International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 5, Issue 6, November - December 2016 **ISSN 2278-6856**

ICT Tool: - 'C' Language Program for Lagrange's Interpolation

Pradip P. Kolhe[1] , Dr Prakash R. Kolhe[2] , Sanjay C. Gawande[3]

[1]Assistant Professor (Computer Sci.), ARIS Cell, Dr. PDKV, Akola (MS), India

[2]Associate Professor (Computer Sci.) (CAS), Dr. BSKKV, Dapoli (MS), India

[3]Assistant Professor (Mathematics), College of Agriculture, Dr. PDKV, Akola

Abstract

In mathematics to estimate an unknown data by analysing the given referenced data Lagrange's Interpolation technique is used. For the given data, (say 'y' at various 'x' in tabulated form), the 'y' value corresponding to 'x' values can be found by interpolation.
Interpolation is the process of estimation of an unknown data by analysing the given reference data. In case of equally spaced 'x' values, a number of interpolation methods are available such as the Newton's forward and backward interpolation, Gauss's forward and backward interpolation, Bessel's formula, Laplace-Everett's formula etc. But, all these methods fail when the spacing of 'x' is unequal. In such case, Lagrange interpolation is one of the best options.
In the era of Information Communication Technology (ICT) .The ICT programming technique, it is easier task. One of the very popular programs in C programming is Lagrange's Interpolation. *This paper discuss* ***Lagrange's Interpolation****in C* ***language, source code and methods with outputs. The source codes of program for Lagrange's Interpolationin C programming are to be compiled. Running them on Turbo C or available version and other platforms might require a few modifications to the code.***

Keywords:- Lagrange's Interpolation, ICT,C Lang., Turbo C.

1.Introduction to Lagrange's Interpolation

One of the very popular programs in C programming is **Lagrange's Interpolation**. whereas a program in C can carry out the operations with short, simple and understandable codes.

In numerical analysis, **Lagrange polynomials** are used for polynomial interpolation. For a given set of distinct points x_j and numbers y_j, the **Lagrange polynomial** is the polynomial of the least degree that at each point x_j assumes the corresponding value y_j (i.e. the functions coincide at each point). The interpolating polynomial of the least degree is unique, however, and it is therefore more appropriate to speak of "the Lagrange form" of that unique polynomial rather than "the Lagrange interpolation polynomial", since the same polynomial can be arrived at through multiple methods. Although named after Joseph Louis Lagrange, who published it in 1795, it was first discovered in 1779 by Edward Waring and it is also an easy consequence of a formula published in 1783 by Leonhard Euler. Lagrange interpolation is susceptible to Runge's phenomenon, and the fact that changing the interpolation points requires recalculating the entire interpolant can make Newton polynomials easier to use. Lagrange polynomials are used in the Newton–Cotes method of numerical integration and in Shamir's secret sharing scheme in cryptography.

Lagrange's Interpolation is defined as
Given a set of *k* + 1 data points

$$(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$$

where no two x_j are the same, the **interpolation polynomial in the Lagrange form** is a linear combination.

$$L(x) := \sum_{j=0}^{k} y_j \ell_j(x)$$

of Lagrange basis polynomials

$$\ell_j(x) := \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)},$$

where $0 \le j \le k$. Note how, given the initial assumption that no two x_i are the same, $x_j - x_m \ne 0$, so this expression is always well-defined. The reason pairs $x_i = x_j$ with $y_i \ne y_j$ are not allowed is that no interpolation function L such that $y_i = L(x_i)$ would exist; a function can only get one value for each argument x_i.

On the other hand, if also $y_i = y_j$, then those two points would actually be one single point.

For all $i \neq j$, $\ell_j(x)$ includes the term $(x - x_i)$ in the numerator, so the whole product will be zero at $x = x_i$:

$$\ell_{j\neq i}(x_i) = \prod_{m\neq j} \frac{x_i - x_m}{x_j - x_m} = \frac{(x_i - x_0)}{(x_j - x_0)} \cdots \frac{(x_i - x_i)}{(x_j - x_i)} \cdots \frac{(x_i - x_k)}{(x_j - x_k)} = 0.$$

On the other hand,

$$\ell_i(x_i) := \prod_{m\neq i} \frac{x_i - x_m}{x_i - x_m} = 1$$

In other words, all basis polynomials are zero at $x = x_i$, except $\ell_i(x)$, for which it holds that $\ell_i(x_i) = 1$, because it lacks the $(x - x_i)$ term.

It follows that $y_i \ell_i(x_i) = y_i$, so at each point x_i,

$$L(x_i) = y_i + 0 + 0 + \dots + 0 = y_i,$$

showing that L interpolates the function exactly.

2.Method for Lagrange's Interpolation:

The function $L(x)$ being sought is a polynomial in x of the least degree that interpolates the given data set; that is, assumes value y_j at the corresponding x_j for all data points j:

$$L(x_j) = y_j \qquad j = 0, \ldots, k$$

Observe that:

1. In $\ell_j(x)$ there are k factors in the product and each factor contains one x, so $L(x)$ (which is a sum of these k-degree polynomials) must also be a k-degree polynomial.

$$\ell_j(x_i) = \prod_{m=0,\, m\neq j}^{k} \frac{x_i - x_m}{x_j - x_m}$$

We consider what happens when this product is expanded. Because the product skips $m = j$, if $i = j$ then all terms are $\frac{x_j - x_m}{x_j - x_m} = 1$ (except where $x_j = x_m$, but that case is impossible, as pointed out in the definition section—in that term, $m = j$, and since $m \neq j$, $i \neq j$, contrary to $i = j$). Also if $i \neq j$ then since $m \neq j$ does not preclude it, one term in the product **will** be for $m = i$, i.e. $\frac{x_i - x_i}{x_j - x_i} = 0$, zeroing the entire product.

So

$$\ell_j(x_i) = \delta_{ji} = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{if } j \neq i \end{cases}$$

where δ_{ij} is the Kronecker delta. So:

$$L(x_i) = \sum_{j=0}^{k} y_j \ell_j(x_i) = \sum_{j=0}^{k} y_j \delta_{ji} = y_i.$$

Thus the function $L(x)$ is a polynomial with degree at most k and where $L(x_i) = y_i$.

Example 1

We wish to interpolate $f(x) = x^2$ over the range $1 \leq x \leq 3$, given these three points:

$$\begin{aligned} x_0 &= 1 & f(x_0) &= 1 \\ x_1 &= 2 & f(x_1) &= 4 \\ x_2 &= 3 & f(x_2) &= 9. \end{aligned}$$

The interpolating polynomial is:

$$\begin{aligned} L(x) &= 1 \cdot \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} + 4 \cdot \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} + 9 \cdot \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} \\ &= x^2. \end{aligned}$$

Example 2

We wish to interpolate $f(x) = x^3$ over the range $1 \leq x \leq 3$, given these three points:

$$\begin{aligned} x_0 &= 1 \; f(x_0) = 1 \\ x_1 &= 2 \; f(x_1) = 8 \\ x_2 &= 3 \; f(x_2) = 27 \end{aligned}$$

The interpolating polynomial is:

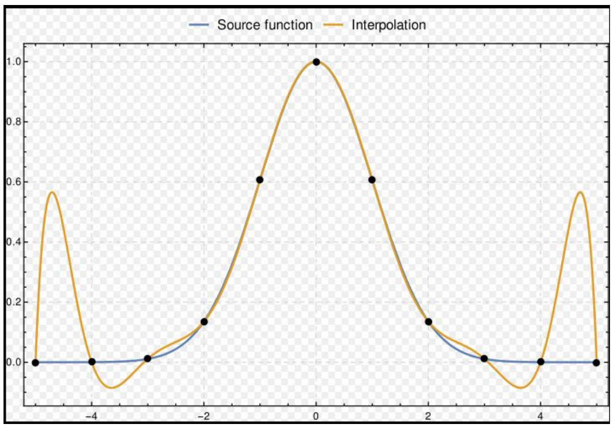
$$\begin{aligned} L(x) &= 1 \cdot \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} + 8 \cdot \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} + 27 \cdot \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} \\ &= 6x^2 - 11x + 6. \end{aligned}$$

Note for Lagrange's Interpolation

The Lagrange form of the interpolation polynomial shows the linear character of polynomial interpolation and the uniqueness of the interpolation polynomial. Therefore, it is preferred in proofs and theoretical arguments. Uniqueness can also be seen from the inheritability of the Vander monde matrix, due to the non-vanishing of the Vander monde determinant.

Lagrange and other interpolation at equally spaced points, as in the example above, yield a polynomial oscillating above and below the true function. This behavior tends to grow with the number of points, leading to a divergence known as Runge's phenomenon; the problem may be eliminated by choosing interpolation points at Chebyshev nodes.

The Lagrange basis polynomials can be used in numerical integration to derive the Newton–Cotes formulas.



Remainder in Lagrange's Interpolation Formula:
When interpolating a given function f by a polynomial of degree n at the nodes $x_0,...,x_n$ we get the remainder which can be expressed as

$$R(x) = f[x_0,\ldots,x_n,x]\ell(x) = \ell(x)\frac{f^{n+1}(\xi)}{(n+1)!}, \qquad x_0 < \xi < x_n,$$

where $f[x_0, \ldots, x_n, x]$ and is the notation for divided differences. Alternatively, the remainder can be expressed as a contour integral in complex domain as

$$R(z) = \frac{\ell(z)}{2\pi i}\int_C \frac{f(t)}{(t-z)(t-z_0)\cdots(t-z_n)}dt = \frac{\ell(z)}{2\pi i}\int_C \frac{f(t)}{(t-z)\ell(t)}dt.$$

The remainder can be bound as

$$|R(x)| \leq \frac{(x_n - x_0)^{n+1}}{(n+1)!}\max_{x_0\leq\xi\leq x_n}|f^{(n+1)}(\xi)|.$$

First Derivative of Lagrange's

The first derivative of the Lagrange's polynomial is given by

$$L'(x) := \sum_{j=0}^{k} y_j \ell'_j(x)$$

where

$$\ell'_j(x) = \ell_j(x) \cdot \sum_{m=0,\, m\neq j}^{n} \frac{1}{x - x_m}$$

The Lagrange polynomial can also be computed in **finite fields**. This has applications in **cryptography**, such as in **Shamir's Secret Sharing scheme.**

Algorithm for Lagrange's Interpolation

```
Step 1:  Read x, n
Step 2: for i = 1 to (n + 1) is steps of 1 do Read xi,fi end for {the above statements reads x,s and the corresponding values of f is}
Step 3:  Sum = 0
Step 4: for i = 1 to (n + 1) in steps of 1 do
Step 5: Profvnc = 1
Step 6: for J = 1 to (n + 1) in steps of 1 do
Step 7: If (j ≠ i) then prodfunc = prodfunc X(x - xj) / (xi - xj) end for
Step 8: Sum = Sum + fi x Prodfunc {sum is the value of f at x} end for
Step 9:  Write x, sum
Step 10: Stop
```

C language source code: - Lagrange's Interpolation

```
#include<stdio.h>
#include<math.h>
main()
{
        float y, x[20], f[20], sum, pf;
        int I, j, n;
        printf("enter the value of n");
        scanf("%d", &n);
        printf("enter the value to be found");
        scanf("%f", &y);
        printf("enter the values of xi's & fi's");
        for(i = 0; i < n; i++)
        {
            pf = 1;
            for(j = 0; j < n; j++)
           {
                 if(j != i)
                 Pf *= (y - x[j])/(x[i] – x[j]);
           }
                 sum += f[i] * pf;
        }
        printf("\nx = %f ", y);
        printf("\n sum =%f ", sum);
}
```

Output of Lagrange's Interpolation

```
enter the value of n 4
enter the value to be found 2.5
enter the values for xi's & fi's
1 1
2 8
3 27
4 64
X = 2.500000
sum = 15.625000
```

Lagrange Polynomial implemented in Python

```
#!/usr/bin/env python
import numpy as np
from matplotlib import pyplot as plt
data_fname = 'knot_points.csv'

# x1,y1
# x2,y2
# ...
 def read_data(fname):
        X = [ ]
        Y = [ ]
with open(fname, 'r') as f:
   for line in f:
   (x, y) = line.split(',')
   X.append(float(x))
   Y.append(float(y))
   return (X, Y)
def langrange_polynomial(X, Y):
        def L(i):
        return lambda x: np.prod([(x-X[j])/(X[i]-X[j]) for
        j in range(len(X)) if i != j]) * Y[i]
        Sx = [L(i) for i in range(len(X))]  # summands
   return lambda x: np.sum([s(x) for s in Sx])
# F = langrange_polynomial(*read_data(data_fname))
(X, Y) = read_data(data_fname)
F = langrange_polynomial(X, Y)
x_range = np.linspace(X[0], X[-1], 100)
plt.plot(X, Y, 'ro')
plt.plot(x_range, map(F, x_range))
plt.xlabel(r'$x$')
plt.ylabel(r'$F(x)$')
plt.title('Lagrange polynomial interpolation')
plt.grid(True)
plt.show()
```

References

[1] Meijering, Erik (2002), "A chronology of interpolation: from ancient astronomy to modern signal and image processing", Proceedings of the IEEE, **90** (3): 319–342

[2] Quarteroni, Alfio; Saleri, Fausto (2003), Scientific Computing with MATLAB, Texts in computational science and engineering, **2**, Springer, p. 66, *ISBN 9783540443636.*

[3] Jean-Paul Berrut &*Lloyd N. Trefethen* (2004). "Barycentric Lagrange Interpolation". *SIAM* Review. **46** (3): 501–517..

[4] Abramowitz and Stegun, "Handbook of Mathematical Functions," p.878

External links

- Hazewinkel, Michiel, ed. (2001), "Lagrange interpolation formula", Encyclopedia of Mathematics, Springer, ISBN 978-1-55608-010-4
- ALGLIB has an implementations in C++ / C# / VBA / Pascal.
- GSL has a polynomial interpolation code in C
- SO has a MATLAB example that demonstrates the algorithm and recreates the first image in this article
- Lagrange Method of Interpolation — Notes, PPT, Mathcad, Mathematica, MATLAB, Maple at Holistic Numerical Methods Institute
- Lagrange interpolation polynomial on www.math-linux.com

- Weisstein, Eric W."Lagrange Interpolating Polynomial". MathWorld.
- Estimate of the error in Lagrange Polynomial Approximation at ProofWiki
- Module for Lagrange Polynomials by John H. Mathews
- Dynamic Lagrange interpolation with JSXGraph
- Numerical computing with functions: The Chebfun Project
- Excel Worksheet Function for Bicubic Lagrange Interpolation.
- Lagrange polynomials in Python.