Polynomial interpolation

From Wikipedia, the free encyclopedia

In numerical analysis, **polynomial interpolation** is the interpolation of a given data set by a polynomial: given some points, find a polynomial which goes exactly through these points.

Applications [edit]

Polynomials can be used to approximate complicated curves, for example, the shapes of letters in typography,[*citation needed*] given a few points. A relevant application is the evaluation of the natural logarithm and trigonometric functions: pick a few known data points, create a lookup table, and interpolate between those data points. This results in significantly faster computations.[*specify*] Polynomial interpolation also forms the basis for algorithms in numerical quadrature and numerical ordinary differential equations.

Polynomial interpolation is also essential to perform sub-quadratic multiplication and squaring such as Karatsuba multiplication and Toom–Cook multiplication, where an interpolation through points on a polynomial which defines the product yields the product itself. For example, given $a = f(x) = a_0x^0 + a_1x^1 + \ldots$ and $b = g(x) = b_0x^0 + b_1x^1 + \ldots$, the product ab is equivalent to $W(x) = f(x)g(x)$. Finding points along $W(x)$ by substituting x for small values in $f(x)$ and $g(x)$ yields points on the curve. Interpolation based on those points will yield the terms of $W(x)$ and subsequently the product ab. In the case of Karatsuba multiplication this technique is substantially faster than quadratic multiplication, even for modest-sized inputs. This is especially true when implemented in parallel hardware.

Definition [edit]

Given a set of $n+1$ data points (x_i, y_i) where no two x_i are the same, one is looking for a polynomial p of degree at most n with the property

$$p(x_i) = y_i, \qquad i = 0, \ldots, n.$$

The unisolvence theorem states that such a polynomial p exists and is unique, and can be proved by the Vandermonde matrix, as described below.

The theorem states that for $n+1$ interpolation nodes (x_i), polynomial interpolation defines a linear bijection

$$L_n : \mathbb{K}^{n+1} \to \Pi_n$$

where Π_n is the vector space of polynomials (defined on any interval containing the nodes) of degree at most n.

Constructing the interpolation polynomial [edit]

Main article: Lagrange polynomial

Suppose that the interpolation polynomial is in the form

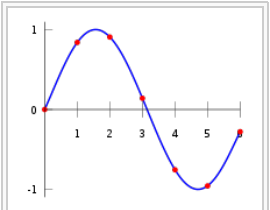
$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2x^2 + a_1x + a_0. \qquad (1)$$

The statement that p interpolates the data points means that

$$p(x_i) = y_i \qquad \text{for all } i \in \{0, 1, \ldots, n\}.$$

If we substitute equation (1) in here, we get a system of linear equations in the coefficients a_k. The system in matrix-vector form reads

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \ldots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \ldots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \ldots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$



The red dots denote the data points (x_k, y_k), while the blue curve shows the interpolation polynomial.

We have to solve this system for a_k to construct the interpolant $p(x)$. The matrix on the left is commonly referred to as a Vandermonde matrix.

The condition number of the Vandermonde matrix may be large,[1] causing large errors when computing the coefficients a_i if the system of equations is solved using Gaussian elimination.

Several authors have therefore proposed algorithms which exploit the structure of the Vandermonde matrix to compute numerically stable solutions in $O(n^2)$ operations instead of the $O(n^3)$ required by Gaussian elimination.[2][3][4] These methods rely on constructing first a Newton interpolation of the polynomial and then converting it to the monomial form above.

Alternatively, we may write down the polynomial immediately in terms of Lagrange polynomials:

$$p(x) = \frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)}y_0 + \frac{(x-x_0)(x-x_2)\cdots(x-x_n)}{(x_1-x_0)(x_1-x_2)\cdots(x_1-x_n)}y_1 + \ldots + \frac{(x-x_0)(x-x_1)\cdots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\cdots(x_n-x_{n-1})}y_n$$

$$= \sum_{i=0}^{n} \left(\prod_{\substack{0 \le j \le n \\ j \ne i}} \frac{x - x_j}{x_i - x_j} \right) y_i$$

For matrix arguments, this formula is called Sylvester's formula and the matrix-valued Lagrange polynomials are the Frobenius covariants.

Uniqueness of the interpolating polynomial [edit]

Proof 1 [edit]

Suppose we interpolate through $n+1$ data points with an at-most n degree polynomial $p(x)$ (we need at least $n+1$ datapoints or else the polynomial cannot be fully solved for). Suppose also another polynomial exists also of degree at most n that also interpolates the $n+1$ points; call it $q(x)$.

Consider $r(x) = p(x) - q(x)$. We know,

1. $r(x)$ is a polynomial
2. $r(x)$ has degree at most n, since $p(x)$ and $q(x)$ are no higher than this and we are just subtracting them.
3. At the $n+1$ data points, $r(x_i) = p(x_i) - q(x_i) = y_i - y_i = 0$. Therefore $r(x)$ has $n+1$ roots.

But $r(x)$ is an polynomial of degree $\leq n$. It has one root too many. Formally, if $r(x)$ is any non-zero polynomial, it must be writable as $r(x) = A(x-x_0)(x-x_1)\cdots(x-x_n)$ for some constant A. By distributivity, the $n+1$ x's multiply together to give leading term Ax^{n+1}, i.e. one degree higher than the maximum we set. So the only way $r(x)$ can exist is if $A = 0$, or equivalently, $r(x) = 0$.

$$r(x) = 0 = p(x) - q(x) \implies p(x) = q(x)$$

So $q(x)$ (which could be any polynomial, so long as it interpolates the points) is identical with $p(x)$, and $p(x)$ is unique.

Proof 2 [edit]

Given the Vandermonde matrix used above to construct the interpolant, we can set up the system

$$Va = y$$

To prove that V is nonsingular we use the Vandermonde determinant formula:

$$\det(V) = \prod_{i,j=0, i<j}^{n} (x_i - x_j)$$

since the $n+1$ points are distinct, the determinant can't be zero as $x_i - x_j$ is never zero, therefore V is nonsingular and the system has a unique solution.

Either way this means that no matter what method we use to do our interpolation: direct, Lagrange etc., (assuming we can do all our calculations perfectly) we will always get the same polynomial.

Non-Vandermonde solutions [edit]

We are trying to construct our unique interpolation polynomial in the vector space Π_n of polynomials of degree n. When using a monomial basis for Π_n we have to solve the Vandermonde matrix to construct the coefficients a_k for the interpolation polynomial. This can be a very costly operation (as counted in clock cycles of a computer trying to do the job). By choosing another basis for Π_n we can simplify the calculation of the coefficients but then we have to do additional calculations when we want to express the interpolation polynomial in terms of a monomial basis.

One method is to write the interpolation polynomial in the Newton form and use the method of divided differences to construct the coefficients, e.g. Neville's algorithm. The cost is $O(n^2)$ operations, while Gaussian elimination costs $O(n^3)$ operations. Furthermore, you only need to do $O(n)$ extra work if an extra point is added to the data set, while for the other methods, you have to redo the whole computation.

Another method is to use the Lagrange form of the interpolation polynomial. The resulting formula immediately shows that the interpolation polynomial exists under the conditions stated in the above theorem. Lagrange formula is to be preferred to Vandermonde formula when we are not interested in computing the coefficients of the polynomial, but in computing the value of $p(x)$ in a given x not in the original data set. In this case, we can reduce complexity to $O(n^2)$.[5]

The Bernstein form was used in a constructive proof of the Weierstrass approximation theorem by Bernstein and has nowadays gained great importance in computer graphics in the form of Bézier curves.

Interpolation error [edit]


This section **may be confusing or unclear to readers**. *(June 2011)*

When interpolating a given function f by a polynomial of degree n at the nodes $x_0,\ldots,x_n$ we get the error

$$f(x) - p_n(x) = f[x_0, \ldots, x_n, x] \prod_{i=0}^{n} (x - x_i)$$

where

$$f[x_0, \ldots, x_n, x]$$

is the notation for divided differences.

If f is $n+1$ times continuously differentiable on a closed interval I and $p_n(x)$ be a polynomial of degree at most n that interpolates f at $n+1$ distinct points $\{x_i\}$ (i=0,1,...,n) in that interval. Then for each x in the interval there exists ξ in that interval such that

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

Proof [edit]

Set the error term as

$$R_n(x) = f(x) - p_n(x)$$

and set up an auxiliary function:

$$Y(t) = R_n(t) - \frac{R_n(x)}{W(x)} W(t)$$

where

$$W(u) = \prod_{i=0}^{n} (u - x_i)$$

Since x_i are roots of f and p_n, we have $Y(x) = Y(x_i) = 0$, which means Y has $n+2$ roots. From Rolle's theorem, $Y'(t)$ has $n+1$ roots, then $Y^{(n+1)}(t)$ has one root ξ, where ξ is in the interval I.

So we can get

$$Y^{(n+1)}(t) = R_n^{(n+1)}(t) - \frac{R_n(x)}{W(x)} (n+1)!$$

Since $p_n(x)$ is a polynomial of degree at most n, then

$$R_n^{(n+1)}(t) = f^{(n+1)}(t)$$

Thus

$$Y^{(n+1)}(t) = f^{(n+1)}(t) - \frac{R_n(x)}{W(x)}\ (n+1)!$$

Since ξ is the root of $Y^{(n+1)}(t)$, so

$$Y^{(n+1)}(\xi) = f^{(n+1)}(\xi) - \frac{R_n(x)}{W(x)}\ (n+1)! = 0$$

Therefore

$$R_n(x) = f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x - x_i)$$

Thus the remainder term in the Lagrange form of the Taylor theorem is a special case of interpolation error when all interpolation nodes x_i are identical.[6] Note that the error will be zero when $x = x_i$ for any *i*. Thus, the maximum error will occur at some point in the interval between two successive nodes.

In the case of equally spaced interpolation nodes where $x_0 = a$ and $x_i = a + ih$ for $i = 1, 2, \ldots, n$, where $h = (b-a)/n$, the product term in the interpolation error formula can be bound as

$$|\prod_{i=0}^{n}(x - x_i)| \leq \prod_{i=0}^{n}|(x - x_i)| \leq \frac{n!}{4}h^{n+1}.$$

Thus the error bound can be given as

$$|R_n(x)| \leq \frac{h^{n+1}}{4(n+1)}\max_{\xi\in[a,b]}|f^{(n+1)}(\xi)|$$

However, this assumes that $f^{(n+1)}(\xi)$ is dominated by h^{n+1}, i.e. $f^{(n+1)}(\xi)h^{n+1} << 1$. In several cases, this is not true and the error actually increases as $n \to \infty$ (see Runge's phenomenon). That question is treated in the section *Convergence properties*.

The above error bound suggests choosing the interpolation points x_i such that the product

$$\left|\prod(x - x_i)\right|,$$

is as small as possible. The Chebyshev nodes achieve this.

Lebesgue constants [edit]

See the main article: Lebesgue constant.

We fix the interpolation nodes $x_0, \ldots, x_n$ and an interval $[a, b]$ containing all the interpolation nodes. The process of interpolation maps the function *f* to a polynomial *p*. This defines a mapping *X* from the space $C([a, b])$ of all continuous functions on $[a, b]$ to itself. The map *X* is linear and it is a projection on the subspace Π_n of polynomials of degree *n* or less.

The Lebesgue constant *L* is defined as the operator norm of *X*. One has (a special case of Lebesgue's lemma):

$$\|f - X(f)\| \leq (L+1)\|f - p^*\|.$$

In other words, the interpolation polynomial is at most a factor $(L + 1)$ worse than the best possible approximation. This suggests that we look for a set of interpolation nodes that makes *L* small. In particular, we have for Chebyshev nodes:

$$L \leq \frac{2}{\pi}\log(n+1) + 1.$$

We conclude again that Chebyshev nodes are a very good choice for polynomial interpolation, as the growth in *n* is exponential for equidistant nodes. However, those nodes are not optimal.

Convergence properties [edit]

It is natural to ask, for which classes of functions and for which interpolation nodes the sequence of interpolating polynomials converges to the interpolated function as $n \to \infty$? Convergence may be understood in different ways, e.g. pointwise, uniform or in some integral norm.

The situation is rather bad for equidistant nodes, in that uniform convergence is not even guaranteed for infinitely differentiable functions. One classical example, due to Carl Runge, is the function $f(x) = 1 / (1 + x^2)$ on the interval $[-5, 5]$. The interpolation error $\|f - p_n\|_\infty$ grows without bound as $n \to \infty$. Another example is the function $f(x) = |x|$ on the interval $[-1, 1]$, for which the interpolating polynomials do not even converge pointwise except at the three points $x = \pm 1, 0$.[7]

One might think that better convergence properties may be obtained by choosing different interpolation nodes. The following result seems to give a rather encouraging answer:

Theorem. For any function *f*(*x*) continuous on an interval [*a*,*b*] there exists a table of nodes for which the sequence of interpolating polynomials $p_n(x)$ converges to *f*(*x*) uniformly on [*a*,*b*].

Proof. It's clear that the sequence of polynomials of best approximation $p_n^*(x)$ converges to *f*(*x*) uniformly (due to Weierstrass approximation theorem). Now we have only to show that each $p_n^*(x)$ may be obtained by means of interpolation on certain nodes. But this is true due to a special property of polynomials of best approximation known from the Chebyshev alternation theorem. Specifically, we know that such polynomials should intersect *f*(*x*) at least $n + 1$ times. Choosing the points of intersection as interpolation nodes we obtain the interpolating polynomial coinciding with the best approximation polynomial.

The defect of this method, however, is that interpolation nodes should be calculated anew for each new function *f*(*x*), but the algorithm is hard to be implemented numerically. Does there exist a single table of nodes for which the sequence of interpolating polynomials converge to any continuous function *f*(*x*)? The answer is unfortunately negative:

Theorem. For any table of nodes there is a continuous function *f*(*x*) on an interval [*a*, *b*] for which the sequence of interpolating polynomials diverges on [*a*,*b*].[8]

The proof essentially uses the lower bound estimation of the Lebesgue constant, which we defined above to be the operator norm of X_n (where X_n is the projection operator on Π_n). Now we seek a table of nodes for which

$$\lim_{n\to\infty} X_n f = f, \text{ for every } f \in C([a, b]).$$

Due to the Banach–Steinhaus theorem, this is only possible when norms of X_n are uniformly bounded, which cannot be true since we know that

$$\|X_n\| \geq \tfrac{2}{\pi}\log(n+1) + C.$$

For example, if equidistant points are chosen as interpolation nodes, the function from Runge's phenomenon demonstrates divergence of such interpolation. Note that this function is not only continuous but even infinitely times differentiable on $[-1, 1]$. For better Chebyshev nodes, however, such an example is much harder to find due to the following result:

Theorem. For every absolutely continuous function on $[-1, 1]$ the sequence of interpolating polynomials constructed on Chebyshev nodes converges to *f*(*x*)

uniformly.[*citation needed*]

Related concepts [edit]

Runge's phenomenon shows that for high values of n, the interpolation polynomial may oscillate wildly between the data points. This problem is commonly resolved by the use of spline interpolation. Here, the interpolant is not a polynomial but a spline: a chain of several polynomials of a lower degree.

Interpolation of periodic functions by harmonic functions is accomplished by Fourier transform. This can be seen as a form of polynomial interpolation with harmonic base functions, see trigonometric interpolation and trigonometric polynomial.

Hermite interpolation problems are those where not only the values of the polynomial p at the nodes are given, but also all derivatives up to a given order. This turns out to be equivalent to a system of simultaneous polynomial congruences, and may be solved by means of the Chinese remainder theorem for polynomials. Birkhoff interpolation is a further generalization where only derivatives of some orders are prescribed, not necessarily all orders from 0 to a k.

Collocation methods for the solution of differential and integral equations are based on polynomial interpolation.

The technique of rational function modeling is a generalization that considers ratios of polynomial functions.

At last, multivariate interpolation for higher dimensions.

See also [edit]

- Newton series

Notes [edit]

1. ^ Gautschi, Walter (1975). "Norm Estimates for Inverses of Vandermonde Matrices". *Numerische Mathematik* **23** (4): 337–347. doi:10.1007/BF01438260.
2. ^ Higham, N. J. (1988). "Fast Solution of Vandermonde-Like Systems Involving Orthogonal Polynomials". *IMA Journal of Numerical Analysis* **8** (4): 473–486. doi:10.1093/imanum/8.4.473.
3. ^ Björck, Å; V. Pereyra (1970). "Solution of Vandermonde Systems of Equations". *Mathematics of Computation* (American Mathematical Society) **24** (112): 893–903. doi:10.2307/2004623. JSTOR 2004623.
4. ^ Calvetti, D and Reichel, L (1993). "Fast Inversion of Vanderomnde-Like Matrices Involving Orthogonal Polynomials". *BIT* **33** (33): 473–484. doi:10.1007/BF01990529.
5. ^ R.Bevilaqua, D. Bini, M.Capovani and O. Menchi (2003). *Appunti di Calcolo Numerico*. Chapter 5, p. 89. Servizio Editoriale Universitario Pisa - Azienda Regionale Diritto allo Studio Universitario.
6. ^ "Errors in Polynomial Interpolation" (PDF).
7. ^ Watson (1980, p. 21) attributes the last example to Bernstein (1912).
8. ^ Watson (1980, p. 21) attributes this theorem to Faber (1914).

References [edit]

- Atkinson, Kendell A. (1988), "Chapter 3.", *An Introduction to Numerical Analysis* (2nd ed.), John Wiley and Sons, ISBN 0-471-50023-2
- Bernstein, Sergei N. (1912), "Sur l'ordre de la meilleure approximation des fonctions continues par les polynômes de degré donné" [On the order of the best approximation of continuous functions by polynomials of a given degree], *Mem. Acad. Roy. Belg.* (in French) **4**: 1–104
- Brutman, L. (1997), "Lebesgue functions for polynomial interpolation — a survey", *Ann. Numer. Math.* **4**: 111–127
- Faber, Georg (1914), "Über die interpolatorische Darstellung stetiger Funktionen" [On the Interpolation of Continuous Functions], *Deutsche Math. Jahr.* (in German) **23**: 192–210
- Powell, M. J. D. (1981), "Chapter 4", *Approximation Theory and Methods*, Cambridge University Press, ISBN 0-521-29514-9
- Schatzman, Michelle (2002), "Chapter 4", *Numerical Analysis: A Mathematical Introduction*, Oxford: Clarendon Press, ISBN 0-19-850279-6
- Süli, Endre; Mayers, David (2003), "Chapter 6", *An Introduction to Numerical Analysis*, Cambridge University Press, ISBN 0-521-00794-1
- Watson, G. Alistair (1980), *Approximation Theory and Numerical Methods*, John Wiley, ISBN 0-471-27706-1

External links [edit]

- Hazewinkel, Michiel, ed. (2001), "Interpolation process", *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
- Interpolating Polynomial by Stephen Wolfram, the Wolfram Demonstrations Project.

Categories: Interpolation | Polynomials