

### §4.1 Polynomial Interpolation

We consider the problem of finding a polynomial that interpolates a given set of values:

$$\begin{array}{c|c|c|c|c} x & x_0 & x_1 & \dots & x_n \\ \hline y & y_0 & y_1 & \dots & y_n \end{array}$$

where the  $x_i$  are all distinct. A polynomial  $p(x)$  is said to interpolate these data if  $p(x_i) = y_i$  for  $i = 0, 1, \dots, n$ . The  $x_i$  values are called “nodes.”

#### Existence

As you might have guessed, for any such set of data, there is an  $n$ -degree polynomial that interpolates it. We present a constructive proof of this fact by use of Lagrange Polynomials.

For a given set of  $n + 1$  nodes  $x_i$ , the Lagrange polynomials are the  $n + 1$  polynomials  $\ell_i$  defined by

$$\ell_i(x_j) = \delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

Then we define the interpolating polynomial as

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

If each Lagrange Polynomial is of degree at most  $n$ , then  $p_n$  also has this property. We claim that we can characterize the Lagrange Polynomials as follows:

$$\ell_i(x) = \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j}.$$

By evaluating this product for each  $x_j$ , we see that this is indeed a characterization of the Lagrange Polynomials. Moreover, each polynomial is clearly the product of  $n$  monomials, and thus has degree no greater than  $n$ .

**Example 1.** Construct the polynomial interpolating the data

$$\begin{array}{c|c|c|c} x & 1 & \frac{1}{2} & 3 \\ \hline y & 3 & -10 & 2 \end{array}$$

by using Lagrange Polynomials.

We construct the Lagrange Polynomials:

$$\begin{aligned} \ell_0(x) &= \frac{(x - \frac{1}{2})(x - 3)}{(1 - \frac{1}{2})(1 - 3)} = -(x - \frac{1}{2})(x - 3) \\ \ell_1(x) &= \frac{(x - 1)(x - 3)}{(\frac{1}{2} - 1)(\frac{1}{2} - 3)} = \frac{4}{5}(x - 1)(x - 3) \\ \ell_2(x) &= \frac{(x - 1)(x - \frac{1}{2})}{(3 - 1)(3 - \frac{1}{2})} = \frac{1}{5}(x - 1)(x - \frac{1}{2}) \end{aligned}$$

Then the interpolating polynomial, in “Lagrange Form” is

$$p_2(x) = -3\left(x - \frac{1}{2}\right)(x - 3) - 8\left(x - 1\right)(x - 3) + \frac{2}{5}\left(x - 1\right)\left(x - \frac{1}{2}\right)$$

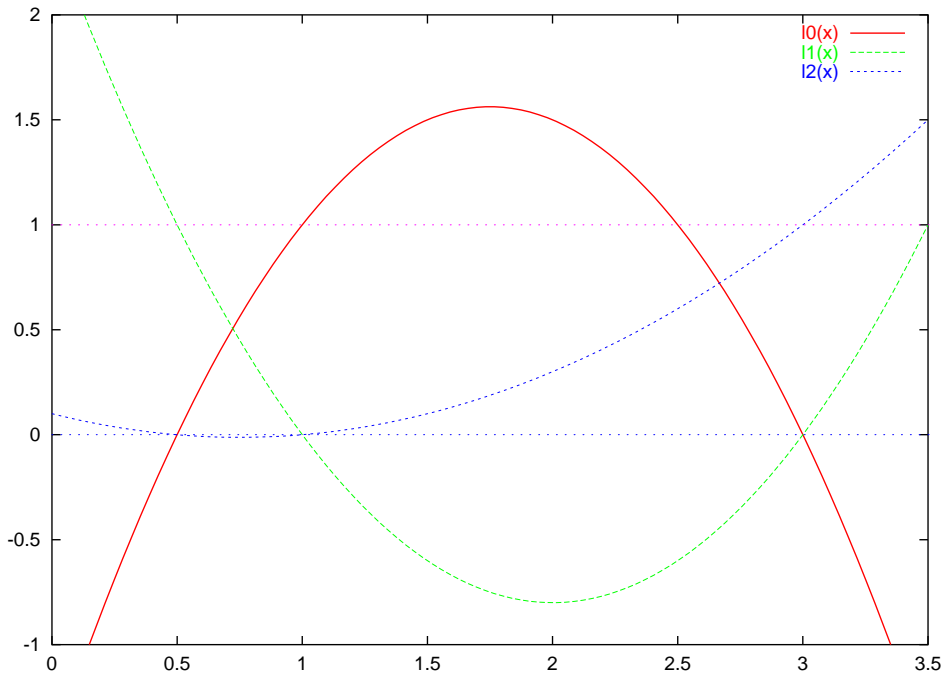


Figure 1: The 3 Lagrange polynomials for the example are shown. Verify, graphically, that these polynomials have the property  $\ell_i(x_j) = \delta_{ij}$  for the nodes  $1, \frac{1}{2}, 3$ .

This gives the theorem

**Theorem 2 (Interpolant Existence).** If  $x_0, x_1, \dots, x_n$  are distinct real values, then for arbitrary real  $y_0, y_1, \dots, y_n$  values, there is some polynomial,  $p(x)$  of degree no greater than  $n$  that interpolates the data, *i.e.*, such that  $p(x_i) = y_i$  for  $i = 0, 1, \dots, n$ .

### Existence (II)

There is another way to prove existence. This method is also constructive, and leads to a different algorithm for constructing the interpolant. One way to view this construction is to imagine how one would update the Lagrangian form of an interpolant. That is, suppose some data were given, and the interpolating polynomial calculated using Lagrange Polynomials; then a new point was given  $(x_{n+1}, y_{n+1})$ , and an interpolant for the augmented set of data is to be found. Each Lagrange Polynomial would have to be updated. This could take a lot of calculation (especially if  $n$  is large).

So the alternative method constructs the polynomials iteratively. Thus we create polynomials  $p_k(x)$  such that  $p_k(x_i) = y_i$  for  $0 \leq i \leq k$ . This is simple for  $k = 0$ , we simply

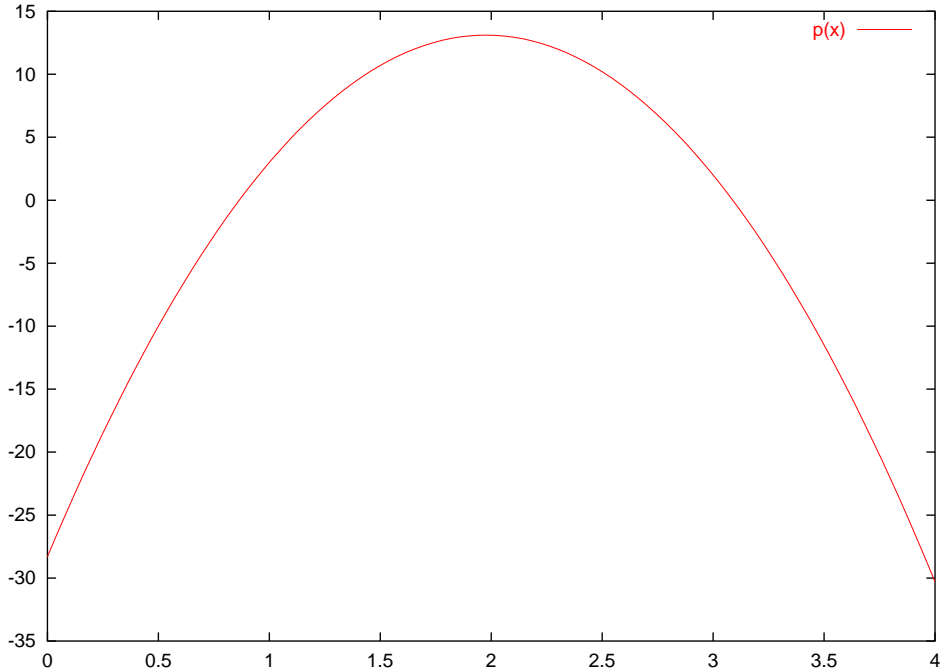


Figure 2: The interpolating polynomial for the example is shown.

let

$$p_0(x) = y_0,$$

the constant polynomial with value  $y_0$ . Then assume we have a proper  $p_k(x)$  and want to construct  $p_{k+1}(x)$ . We claim that the following construction works:

$$p_{k+1}(x) = p_k(x) + c(x - x_0)(x - x_1) \cdots (x - x_k),$$

for some constant  $c$ . Note that the second term will be zero for any  $x_i$  for  $0 \leq i \leq k$ , so  $p_{k+1}(x)$  will interpolate the data at  $x_0, x_1, \dots, x_k$ . To get the value of the constant we calculate  $c$  such that

$$y_{k+1} = p_{k+1}(x_{k+1}) = p_k(x_{k+1}) + c(x_{k+1} - x_0)(x_{k+1} - x_1) \cdots (x_{k+1} - x_k).$$

This construction is known as Newton's Algorithm, and the resultant form is Newton's form of the interpolant

**Example 3.** Construct the polynomial interpolating the data

$$\begin{array}{c|c|c|c} x & 1 & \frac{1}{2} & 3 \\ \hline y & 3 & -10 & 2 \end{array}$$

by using Newton's Algorithm.

We construct the polynomial iteratively:

$$\begin{aligned} p_0(x) &= 3 \\ p_1(x) &= 3 + c(x - 1) \end{aligned}$$

We want  $-10 = p_1(\frac{1}{2}) = 3 + c(-\frac{1}{2})$ , and thus  $c = 26$ . Then

$$p_2(x) = 3 + 26(x - 1) + c(x - 1)(x - \frac{1}{2})$$

We want  $2 = p_2(3) = 3 + 26(2) + c(2)(\frac{5}{2})$ , and thus  $c = \frac{-53}{5}$ . Then we get

$$p_2(x) = 3 + 26(x - 1) + \frac{-53}{5}(x - 1)(x - \frac{1}{2}).$$

### Uniqueness

Does Newton's Algorithm give a different polynomial? It is easy to show, by induction, that the degree of  $p_n(x)$  is no greater than  $n$ . Suppose that  $q(x), p(x)$  are two distinct polynomials of degree no greater than  $n$  that both interpolate a set of data. Let  $r(x) = p(x) - q(x)$ . Note that  $r(x)$  can have degree no greater than  $n$ , yet it has roots at  $x_0, x_1, \dots, x_n$ . The only polynomial of degree  $\leq n$  that has  $n + 1$  distinct roots is the zero polynomial, *i.e.*,  $0 \equiv r(x) = p(x) - q(x)$ .

Thus Newton's Algorithm and the Lagrange method give the same polynomial. Which should we use? Newton's Algorithm seems more flexible—it can deal with adding new data. There also happens to be a way of storing Newton's form of the interpolant that makes the polynomial simple to evaluate (in the sense of number of calculations required).

### Newton's Nested Form

Recall the iterative construction of Newton's Form:

$$p_{k+1}(x) = p_k(x) + c_k(x - x_0)(x - x_1) \cdots (x - x_k).$$

The previous iterate  $p_k(x)$  was constructed similarly, so we can write:

$$p_{k+1}(x) = [p_{k-1}(x) + c_{k-1}(x - x_0)(x - x_1) \cdots (x - x_{k-1})] + c_k(x - x_0)(x - x_1) \cdots (x - x_k).$$

Continuing in this way we see that we can write

$$p_n(x) = \sum_{k=0}^n c_k \left[ \prod_{0 \leq j < k} (x - x_j) \right],$$

where an empty product has value 1 by convention. This can be rewritten in a funny form, where a monomial is factored out of each successive summand:

$$p_n(x) = c_0 + (x - x_0)[c_1 + (x - x_1)[c_2 + (x - x_2)[\dots]]]$$

Supposing for an instant that the constants  $c_k$  were known, this provides a better way of calculating  $p_n(t)$  at arbitrary  $t$ . By "better" we mean requiring few multiplications and additions. This nested calculation is performed iteratively:

$$\begin{aligned}v_0 &= c_n \\v_1 &= c_{n-1} + (t - x_{n-1})v_0 \\v_2 &= c_{n-2} + (t - x_{n-2})v_1 \\&\vdots \\v_n &= c_0 + (t - x_0)v_{n-1}\end{aligned}$$

This requires only  $n$  multiplications and  $2n$  additions. Compare this with the number required for using the Lagrange form: at least  $n^2$  additions and multiplications.

**We got this far in class. The following pages are for Friday 2003/10/10.**

## Divided Differences

It turns out that the coefficients  $c_k$  for Newton's nested form can be calculated relatively easily. For whatever reason that's never been explained, the notation for these things is of the form

$$f[x_0, x_1, \dots, x_k].$$

Where does “ $f$ ” come from? Well, for some reason, when talking about polynomial interpolation, people change the original problem to assume that they are interpolating the data

$$\begin{array}{c|c|c|c|c} x & x_0 & x_1 & \dots & x_n \\ \hline f(x) & f(x_0) & f(x_1) & \dots & f(x_n) \end{array}$$

for some unknown function  $f$ , and distinct  $x_i$ .

This thing  $f[x_0, x_1, \dots, x_k]$  is called a *divided difference of order  $k$*  for  $f$ . We briefly note that the  $x_i$  need not be consecutive, and the divided difference is still defined. That is

$$f[x_3, x_{17}, x_2]$$

is well defined (for  $n \geq 17$ .)

There's a whole lot of theory around divided differences, and they're really cool and stuff, especially if you are a mathematician. All you really need to know I summarize here:

1. The coefficients for the nested form are the divided differences. That is

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

2. The 0<sup>th</sup> order divided differences are simple:

$$f[x_i] = f(x_i).$$

3. The  $k$ <sup>th</sup> order divided differences are defined recursively:

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

The graphical way to calculate these things is a “pyramid scheme”<sup>1</sup>, where you fill in the following table:

$x$	$f[ ]$	$f[ , ]$	$f[ , , ]$
$x_0$	$f[x_0]$		
$x_1$	$f[x_1]$	$f[x_0, x_1]$	
$x_2$	$f[x_2]$	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$

We drag out our example one last time:

---

<sup>1</sup>That's supposed to be a joke.

**Example 4.** Find the divided differences for the following data

$x$	1	$\frac{1}{2}$	3
$f(x)$	3	-10	2

We have

$x$	$f[ ]$	$f[ , ]$	$f[ , , ]$
1	3		
$\frac{1}{2}$	-10	$\frac{-13}{-\frac{1}{2}}$	$\frac{-53}{5}$
3	2	$\frac{12}{\frac{5}{2}}$	

You should verify that along the top line of this pyramid you can read off the coefficients for Newton's form, as found in the previous incarnation of this example. It works!