error Rate: 1.79831e-009
The actual value of x is: 0.607102
```

Discussion

Here in the above code, two functions named `f()` and `fdiff()` were declared that returned double type value of the function for the parameter `x`. Then in the `main()` function, in a do while loop, the general formula of Newton-Raphson method was applied until the value of root `x` had the error rate of 0.000001.

The Ramanujan Method

Theory

Srinivasa Ramanujan (1887 – 1920) described an iterative method which can be used to determine the smallest root of the equation $f(x) = 0$ where $f(x)$ is the form of,

$$f(x) = 1 - (a_1x + a_2x^2 + a_3x^3 + a_4x^4 + \dots) \dots\dots(1)$$

for smaller values of x , it can be written as,

$$(1 - (a_1x + a_2x^2 + a_3x^3 + a_4x^4 + \dots))^{-1} = (b_1 + b_2x + b_3x^2 + \dots) \dots\dots(2)$$

Expanding the left side of the equation (2) TO binomial theorem, it can be obtained that,

$$1 + (a_1x + a_2x^2 + a_3x^3 + \dots) + (a_1x + a_2x^2 + a_3x^3 + \dots)^2 + \dots = b_1 + b_2x + b_3x^2 + \dots \dots\dots(3).$$

Comparing the co-efficients of like powers of x on both sides of equation no. (3), it is obtained that,

$$b_1 = 1,$$

$$b_2 = a_1 = a_1b_1,$$

$$b_3 = a_1b_2 + a_2b_1,$$

$$b_4 = a_1b_3 + a_2b_2 + a_3b_1,$$

.

.

$$b_n = a_1b_{n-1} + a_2b_{n-2} + \dots + a_{n-1}b_1 \dots\dots(4); n = 2,3,4,\dots$$

Without any proof Ramanujan states that, the successive convergents, (b_n/b_{n+1}) is the root of equation $f(x) = 0$.

Code

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

void Ramanujan(double a1,double a2,double a3)
{
    double a[100],b[100];
    a[0]=1.0;
    a[1]=a1;
    a[2]=a2;
    a[3]=a3;
```

```

for(int i=4;i<38;i++)
    a[i]=0;

b[0]=1.0;
b[1]=1.0;

int i,j,k,x=1,y=2,m,count=0;
i=1,j=1;
int n=2;
double result,temp,error,sum=0;

cout<<"b"<<"1"<<" = "<<b[1]<<endl;

while(n<18)
{
    count++;
    i=1,j=1;
    while(1)
    {
        sum += a[j]*b[n-i];
        if((n-i)==1 && j==(n-1)){
            break;
        }
        i++;
        j++;
    }
    if((n-i)==1 && j==(n-1)){
        b[n]=sum;

        cout<<"b"<<n<<" = "<<b[n]<<endl;
        n++;
        sum=0;
    }
    result = (double)(b[x]/b[y]);

    if(count==1)
        temp=result;
    else
        error = abs(result-temp);

    cout<<"b"<<x<<"/"<<"b"<<y<<" = "<<result<<" Error: "<<error<<endl;

    temp = result;
    m=y;
    x=y;
    y=m+1;
}

```

```

        if(error==0.00001)
            break;
    }
    cout<<"\nThe Smallest Root is: "<<result<<endl;
}

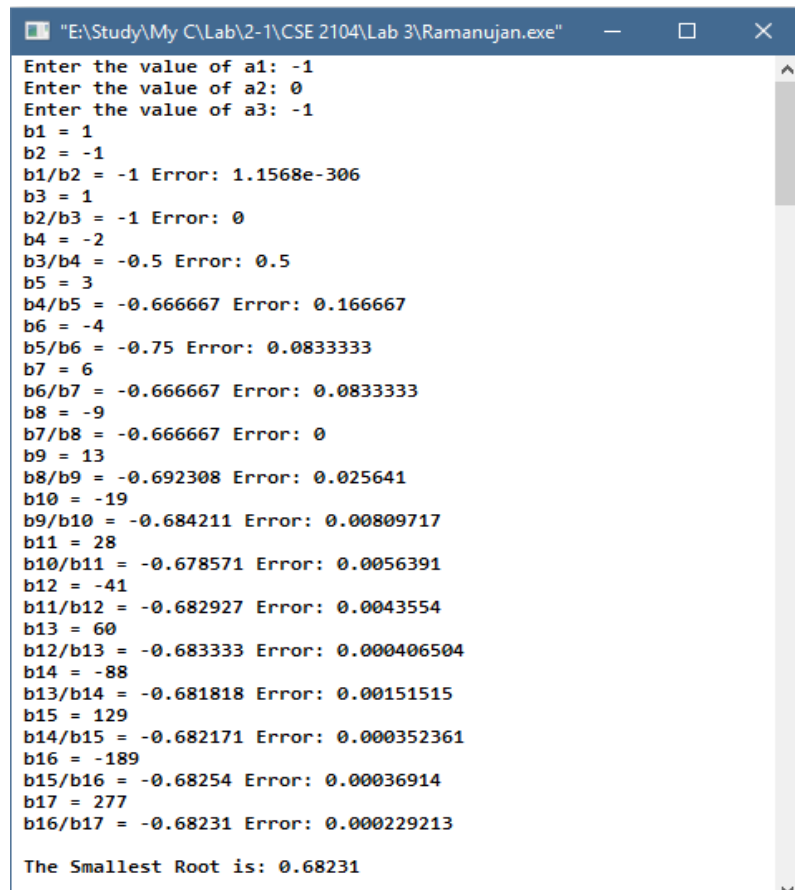
int main()
{
    double a1,a2,a3;
    cout<<"Enter the value of a1: ";
    cin>>a1;
    cout<<"Enter the value of a2: ";
    cin>>a2;
    cout<<"Enter the value of a3: ";
    cin>>a3;

    Ramanujan(a1,a2,a3);

    return 0;
}

```

Output



```

E:\Study\My C\Lab\2-1\CSE 2104\Lab 3\Ramanujan.exe
Enter the value of a1: -1
Enter the value of a2: 0
Enter the value of a3: -1
b1 = 1
b2 = -1
b1/b2 = -1 Error: 1.1568e-306
b3 = 1
b2/b3 = -1 Error: 0
b4 = -2
b3/b4 = -0.5 Error: 0.5
b5 = 3
b4/b5 = -0.666667 Error: 0.166667
b6 = -4
b5/b6 = -0.75 Error: 0.0833333
b7 = 6
b6/b7 = -0.666667 Error: 0.0833333
b8 = -9
b7/b8 = -0.666667 Error: 0
b9 = 13
b8/b9 = -0.692308 Error: 0.025641
b10 = -19
b9/b10 = -0.684211 Error: 0.00809717
b11 = 28
b10/b11 = -0.678571 Error: 0.0056391
b12 = -41
b11/b12 = -0.682927 Error: 0.0043554
b13 = 60
b12/b13 = -0.683333 Error: 0.000406504
b14 = -88
b13/b14 = -0.681818 Error: 0.00151515
b15 = 129
b14/b15 = -0.682171 Error: 0.000352361
b16 = -189
b15/b16 = -0.68254 Error: 0.00036914
b17 = 277
b16/b17 = -0.68231 Error: 0.000229213

The Smallest Root is: 0.68231

```

Discussion

Here in the above code, a function named Ramanujan() was declared where a1,a2,a3 went as parameters denoting the co-efficient of the x, x^2, x^3 respectively. In that very function, the equation no (4) from theory section was performed using loop and conditional logic. Thus the error rate was shown in output along with the roots. And finally the smallest root was acquired and shown in the output.