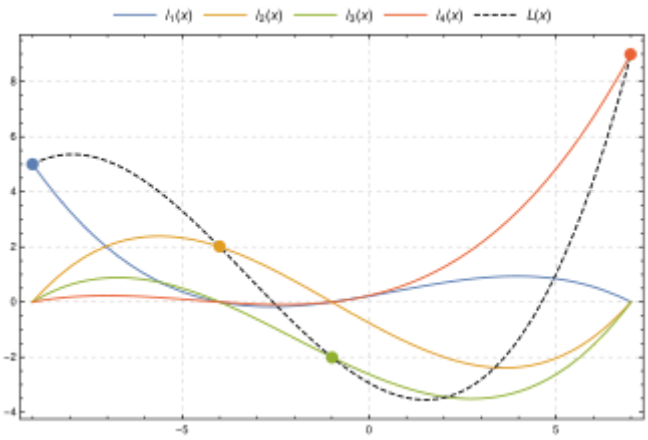
Lagrange polynomial

In numerical analysis, the **Lagrange interpolating polynomial** is the unique polynomial of lowest degree that interpolates a given set of data.

Given a data set of coordinate pairs (x_j, y_j) with $0 \leq j \leq k$, the x_j are called *nodes* and the y_j are called *values*. The Lagrange polynomial $L(x)$ has degree $\leq k$ and assumes each value at the corresponding node, $L(x_j) = y_j$.

Although named after Joseph-Louis Lagrange, who published it in 1795,[1] the method was first discovered in 1779 by Edward Waring.[2] It is also an easy consequence of a formula published in 1783 by Leonhard Euler.[3]

Uses of Lagrange polynomials include the Newton–Cotes method of numerical integration, Shamir's secret sharing scheme in cryptography, and Reed–Solomon error correction in coding theory.



This image shows, for four points ((−9, 5), (−4, 2), (−1, −2), (7, 9)), the (cubic) interpolation polynomial $L(x)$ (dashed, black), which is the sum of the *scaled* basis polynomials $y_0\ell_0(x)$, $y_1\ell_1(x)$, $y_2\ell_2(x)$ and $y_3\ell_3(x)$. The interpolation polynomial passes through all four control points, and each *scaled* basis polynomial passes through its respective control point and is 0 where x corresponds to the other three control points.

For equispaced nodes, Lagrange interpolation is susceptible to Runge's phenomenon of large oscillation.

Definition

Given a set of $k+1$ nodes $\{x_0, x_1, \ldots, x_k\}$, which must all be distinct, $x_j \neq x_m$ for indices $j \neq m$, the **Lagrange basis** for polynomials of degree $\leq k$ for those nodes is the set of polynomials $\{\ell_0(x), \ell_1(x), \ldots, \ell_k(x)\}$ each of degree k which take values $\ell_j(x_m) = 0$ if $m \neq j$ and $\ell_j(x_j) = 1$. Using the Kronecker delta this can be written $\ell_j(x_m) = \delta_{jm}$. Each basis polynomial can be explicitly described by the product:

$$\begin{aligned}\ell_j(x) &= \frac{(x-x_0)}{(x_j-x_0)} \cdots \frac{(x-x_{j-1})}{(x_j-x_{j-1})} \frac{(x-x_{j+1})}{(x_j-x_{j+1})} \cdots \frac{(x-x_k)}{(x_j-x_k)} \\ &= \prod_{\substack{0 \leq m \leq k \\ m \neq j}} \frac{x-x_m}{x_j-x_m}.\end{aligned}$$

Notice that the numerator $\prod_{m\neq j}(x-x_m)$ has k roots at the nodes $\{x_m\}_{m\neq j}$ while the denominator $\prod_{m\neq j}(x_j-x_m)$ scales the resulting polynomial so that $\ell_j(x_j)=1$.

The Lagrange interpolating polynomial for those nodes through the corresponding *values* $\{y_0, y_1, \ldots, y_k\}$ is the linear combination:

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

Each basis polynomial has degree k, so the sum $L(x)$ has degree $\leq k$, and it interpolates the data because $L(x_m) = \sum_{j=0}^{k} y_j \ell_j(x_m) = \sum_{j=0}^{k} y_j \delta_{mj} = y_m$.

The interpolating polynomial is unique. Proof: assume the polynomial $M(x)$ of degree $\leq k$ interpolates the data. Then the difference $M(x) - L(x)$ is zero at $k+1$ distinct nodes $\{x_0, x_1, \ldots, x_k\}$. But the only polynomial of degree $\leq k$ with more than k roots is the constant zero function, so $M(x) - L(x) = 0$, or $M(x) = L(x)$.

Barycentric form

Each Lagrange basis polynomial $\ell_j(x)$ can be rewritten as the product of three parts, a function $\ell(x) = \prod_m (x - x_m)$ common to every basis polynomial, a node-specific constant $w_j = \prod_{m\neq j}(x_j - x_m)^{-1}$ (called the *barycentric weight*), and a part representing the displacement from x_j to x:[4]

$$\ell_j(x) = \ell(x)\frac{w_j}{x - x_j}$$

By factoring $\ell(x)$ out from the sum, we can write the Lagrange polynomial in the so-called *first barycentric form*:

$$L(x) = \ell(x)\sum_{j=0}^{k}\frac{w_j}{x-x_j}y_j.$$

If the weights w_j have been pre-computed, this requires only $\mathcal{O}(k)$ operations compared to $\mathcal{O}(k^2)$ for evaluating each Lagrange basis polynomial $\ell_j(x)$ individually.

The barycentric interpolation formula can also easily be updated to incorporate a new node x_{k+1} by dividing each of the w_j, $j = 0 \ldots k$ by $(x_j - x_{k+1})$ and constructing the new w_{k+1} as above.

For any x, $\sum_{j=0}^{k} \ell_j(x) = 1$ because the constant function $g(x) = 1$ is the unique polynomial of degree $\leq k$ interpolating the data $\{(x_0, 1), (x_1, 1), \ldots, (x_k, 1)\}$. We can thus further simplify the barycentric formula by dividing $L(x) = L(x)/g(x)$:

$$L(x) = \ell(x) \sum_{j=0}^{k} \frac{w_j}{x - x_j} y_j \Bigg/ \ell(x) \sum_{j=0}^{k} \frac{w_j}{x - x_j}$$

$$= \sum_{j=0}^{k} \frac{w_j}{x - x_j} y_j \Bigg/ \sum_{j=0}^{k} \frac{w_j}{x - x_j}.$$

This is called the *second form* or *true form* of the barycentric interpolation formula.

This second form has advantages in computation cost and accuracy: it avoids evaluation of $\ell(x)$; the work to compute each term in the denominator $w_j/(x - x_j)$ has already been done in computing $\big(w_j/(x - x_j)\big)y_j$ and so computing the sum in the denominator costs only $k - 1$ addition operations; for evaluation points x which are close to one of the nodes x_j, catastrophic cancelation would ordinarily be a problem for the value $(x - x_j)$, however this quantity appears in both numerator and denominator and the two cancel leaving good relative accuracy in the final result.

Using this formula to evaluate $L(x)$ at one of the nodes x_j will result in the indeterminate $\infty y_j/\infty$; computer implementations must replace such results by $L(x_j) = y_j$.

Each Lagrange basis polynomial can also be written in barycentric form:

$$\ell_j(x) = \frac{w_j}{x - x_j} \Bigg/ \sum_{m=0}^{k} \frac{w_m}{x - x_m}.$$

A perspective from linear algebra

Solving an interpolation problem leads to a problem in linear algebra amounting to inversion of a matrix. Using a standard monomial basis for our interpolation polynomial $L(x) = \sum_{j=0}^{k} x^j m_j$, we must invert the Vandermonde matrix $(x_i)^j$ to solve $L(x_i) = y_i$ for the coefficients m_j of $L(x)$. By choosing a better basis, the Lagrange basis, $L(x) = \sum_{j=0}^{k} l_j(x) y_j$, we merely get the identity matrix, δ_{ij}, which is its own inverse: the Lagrange basis automatically *inverts* the analog of the Vandermonde matrix.

This construction is analogous to the Chinese remainder theorem. Instead of checking for remainders of integers modulo prime numbers, we are checking for remainders of polynomials when divided by linears.

Furthermore, when the order is large, Fast Fourier transformation can be used to solve for the coefficients of the interpolated polynomial.

Example

We wish to interpolate $f(x) = x^2$ over the domain $1 \leq x \leq 3$ at the three nodes $\{1, 2, 3\}$:

$$\begin{aligned}
x_0 &= 1, & y_0 &= f(x_0) = 1,\\
x_1 &= 2, & y_1 &= f(x_1) = 4,\\
x_2 &= 3, & y_2 &= f(x_2) = 9.
\end{aligned}$$

The node polynomial ℓ is

$$\ell(x) = (x-1)(x-2)(x-3) = x^3 - 6x^2 + 11x - 6.$$

The barycentric weights are

$$\begin{aligned}
w_0 &= (1-2)^{-1}(1-3)^{-1} = \tfrac{1}{2},\\
w_1 &= (2-1)^{-1}(2-3)^{-1} = -1,\\
w_2 &= (3-1)^{-1}(3-2)^{-1} = \tfrac{1}{2}.
\end{aligned}$$

The Lagrange basis polynomials are

$$\begin{aligned}
\ell_0(x) &= \frac{x-2}{1-2}\cdot\frac{x-3}{1-3} = \tfrac{1}{2}x^2 - \tfrac{5}{2}x + 3,\\
\ell_1(x) &= \frac{x-1}{2-1}\cdot\frac{x-3}{2-3} = -x^2 + 4x - 3,\\
\ell_2(x) &= \frac{x-1}{3-1}\cdot\frac{x-2}{3-2} = \tfrac{1}{2}x^2 - \tfrac{3}{2}x + 1.
\end{aligned}$$

The Lagrange interpolating polynomial is:

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

In (second) barycentric form,

$$L(x) = \frac{\displaystyle\sum_{j=0}^{2}\frac{w_j}{x-x_j}y_j}{\displaystyle\sum_{j=0}^{2}\frac{w_j}{x-x_j}} = \frac{\dfrac{\frac{1}{2}}{x-1} + \dfrac{-4}{x-2} + \dfrac{\frac{9}{2}}{x-3}}{\dfrac{\frac{1}{2}}{x-1} + \dfrac{-1}{x-2} + \dfrac{\frac{1}{2}}{x-3}}.$$

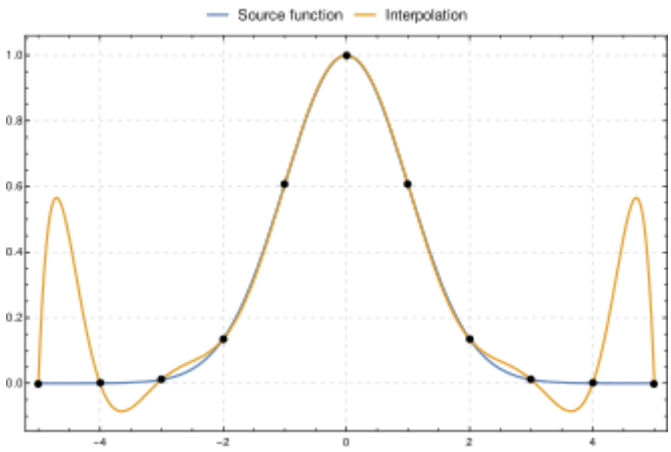
Notes

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 invertibility of the Vandermonde matrix, due to the non-vanishing of the Vandermonde determinant.

But, as can be seen from the construction, each time a node x_k changes, all Lagrange basis polynomials have to be recalculated. A better form of the interpolation polynomial for practical (or computational) purposes is the barycentric form of the Lagrange interpolation (see below) or Newton polynomials.

Lagrange and other interpolation at equally spaced points, as in the example above, yield a polynomial oscillating above and below the true function. This behaviour 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.[5]

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



Example of interpolation divergence for a set of Lagrange polynomials.

Remainder in Lagrange interpolation formula

When interpolating a given function f by a polynomial of degree k at the nodes $x_0, \ldots, x_k$ we get the remainder $R(x) = f(x) - L(x)$ which can be expressed as[6]

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

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

$$R(x) = \frac{\ell(x)}{2\pi i}\int_C \frac{f(t)}{(t-x)(t-x_0)\cdots(t-x_k)}dt = \frac{\ell(x)}{2\pi i}\int_C \frac{f(t)}{(t-x)\ell(t)}dt.$$

The remainder can be bound as

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

Derivation[7]

Clearly, $R(x)$ is zero at nodes. To find $R(x)$ at a point x_p, define a new function $F(x) = R(x) - \tilde{R}(x) = f(x) - L(x) - \tilde{R}(x)$ and choose $\tilde{R}(x) = C \cdot \prod_{i=0}^{k}(x - x_i)$ where C is the constant we are required to determine for a given x_p. We choose C so that $F(x)$ has $k+2$ zeroes (at all nodes and x_p) between x_0 and x_k (including endpoints). Assuming that $f(x)$ is $k+1$-times differentiable, since $L(x)$ and $\tilde{R}(x)$ are polynomials, and therefore, are infinitely differentiable, $F(x)$ will be $k+1$-times differentiable. By Rolle's theorem, $F^{(1)}(x)$ has $k+1$ zeroes, $F^{(2)}(x)$ has k zeroes... $F^{(k+1)}$ has 1 zero, say ξ, $x_0 < \xi < x_k$. Explicitly writing $F^{(k+1)}(\xi)$:

$$F^{(k+1)}(\xi) = f^{(k+1)}(\xi) - L^{(k+1)}(\xi) - \tilde{R}^{(k+1)}(\xi)$$

$L^{(k+1)} = 0, \tilde{R}^{(k+1)} = C \cdot (k+1)!$ (Because the highest power of x in $\tilde{R}(x)$ is $k+1$)

$$0 = f^{(k+1)}(\xi) - C \cdot (k+1)!$$

The equation can be rearranged as

$$C = \frac{f^{(k+1)}(\xi)}{(k+1)!}$$

Since $F(x_p) = 0$ we have $R(x_p) = \tilde{R}(x_p) = \frac{f^{k+1}(\xi)}{(k+1)!} \prod_{i=0}^{k} (x_p - x_i)$

Derivatives

The dth derivative of a Lagrange interpolating polynomial can be written in terms of the derivatives of the basis polynomials,

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

Recall (see § Definition above) that each Lagrange basis polynomial is

$$\ell_j(x) = \prod_{\substack{m=0 \\ m \neq j}}^{k} \frac{x - x_m}{x_j - x_m}.$$

The first derivative can be found using the product rule:

$$\begin{aligned} \ell_j'(x) &= \sum_{\substack{i=0 \\ i \neq j}}^{k} \left[\frac{1}{x_j - x_i} \prod_{\substack{m=0 \\ m \neq (i,j)}}^{k} \frac{x - x_m}{x_j - x_m} \right] \\ &= \ell_j(x) \sum_{\substack{i=0 \\ i \neq j}}^{k} \frac{1}{x - x_i}. \end{aligned}$$

The second derivative is

$$\ell_j''(x) = \sum_{\substack{i=0\\ i\neq j}}^{k} \frac{1}{x_j - x_i} \left[\sum_{\substack{m=0\\ m\neq (i,j)}}^{k} \left(\frac{1}{x_j - x_m} \prod_{\substack{n=0\\ n\neq (i,j,m)}}^{k} \frac{x - x_n}{x_j - x_n} \right) \right]$$

$$= \ell_j(x) \sum_{0\leq i<m\leq k} \frac{2}{(x - x_i)(x - x_m)}$$

$$= \ell_j(x) \left[\left(\sum_{\substack{i=0\\ i\neq j}}^{k} \frac{1}{x - x_i} \right)^2 - \sum_{\substack{i=0\\ i\neq j}}^{k} \frac{1}{(x - x_i)^2} \right].$$

The third derivative is

$$\ell_j'''(x) = \ell_j(x) \sum_{0\leq i<m<n\leq k} \frac{3!}{(x - x_i)(x - x_m)(x - x_n)}$$

and likewise for higher derivatives.

Finite fields

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

See also

- Neville's algorithm
- Newton form of the interpolation polynomial
- Bernstein polynomial
- Carlson's theorem
- Lebesgue constant (interpolation)
- The Chebfun system
- Table of Newtonian series
- Frobenius covariant
- Sylvester's formula
- Finite difference coefficient
- Hermite interpolation

References

1. Lagrange, Joseph-Louis (1795). "Leçon Cinquième. Sur l'usage des courbes dans la solution des problèmes". *Leçons Elémentaires sur les Mathématiques* (in French). Paris. Republished in Serret, Joseph-Alfred, ed. (1877). *Oeuvres de Lagrange*. Vol. 7. Gauthier-Villars. pp. 271–287 (https://archive.org/details/oeuvresdelagrang07lagr/page/271). Translated as "Lecture V. On the Employment of Curves in the Solution of Problems" (https://archive.org/details/lecturesonelemen00lagriala/page/127). *Lectures on Elementary Mathematics*. Translated by McCormack, Thomas J. (2nd ed.). Open Court. 1901. pp. 127–149.

2. Waring, Edward (1779). "Problems concerning interpolations" (https://archive.org/details/philosophicaltra6917roya/page/59). *Philosophical Transactions of the Royal Society*. **69**: 59–67. doi:10.1098/rstl.1779.0008 (https://doi.org/10.1098%2Frstl.1779.0008).
3. Meijering, Erik (2002). "A chronology of interpolation: from ancient astronomy to modern signal and image processing" (http://bigwww.epfl.ch/publications/meijering0201.pdf) (PDF). *Proceedings of the IEEE*. **90** (3): 319–342. doi:10.1109/5.993400 (https://doi.org/10.1109%2F5.993400).
4. Berrut, Jean-Paul; Trefethen, Lloyd N. (2004). "Barycentric Lagrange Interpolation" (https://people.maths.ox.ac.uk/trefethen/barycentric.pdf) (PDF). *SIAM Review*. **46** (3): 501–517. Bibcode:2004SIAMR..46..501B (https://ui.adsabs.harvard.edu/abs/2004SIAMR..46..501B). doi:10.1137/S0036144502417715 (https://doi.org/10.1137%2FS0036144502417715).
5. Quarteroni, Alfio; Saleri, Fausto (2003). *Scientific Computing with MATLAB* (https://books.google.com/books?id=fE1W5jsU4zoC&pg=PA66). Texts in computational science and engineering. Vol. 2. Springer. p. 66. ISBN 978-3-540-44363-6..
6. Abramowitz, Milton; Stegun, Irene Ann, eds. (1983) [June 1964]. "Chapter 25, eqn 25.2.3" (http://www.math.ubc.ca/~cbm/aands/page_878.htm). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Applied Mathematics Series. Vol. 55 (Ninth reprint with additional corrections of tenth original printing with corrections (December 1972); first ed.). Washington D.C.; New York: United States Department of Commerce, National Bureau of Standards; Dover Publications. p. 878. ISBN 978-0-486-61272-0. LCCN 64-60036 (https://lccn.loc.gov/64-60036). MR 0167642 (https://mathscinet.ams.org/mathscinet-getitem?mr=0167642). LCCN 65-12253 (https://lccn.loc.gov/65012253).
7. "Interpolation" (https://sam.nitk.ac.in/sites/default/Numerical_Methods/Interpolation/interpolation.pdf) (PDF).

External links

- "Lagrange interpolation formula" (https://www.encyclopediaofmath.org/index.php?title=Lagrange_interpolation_formula), *Encyclopedia of Mathematics*, EMS Press, 2001 [1994]
- ALGLIB (http://www.alglib.net/interpolation/polynomial.php) has an implementations in C++ / C# / VBA / Pascal.
- GSL (https://www.gnu.org/software/gsl/) has a polynomial interpolation code in C
- SO (https://stackoverflow.com/questions/11029615/lagrange-interpolation-method/11552763) 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 (http://numericalmethods.eng.usf.edu/topics/lagrange_method.html) at Holistic Numerical Methods Institute (http://numericalmethods.eng.usf.edu)
- Lagrange interpolation polynomial (http://www.math-linux.com/spip.php?article71) on www.math-linux.com
- Weisstein, Eric W. "Lagrange Interpolating Polynomial" (https://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html). *MathWorld*.
- Lagrange polynomial at ProofWiki
- Dynamic Lagrange interpolation with JSXGraph (http://jsxgraph.uni-bayreuth.de/wiki/index.php/Lagrange_interpolation)
- Numerical computing with functions: The Chebfun Project (https://web.archive.org/web/20101013180326/http://www2.maths.ox.ac.uk/chebfun/)
- Excel Worksheet Function for Bicubic Lagrange Interpolation (http://mathformeremortals.wordpress.com/2013/01/15/bicubic-interpolation-excel-worksheet-function/)

- Lagrange polynomials in Python (https://learn.64bitdragon.com/articles/computer-science/numerical-analysis/lagrange-interpolation)