RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Lab report: 07

Date of Experiment: 01.04.2018
Date of Submission: 08.04.2018

Submitted to:
Shyla Afroge
Assistant Professor,
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Submitted by:
Riyad Morshed Shoeb
Roll No: 1603013
Section: A
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Name of the Experiment: Implementation of Numerical Differentiation

Theory:

Considering Newton's Forward difference formula-

$$y_n(x) = y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots, where, x = x_0 + ph$$

Then,

$$\frac{dy}{dx} = \frac{dy}{dp}.\frac{dp}{dx} = \frac{1}{h}\left(\Delta y_0 + \frac{2p-1}{2}\Delta^2 y_0 + \frac{3p^2-6p+2}{6}\Delta^3 y_0 + \cdots\right)$$

Differentiating again, we get,

$$\frac{d^2y}{dx^2} = \frac{1}{h^2}\left(\Delta^2 y_0 + \frac{6p-6}{6}\Delta^3 y_0 + \frac{12p^2-36p+22}{24}\Delta^4 y_0 + \cdots\right)$$

This formula can be used to compute the value of $\frac{dy}{dx}$ and $\frac{d^2y}{dx^2}$ for non-tabular values of x.
For tabular values, we set,

$$x = x_0 \quad and\ hence, p = 0$$

Hence, the formula becomes,

$$\left[\frac{dy}{dx}\right]_{x=x_0} = \frac{1}{h}\left[\Delta y_0 - \frac{1}{2}\Delta^2 y_0 + \frac{1}{3}\Delta^3 y_0 - \frac{1}{4}\Delta^4 y_0 + \cdots\right]\ and$$

$$\left[\frac{d^2y}{dx^2}\right]_{x=x_0} = \frac{1}{h}\left[\Delta^2 y_0 - \Delta^3 y_0 + \frac{11}{12}\Delta^4 y_0 + \cdots\right]$$

Again, using Newton's backward difference formula,

$$\left[\frac{dy}{dx}\right]_{x=x_n} = \frac{1}{h}\left(\nabla y_n + \frac{1}{2}\nabla^2 y_n + \frac{1}{3}\nabla^3 y_n + \cdots\right)$$

$$\left[\frac{d^2y}{dx^2}\right]_{x=x_n} = \frac{1}{h}\left(\nabla^2 y_n + \nabla^3 y_n + \frac{11}{12}\nabla^4 y_n + \frac{5}{6}\nabla^5 y_n + \cdots\right)$$

Code:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cmath>
using namespace std;

int main(void)
{
    int i,k,n;
    int j=1;
    double h,sum1,sum2;
    double x_check;
    double x[10];
    double y[10];
    double d_y[10],d2_y[10],d3_y[10],d4_y[10],d5_y[10],d6_y[10];

    printf("Enter the length of your data: ");
    cin>>n;
    printf("Enter the values:\n x | y\n");
    for(i=0;i<n;i++)
        cin>>x[i]>>y[i];
    h=x[1]-x[0];
```

```
printf("Forward difference table\n");
printf("\ndy\n");
for(i=0;i<n-j;i++)
{
    d_y[i]=y[i+1]-y[i];
    cout<<d_y[i];
    printf("\n");
}
j++;

printf("\nd2_y\n");
for(i=0;i<n-j;i++)
{
    d2_y[i]=d_y[i+1]-d_y[i];
    cout<<d2_y[i];
    printf("\n");
}
j++;

printf("\nd3_y\n");
for(i=0;i<n-j;i++)
{
    d3_y[i]=d2_y[i+1]-d2_y[i];
    cout<<d3_y[i];
    printf("\n");
}
j++;

printf("\nd4_y\n");
for(i=0;i<n-j;i++)
{
    d4_y[i]=d3_y[i+1]-d3_y[i];
    cout<<d4_y[i];
    printf("\n");
}
j++;

printf("\nd5_y\n");
for(i=0;i<n-j;i++)
{
    d5_y[i]=d4_y[i+1]-d4_y[i];
    cout<<d5_y[i];
    printf("\n");
}
j++;

printf("\nd6_y\n");
for(i=0;i<n-j;i++)
{
    d6_y[i]=d5_y[i+1]-d5_y[i];
    cout<<d6_y[i];
    printf("\n");
}

printf("Enter a value of x to check: ");
cin>>x_check;
```

```
    for(i=0;i<n;i++)
    {
        if(x_check==x[i])
            break;
    }

    k=i;
    sum1=(1/h)*(d_y[k]-((d2_y[k])/2)+((d3_y[k])/3)-
((d4_y[k])/4)+((d5_y[k])/5));
    sum2=(1/(h*h))*(d2_y[k]-d3_y[k]+(.9167*(d4_y[k]))-
(.833*(d5_y[k])));
    printf("dy/dx: ");
    cout<<sum1<<endl;
    printf("d2y/dx2: ");
    cout<<sum2<<endl;
    return 0;
}
```

Output:

```
"D:\2nd year odd sem\CSE 2104\Lab_7\Newton's_Forward_Difference_Differentiation.exe"
Enter the length of your data: 7
Enter the values:
 x | y
1.0 2.7183
1.2 3.3201
1.4 4.0552
1.6 4.9530
1.8 6.0496
2.0 7.3891
2.2 9.0250
Forward difference table

dy
0.6018
0.7351
0.8978
1.0966
1.3395
1.6359

d2_y
0.1333
0.1627
0.1988
0.2429
0.2964

d3_y
0.0294
0.0361
0.0441
0.0535

d4_y
0.0067
0.008
0.0094

d5_y
0.0013
0.0014

d6_y
0.0001
Enter a value of x to check: 1.2
dy/dx: 3.32032
d2y/dx2: 3.31919

Process returned 0 (0x0)   execution time : 125.299 s
Press any key to continue.
```

Name of the Experiment: Implementation of Numerical method to calculate maximum and minimum values of a tabulated function.

Theory:

It is known that the maximum and minimum values of a function can be found by equating the first derivative to zero and solving for the variable. The same procedure can be applied to determine the maxima and minima of a tabulated function.
Considering Newton's Forward difference formula-

$$y_n(x) = y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots, where, x = x_0 + ph$$

Differentiating this with respect to p, we get,

$$\frac{dy}{dx} = \frac{dy}{dp}.\frac{dp}{dx} = \frac{1}{h}\left(\Delta y_0 + \frac{2p-1}{2}\Delta^2 y_0 + \frac{3p^2-6p+2}{6}\Delta^3 y_0 + \cdots\right)$$

For maxima and minima $\frac{dy}{dp} = 0$. Hence, terminating the right hand side, for simplicity, after the third difference and equating it to zero, we obtain the quadratic for p,

$$c_0 + c_1 p + c_2 p^2$$

Where,

$$c_0 = \Delta y_0 - \frac{1}{2}\Delta^2 y_0 + \frac{1}{3}\Delta^3 y_0$$
$$c_1 = \Delta^2 y_0 - \Delta^3 y_0$$
$$c_2 = \frac{1}{2}\Delta^3 y_0$$

Code:
```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cmath>
#include<algorithm>
#include<cstring>
using namespace std;

int main(void)
{
    int i,n,c,a;
    int j=1;
    double b;
    double x[10];
    double y[10];
    double d_y[10],d2_y[10],d3_y[10],d4_y[10];
    double minima;
    double maxima;
    double h,p,p1;
    double xx;
    printf("Enter the length of your data: ");
    cin>>n;
    printf("Enter the values:\n x | y\n");
    for(i=0;i<n;i++)
        cin>>x[i]>>y[i];
    h=x[1]-x[0];
    printf("\nDifference table\n");
    printf("dy\n");
    for(i=0;i<n-j;i++)
```

```
    {
        d_y[i]=y[i+1]-y[i];
        cout<<d_y[i];
        printf("\n");
    }
    j++;
    printf("\nd2_y\n");
    for(i=0;i<n-j;i++)
    {
        d2_y[i]=d_y[i+1]-d_y[i];
        cout<<d2_y[i];
        printf("\n");
    }
    j++;
    printf("\nd3_y\n");
    for(i=0;i<n-j;i++)
    {
        d3_y[i]=d2_y[i+1]-d2_y[i];
        cout<<d3_y[i];
        printf("\n");
    }
    p=0.5-(d_y[0]/d2_y[0]);
    xx=x[0]+p*h;
    printf("Maximum value of the tabulated function will be found for
the value of x= ");
    cout<<xx<<endl;
    maxima=y[0]+p*d_y[0]+(p*(p-1)*d2_y[0]*0.5)+((p*(p-1)*(p-
2)*d3_y[0])/6);
    printf("And the maximum value will be: ");
    cout<<maxima<<endl;
}
```

Output:

```
"D:\2nd year odd sem\CSE 2104\Lab_7\Maxima.exe"
Enter the length of your data: 5
Enter the values:
 x | y
1.2 0.9320
1.3 0.9636
1.4 0.9855
1.5 0.9975
1.6 0.9996

Difference table
dy
0.0316
0.0219
0.012
0.0021

d2_y
-0.0097
-0.0099
-0.0099

d3_y
-0.0002
0
Maximum value of the tabulated function will be found for the value of x= 1.57577
And the maximum value will be: 0.999877

Process returned 0 (0x0)   execution time : 12.126 s
Press any key to continue.
```