**ASSIGNMENT ON CSE 2104**

**Newton's Interpolation Formula**

***Submitted By***

***Name :Alam***

***Roll :133027***

***Department : CSE***

***Section : A***

[**Newton's Interpolation Formula: Difference between the forward and the backward formula**](http://math.stackexchange.com/questions/624894/newtons-interpolation-formula-difference-between-the-forward-and-the-backward)

Here are the **formulas**:

Given a set of *k* + 1 data points

(x_0, y_0),\ldots,(x_k, y_k)

where no two *xj* are the same, the interpolation polynomial in the **Newton form** is a [linear combination](https://en.wikipedia.org/wiki/Linear_combination) of **Newton basis polynomials**

N(x) := \sum_{j=0}^{k} a_{j} n_{j}(x)

with the **Newton basis polynomials** defined as

n_j(x) := \prod_{i=0}^{j-1} (x - x_i)

for *j* > 0 and n_0(x) \equiv 1.

The coefficients are defined as

a_j := [y_0,\ldots,y_j]

where

[y_0,\ldots,y_j]

is the notation for [divided differences](https://en.wikipedia.org/wiki/Divided_differences).

Thus the **Newton polynomial** can be written as

![N(x) = [y_0] + [y_0,y_1](x-x_0) + \cdots + [y_0,\ldots,y_k](x-x_0)(x-x_1)\cdots(x-x_{k-1}).](data:image/png;base64,)

The **Newton Polynomial** above can be expressed in a simplified form when x_0, x_1, \dots, x_k are arranged consecutively with equal space. Introducing the notation h = x_{i+1}-x_i for each i=0,1,\dots,k-1 and x=x_0+sh, the difference x-x_i can be written as (s-i)h. So the **Newton Polynomial** above becomes:

\begin{align}
N(x) &= [y_0] + [y_0,y_1]sh + \cdots + [y_0,\ldots,y_k] s (s-1) \cdots (s-k+1){h}^{k} \\
&= \sum_{i=0}^{k}s(s-1) \cdots (s-i+1){h}^{i}[y_0,\ldots,y_i] \\
&= \sum_{i=0}^{k}{s \choose i}i!{h}^{i}[y_0,\ldots,y_i]
\end{align}

is called the **Newton Forward Divided Difference Formula**.

If the nodes are reordered as {x}_{k},{x}_{k-1},\dots,{x}_{0}, the **Newton Polynomial** becomes:

![N(x)=[y_k]+[{y}_{k}, {y}_{k-1}](x-{x}_{k})+\cdots+[{y}_{k},\ldots,{y}_{0}](x-{x}_{k})(x-{x}_{k-1})\cdots(x-{x}_{1})](data:image/png;base64,)

If {x}_{k},\;{x}_{k-1},\;\dots,\;{x}_{0} are equally spaced with x={x}_{k}+sh and {x}_{i}={x}_{k}-(k-i)h for *i* = 0, 1, ..., *k*, then,

\begin{align}
N(x) &= [{y}_{k}]+ [{y}_{k}, {y}_{k-1}]sh+\cdots+[{y}_{k},\ldots,{y}_{0}]s(s+1)\cdots(s+k-1){h}^{k} \\
&=\sum_{i=0}^{k}{(-1)}^{i}{-s \choose i}i!{h}^{i}[{y}_{k},\ldots,{y}_{k-i}]
\end{align}

is called the **Newton Backward Divided Difference Formula**.

code for forward:

#include<bits/stdc++.h>

using namespace std;

int fact(int n)

{

int i=0;

int sum=1;

for(i=2;i<=n;i++)

sum\*=i;

return sum;

}

int main()

{

freopen("input.txt","r",stdin);

double x[20]={0},y[20][20]={0},p,a,sum=0,arr[10],h,y0;

int i,j,n;

scanf("%d",&n);

for(i=0;i<n;i++){

scanf("%lf %lf",&x[i],&y[i][0]);

}

for(i=0;i<n;i++){

for(j=1;j<n-i; j++){

y[j-1][i+1]=y[j][i]-y[j-1][i];

}

}

printf("x- - - - - - - -y0- - - - - - -y1- - - - - - -y2- - - - - -y3- - - - - -y4\n");

for(i=0;i<n;i++){

printf("%.4lf\t",x[i]);

for(j=0;j<n-i; j++){

if(i==0)

{

arr[j]=y[i][j+1];

}

printf("%.4lf\t",y[i][j]);

}

printf("\n");

}

for(i=0;i<n-1;i++)

{

printf("%.4lf ",arr[i]);

}

h=x[1]-x[0];

y0=y[0][0];

printf("\n%.4lf ",h);

printf("\n%.4lf ",y0);

for(i=0;i<n-1;i++)

{

if(i%2==0)

sum=sum+(arr[i]/(i+1));

else

sum=sum+(arr[i]/(i+1)\*(-1));

}

printf("\nResult:%.4lf ",sum);

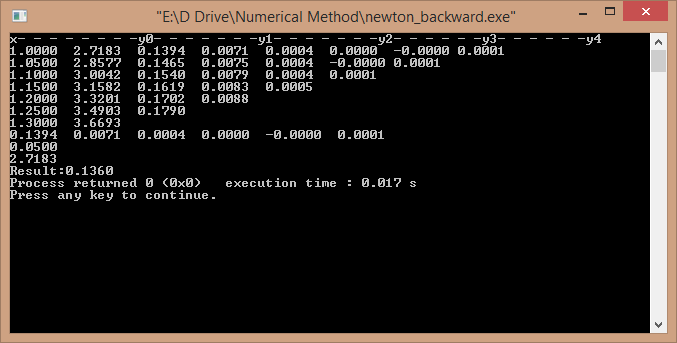
/\*printf("Enter the number\n");

scanf("%lf",&a);\*/

return 0;

}

Output:



Code for backward:

#include <stdio.h>

main()

{

float a[10][10],x[10];

int i,j,fact=1,m;

float xm,p,psum,inter;

printf ("enter number of data points: ");

scanf ("%d", &m);

for (i=0; i<m; i++)

{

for (j=0; j<=m; j++)

{

a[i][j] = 0;

}

}

printf ("\nenter the data (x <space> y)\n");

for (i=0; i<m; i++)

scanf ("%f%f", &x[i],&a[i][0]);

for (j=1; j<m; j++)

{

for (i=1; i<m; i++)

{

if (j<=i)

a[i][j]=a[i][j-1]-a[i-1][j-1];

}

}

printf ("\nbackward difference table\n");

for (i=0; i<m; i++)

{

printf("%.3f", x[i]);

for (j=0; j<=i; j++)

{

printf ("\t%.3f", a[i][j]);

}

printf ("\n");

}

printf ("\nenter x: ");

scanf ("%f", &xm);

p=(xm-x[m-1])/(x[1]-x[0]);

inter=a[m-1][0]+(p\*a[m-1][1]);

printf ("\n");

psum=p;

for(i=2; i<m; i++)

{

psum=psum\*(p+i-1);

fact=fact\*i;

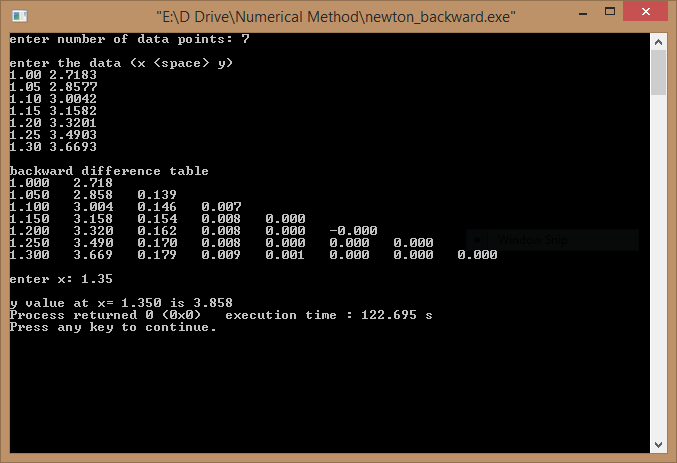
inter=inter+((psum\*a[m-1][i])/fact);

}

printf("y value at x= %.3f is %.3f", xm,inter);

}

Output:



Ovjerbation:

I was taught that the forward formula should be used when calculating the value of a point near x0x0 and the backward one when calculating near xnxn. However, the interpolation polynomial is unique, so the value should be the same. So is there any difference between the two, or my lecturer is wrong?