Haven's Light is Our Guide

Rajshahi University of Engineering & Technology

Department of Computer Science & Engineering

Lab Report

Course Code:	CSE 2204
Course Title:	Numerical Methods Sessional
Experiment No:	02
Experiment Name:	Solving Equations using Iteration and Newton-Rapson Method

Date:

January 14, 2024

Submitted By:	Submitted To:
Name: Md. Abdullah Al Mamun	Shyla Afroge
Section: A	Assistant Professor
Roll No: 2003028	Computer Science & Engineering
Year: 2nd Year Odd Semester	Rajshahi University of Engineering & Technology

Experiment No: 02

Experiment Name: **Solving Equations using Iteration and Newton-Rapson Method**

Theory:

Iteration Method:

Iteration is a fundamental concept in mathematics and numerical methods, involving the process of repeatedly applying a rule or procedure to achieve a desired outcome. In the context of numerical analysis, iteration is commonly used to approximate solutions to equations, particularly when analytical solutions are challenging or impossible to find. The iterative process starts with an initial guess and refines it through successive iterations until a sufficiently accurate solution is obtained.

Algorithm:

1. Initial Guess: Begin with an initial guess or estimate for the solution.
2. Iterative Process: Apply a rule or procedure to refine the estimate through successive iterations.
3. Convergence Criteria: Continue the iterations until a specified convergence criterion is met, indicating that the solution is sufficiently accurate.

Newton-Raphson Method:

The Newton-Raphson Method is a powerful iterative technique for finding successively better approximations to the roots of a real-valued function. It is based on linear approximation and tangent lines. Starting with an initial guess x_0, the method iteratively refines the approximation using the formula $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, where $f(x)$ is the function and $f'(x)$ is its derivative. The method converges rapidly when the initial guess is close to the actual root, making it an efficient tool for root-finding. However, it may fail to converge or converge to an undesired root under certain conditions or if the function has peculiar characteristics. Despite these considerations, the Newton-Raphson Method is widely utilized in various fields for its speed and effectiveness in finding solutions to nonlinear equations.

Algorithm:

1. Initial Guess: Start with an initial guess x_0 for the root.
2. Iterative Formula: Apply the iterative formula $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, where $f(x)$ is the function and $f'(x)$ is its derivative.
3. Successive Refinement: Repeat the iterative process to successively refine the approximation to the root.
4. Convergence Criteria: Continue the iterations until a specified convergence criterion is met, indicating that the root approximation is sufficiently accurate.

Program:

Listing 1: Newton-Rapson Method

```
double newtonRapsonMethod(double x, double error)
{
    double x1 = x - function(x) / functionDerivative(x);
    double x2 = x1 - function(x1) / functionDerivative(x1);

    cout << "x0: " << x << "\tx1: " << x1 << "\n";

    int i = 1;
    while (abs(x2 - x1) > error)
    {
        x1 = x2;
        x2 = x1 - function(x1) / functionDerivative(x1);
        cout << "x" << i << ": " << x1 << "\tx" << (i + 1) << ": " <<
            x2 << "\n";
        i++;
    }

    return x2;
}
```

Listing 2: Iteration Method

```
double phiFunction(double x, int coeff[], int degree)
{
    double sum = 0;
    for (int i = 0; i <= degree; i++)
    {
        sum += coeff[i] * pow(x, i);
    }
    return sum;
}

double iterarionMethod(double x, double error)
{
    int degree;
    cout << "Enter degree for phi function: ";
    cin >> degree;
    int coeff[degree + 1];
    cout << "Enter " << degree + 1 << " Coefficients: ";
    for (int i = 0; i <= degree; i++)
        cin >> coeff[i];

    double x1 = phiFunction(x, coeff, degree);
    double x2 = phiFunction(x1, coeff, degree);

    while (abs(x2 - x1) > error)
    {
        x1 = x2;
        x2 = phiFunction(x1, coeff, degree);
        cout << "x1: " << x1 << "\tx2: " << x2 << "\n";
    }

    return x2;
}
```

Listing 3: functionInput.h helper file

```
#ifndef FUNCTIONINPUT_H
#define FUNCTIONINPUT_H
#include <iostream>
#include <cmath>
```

```
5
6       using namespace std;
7
8       int degree;
9       int *coefficients = new int[degree + 1];
10
11      void takeInputForFunction()
12      {
13          int deg;
14          cout << "Enter the degree of the polynomial: ";
15          cin >> deg;
16          int *coef = new int[deg + 1];
17          cout << "Enter " << deg + 1 << " coefficients of the polynomial: ";
18          for (int i = 0; i <= deg; i++)
19              cin >> coef[i];
20          delete[] coefficients; // free the previously allocated memory
21          coefficients = coef;
22          degree = deg;
23      }
24
25      double function(double x)
26      {
27          double sum = 0;
28          for (int i = 0; i <= degree; i++)
29          {
30              sum += coefficients[i] * pow(x, i);
31          }
32          return sum;
33      }
34
35      double functionDerivative(double x)
36      {
37          double sum = 0;
38          for (int i = 1; i <= degree; i++)
39              sum += i * coefficients[i] * pow(x, i - 1);
40          return sum;
41      }
42
43      #endif // !FUNCTIONINPUT_H
```

Listing 4: Main Program

```
1       #include <iostream>
2       #include "functionInput.h"
3       using namespace std;
4
5       int main()
6       {
7           takeInputForFunction();
8           double a;
9
10          cout << "Enter guess: ";
11          cin >> a;
12
13          cout << "\n1. Newton Rapson Method\n2. Iteration Method\n";
14          cout << "Choose the method you want to use: ";
15
16          int choice;
17          cin >> choice;
18          switch (choice)
19          {
20              {
21              case 1:
22              {
23                  double sol = newtonRapsonMethod(a, 0.0001);
24                  cout << "The solution using Newton-Rapson Method: " << sol
```

```
                    << endl;
                break;
            }

            case 2:
            {
                double sol = iterarionMethod(a, 0.0001);
                cout << "The solution using Iteration Method: " << sol <<
                    endl;
                break;
            }

            default:
                break;
            }
        }
    }
```

Result:

The Newton-Raphson Method and the Iteration Method are both iterative techniques used for approximating solutions to mathematical problems. The Newton-Raphson Method utilizes function derivatives for rapid convergence, especially when initial guesses are close to roots, but it may be sensitive to specific conditions. In contrast, the Iteration Method is a more general approach, suitable for a wide range of functions, although it tends to converge more slowly. The choice between these methods depends on the specific characteristics of the problem and the quality of the initial guess.

```
┌──(shohag㉿Shohag-Ubuntu)-[~/Documents/LABs/CSE-2204]
└─$ cd "/home/shohag/Documents/LABs/CSE-2204/" && g++ lab02.cpp -o lab02 && "/home/shohag/Documents/LABs/CSE-2204/"lab02
Enter the degree of the polynomial: 2
Enter 3 coefficients of the polynomial: 1 -2 1
Enter guess: 0.6

1. Newton Rapson Method
2. Iteration Method
Choose the method you want to use:1
x0: 0.6 x1: 0.8
x1: 0.9 x2: 0.95
x2: 0.95        x3: 0.975
x3: 0.975       x4: 0.9875
x4: 0.9875      x5: 0.99375
x5: 0.99375     x6: 0.996875
x6: 0.996875    x7: 0.998438
x7: 0.998438    x8: 0.999219
x8: 0.999219    x9: 0.999609
x9: 0.999609    x10: 0.999805
x10: 0.999805   x11: 0.999902
The solution using Newton-Rapson Method: 0.999902
```

Figure 1: Output of the Program