CMPSC/Math 451, Numerical Computation

Wen Shen

Department of Mathematics, Penn State University

Chapter 2: Polynomial Interpolation

In this Chapter we study how to interpolate a data set with a polynomial.

Problem description:
Given $(n+1)$ points, say (x_i, y_i), where $i = 0, 1, 2, \cdots, n$, with distinct x_i, not necessarily sorted, we want to find a polynomial of degree n,

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

such that it interpolates these points, i.e.,

$$P_n(x_i) = y_i, \qquad i = 0, 1, 2, \cdots, n$$

The goal is to determine the coefficients $a_n, a_{n-1}, \cdots, a_1, a_0$.

NB! The total number of data points is 1 larger than the degree of the polynomial.

Why should we do this? Here are some reasons:

- Find the values between the points for discrete data set;
- To approximate a (probably complicated) function by a polynomial;
- Then, it is easier to do computations such as derivative, integration etc.

Example 1. Interpolate the given data set with a polynomial of degree 2:

x_i	0	1	2/3
y_i	1	0	0.5

Answer. Let

$$P_2(x) = a_2x^2 + a_1x + a_0$$

We need to find the coefficients a_2, a_1, a_0.
By the interpolating properties, we have 3 equations:

$$\begin{aligned} x = 0,\ y = 1 \quad &: \quad P_2(0) = a_0 = 1 \\ x = 1,\ y = 0 \quad &: \quad P_2(1) = a_2 + a_1 + a_0 = 0 \\ x = 2/3,\ y = 0.5 \quad &: \quad P_2(2/3) = (4/9)a_2 + (2/3)a_1 + a_0 = 0.5 \end{aligned}$$

Here we have 3 linear equations and 3 unknowns (a_2, a_1, a_0).

The equations:

$$\begin{aligned} a_0 &= 1 \\ a_2 + \quad a_1 + a_0 &= 0 \\ \frac{4}{9}a_2 + \frac{2}{3}a_1 + a_0 &= 0.5 \end{aligned}$$

In matrix-vector form

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ \frac{4}{9} & \frac{2}{3} & 1 \end{pmatrix} \begin{pmatrix} a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0.5 \end{pmatrix}$$

Easy to solve in Matlab, or do it by hand:

$$a_2 = -3/4, \quad a_1 = -1/4, \quad a_0 = 1.$$

Then

$$P_2(x) = -\frac{3}{4}x^2 - \frac{1}{4}x + 1.$$

The general case. For the general case with $(n+1)$ points, we have

$$P_n(x_i) = y_i, \qquad i = 0, 1, 2, \cdots, n$$

We will have $(n+1)$ equations and $(n+1)$ unknowns:

$$\begin{array}{lcl} P_n(x_0) = y_0 & : & x_0^n a_n + x_0^{n-1} a_{n-1} + \cdots + x_0 a_1 + a_0 \;=\; y_0 \\ P_n(x_1) = y_1 & : & x_1^n a_n + x_1^{n-1} a_{n-1} + \cdots + x_1 a_1 + a_0 \;=\; y_1 \\ \vdots & & \\ P_n(x_n) = y_n & : & x_n^n a_n + x_n^{n-1} a_{n-1} + \cdots + x_n a_1 + a_0 \;=\; y_n \end{array}$$

Putting this in matrix-vector form

$$\begin{pmatrix} x_0^n & x_0^{n-1} & \cdots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n^n & x_n^{n-1} & \cdots & x_n & 1 \end{pmatrix} \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

i.e.

$$\mathbf{X}\,\vec{a} = \vec{y}$$

$$\mathbf{X}\,\vec{a} = \vec{y}$$

$\mathbf{X}$: $(n+1)\times(n+1)$ matrix, given
(It's called the van der Monde matrix)
$\vec{a}$: unknown vector, with length $(n+1)$
$\vec{y}$: given vector, with length $(n+1)$

Theorem: If x_i's are distinct, then $\mathbf{X}$ is invertible, therefore $\vec{a}$ has a unique solution.

In Matlab, the command `vander(x)`, where is a vector that contains the interpolation points x=$[x_1, x_2, \cdots, x_n]$), will generate this matrix.

Bad news: $\mathbf{X}$ has very large condition number for large n, therefore not effective to solve if n is large.

Other more efficient and elegant methods include

- Lagrange polynomials
- Newton's divided differences

Lagrange interpolation polynomials

Given points: $x_0, x_1, \cdots, x_n$

Define the cardinal functions $l_0, l_1, \cdots, l_n :\in \mathcal{P}^n$, satisfying the properties

$$l_i(x_j) = \delta_{ij} = \begin{cases} 1 & , \quad i = j \\ 0 & , \quad i \neq j \end{cases} \qquad i = 0, 1, \cdots, n$$

Here δ_{ij} is called the Kronecker's delta.

Locally supported in discrete sense.

The cardinal functions $l_i(x)$ can be written as

$$
\begin{aligned}
l_i(x) &= \prod_{j=0, j\neq i}^{n} \left(\frac{x - x_j}{x_i - x_j} \right) \\
&= \frac{x - x_0}{x_i - x_0} \cdot \frac{x - x_1}{x_i - x_1} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n}
\end{aligned}
$$

Verify:

$$l_i(x_i) = 1$$

and for $i \neq k$

$$l_i(x_k) = 0$$

$$l_i(x_k) = \delta_{ik}.$$

Lagrange form of the interpolation polynomial can be simply expressed as

$$P_n(x) = \sum_{i=0}^{n} l_i(x) \cdot y_i.$$

It is easy to check the interpolating property:

$$P_n(x_j) = \sum_{i=0}^{n} l_i(x_j) \cdot y_i = y_j, \qquad \text{for every} \quad j.$$

Example 2. Write the Lagrange polynomial for the data (same as in Example 1)

x_i	0	2/3	1
y_i	1	0.5	0

Answer. The data set corresponds to

$$x_0 = 0, \quad x_1 = 2/3, \quad x_2 = 1, \qquad y_0 = 1, \quad y_1 = 0.5, \quad y_2 = 0.$$

We first compute the cardinal functions

$$\begin{aligned} l_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-2/3)}{(0-2/3)}\frac{(x-1)}{(0-1)} = \frac{3}{2}\left(x-\frac{2}{3}\right)(x-1) \\ l_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-0)}{(2/3-0)}\frac{(x-1)}{(2/3-1)} = -\frac{9}{2}x(x-1) \\ l_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-0)}{(1-0)}\frac{(x-2/3)}{(1-2/3)} = 3x\left(x-\frac{2}{3}\right) \end{aligned}$$

so

$$\begin{aligned} P_2(x) &= l_0(x)y_0 + l_1(x)y_1 + l_2(x)y_2 = \frac{3}{2}\left(x-\frac{2}{3}\right)(x-1) - \frac{9}{2}x(x-1)(0.5) + 0 \\ &= \cdots = -\frac{3}{4}x^2 - \frac{1}{4}x + 1, \qquad \text{same as in Example 1} \end{aligned}$$

Pros and cons of Lagrange polynomial:

(+) Elegant formula,

(-) Slow to compute, each $l_i(x)$ is different,

(-) Not flexible: if one changes a points x_j, or add on an additional point x_{n+1}, one must re-compute all l_i's.

Newton's divided differences

Given a data set

x_i	x_0	x_1	$\cdots$	x_n
y_i	y_0	y_1	$\cdots$	y_n

We will describe an algorithm in a recursive form.

Main idea:
Given $P_k(x)$ that interpolates $k+1$ data points $\{x_i, y_i\}$, $i = 0, 1, 2, \cdots, k$, compute $P_{k+1}(x)$ that interpolates one extra point, $\{x_{k+1}, y_{k+1}\}$, by using P_k and adding an extra term.

- For $n = 0$, we set $P_0(x) = y_0$. Then $P_0(x_0) = y_0$.
- For $n = 1$, we set
$$P_1(x) = P_0(x) + a_1(x - x_0) \tag{1}$$
where a_1 is to be determined.
Then, $P_1(x_0) = P_0(x_0) + 0 = y_0$, for any a_1.
Find a_1 by the interpolation property $y_1 = P_1(x_1)$, we have
$$y_1 = P_0(x_1) + a_1(x_1 - x_0) = y_0 + a_1(x_1 - x_0).$$
This gives us
$$a_1 = \frac{y_1 - y_0}{x_1 - x_0}.$$

- For $n = 2$, we set

$$P_2(x) = P_1(x) + a_2(x - x_0)(x - x_1).$$

Then, $P_2(x_0) = P_1(x_0) = y_0, P_2(x_1) = P_1(x_1) = y_1$.

Determine a_2 by the interpolating property $y_2 = P_2(x_2)$.

$$y_2 = P_1(x_2) + a_2(x_2 - x_0)(x_2 - x_1),$$

Then

$$a_2 = \frac{y_2 - P_1(x_2)}{(x_2 - x_0)(x_2 - x_1)}.$$

We would like to express a_2 in a different way. Recall

$$P_1(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0).$$

Then

$$\begin{aligned} P_1(x_2) &= y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0) \\ &= y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_1) + \frac{y_1 - y_0}{x_1 - x_0}(x_1 - x_0) \\ &= y_1 + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_1). \end{aligned}$$

Then, a_2 can be rewritten as

$$a_2 = \frac{y_2 - P_1(x_2)}{(x_2 - x_0)(x_2 - x_1)} = \frac{y_2 - y_1 - \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}.$$

The general case for a_n:
Assume that $P_{n-1}(x)$ interpolates (x_i, y_i) for $i = 0, 1, \cdots, n-1$.
Let

$$P_n(x) = P_{n-1}(x) + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1})$$

Then for $i = 0, 1, \cdots, n-1$, we have

$$P_n(x_i) = P_{n-1}(x_i) = y_i.$$

Find a_n by the property $P_n(x_n) = y_n$,

$$y_n = P_{n-1}(x_n) + a_n(x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1})$$

then

$$a_n = \frac{y_n - P_{n-1}(x_n)}{(x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1})}$$

Newton's form for the interpolation polynomial:

$$\begin{aligned} P_n(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots \\ &\quad + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}), \end{aligned}$$

Newton's divided differences, recursive computation

The recursion is initiated with

$$f[x_i] = y_i, \qquad i = 0, 1, 2, \cdots$$

Then

$$f[x_0, x_1] = \frac{f[x_1] - f[x_0]}{x_1 - x_0}, \quad f[x_1, x_2] = \frac{f[x_2] - f[x_1]}{x_2 - x_1}, \quad \cdots$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_1, x_0]}{x_2 - x_0}, \quad f[x_1, x_2, x_3] = \frac{f[x_3, x_2] - f[x_2, x_1]}{x_3 - x_1}, \quad \cdots$$

For the general step, we have

$$f[x_0, x_1, \cdots, x_k] = \frac{f[x_1, x_2, \cdots, x_k] - f[x_0, x_1, \cdots, x_{k-1}]}{x_k - x_0}$$

The constants a_k's in the Newton's form are computed as

$$a_0 = f[x_0], \quad a_1 = f[x_0, x_1], \quad \cdots \quad a_k = f[x_0, x_1, \cdots, x_k]$$

Computation of the divided differences

We compute the $f[\cdots]$'s through the following table:

$$
\begin{array}{c|c|c|c|c|c}
x_0 & f[x_0] = y_0 & & & & \\
x_1 & f[x_1] = y_1 & f[x_0, x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0} & & & \\
x_2 & f[x_2] = y_2 & f[x_1, x_2] = \frac{f[x_2]-f[x_1]}{x_2-x_1} & f[x_0, x_1, x_2] & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & \\
x_n & f[x_n] = y_n & f[x_{n-1}, x_n] = \frac{f[x_n]-f[x_{n-1}]}{x_n-x_{n-1}} & f[x_{n-2}, x_{n-1}, x_n] & \ldots & f[x_0, x_1, \cdots, x_n]
\end{array}
$$

The diagonal elements give us the a_i's.

Example 3. Write Newton's form of interpolation polynomial for the data

x_i	0	1	2/3	1/3
y_i	1	0	1/2	0.866

Answer. Set up the triangular table for computation

0	$\boxed{1}$			
1	0	$\boxed{-1}$		
2/3	0.5	-1.5	$\boxed{-0.75}$	
1/3	0.8660	-1.0981	-0.6029	$\boxed{0.4413}$

So we have

$$a_0 = 1, \quad a_1 = -1, \quad a_2 = -0.75, \quad a_3 = 0.4413.$$

Then

$$P_3(x) = \boxed{1} + \boxed{-1}\,x + \boxed{-0.75}\,x(x-1) + \boxed{0.4413}\,x(x-1)(x-\frac{2}{3}).$$

Flexibility of Newton's form: easy to add additional points to interpolate.

Nested form of Newton's polynomial:

$$
\begin{aligned}
P_n(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots \\
&\qquad + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}) \\
&= a_0 + (x - x_0)\left(a_1 + (x - x_1)(a_2 + (x - x_2)(a_3 + \cdots + a_n(x - x_{n-1})))\right)
\end{aligned}
$$

Effective coding:
Given the data x_i and a_i for $i = 0, 1, \cdots, n$
the following pseudo-code evaluates the Newton's polynomial $p = P_n(x)$

- $p = a_n$
- for $k = n - 1, n - 2, \cdots, 0$
  - $p = p(x - x_k) + a_k$
- end

This requires only $3n$ flops.

Existence and Uniqueness theorem for polynomial interpolation

Theorem. (Fundamental Theorem of Algebra)
Every polynomial of degree n that is not identically zero, has maximum n roots (including multiplicities). These roots may be real of complex. In particular, this implies that if a polynomial of degree n has more than n roots, then it must be identically zero.

Theorem. **(Existence and Uniqueness of Polynomial Interpolation)** *Given $(x_i, y_i)_{i=0}^n$, with x_i's distinct. There exists one and only polynomial $P_n(x)$ of degree $\leq n$ such that $P_n(x_i) = y_i$* for $i = 0, 1, \cdots, n$.

Proof. The existence: by construction.
Uniqueness: Assume we have two polynomials $p(x), q(x) \in \mathcal{P}_n$, such that

$$p(x_i) = y_i, \qquad q(x_i) = y_i, \qquad i = 0, 1, \cdots, n$$

Now, let $g(x) = p(x) - q(x)$, a polynomial of degree $\leq n$.

$$g(x_i) = p(x_i) - q(x_i) = y_i - y_i = 0, \qquad i = 0, 1, \cdots, n$$

So $g(x)$ has $n + 1$ zeros. By the Fundamental Theorem of Algebra, we must have $g(x) \equiv 0$, therefore $p(x) \equiv q(x)$.

Errors in Polynomial Interpolation

Given a function $f(x)$ on $x \in [a, b]$, and a set of distinct points $x_i \in [a, b]$, $i = 0, 1, \cdots, n$. Let $P_n(x) \in \mathcal{P}_n$ s.t.,

$$P_n(x_i) = f(x_i), \qquad i = 0, 1, \cdots, n$$

$$\text{error function :} \qquad e(x) = f(x) - P_n(x), \qquad x \in [a, b].$$

Theorem. *There exists some value $\xi \in [a, b]$, such that*

$$e(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (x - x_i), \qquad \text{for all} \quad x \in [a, b]. \qquad (1)$$

Proof. If $f \in \mathcal{P}_n$, then by Uniqueness Theorem of polynomial interpolation we must have $f(x) = P_n(x)$. Then $e(x) \equiv 0$ and the proof is trivial.

Now assume $f \notin \mathcal{P}_n$. If $x = x_i$ for some i, we have $e(x_i) = f(x_i) - P_n(x_i) = 0$, and the result holds.

Now consider $x \neq x_i$ for any i.

$$W(x) = \prod_{i=0}^{n}(x - x_i) \quad \in \mathcal{P}_{n+1},$$

it holds

$$W(x_i) = 0, \qquad W(x) = x^{n+1} + \cdots, \qquad W^{(n+1)} = (n+1)!.$$

Fix an y such that $a \leq y \leq b$ and $y \neq x_i$ for any i. We define a constant

$$c = \frac{f(y) - P_n(y)}{W(y)},$$

and another function

$$\varphi(x) = f(x) - P_n(x) - cW(x).$$

We find all the zeros for $\varphi(x)$. We see that x_i's are zeros since

$$\varphi(x_i) = f(x_i) - P_n(x_i) - cW(x_i) = 0, \qquad i = 0, 1, \cdots, n$$

and also y is a zero because

$$\varphi(y) = f(y) - P_n(y) - cW(y) = 0$$

So, φ has at least $(n+2)$ zeros.

Here goes our deduction:

$$\begin{array}{rccl}
\varphi(x) \text{ has at least} & n+2 & \text{zeros on } [a,b]. \\
\varphi'(x) \text{ has at least} & n+1 & \text{zeros on } [a,b]. \\
\varphi''(x) \text{ has at least} & n & \text{zeros on } [a,b]. \\
 & \vdots & \\
\varphi^{(n+1)}(x) \text{ has at least} & 1 & \text{zero on } [a,b]. & \boxed{\text{Call it } \xi \text{ s.t. } \varphi^{(n+1)}(\xi) = 0.}
\end{array}$$

So we have

$$\varphi^{(n+1)}(\xi) = f^{(n+1)}(\xi) - 0 - cW^{(n+1)}(\xi) = 0.$$

Recall $W^{(n+1)} = (n+1)!$, we have, for every y,

$$f^{(n+1)}(\xi) = cW^{(n+1)}(\xi) = \frac{f(y) - P_n(y)}{W(y)}(n+1)!.$$

Writing y into x, we get

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

for some $\xi \in [a,b]$.

Recall the error formula: $$e(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (x - x_i)$$

Example 1. If $n = 1$, $x_0 = a, x_1 = b$, $b > a$, find an upper bound for error.

Answer. Let

$$M = \max_{a \le x \le b} |f''(x)| = \|f''\|_\infty$$

and observe

$$\max_{a \le x \le b} |(x-a)(x-b)| = \cdots = \frac{(b-a)^2}{4}.$$

For $x \in [a, b]$, we have

$$|e(x)| = \frac{1}{2} |f''(\xi)| \cdot |(x-a)(x-b)| \le \frac{1}{2} \|f''\|_\infty \frac{(b-a)^2}{4} = \frac{1}{8} \|f''\|_\infty (b-a)^2.$$

NB! Error depends on the distribution of nodes x_i.

Uniform grid

Equally distribute the nodes (x_i): on $[a, b]$, with $n+1$ nodes.

$$x_i = a + ih, \qquad h = \frac{b-a}{n}, \qquad i = 0, 1, \cdots, n.$$

One can show that for $x \in [a, b]$, it holds

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

Proof. If $x = x_i$ for some i, then $x - x_i = 0$ and the product is 0, so it trivially holds.
Now assume $x_i < x < x_{i+1}$ for some i. We have

$$\max_{x_i < x < x_{i+1}} |(x - x_i)(x - x_{i+1})| = \frac{1}{4}(x_{i+1} - x_i)^2 = \frac{h^2}{4}.$$

Now consider the other terms in the product, say $x - x_j$, for either $j > i + 1$ or $j < i$. Then $|x - x_j| \le h(j - i)$ for $j > i + 1$ and $|x - x_j| \le h(i + 1 - j)$ for $j < i$. In all cases, the product of these terms are bounded by $h^{n-1} n!$, proving the result.

We have the error estimate

$$|e(x)| \leq \frac{1}{4(n+1)} \left| f^{(n+1)}(x) \right| h^{n+1} \leq \frac{M_{n+1}}{4(n+1)} h^{n+1}$$

where

$$M_{n+1} = \max_{x \in [a,b]} \left| f^{(n+1)}(x) \right| = \left\| f^{(n+1)} \right\|_\infty$$

Example 2, Consider interpolating $f(x) = \sin(\pi x)$ with polynomial on the interval $[-1, 1]$ with uniform nodes. Give an upper bound for error.
Answer. Since

$$f'(x) = \pi \cos \pi x, \quad f''(x) = -\pi^2 \sin \pi x, \quad f'''(x) = -\pi^3 \cos \pi x$$

we have

$$\left| f^{(n+1)}(x) \right| \leq \pi^{n+1}, \qquad M_{n+1} = \pi^{n+1}$$

so the upper bound for error is

$$|e(x)| \leq \frac{M_{n+1}}{4(n+1)} h^{n+1} \leq \frac{\pi^{n+1}}{4(n+1)} \left(\frac{2}{n} \right)^{n+1}.$$

Simulation data:

n	error bound	measured error
4	4.8×10^{-1}	1.8×10^{-1}
8	3.2×10^{-3}	1.2×10^{-3}
16	1.8×10^{-9}	6.6×10^{-10}