Heaven's Light is Our Guide

Rajshahi University of Engineering & Technology

Department of Computer Science & Engineering

Lab Report

Topic : Solution of Algebraic Equation

Course Title : Sessional Based on CSE 2103
Course Code : CSE 2104

Submitted By :

Md. Tanzid Hasan
Section : A
Roll No : 1603054

Submitted To :

Shyla Afroge

Assistant Professor

Dept. of Computer Science & Engineering

Date : 18-03-2018

Experiment No : 01

Experiment Name : Interpolation

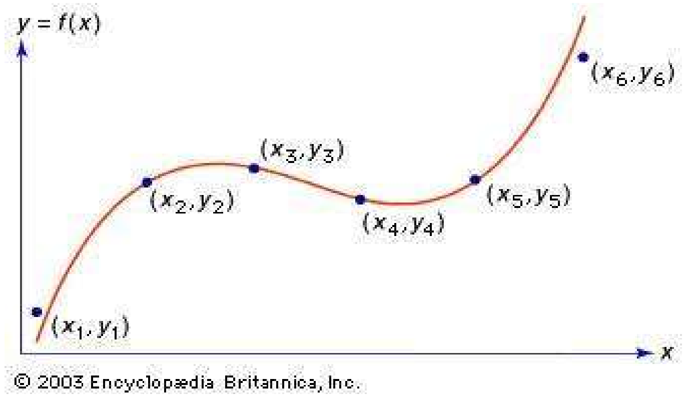
Theory :

In the mathematical field of numerical analysis, interpolation is a method of constructing new data points within the range of a discrete set of known data points.

In engineering and science, one often has a number of data points, obtained by sampling or experimentation, which represent the values of a function for a limited number of values of the independent variable. It is often required to interpolate (i.e., estimate) the value of that function for an intermediate value of the independent variable.

Interpolation, in mathematics, the determination or estimation of the value of $f(x)$, or a function of x, from certain known values of the function. If $x_0 < \ldots < x_n$ and $y_0 = f(x_0), \ldots, y_n = f(x_n)$ are known, and if $x_0 < x < x_n$, then the estimated value of $f(x)$ is said to be an interpolation. If $x < x_0$ or $x > x_n$, the estimated value of $f(x)$ is said to be an extrapolation.
If $x_0, \ldots, xn$ are given, along with corresponding values $y_0, \ldots, y_n$ (see the figure), interpolation may be regarded as the determination of a function $y = f(x)$ whose graph passes through the $n + 1$ points, (x_i, y_i) for $i = 0, 1, \ldots, n$. There are infinitely many such functions, but the simplest is a polynomial interpolation function $y = p(x) = a_0 + a_1x + \ldots\ldots\ldots\ldots + a_nx_n$ with constant a_i's such that $p(x_i) = y_i$ for $i = 0, \ldots\ldots, n$. There is exactly one such interpolating polynomial of degree n or less. If the xi's are equally spaced, say by some factor h, then the following formula of Isaac Newton produces a polynomial function that fits the data :

$$f(x)=a_0+a_1(x - x_0)/h + a_2(x - x_0)(x - x_1)/2!h^2 + \ldots + an(x - x_0)\cdots(x - x_{n-1})/n!hn$$



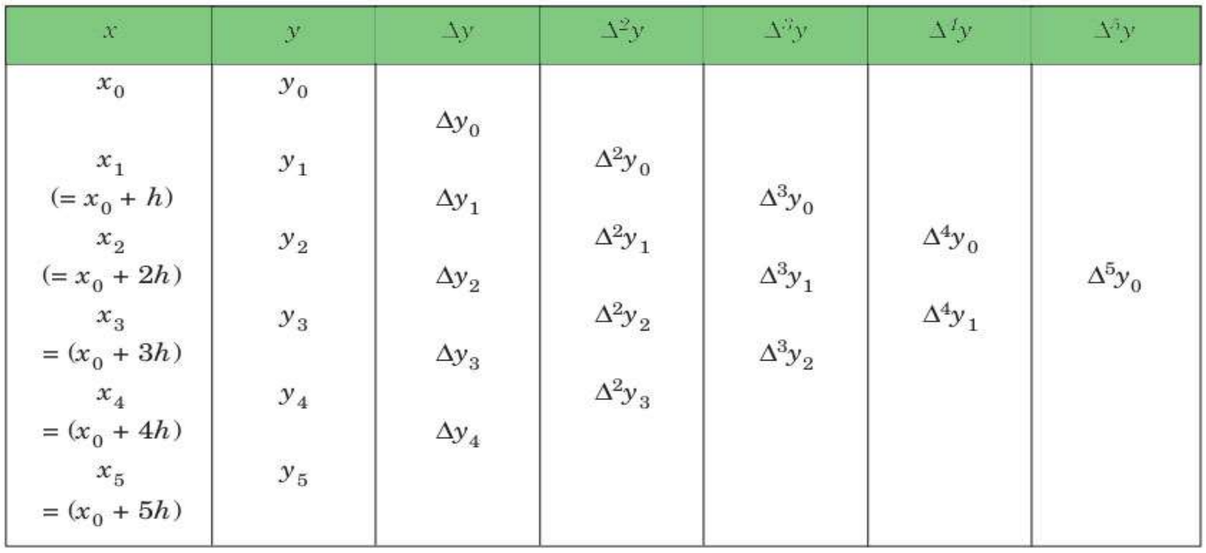
Polynomial approximation is useful even if the actual function $f(x)$ is not a polynomial, for the polynomial $p(x)$ often gives good estimates for other values of $f(x)$.

There are two types in interpolation method. Such as, 1. Forward, 2. Backward.

1. Forward interpolation method :

The differences $y_1 - y_0$, $y_2 - y_1$, $y_3 - y_2$, ……, $y_n - y_n–1$ when denoted by dy_0, dy_1, dy_2, ……, dy_{n-1} are respectively, called the first forward differences. Thus the first forward differences are :

Forward difference table

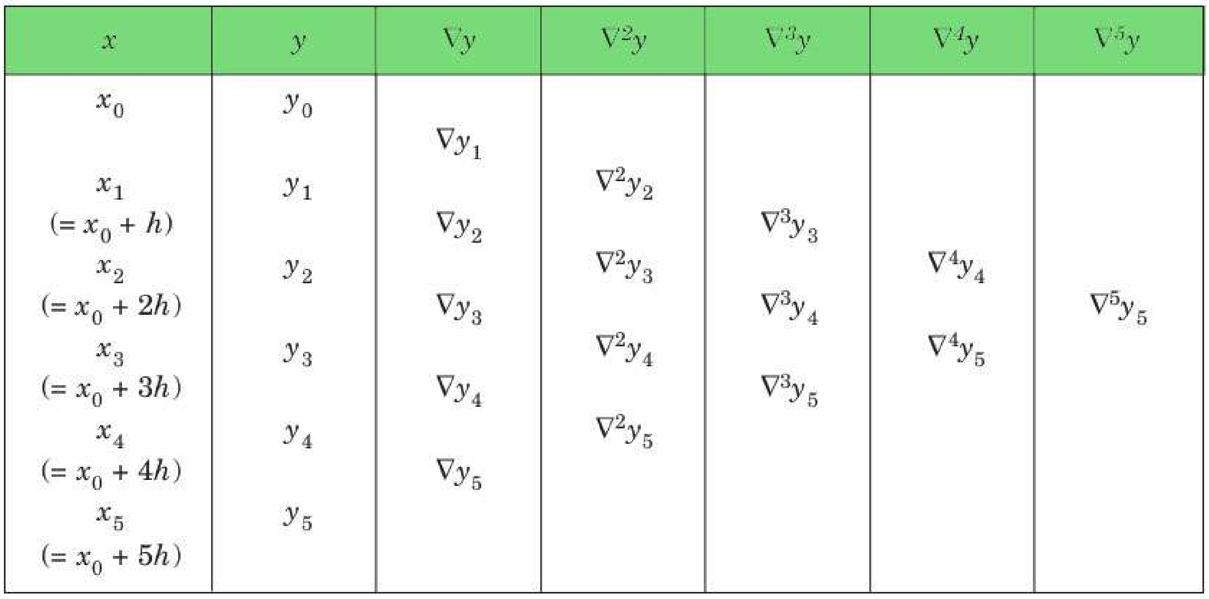


x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
x_0	y_0					
		Δy_0				
x_1 $(= x_0 + h)$	y_1		$\Delta^2 y_0$			
		Δy_1		$\Delta^3 y_0$		
x_2 $(= x_0 + 2h)$	y_2		$\Delta^2 y_1$		$\Delta^4 y_0$	
		Δy_2		$\Delta^3 y_1$		$\Delta^5 y_0$
x_3 $= (x_0 + 3h)$	y_3		$\Delta^2 y_2$		$\Delta^4 y_1$	
		Δy_3		$\Delta^3 y_2$		
x_4 $= (x_0 + 4h)$	y_4		$\Delta^2 y_3$			
		Δy_4				
x_5 $= (x_0 + 5h)$	y_5					

2. Backward interpolation method :

The differences $y_1–y_0$, $y_2–y_1$, ……, $y_n–y_{n-1}$ when denoted by dy_1, dy_1, ……, dy_n, respectively, are called first backward difference. Thus the first backward differences are :

Backward difference table



x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$	$\nabla^5 y$
x_0	y_0					
		∇y_1				
x_1 $(= x_0 + h)$	y_1		$\nabla^2 y_2$			
		∇y_2		$\nabla^3 y_3$		
x_2 $(= x_0 + 2h)$	y_2		$\nabla^2 y_3$		$\nabla^4 y_4$	
		∇y_3		$\nabla^3 y_4$		$\nabla^5 y_5$
x_3 $(= x_0 + 3h)$	y_3		$\nabla^2 y_4$		$\nabla^4 y_5$	
		∇y_4		$\nabla^3 y_5$		
x_4 $(= x_0 + 4h)$	y_4		$\nabla^2 y_5$			
		∇y_5				
x_5 $(= x_0 + 5h)$	y_5					

Code :

```
#include <bits/stdc++.h>
using namespace std;
int fact(int n)
{
   int multi=1;
   for(int i=1;i<=n;i++)
      multi=multi*i;
   return multi;
}

void inter_forward()
{
   int n;
   double m,h;
   cout<<"How many elements you want to input : ";
   cin>>n;
   double a[n];
   double b[n][n];
   double q[n];
   cout<<"Enter x and corresponding y : "<<endl;
   for(int i=0;i<n;i++)
   {
      cout<<"x"<<i<<" : ";
      cin>>a[i];
      cout<<"y"<<i<<" : ";
      cin>>b[0][i];
   }
   cout<<"For which x you want to determine the value : ";
```

```
    cin>>m;
    double sum=0;
    h=a[1]-a[0];
    double p=((m-a[0])*1.0)/h;
    for(int i=0;i<n;i++)
        for(int j=0;j<n-i;j++)
            b[i+1][j]=b[i][j+1]-b[i][j];
    q[0]=1;q[1]=p;
    double g=p;
    for(int i=2;i<n;i++)
    {
        g=g*(p-i+1);
        q[i]=(g*1.0)/fact(i);
    }
    for(int i=0;i<n;i++)
    {
        sum=sum+(q[i]*b[i][0]);
    }
    cout<<"Answer : "<<sum<<endl;
}

void inter_backward()
{
    int n;
    double m,h;
    cout<<"How many elements you want to input : ";
    cin>>n;
    double a[n];
    double b[n][n];
    double q[n];
    cout<<"Enter x and corresponding y : "<<endl;
    for(int i=0;i<n;i++)
    {
        cout<<"x"<<i<<" : ";
        cin>>a[i];
        cout<<"y"<<i<<" : ";
        cin>>b[0][i];
    }
    cout<<"For which x you want to determine the value : ";
    cin>>m;
    double sum=0;
    h=a[1]-a[0];
    double p=((m-a[n-1])*1.0)/h;
    for(int i=0;i<n;i++)
        for(int j=i;j<n;j++)
            b[i+1][j+1]=b[i][j+1]-b[i][j];
    q[0]=1;q[1]=p;
    double g=p;
    for(int i=2;i<n;i++)
    {
        g=g*(p-i+1);
        q[i]=(g*1.0)/fact(i);
    }
    for(int i=0;i<n;i++)
        sum=sum+(q[i]*b[i][n-1]);

    cout<<"Answer : "<<sum<<endl;
}
```

```
int main()
{
   int k;
   for(;;)
   {
      cout<<"1. Interpolation Forward Method."<<endl<<"2. Interpolation
Backward Method."<<endl<<"3. Exit."<<endl<<"    Enter the option(1-3) : ";
      cin>>k;
      if(k>3)
         cout<<"Invalid input."<<endl<<endl;
      if(k==3)
         break;
      switch(k)
      {
      case 1 :
            {
               cout<<endl<<"1. Interpolation Forward Method : "<<endl;
               inter_forward();
               cout<<endl<<endl;
            }break;
      case 2 :
            {
               cout<<endl<<"2. Interpolation Backward Method : "<<endl;
               inter_backward();
               cout<<endl<<endl;
            }
      }
   }
   return 0;}
```

Input & Output :

```
1. Interpolation Forward Method.
2. Interpolation Backward Method.
3. Exit.
   Enter the option(1-3) : 1

1. Interpolation Forward Method :
How many elements you want to input : 4
Enter x and corresponding y :
x0 : 1
y0 : 24
x1 : 3
y1 : 120
x2 : 5
y2 : 336
x3 : 7
y3 : 720
For which x you want to determine the value : 8
Answer : 990


1. Interpolation Forward Method.
2. Interpolation Backward Method.
3. Exit.
   Enter the option(1-3) : 2

2. Interpolation Backward Method :
How many elements you want to input : 4
Enter x and corresponding y :
```

```
x0 : 1
y0 : 24
x1 : 3
y1 : 120
x2 : 5
y2 : 336
x3 : 7
y3 : 720
For which x you want to determine the value : 8
Answer : 894


1. Interpolation Forward Method.
2. Interpolation Backward Method.
3. Exit.
   Enter the option(1-3) : 3
```

Discussion : This is a menu problem for interpolation. In this problem, two different type of interpolation was used. One is forward and other is backward. Forward method was followed as it was stated in the algorithm. Backward method was followed its algorithm. In curve fitting problems, the constraint that the interpolant has to go exactly through the data points is relaxed. It is only required to approach the data points as closely as possible (within some other constraints). This requires parameterizing the potential interpolants and having some way of measuring the error. In the simplest case this leads to least squares approximation.