Lagrange's Formula

Theory

This is again an N^{th} degree polynomial approximation formula to the function **f(x)**, which is known at discrete points $\mathbf{x_i}$, i = 0, 1, 2 **. . . N^{th}**. The formula can be derived from the Vandermonds determinant but a much simpler way of deriving this is from Newton's divided difference formula. If f(x) is approximated with an N^{th} degree polynomial then the N^{th} divided difference of f(x) constant and $(N+1)^{th}$ divided difference is zero. That is

$$f\,[x0, x1, \ldots xn, x] = 0$$

From the second property of divided difference we can write

$$\frac{f_0}{(x_0 - x_1)\ldots(x_0 - x_n)(x_0 - x)} + \frac{f_n}{(x_n - x_0)\ldots(x_n - x_{n-1})(x_n - x)} + \ldots + \frac{f_x}{(x - x_0)\ldots(x - x_n)} = 0$$

or

$$f(x) = \frac{(x - x_1)\ldots(x - x_n)}{(x_0 - x_1)\ldots(x_0 - x_n)} f_0 + \ldots + \frac{(x - x_0)\ldots(x - x_{n-1})}{(x_n - x_0)\ldots(x_n - x_{n-1})} f_n$$

$$\sum_{i=0}^{n} \left(\prod_{\substack{j=0 \\ j \neq 1}}^{n} \frac{x - x_j}{(x_i - x_j)} \right) f_i$$

Since Lagrange's interpolation is also an N^{th} degree polynomial approximation to f(x) and the N^{th} degree polynomial passing through (N+1) points is unique hence the Lagrange's and Newton's divided difference approximations are one and the same. However, Lagrange's formula is more convenient to use in computer programming and Newton's divided difference formula is more suited for hand calculations.

Code

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int n,x[10],y[10],X;
    double result=0;

    cout<<"How many pairs of x and y: ";
    cin>>n;
    cout<<endl;

    for(int i=0; i<n ; i++)
    {
        cout<<"Enter the value of x"<<i<<" : ";
        cin>>x[i];
        cout<<"Enter the value of y"<<i<<" : ";
        cin>>y[i];
    }
    cout<<"\nEnter the value of x for which the value of y will be determined: ";
    cin>>X;

    int i=0;
    do
    {
        int j=0;
        double numerator=1,denominator=1;

        do
        {
            if(i!=j)
            {
                numerator = numerator*(X-x[j]);
                denominator = denominator*(x[i]-x[j]);
            }
            j = j+1;
        }while(j<n);
        result = result + (numerator/denominator)*y[i];
        i = i+1;
    }while(i<n);

    cout<<"\nThe value of y("<<X<<") is: "<<result<<endl;

    return 0;
}
```

Output

```
"E:\Study\My C\Lab\2-1\CSE 2104\Lab 7\Lagranges' Formula.exe"
How many pairs of x and y: 5

Enter the value of x0 : 0
Enter the value of y0 : 0
Enter the value of x1 : 1
Enter the value of y1 : 1
Enter the value of x2 : 3
Enter the value of y2 : 81
Enter the value of x3 : 4
Enter the value of y3 : 256
Enter the value of x4 : 5
Enter the value of y4 : 625

Enter the value of x for which the value of y will be determined: 2

The value of y(2) is: 16
```

Discussion

In this above code, lagrange's Interpolation formula was applied where the the pairs of the values of x and y were taken as input and for a particular value of x, the value of y was determined using this formula. In this program, a only do while loop was used to execute the main formula of lagrange. Thus in this process, the value of y for x has been calculated.