Math 104A - Numerical Analysis I

Approximation of functions

Jea-Hyun Park

University of California Santa Barbara

Fall 2022

Introduction

Polynomial Interpolation

Throughout the section, we want to answer the problem (and some important properties): given the data below, find a polynomial $y = p(x)$ of minimal degree *interpolating* it.

x	x_0	x_1	x_2	$\cdots$	x_n
y	y_0	y_1	y_2	$\cdots$	y_n

- **Subjective questions**: Does this problem make sense? What should we check to make this problem meaningful? What do your guts tell you before even start studying it?

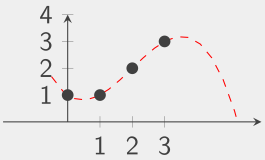
Theorem (Unique interpolation theorem)

If $x_0, x_1, \cdots, x_n$ are distinct real, for arbitrary values $y_0, y_1, \cdots, y_n$, there is a unique polynomial $p \in \Pi_n$ such that $p(x_i) = y_i$ $(0 \leq i \leq n)$.

Proof 1.

Vandermonde – next few slides . $\square$

- Notation: $\Pi_n := \{$ polynomials of degree at most $n\}$.
- Notice that the degrees of freedom match: (n+1) values to interpolate and (n+1) coefficients we can tune in $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$.



Vandermonde matrix

Idea: Brute force.
Set $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$, and require the conditions.

$$
\begin{aligned}
p(x_0) &= a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n = y_0 \\
p(x_1) &= a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n = y_1 \\
&\vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
p(x_n) &= a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n = y_n
\end{aligned}
$$

In matrix form,

$$
\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}
=
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}
$$

- The coefficient matrix is called **Vandermonde matrix**.

Vandermonde matrix

Theorem

$\det(V) = \prod_{0\le i<j\le n}(x_j - x_i)$, *where* V *is the Vandermonde matrix:*

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix}.$$

Proof.

1. $\det(V)$ is a polynomial in $x_0, x_1, \cdots, x_n$.
2. If $x_0 = x_1$, the first two rows are identical. Thus, $\det(V) = 0$. The factor theorem asserts $(x_0 - x_1)$ divides $\det(V)$. (Think of x_0 as a variable, say x, and x_1 as a number, say 1, and plug in $x = 1$.) Do the same for $x_i = x_j$ ($i \neq j$) and conclude $(x_i - x_j)$ divides $\det(V)$. Therefore, $\det(V) = (\text{something})\prod_{0\le i<j\le n}(x_j - x_i)$.

Proof.

3. Recall Liebniz formula for the determinant: sum of $\pm$(product of entries taken from distinct columns while scanning rows from the top to the bottom). '$+$' is assigned when the order of column index chosen is an *even permutation* of $(0, 1, \cdots, n)$ and '$-$' when it is an *odd permutation*.
4. Observe that 'something' must be a constant since the order of the polynomial is $n(n+1)/2 = 0 + 1 + 2 + \cdots + n$ from both Liebniz formula and the product form.
5. Comparing the term $x_1 x_2^2 \cdots x_n^n$, we realize that the constant must be 1: this term appears only once with '$+$' in the Liebniz formula and we have (something)$x_1 x_2^2 \cdots x_n^n$ by expanding (something) $\prod_{0 \le i < j \le n} (x_j - x_i)$ choosing only x_j's.

$\square$

Theorem (Unique interpolation theorem)

If $x_0, x_1, \cdots, x_n$ are distinct real, for arbitrary values $y_0, y_1, \cdots, y_n$, there is a unique polynomial $p \in \Pi_n$ such that $p(x_i) = y_i$ $(0 \leq i \leq n)$.

Proof 1.

Since the nodes are distinct, the determinant of the Vandermonde matrix $\det(V) = \prod_{0 \leq i < j \leq n} (x_j - x_i)$ nonzero, hence the matrix is invertible. Therefore, we have a unique solution $[a_0, a_1, \cdots, a_n]^T$ in the Vandermonde system for any prescribed $[y_0, y_1, \cdots, y_n]^T$. That is, $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \in \Pi_n$ is the unique polynomial we want. $\square$

Theorem (Unique interpolation theorem - duplicate)

If $x_0, x_1, \cdots, x_n$ are distinct real, for arbitrary values $y_0, y_1, \cdots, y_n$, there is a unique polynomial $p \in \Pi_n$ such that $p(x_i) = y_i$ $(0 \le i \le n)$.

Proof 2.

Board work. □

Example

Following the previous proof, find a polynomial (of minimal degree) interpolating

x	0	1	2	3
y	1	1	2	3

- The way the polynomial organized in the proof is called **Newton form**.

Horner's algorithm: evaluating polynomials

By storing coefficients, we can only *encode* a polynomial

$$\begin{aligned} p(x) &= 1 + 0 \cdot x + \frac{1}{2}x(x-1) - \frac{1}{6}x(x-1)(x-2) \\ &= -\frac{1}{6}x^3 + x^2 - \frac{5}{6}x + 1. \end{aligned}$$

We need to compute the output just to know one function value.

For practical reasons, the following **nested multiplication** or **Horner's algorithm** is better than following the math expression.

$$\left(\left(-\frac{1}{6}(x-2) + \frac{1}{2}\right)(x-1)\right)x + 1$$

In algorithm form, this reads much nicer:

$u \leftarrow c_k$;
for $i \leftarrow k-1$ **to** 0 **do**
$\quad u \leftarrow (t - x_i)u + c_k$;
end

- This is **only for evaluating** a polynomial after finding an interpolation. Don't mix this with how to find Newton form interpolations.
- Multiplications are more expensive than additions in computing. Count the multiplications to see the difference.
- This is purely computational. Mathematically, they are the same.

Idea: Find a basis of Π_n that makes interpolating procedure simple.
In particular, if we can find $\ell_i(x) \in \Pi_n$ such that

$$\ell_i(x_j) = \delta_{ij}, \tag{1}$$

then, (we will call the way it's written **Lagrange form**)

$$p(x) = \sum_{i=0}^{n} y_i \ell_i(x).$$

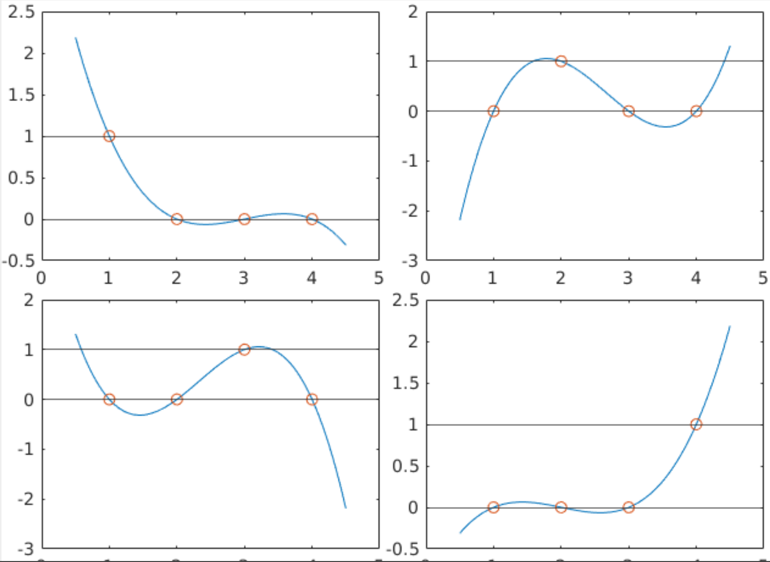
Definition (Lagrange basis or cardinal functions)

For a given set of distinct abscissas $\{x_i\}_{i=0}^n$,

$$\ell_i = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}, \qquad (0 \leq i \leq n) \tag{2}$$

are called **Lagrange basis** or **cardinal functions** associated to/subordinate to $\{x_i\}_{i=0}^n$.

- Π_n is a vector space.
- **Notation**: (**Kronecker delta**)
  $$\delta_{ij} = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$$
- 'abscissas' means horizontal coordinates $\{x_i\}_{i=0}^n$, .



- ℓ_0 (top left), ℓ_1 (top right), ℓ_2 (bottom left), ℓ_3 (bottom right)
- **Notation**: ℓ_i's will be reserved to be Lagrange basis functions from now on.

Lagrange interpolation

Theorem

If a set of functions $\{f_i(x)\}_{i=0}^n$ satisfies $f_i(x_j) = \delta_{ij}$, then it is linearly independent.

Proof.

Board work. $\square$

Corollary

Lagrange basis is indeed a basis.

Proof.

Since $\dim(\Pi_n) = \#\{\ell_i(x)\}_{i=0}^n = n+1$, it suffices to show linear independence. Observe $\ell_i(x_j) = \prod_{k\neq i} \frac{x_j - x_k}{x_i - x_k} = 0$ if $j \neq i$ (one of the numerator is zero) and, if $j = i$, $\ell_i(x_i) = \prod_{k\neq i} \frac{x_i - x_k}{x_i - x_k} = 1$. Linear independence follows from the previous theorem. $\square$

- $p(x) = \sum_{i=0}^n y_i \ell_i(x)$ means that just putting the data as coordinates (or coefficients) of the Lagrange basis, you have the interpolation.
- This result can be considered 3rd proof of the polynomial interpolation theorem.

Example

Following the previous proof, find a polynomial (of minimal degree) interpolating

x	0	1	2	3
y	1	1	2	3

Polynomial interpolation comparison

	Vandermonde	Lagrange	Newton
Basis	$1, x, x^2 \cdots$	$\ell_i = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$	$1, (x - x_0), (x - x_0)(x - x_1), \cdots$
Theory	Some algebraic beauty	Convenient for **Lagrange interpolation** (i.e., only function values involved)	Effective for **Hermite interpolation** (i.e., also derivatives involved)
Numerical	Inaccurate and inefficient: the matrix is *ill-conditioned* and inverting a matrix is among expensive computations	Efficient when nodes are fixed but possibly the data to fit changes (Lagrange basis depends only on the nodes)	Efficient when nodes gets added (a newly added term does not affect the previous interpolations). Also, finding coefficients can be efficient when equipped with **divided difference**.[1]
Evaluation algorithm	Horner	Some algorithms exist	Horner

[1] Text in blue: next topics.

Theorem

Let $x_0, x_1, \cdots, x_n \in [a, b]$ be distinct nodes, $f \in C^{n+1}[a, b]$, and $p \in \Pi_n$ interpolating f at the nodes. For each $x \in [a, b]$, there is $\xi_x \in (a, b)$ such that

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

Proof.

Board work. $\square$

- This theorem is of great importance later on when we analyze numerical methods. There will be no waste of time appreciating detailed aspects of this theorem.

Example

Find a bound on errors made by the polynomial interpolation of $f(x) = \sin(x)$ at 11 distinct nodes on $[0, 1]$. What if we require the nodes to be equally spaced?

- **Subjective question**: Can you conjecture how good interpolations are depending on the choice of nodes? Feel free to say what your guts tell you, then modify it if needed.

Question: For a very smooth function, say, $f \in C^\infty[-1, 1]$, imagine what polynomial interpolations will be like if you use equally spaced nodes? What will happen as we increase the nodes?

Example (Runge's phenomenon)

Dynamic example of

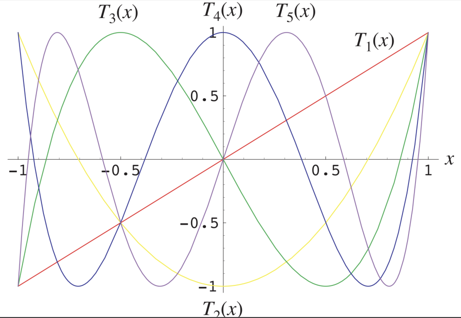
$$\frac{1}{1+25x^2}$$

- Thank you, Cleve Moler, for this example.

Motivation: Though we will not be able to discuss the full picture, some "best" interpolation is related to **Chebyshev polynomials**.

Definition (Chebyshev polynomials - 1st kind)

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad (n \geq 1)$$



- When we talk about Chebyshev polynomials, we are interested in the domain $[-1, 1]$ though they are defined everywhere.
- First few are:

$$T_2(x) = 2x^2 - 1$$
$$T_3(x) = 4x^3 - 3x$$
$$T_4(x) = 8x^4 - 8x^2 + 1$$

- We omit "1st kind" from now on.

Chebyshev polynomials

Theorem

For $x \in [-1,1]$, we have

$$T_n(x) = \cos\left(n \cos^{-1} x\right) \quad (n \geq 0)$$

Proof.

Board work. $\square$

Corollary

$$|T_n(x)| \leq 1 \qquad (-1 \leq x \leq 1)$$

$$T_n\left(\cos \frac{j\pi}{n}\right) = (-1)^j \qquad (0 \leq j \leq n)$$

$$T_n\left(\cos \frac{2j-1}{2n}\pi\right) = 0 \qquad (1 \leq j \leq n)$$

Theorem

Monic polynomials satisfy

$$\|p\|_\infty = \max_{-1\le x\le 1} |p(x)| \ge 2^{1-n}.$$

Proof.

Board work. □

Theorem

If the nodes are zeros of T_{n+1}, then we have, for $|x| \le 1$,

$$|f(x) - p(x)| \le \frac{1}{2^n(n+1)!} \max_{|t|\le 1} \left| f^{(n+1)}(t) \right|$$

Proof.

Board work. □

Question: We saw a bad interpolating result for the Runge's function. What if we use Chebyshev nodes?

Example (Runge's phenomenon)

Dynamic example of

$$\frac{1}{1+25x^2}$$

- As the name suggests, **Chebyshev nodes** are the ones consist of the zeros of Chebyshev polynomials.

Here is some high level summary, which I believe is good enough for the very first course of numerical analysis.

- If a function is very well-behaving (like $\sin(x)$), reasonable polynomial interpolation (e.g., equally spaced ones) works well.
- Even if a function looks well-behaving (like $1/(1+25x^2)$), equally spaced nodes may not work. (To distinguish these two, we need to look through complex analysis lens.)
- If we choose a good set of nodes, the interpolation can be very satisfying (e.g., Runge's function with Chebyshev nodes)
- (**Weierstrass Approximation Theorem**) For any continuous function, we can find as good polynomial approximations as we please. (But it does not tell us how.) That is, let $f \in C[a, b]$, then, for $\forall \epsilon > 0$, there is a polynomial p such that $\|f - p\|_\infty < \epsilon$.

- As you have seen, polynomial interpolation is subtle and requires a deep dive for a better picture.
- Noticed $\frac{1}{1+25x^2}$ has singularities at $\pm\frac{\sqrt{-1}}{5}$. (We don't pursue this any further.)
- "Fix nodes first, then you can always find a bad function. Conversely, fix a function, then you can always find good nodes."

Polynomial Interpolation

Divided Differences

Setting: given a function f and nodes $x_0, x_1, \cdots, x_n$, find $p \in \Pi_n$ interpolating f, i.e., $p(x_i) = f(x_i)$, $(0 \leq i \leq n)$.

Example

Given nodes x_0, x_1, x_2 find the interpolating polynomial (of minimal degree) in Newton form. What does each coefficient depends on.

- Refresher: There is a unique interpolating polynomial given nodes and data. But the way it is written makes a huge (practical) difference.

Definition (Divided differences)

Given a function f and nodes $x_0, x_1, \cdots, x_n$, suppose $p(x) = \sum_{k=0}^{n} c_k q_k(x)$ is the polynomial interpolating f at the nodes in Newton form, where $q_k(x) = \prod_{j=0}^{k-1}(x - x_j)$, $(0 \leq k \leq n)$ is the basis of Newton form. Then, **divided differences** are defined to be the coefficients

$$f[x_0, x_1, \cdots, x_k] := c_k.$$

Corollary

Under the same assumptions as above,

$$\begin{aligned} p(x) &= \sum_{k=0}^{n} f[x_0, x_1, \cdots, x_k] q_k(x) \\ &= \sum_{k=0}^{n} f[x_0, x_1, \cdots, x_k] \prod_{j=0}^{k-1}(x - x_j). \end{aligned}$$

- **Convention**: $\sum_{k=0}^{-1} a_k = 0$ and $\prod_{k=0}^{-1} a_k = 1$. In words, "if a product or a sum does not make sense, assign it a value that has the same effect of doing nothing."
- The notation $f[x_0, x_1, \cdots, x_k]$ emphasizes it depend on f and the nodes only up to index k.

Newton form and divided differences

Theorem (Recursive relation of divided differences)

$$f\left[x_0, x_1, \dots, x_n\right] = \frac{f\left[x_1, x_2, \dots, x_n\right] - f\left[x_0, x_1, \dots, x_{n-1}\right]}{x_n - x_0}$$

Proof.

Board work. $\square$

Example

Use the above formula to find a polynomial (of minimal degree) interpolating

x	0	1	2	3
y	1	1	2	3

- Mnemonic device: (a) looks similar to finite difference approximation of derivatives (in fact, this is exactly true for $f[x_0, x_1]$), (b) numerator index – last n minus first; numerator index – last one minus first one.

Algorithm for divided differences is very efficient.

Algorithm 1: Divided differences

```
for i = 0 to n do
    d_i ← f(x_i);
end
for j = 0 to n do
    for i = n to j do
        d_i ← (d_i − d_{i−1})/(x_i − x_{i−j});
    end
end
```

Then, the interpolating polynomial is

$$p(x) = \sum_{i=0}^{n} d_i \prod_{j=0}^{i-1} (x - x_j).$$

- See the textbook pp. 331-332 for details of algorithm. We focus on other properties and applications of divided differences.

Theorem (Symmetry of divided differences)

If $(z_0, z_1, \cdots, z_n)$ *is a permutation of* $(x_0, x_1, \cdots, x_n)$, *then*

$$f[z_0, z_1, \cdots, z_n] = f[x_0, x_1, \cdots, x_n]$$

Proof.

Board work. □

Theorem (Error of polynomial interpolation)

Let $x_0, x_1, \cdots, x_n \in [a, b]$ *be distinct nodes and be* $p \in \Pi_n$ *interpolating* f *at the nodes. For each* $t \in [a, b]$ *different from the nodes, we have*

$$f(x) - p(x) = f[x_0, x_1, \cdots, x_n, t] \prod_{i=0}^{n} (x - x_i)$$

- Permutation means a shuffle.

Properties of divided differences

Theorem (Error of polynomial interpolation)

Let $x_0, x_1, \cdots, x_n \in [a, b]$ be distinct nodes and be $p \in \Pi_n$ interpolating f at the nodes. For each $t \in [a, b]$ different from the nodes, we have

$$f(t) - p(t) = f[x_0, x_1, \cdots, x_n, t] \prod_{i=0}^{n} (t - x_i)$$

Proof.

Board work. □

Theorem (Discrete derivatives)

Let $x_0, x_1, \cdots, x_n \in [a, b]$ be distinct nodes. If $f \in C^n[a, b]$, there is $\xi \in (a, b)$ such that

$$f[x_0, x_1, \cdots, x_n] = \frac{1}{n!} f^{(n)}(\xi)$$

- Permutation means a shuffle.

Polynomial Interpolation

Hermite Interpolation

Motivation: We may want to interpolate not only the function values but also its slopes or curvatures too to get a high quality approximation. Dynamic example

Example

Find a polynomial that "interpolates" $f \in C^1$ satisfying $f(0) = 0, f(1) = 1, f'(\frac{1}{2}) = 2$.

- This example shows interpolating derivatives must be posed in a certain way if we want a simple, systematic solution.

Hermite interpolation

Theorem

Given distinct nodes $x_0, x_1, \cdots, x_n$, for any $c_{ij} \in \mathbb{R}$ ($\forall i, j$ that makes sense), there exists a unique polynomial $p \in \Pi_m$ satisfying

$$p^{(j)}(x_i) = c_{ij}, \qquad (0 \leq j \leq k_i - 1,\ 0 \leq i \leq n),$$

where $k_i \geq 1$ and $m + 1 = k_0 + k_1 + \cdots + k_n$.

Proof.

Board work. $\square$

Example

Find a Hermite interpolation with only one node: $p^j(a) = c_j$, $(0 \leq j \leq n)$

- Notice that the degrees of freedom match: $\dim(\Pi_m) = \deg(p) + 1$ is equal to #conditions prescribed.
- The most useful special case is when $k_i = 2$ $(\forall i)$: there is a unique $p \in \Pi_{2n-1}$ such that $p(x_i) = y_i$ and $p'(x_i) = y_i'$.

Lemma

If $f \in C^1[a,b]$,

$$\lim_{x \to x_0} f[x_0, x] = f'(x_0)$$

Proof.

$$\lim_{x \to x_0} f[x_0, x] = \frac{f(x) - f(x_0)}{x - x_0} = f'(x_0)$$

□

This motivates the following definition.

Definition (Divided differences with repeated nodes)

If $f \in C^1[a,b]$, $f[x_0, x_0] := f'(x_0)$.

Similarly,

Lemma

If $f \in C^k[a,b]$ and $a \le x_0 \le x_1 \le \cdots \le x_k \le b$,

$$\lim_{x_k \to x_0} f[x_0, x_1, \cdots, x_k] = \frac{f^{(k)}(x_0)}{k!}$$

Proof.

Board work. □

Definition (Divided differences with repeated nodes)

For $k \ge 0$, if $f \in C^k[a,b]$,

$$f[\underbrace{x_0, \cdots, x_0}] := \frac{f^{(k)}(x_0)}{k!}.$$

Example

Find the polynomial of minimal degree such that $p(x_0) = f(x_0)$, $p'(x_0) = f'(x_0)$, $p(x_1) = f(x_1)$, $p'(x_1) = f'(x_1)$.

Proof.

Boord work. $\square$

Example

Find the polynomial of minimal degree such that $p(1) = 2$, $p'(1) = 3$, $p(2) = 6$, $p'(2) = 7$, $p''(2) = 8$.

Proof.

Boord work. $\square$

- Be careful when working with second or higher degree: $f^{(k)}(x_i)/k!$ must be fed instead of $f^{(k)}(x_i)$!

Theorem

Let $x_0, x_1, \cdots, x_n \in [a, b]$ be distinct nodes and $f \in C^{2n+2}[a, b]$. If $p \in \Pi_{2n+1}$ such that $p(x_i) = f(x_i)$ and $p'(x_i) = f'(x_i)$ $(0 \leq k \leq n)$. Then, for any $x \in [a, b]$, there is $\xi \in (a, b)$ such that

$$f(x) - p(x) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \prod_{i=0}^{n} (x - x_i)^2 .$$

Proof.

Skip. It is very similar to the Lagrange interpolation case. $\square$