NTNU

Norwegian University of Science and Technology

TMA4125 Matematikk 4N

Polynomial interpolation: Methods

Ronny Bergmann

Department of Mathematical Sciences, NTNU.

January 12, 2022

Interpolation: basic idea

Often we get some discrete measurement data $(x_i, y_i), i = 0, \ldots, n$.

Goal. Find a function f that describes this data, i. e.

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

and that f is from a certain class (smoothness, polynomial,...) – let's say "nice" function

Task. If we have a complicated function g, we might take $(x_i, g(x_i))$, find a "nice" function (e. g. easy to integrate) and use the approach from above.

Polynomial interpolation

Task. Given $n+1$ points $(x_i, y_i), i = 0, \ldots, n$, find a polynomial $p(x)$ of lowest possible degree satisfying the interpolation condition

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

The solution $p(x)$ is called interpolation polynomial.

The values x_i are called nodes, the points (x_i, y_i) are called interpolation points.

Example of an interpolation problem

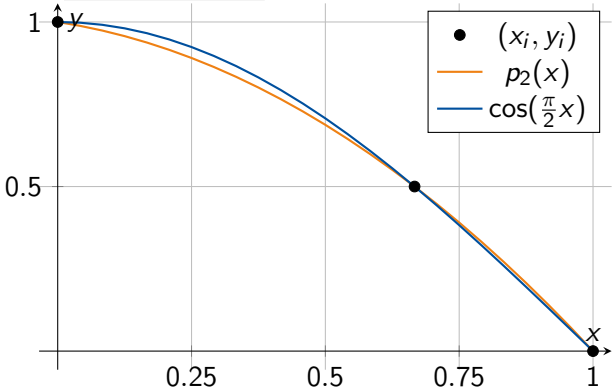
For given data

i	0	1	2
x_i	0	$\frac{2}{3}$	1
y_i	1	$\frac{1}{2}$	0

The corresponding interpolation polynomial is

$$p_2(x) = \frac{1}{4}\left(-3x^2 - x + 4\right).$$

The data are sample values of $\cos(\frac{\pi}{2}x)$ on $[0, 1]$.



- p_2 interpolates the data.
- Locally (on $[0, 1]$) p_2 explains the function $\cos(\frac{\pi}{2}x)$ quite well.

Roadmap

We will discuss the following

- **Method.** How to compute the interpolation polynomial?
- Existence and uniqueness results
- **Error analysis.** If the polynomial is used to approximate a function, how good is the approximation?
- **Improvements.** If the nodes x_i can be chosen freely, how should we do it in order to reduce the error?

Polynomials: some useful facts

We already learned about

- $\mathbb{P}_n$ the set of polynomials of degree n or less
- $C^m[a, b]$ the set of all continuous functions that have continuous first m derivatives

and a polynomial of degree n we denote by $p_n \in \mathbb{P}_n$ written as

$$p_n(x) = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0 = \sum_{i=0}^{n} c_i x^i,$$

where $c_i \in \mathbb{R}$, $i = 0, \ldots, n$, are some real coefficients.

Roots of a polynomial

The value r is a root or zero of a polynomial p if $p(r) = 0$.

A nonzero polynomial p_n of dergree n can never have more than n real roots (there are maybe less).

A polynomial p_n of degree n with n real roots $r_1, r_2, \ldots, r_n \in \mathbb{R}$ can be written as

$$p_n(x) = c(x - r_1)(x - r_2) \cdot \ldots \cdot (x - r_n) = c \prod_{i=1}^{n} (x - r_i).$$

Direct method

For a polynomial p_n of degree n we can write down the interpolation conditions that

$$p_n(x_j) = \sum_{i=0}^{n} c_i x_j^i = y_j, \qquad \text{for } j = 0, \ldots, n$$

has to hold.
These are $n+1$ equations and we have $n+1$ unknowns $c_0, c_1, \ldots, c_n$.

Usually you do not do this approach for larger n, since numerically this is hard to solve.

Lagrange interpolation

Given $n+1$ points (x_i, y_i), $i = 0, \ldots, n$, with distinct values of x_i.
Then the cardinal functions ℓ_i, $i = 0, \ldots, n$, are defined by

$$\ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} = \frac{x - x_0}{x_i - x_0} \cdot \frac{x - x_1}{x_i - x_1} \cdot \ldots \cdot \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdot \ldots \cdot \frac{x - x_n}{x_i - x_n}$$

They have the following properties

- $\ell_i \in \mathbb{P}_n$ for $i = 0, \ldots, n$
- $\ell_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{when } i = j, \\ 0 & \text{else} \end{cases}$
- they are constructed solely from the nodes x_i (no y_i involved)
- they are linearly independent $\Rightarrow$ form a basis of $\mathbb{P}_n$.
- they are also called Lagrange polynomials

Interplation with cardinal functions

The interpolation polynomial is now given by

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

since we have

$$p_n(x_j) = \sum_{i=0}^{n} y_i \ell_i(x_j) = y_j \ell_j(x_j) = y_j \qquad \text{for } j = 0, \ldots, n.$$

Example of cardinal functions

For given data

i	0	1	2
x_i	0	1	3
y_i	3	8	6

we get

$$\ell_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-1)(x-3)}{(0-1)(0-3)} = \frac{1}{3}x^2 - \frac{4}{3}x + 1$$

$$\ell_1(x) = \frac{(x-0)(x-3)}{(1-0)(1-3)} = -\frac{1}{2}x^2 + \frac{3}{2}x$$

$$\ell_2(x) = \frac{(x-0)(x-1)}{(3-0)(3-1)} = \frac{1}{6}x^2 - \frac{4}{6}x$$

and the corresponding interpolation polynomial looks like

$$\begin{aligned} p_2(x) &= 3\ell_0(x) + 8\ell_1(x) + 6\ell_2(x) \\ &= -2x^2 + 7x + 3 \end{aligned}$$

Implementation

The method above is implemented as two functions

- `cardinal(xdata, x)` creates a list of cardinal functions $\ell_i(x)$ evaluated in x.
- `lagrange(ydata, l)` creates the interpolation polynomial $p_n(x)$.

Here,

- `xdata` and `ydata` are arrays with the interpolation points x_i and y_i, respectively,
- `x` is an array of values in which the polynomials are evaluated.
- `l` is what `cardinal(xdata,x)` returns

You are not required to understand the implementation of these functions, but you should know how to use them.

Existence and uniqueness of interpolation polynomials

We have already proved the existence of such polynomials, simply by constructing them. But are they unique? The answer is yes!

Theorem. (Existence and uniqueness.)
Given $n+1$ points $(x_i, y_i)_{i=0}^n$ with distinct x values. Then there is one and only one polynomial $p_n(x) \in \mathbb{P}_n$ satisfying the interpolation condition

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

Proof.
Suppose there exist two different interpolation polynomials p_n and q_n of degree n interpolating the same $n+1$ points. The polynomial $r(x) = p_n(x) - q_n(x)$ is of degree n with zeros in all the nodes x_i, that is a total of $n+1$ zeros. But then $r \equiv 0$, and the two polynomials p_n and q_n are identical.

Lagrange – final notes

Advantage.
Given nodes $x_0, \ldots, x_n$ we can directly compute the cardinal functions $\ell_i(x)$, $i = 0, \ldots, n$

This is independent of $y_0, \ldots, y_n$, so we can **reuse** the cardinal functions: given new/second values $\tilde{y}_0, \ldots, \tilde{y}_n$ (to the same nodes)!
(just call `lagrange(tildeydata, l)`)

Disadvantage.
All $\ell_0, \ldots, \ell_n$ have as highest monomial x^n.

Given the nodes $x_0, \ldots, x_n$ and their cardinal functions $\ell_i(x)$, $i = 1, \ldots, n$, and someone comes along with an **additional** node x_{n+1}
$\Rightarrow$ I have to compute new cardinal function $\tilde{\ell}_0, \ldots, \tilde{\ell}_n$ and a new $\tilde{\ell}_{n+1}$

Newton interpolation

Let's look at alternative approach to find the interpolation polynomial. Let $x_0, x_1, \ldots, x_n$ be $n+1$ distinct real numbers.

Then instead of the cardinal polynomials to form p_n, we will employ the so-called Newton polynomials ω_i, $i = 0, \ldots, n$ defined by

$$
\begin{aligned}
\omega_0(x) &= 1, \\
\omega_1(x) &= (x - x_0), \\
\omega_2(x) &= (x - x_0)(x - x_1), \\
&\vdots \\
\omega_n(x) &= (x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-1}),
\end{aligned}
$$

or compact: $\omega_i(x) = \prod_{k=0}^{i-1} (x - x_k), \quad \text{for } i = 0, \ldots, n.$

Newton form of the interpolation polynomial

The so-called Newton form of a polynomial of degree n is an expansion of the form

$$p(x) = \sum_{i=0}^{n} c_i \omega_i(x)$$

or more explicitly

$$\begin{aligned} p(x) = {} & c_n(x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-1}) \\ & + c_{n-1}(x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-2}) \\ & + \cdots \\ & + c_2(x - x_0)(x - x_1) \\ & + c_1(x - x_0) \\ & + c_0. \end{aligned}$$

Computing the coefficients in Newton form

Let us start with $y_j = f(x_j)$ (for a nicer form) and a single node x_0.
Then $y_0 = f(x_0) = p(x_0) = c_0$.

One step further: Consider consider two nodes x_0, x_1. Then we see that $f(x_0) = p(x_0) = c_0$ (as before) and

$$f(x_1) = p(x_1) = c_0 + c_1(x_1 - x_0).$$

Since we know $c_0 = f(x_0)$ we can rearrange this

$$c_1 = \frac{f(x_1) - c_0}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

Computing the coefficients in Newton form II

Given three nodes x_0, x_1, x_2 yields the coefficients c_0, c_1 as defined before, and from

$$f(x_2) = p(x_2) = c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1)$$

we get by rearranging and plugging in c_0, c_1

$$c_2 = \frac{f(x_2) - c_0 - c_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} = \frac{f(x_2) - f(x_0) - \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)}.$$

We can divide numerator and denominator by $(x_2 - x_1)$.
Then simplifying the numerator

$$c_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}.$$

This is a finite difference of finite differences.

Iterated finite differences

Continuing this yields a so-called triangular systems that permits to define the remaining coefficients $c_3, \ldots, c_n$.
We easily see that c_k only depends on the interpolation points $(x_0, y_0), \ldots, (x_k, y_k)$, where $y_i := f(x_i)$, $i = 0, \ldots, n$.

We introduce the finite difference notation for a function f:

0th order $f[x_0] := f(x_0)$

1st order $f[x_0, x_1] := \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$

2nd order $f[x_0, x_1, x_2] := \dfrac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$.

nth order $f[x_0, x_1, \ldots, x_n] := \dfrac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}$.

These are also called nth Newton divided differences.

Newtons method for interpolation – summary

Given $n+1$ interpolation points $(x_0, y_0), \ldots, (x_n, y_n)$, $y_i := f(x_i)$.
Expressing the order n interpolation polynomial p_n in Newton's form

$$\begin{aligned} p_n(x) = c_n(x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-1}) \\ + c_{n-1}(x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-2}) \\ + \cdots + c_1(x - x_0) + c_0, \end{aligned}$$

yields that the coefficients are given by

$$c_k = f[x_0, x_1, \ldots, x_k], \qquad k = 0, 1, \ldots, n.$$

In fact, a recursion is in place

$$p_n(x) = p_{n-1}(x) + f[x_0, \ldots, x_n](x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

The finite differences table

It is common to write the finite differences in a table, which for $n = 3$ will look like:

$$
\begin{array}{c|cccc}
x_0 & f[x_0] & & & \\
 & & f[x_0, x_1] & & \\
x_1 & f[x_1] & & f[x_0, x_1, x_2] & \\
 & & f[x_1, x_2] & & f[x_0, x_1, x_2, x_3] \\
x_2 & f[x_2] & & f[x_1, x_2, x_3] & \\
 & & f[x_2, x_3] & & \\
x_3 & f[x_3] & & &
\end{array}
$$

Newton divided differences for the first example

Given the points in Example 1.
The corresponding table of divided differences becomes:

$$\begin{array}{c|c|c|c} 0 & 1 & & \\ & & -\frac{3}{4} & \\ \frac{2}{3} & \frac{1}{2} & & -\frac{3}{4} \\ & & -\frac{3}{2} & \\ 1 & 0 & & \end{array}$$

Let's take a look how we get to these numbers.
The final interpolation polynomial then reads

$$p_2(x) = 1 - \frac{3}{4}(x-0) - \frac{3}{4}(x-0)(x-\frac{2}{3}) = 1 - \frac{1}{4}x - \frac{3}{4}x^2.$$

Implementation

The method above is implemented as two functions:

- `divdiff(xdata, ydata)` Create the table of divided differences
- `newton_interpolation(F, xdata, x)` Evaluate the interpolation polynomial.

Here,

- `xdata` and `ydata` are arrays with the interpolation points,
- `F` is the result from the first function, and
- `x` is an array of values in which the polynomial is evaluated.