Newton Forward and Backward Interpolation Method

VISHAL VAMAN MEHTRE[1], MOHIT PRAJAPATI[2]

[1]*Assistant Professor, Department of Electrical Engineering Bharati Vidyapeeth Deemed College of Engineering, PUNE*

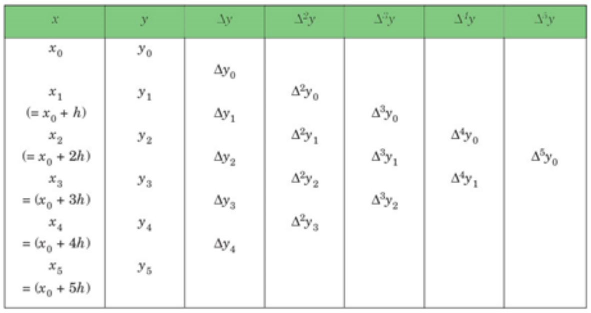
[2] *Department of Electrical Engineering Bharati Vidyapeeth Deemed College of Engineering, PUNE*

Abstract- This paper gives us the basic information of "Newton forward and backward interpolation method". These methods are used to solve problem on newton interpolation by forward or backward interpolation method. For different problem we have different method, this is explained by solving problem below on both newton forward and backward interpolation method. We are also deriving formula for newton forward interpolation method and newton backward interpolation method.

I. INTRODUCTION

- Interpolation is the method of finding the value of a function for any intermediate value for independent variable, while the process of calculating the value of the function outside the given range is called extrapolation. [4]
- Forward Differences: The differences y1 – y0, y2 – y1, y3 – y2… yn – yn–1 when denoted by dy0, dy1, dy2… dyn–1 are respectively, called the first forward differences.

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

II. NEWTON'S GREGORY FORWARD INTERPOLATION METHOD

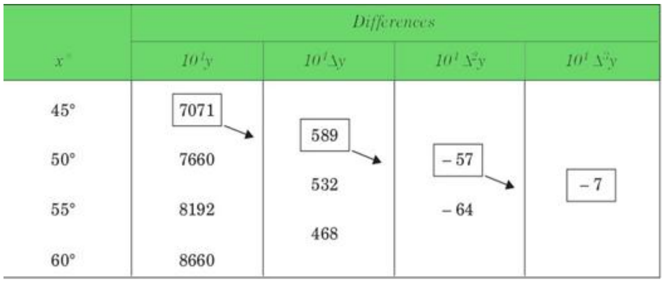
This formula is mainly useful for interpolating the values of f(x) near the starting of the set of values given. "h" is the common difference and u = (x – a) / h, here "a" is first term.

Example: -

Input: - value of sin 52

a	45°	50°	55°	60°
a°	0.7071	0.7660	0.8192	0.8660

Output:



$x°$	Differences			
	$10^4 y$	$10^4 \Delta y$	$10^4 \Delta^2 y$	$10^4 \Delta^3 y$
45°	$\boxed{7071}$			
		$\boxed{589}$		
50°	7660		$\boxed{-57}$	
		532		$\boxed{-7}$
55°	8192		– 64	
		468		
60°	8660			

Value at Sin 52 is 0.788003

Below is the implementation of newton forward interpolation method.

```
CPP Program to interpolate using
 newton forward interpolation
#include <bits/stdc++.h>
using namespace std;
 // calculating u mentioned in the formula
float u_cal(float u, int n)
{
   float temp = u;
   for (int i = 1; i < n; i++)
      temp = temp * (u - i);
   return temp;
}
// calculating factorial of given number n
int fact(int n)
{
   int f = 1;
   for (int i = 2; i <= n; i++)
      f *= i;
   return f;
}
int main()
```

```
{
    // Number of values given
    int n = 4;
    float x[] = { 45, 50, 55, 60 };
    // y[][] is used for difference table
    // with y[][0] used for input
    float y[n][n];
    y[0][0] = 0.7071;
    y[1][0] = 0.7660;
    y[2][0] = 0.8192;
    y[3][0] = 0.8660;
    // Calculating the forward difference
    // table
    for (int i = 1; i < n; i++) {
        for (int j = 0; j < n - i; j++)
            y[j][i] = y[j + 1][i - 1] - y[j][i - 1];
    }
    // Displaying the forward difference table
    for (int i = 0; i < n; i++) {
        cout << setw(4) << x[i]
             << "\t";
        for (int j = 0; j < n - i; j++)
            cout << setw(4) << y[i][j]
                 << "\t";
        cout << endl;
    }
    // Value to interpolate at
    float value = 52;
    // initializing u and sum
    float sum = y[0][0];
    float u = (value - x[0]) / (x[1] - x[0]);
    for (int i = 1; i < n; i++) {
        sum = sum + (u_cal(u, i) * y[0][i]) /
                                 fact(i);
    }
    cout << "\n Value at " << value << " is "
         << sum << endl;
    return 0;
}
```

Output

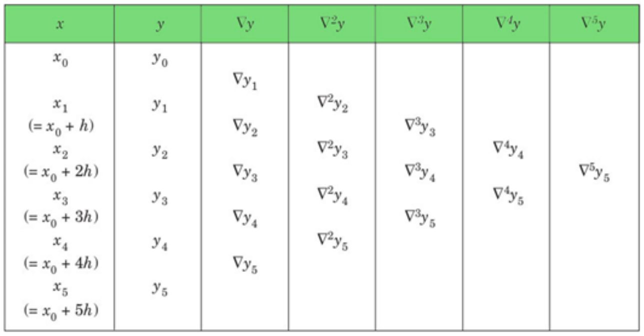
45	0.7071	0.0589	- 0.00569999	- 0.000699997
50	0.766	0.0532	- 0.00639999	
55	0.8192	0.0468		
60	0.866			

Value at 52 is 0.788003

- Backward Differences

The differences y1 – y0, y2 – y1, ……, yn – yn–1 when denoted by dy1, dy2, ……, dyn, respectively, are named as 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					

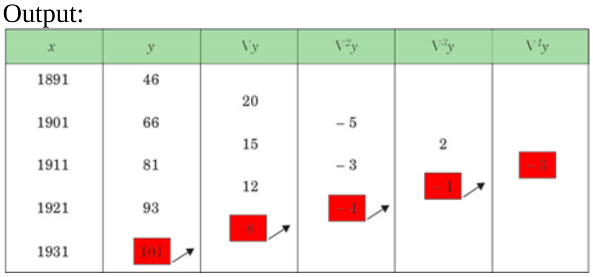
III. NEWTON'S GREGORY BACKWARD INTERPOLATION FORMULA

This formula is used when the value of f(x) is required at the end of the table. h is known as the common difference and u = (x – an) / h, Here an is last term in table.[3]

Example:Input: Population in 1925

Year(x):	1891	1901	1911	1921	1931
Population(y): (in thousands)	46	66	81	93	101

Output:



x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$
1891	46				
		20			
1901	66		– 5		
		15		2	
1911	81		– 3		– 3
		12		– 1	
1921	93		– 4		
		8			
1931	101				

Value in 1925 is 96.8368

Below is the implementation of newton backward interpolation method.

```
// CPP Program to interpolate using
// newton backward interpolation
#include <bits/stdc++.h>
```

```
using namespace std;
// Calculation of u mentioned in formula
float u_cal(float u, int n)
{
    float temp = u;
    for (int i = 1; i < n; i++)
        temp = temp * (u + i);
    return temp;
}
// Calculating factorial of given n
int fact(int n)
{
    int f = 1;
    for (int i = 2; i <= n; i++)
        f *= i;
    return f;
}
int main()
{
    // number of values given
    int n = 5;
    float x[] = { 1891, 1901, 1911,
                  1921, 1931};
    // y[][] is used for difference
    // table and y[][0] used for input
    float y[n][n];
    y[0][0] = 46;
    y[1][0] = 66;
    y[2][0] = 81;
    y[3][0] = 93;
    y[4][0] = 101;
    // Calculating the backward difference table
    for (int i = 1; i < n; i++) {
        for (int j = n - 1; j >= i; j--)
            y[j][i] = y[j][i - 1] - y[j - 1][i - 1];
    }
    // Displaying the backward difference table
    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= i; j++)
            cout << setw(4) << y[i][j]
                 << "\t";
        cout << endl;
    }
    // Value to interpolate at
    float value = 1925;
    // Initializing u and sum
    float sum = y [n - 1][0];
    float u = (value - x [n - 1]) / (x [1] - x[0]);
    for (int i = 1; i < n; i++) {
        sum = sum + (u_cal(u, i) * y[n - 1][i]) /
                             fact(i);
    }
    cout << "\n Value at " << value << " is "
         << sum << endl;
    return 0;
}
```

Output:

46				
66	20			
81	15	-5		
93	12	-3	2	
101	8	-4	-1	-3

Value at 1925 is 96.8368

IV. USE OF NEWTON FORWARD AND BACKWARD INTERPOLATION METHOD

Forward interpolation formula is used to interpolate the values of y nearer to the beginning value of the given table. Also this formula is applicable if in case where h (difference in in travel) is constant.

Now coming to backward difference formula. [1]

This formula is used to interpolate the values of y nearer to the end of the table (or) tabular values.

These are the places, where we need to apply those formulas.
These formulas are used when the in travels for computing a value of a function are equidistant.

CONCLUSION

The degree of the polynomial is one less than the number of pairs of observations. The polynomial that represents the given set of numerical data can be used for interpolation at any position of the independent variable lying within its two extreme values. The approach of interpolation, described here, can be suitably applied in inverse interpolation also. Newton's forward interpolation formula is valid for estimating the value of the dependent variable under the following two conditions [2]: 1. The given values of the independent variable are at equal interval. 2. The value of the independent variable corresponding to which the value of the dependent variable is to be estimated lies in the first half of the series of the given values of the independent variable.[2]

Therefore, the formula derived here is valid for representing a set of numerical data on a pair of variables by a polynomial under these two conditions only. Consequently, there is necessity of searching for some formula for representing a set of numerical data on a pair of variables by a polynomial if the value of the independent variable corresponding to which the value of the dependent variable is to be estimated lies in the last half of the series of the given values, which are at equal interval, of the independent variable. Moreover, there is also necessity of searching for some formula for representing a set of numerical data on a pair of variables by a polynomial if the given values of the independent variable are not at equal interval.

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to Dr. D.S Bankar Head of Department of Electrical Engineering for their able guidance and support for completing my research paper. I would also like to thank the faculty members of Department of Electrical Engineering would helped us with extended support.

REFERENCES

[1] Biswajit Das and Dhritikesh Chakrabarty, “A Study on Newton Forward and Backward Interpolation”, Published By: - International Journal of Statistics and Applied Mathematics 2016; 1(2): 36-41, issued on 12, June, 2016.

[2] Prof. Danny Newton Forward and Backward Interpolation, sixth edition.

[3] “Polymorphism Measures for early risk prediction”, “Saida Benlarbi”.

[4] Study of Measurements using Newton Forward and Backward Interpolation Method “Shivam” issue 2nd July 2013.