INTERPOLATION

COLTON GRAINGER

1. Polynomial approximation

To take intuition from Wikipedia[1]:

> Polynomials can be constructed from constants and indeterminates using only addition and multiplication.

The implementation of said addition and multiplication, of course, should optimized according to the set from which these constants are chosen.[2]

We'll make concrete use of the Weierstrass Approximation Theorem[4] (only later generalized to $C(X)$, the ring of continuous functions on a compactum X with the topology of uniform convergence, by Stone[5]):

!THM (Weierstrass Approximation) Given a continuous function $f\colon [a,b] \to \mathbf{R}$ for all $\varepsilon > 0 \ldots$ there exists a polynomial $p \in \mathbf{R}[x]$ such that for all $x \in [a,b]$ we have $|f(x) - p(x)| < \varepsilon$.

For notation's sake, we also define a general polynomial ring[6].

!DEF (The polynomial ring $\mathbf{K}[X]$) Let $\mathbf{K}$ be a field. By the ring of polynomials in the indeterminate, X, written as $\mathbf{K}[X]$, we mean ... the set of all symbols $\sum_{i=1}^{n} a_i X^i$ where $n \in \mathbf{N}$ and the coefficients a_i are elements of the field $\mathbf{K}$.

Here's the application of polynomial approximation in numerical quadrature:

- I desire $\int_a^b f(x)\,dx$.
- I can bound $|\int_a^b f(x)\,dx - \int_a^b p(x)\,dx| \leq \varepsilon(b-a)$.
- To obtain the desired precision, I ought to control ε.

The set of Taylor polynomials is an obvious candidate to approximate a differentiable function. Yet, looking at Taylor's theorem, we might not have a promising error bound our interval of integration.

!FACT (Taylor error bounds) Let $f(x)$ be defined on $[a,b]$ and suppose $f^{(n+1)}$ is (defined and) continuous on $[a,b]$. For any $x, x_0 \in [a,b]$ there exists $\xi(x) = \xi$ between x and x_0 such that, where p_n is the nth degree Taylor polynomial

---

Compiled: 2018-09-30.

[1] https://en.wikipedia.org/wiki/Polynomial

[2] R. Brent and P. Zimmermann, *Modern Computer Arithmetic*[3]. Version 0.5.9. Retrieved July 17, 2018.

> We shall see that many algorithms for polynomial arithmetic are similar to the corresponding algorithms for integer arithmetic, but simpler due to the lack of carries in polynomial arithmetic. Consider for example addition: the sum of two polynomials of degree n always has degree at most n, whereas the sum of two n -digit integers may have $n + 1$ digits. Thus, we often describe algorithms for polynomials as an aid to understanding the corresponding algorithms for integers. (p. 1)
>
> A strong link exists between modular arithmetic and arithmetic on polynomials. One way of implementing finite fields $\mathbf{F}_q$ with $q = p^n$ elements is to work with polynomials in $\mathbf{F}_p[x]$, which are reduced modulo a monic irreducible polynomial $f(x) \in \mathbf{F}_p[x]$ of degree n. In this case, modular reduction happens both at the coefficient level (in F_p) and at the polynomial level (modulo $f(x)$). (p. 49)

[4] https://en.wikipedia.org/wiki/Stone%E2%80%93Weierstrass_theorem

[5] https://en.wikipedia.org/wiki/Marshall_Harvey_Stone

[6] https://en.wikipedia.org/wiki/Polynomial_ring

centered at x_0, the remainder $r = f - p_n$ is given ...

$$r(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}.$$

For example, the Taylor series representation of the function $f(x) = \frac{1}{1+25x^2}$ only converges uniformly for $|x| < 0.2$.

2. Lagrange form

Consider $f \in C[a, b]$ (the set of continuous real-valued functions on the compact interval $[a, b]$), and let $x_0, \ldots, x_n$ be $n + 1$ distinct points. Does there exist polynomial of least degree $p \in \mathbf{R}[x]$ which *interpolates* f at the given points? such that $f(x_i) = p(x_i)$ for all $i \in \{0, 1, \ldots, n\}$? Is it unique?

The fundamental theorem of algebra guarantees uniqueness,[7] and Lagrange's construction demonstrates existence.

DEF! Having $n + 1$ distinct points x_i specified, the $n + 1$ Lagrange polynomials ℓ_k are written as the product ...

$$\ell_k(x) = \prod_{i \neq k} \frac{x - x_i}{x_k - x_i} \text{ for all } k \in \{0, 1, \ldots, n\}.$$

It can be shown that the set of $n + 1$ Lagrange polynomials spans $\mathcal{P}_n$—the vector space of polynomials of degree at most n over the field $\mathbf{R}$. We note the Lagrange polynomials form a basis for $\mathcal{P}_n$ since $\dim(\mathcal{P}_n) = n + 1$.

Now for Lagrange's construction, which specifies the linear combination of basis vectors $\{\ell_k : k = 0, 1, \ldots, n\}$.

DEF! (Lagrange form) Let $f \in C[a, b]$, and specify $n + 1$ distinct points x_i. The interpolating polynomial in Lagrange form is $p_n \in \mathcal{P}_n$ where ...

$$p_n(x) = \sum_{k=0}^{n} f(x_k)\ell_k(x).$$

FACT! The degree of a Lagrange polynomial generated by $n + 1$ points is ... n.

FACT! Evaluated at any of the $n + 1$ generating points x_i, the Lagrange kth polynomial is equal to ... the Kroenecker delta

$$\delta_{ik} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{else.} \end{cases}$$

The least degree interpolating polynomial we sought the existence of is p_n. Moreover, p_n is a unique vector in $\mathcal{P}_n$ (though it may have different representations, which can be more or less computationally efficient, by change of bases), so existence proof for interpolating polynomials (generated by a finite number of distinct points) is finished.

Coming up:

- What is the computational cost to evaluate $p_n(x)$ at $x \neq x_i$?
- How large is the error $|f(x) - p_n(x)|$ at $x \neq x_i$?
- Can we guarantee the error bound shrinks as we increment the number of generating points x_i?
- What is the computational complexity of an algorithm to generate the Lagrange form of the interpolating polynomial?
- Can we reduce computational complexity by iteratively generating more accurate approximations to a given function? that is, by using successively higher degree interpolating polynomials?
- What criteria do we have to choose higher-degree polynomials (from an otherwise underdetermined solution space)?

[7] The difference between any two least degree interpolating polynomials is the zero polynomial.

3. Newton form

An example first. Consider the linear interpolant (degree $n = 1$) between $(x_0, f(x_0))$ and $(x_1, f(x_1))$.

$$p_1(x) = f(x_1)\ell_1(x) + f(x_1)\ell_1(x) \tag{1}$$

$$= f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0} \qquad \text{(Lagrange form)} \tag{2}$$

$$= f(x_0) + f(x_1)\frac{f(x_1) - f(x_0)}{x_1 - x_0} \qquad \text{(Newton form)} \tag{3}$$

Newton's form allows us to compute p_n from p_{n-1} recursively. That is, we define p_n with the same $n - 1$ basis vectors and $n - 1$ coordinates as p_{n-1}, but add a basis vector that's 0 at all points x_i with $i < n$.

Namely,

$$p_n(x) = p_{n-1}(x) + a_n \prod_{i<n}(x - x_i)$$

with

$$a_n = (f(x_n) - p_{n-1}(x_n))\left(\prod_{i<n}(x_n - x_i)\right)^{-1}.$$

In practice we express the coordinate a_n in recursive *divided differences*, i.e.,

$$a_n = f[x_0, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}$$

where the identification $f[x] = f(x)$ for all x is the base for recursion.

The Newton form of the interpolating polynomial is

$$p_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1})$$

or, from another view,

$$p_n(x) = a_0 + (x - x_0)\Big(a_1 + (x - x_1)\Big(a_2 + \cdots + a_n(x - x_{n-1})\cdots\Big)\Big).$$

Redistributing the parentheses by nesting them, we can evaluate $p_n(x)$ with $O(n)$ operations, namely, n multiplications and n additions.

4. Matrix interpretation

For $n + 1$ generating points $(x_i, f(x_i))$, there is a unique interpolating polynomial p_n of degree n.

But p_n is just point in the vector space $\mathcal{P}_n$, so we should try to express the constraint that $p_n(x_i) = f(x_i)$ for all $i \in \{0, 1, \ldots, n\}$ as $n + 1$ linear equations to determine the coordinates of p_n.

We are required to choose a basis for $\mathcal{P}_n$, and we've thus far seen three:

- the standard basis $\{1, x, x^2, \ldots, x^n\}$
  - yields a power series approximation[8] to f
- the Lagrange polynomials $\{\ell_0, \ell_1, \ldots, \ell_n\}$
  - produces the Lagrange form of the interpolating polynomial

[8]Indeed, the standard basis *is a minimal spanning set* for $\mathcal{P}_n$, so the divergence of Taylor series representation cannot be blamed on having the wrong vectors to create p_n. Instead, we find that the Taylor series representation implements a differential (rather than linear algebraic) algorithm for reducing a highly ill-conditioned system, which, in some cases, as when the function to be interpolated is analytic, proceeds, but fails for merely continuous functions.

- the basis of the Newton form $\{1, (x - x_0), \ldots, (x - x_0) \cdots (x - x_n)\}$
  - gives rise to the Newton/nested form

The constraint as a matrix equation (in any basis) is

$$A \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

For the standard basis, A is the Vandermode matrix, which tends to be ill-conditioned.

For the Lagrange polynomial basis, we have simply[9] $A = I$.

For the basis of the Newton form, A is a lower triangular matrix. Solving $A\vec{a} = \vec{f}$ for the coordinates $\vec{a}$ requires $n^2/2$ operations—the same required to compute with a divided difference table. In practice, we use divided differences to avoid storing the matrix elements in memory.

5. Error bounds

THEOREM! Let $f \in C^{n+1}[a, b]$, the x_i be $n + 1$ distinct points in $[a, b]$, and suppose p_n is a polynomial of degree at most n that interpolates f at the x_i. Given $x \in [a, b]$, there exists a point $\xi \in (a, b)$ such that ...

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

[9] It's computing the basis that's expensive!