

§4.1 Polynomial Interpolation (Continued)

We consider a slight alteration of the problem of interpolation. Suppose we have some function $f(x)$ which we want to approximate by a polynomial $p(x)$. We do this by finding the unique polynomial of degree $\leq n$ that interpolates the data

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

where the nodes x_i are distinct.

Divided Differences

Recall that using Newton's Algorithm we can write the polynomial $p(x)$ as

$$\begin{aligned} p_n(x) &= \sum_{k=0}^n c_k \left[\prod_{0 \leq j < k} (x - x_j) \right] \\ &= c_0 + (x - x_0)[c_1 + (x - x_1)[c_2 + (x - x_2)[\dots [c_n]]]]. \end{aligned}$$

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.

It turns out that the coefficients c_k for Newton's nested form can be calculated relatively easily. In fact, we can give them a new name:

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

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 $n \geq 17$.)

All you really need to know about divided differences I summarize here:

1. The 0th order divided differences are simply defined:

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

2. The k^{th} 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”¹, 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:

Example 1. 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	$\frac{-13}{-\frac{1}{2}}$	
3	2	$\frac{12}{5/2}$	$\frac{-53}{5}$

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!

§4.2 Errors in Polynomial Interpolation

We now consider two questions:

1. If we want to interpolate some function $f(x)$ at $n + 1$ nodes over some closed interval, how should we pick the nodes?
2. How accurate can we make a polynomial interpolant over a closed interval?

¹That’s supposed to be a joke.

You may be surprised to find that the answer to the first question is *not* that we should make the x_i equally spaced over the closed interval. In fact if we let

$$f(x) = (1 + x^2)^{-1},$$

(known as the *Runge Function*), and let $p_n(x)$ interpolate f on n equally spaced nodes, including the endpoints, on $[-5, 5]$, then

$$\lim_{n \rightarrow \infty} \max_{x \in [-5, 5]} |p_n(x) - f(x)| = \infty.$$

It turns out that a much better choice is related to the *Chebyshev Polynomials* (“of the first kind”). If our closed interval is $[-1, 1]$, then we want to define our nodes as

$$x_i = \cos \left[\left(\frac{2i + 1}{2n + 2} \right) \pi \right], \quad 0 \leq i \leq n.$$

Note that this is different than what appears in your book. This book is in error. If you do not trust me, look at the fifth edition, or look at equation (9) of

<http://mathworld.wolfram.com/ChebyshevPolynomialoftheFirstKind.html>

Literally interpreted, these *Chebyshev Nodes* are the projections of points uniformly spaced on a semi circle; see Figure 1

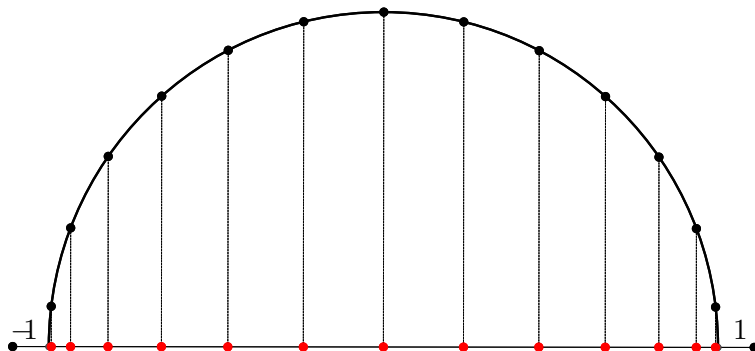


Figure 1: The Chebyshev nodes are the projections of nodes equally spaced on a semi circle.

Interpolation Error Theorem

We did not just invent the Chebyshev nodes. The fact that they are “good” for interpolation follows from the following theorem:

Theorem 2 (Interpolation Error Theorem). Let p be the polynomial of degree at most n interpolating function f at the $n + 1$ distinct nodes x_0, x_1, \dots, x_n on $[a, b]$. Let $f^{(n+1)}$ be continuous. Then for each $x \in [a, b]$ there is some $\xi \in [a, b]$ such that

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

You should be thinking that the term on the right hand side resembles the error term in Taylor's Theorem.

Proof. First consider when x is one of the nodes x_i ; in this case both the LHS and RHS are zero. So assume x is not a node. Make the following definitions

$$\begin{aligned} w(t) &= \prod_{i=0}^n (t - x_i), \\ c &= \frac{f(x) - p(x)}{w(x)}, \\ \phi(t) &= f(t) - p(t) - cw(t). \end{aligned}$$

Since x is not a node, $w(x)$ is nonzero. Now note that $\phi(x_i)$ is zero for each node x_i , and that by definition of c , that $\phi(x) = 0$ for our x . That is $\phi(t)$ has $n + 2$ roots.

Some other facts: f, p, w have $n + 1$ continuous derivatives, by assumption and definition; thus ϕ has this many continuous derivatives. Apply Rolle's Theorem to $\phi(t)$ to find that $\phi'(t)$ has $n + 1$ roots. Then apply Rolle's Theorem again to find $\phi''(t)$ has n roots. In this way we see that $\phi^{(n+1)}(t)$ has a root, call it ξ . That is

$$0 = \phi^{(n+1)}(\xi) = f^{(n+1)}(\xi) - p^{(n+1)}(\xi) - cw^{(n+1)}(\xi).$$

But $p(t)$ is a polynomial of degree $\leq n$, so its $n + 1^{\text{th}}$ derivative is zero. And $w(t)$ is a polynomial of degree $n + 1$ in t , so its $n + 1^{\text{th}}$ derivative is easily seen to be $(n + 1)!$. Thus

$$\begin{aligned} 0 &= f^{(n+1)}(\xi) - c(n + 1)! \\ c(n + 1)! &= f^{(n+1)}(\xi) \\ \frac{f(x) - p(x)}{w(x)} &= \frac{1}{(n + 1)!} f^{(n+1)}(\xi) \\ f(x) - p(x) &= \frac{1}{(n + 1)!} f^{(n+1)}(\xi) w(x), \end{aligned}$$

which is what was to be proven. □

We got this far in class. The following pages are for Monday 2003/10/13.

The Chebyshev nodes on $[-1, 1]$ have the remarkable property that

$$\left| \prod_{i=0}^n (t - x_i) \right| \leq 2^{-n}$$

for any $t \in [-1, 1]$. Moreover, it can be shown that for *any* choice of nodes x_i that

$$\max_{t \in [-1, 1]} \left| \prod_{i=0}^n (t - x_i) \right| \geq 2^{-n}.$$

In this sense the Chebyshev nodes are considered the best for polynomial interpolation.

Interpolation Error for Equally Spaced Nodes

Despite the proven superiority of Chebyshev Nodes, and the problems with the Runge Function, equally spaced nodes are frequently used. We now consider bounding

$$\max_{x \in [a, b]} \prod_{i=0}^n |x - x_i|,$$

where

$$x_i = a + hi = a + \frac{(b-a)}{n}i, \quad i = 0, 1, \dots, n.$$

Start by picking an x . We can assume x is not one of the nodes, otherwise the product in question is zero. Then x is between some x_j, x_{j+1} . We can show that

$$|x - x_j| |x - x_{j+1}| \leq \frac{h^2}{4}.$$

by simple calculus.

Now we claim that $|x - x_i| \leq (j - i + 1)h$ for $i < j$, and $|x - x_i| \leq (i - j)h$ for $j + 1 < i$. Then

$$\prod_{i=0}^n |x - x_i| \leq \frac{h^2}{4} [(j+1)!h^j] [(n-j)!h^{n-j-1}].$$

Through use of mathemagic, we get an overall bound

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

The interpolation theorem then gives us

$$f(x) - p(x) \leq \frac{h^{n+1}}{4(n+1)} f^{(n+1)}(\xi).$$

The reason this does not seem to apply to Runge's Function is that $f^{(n)}$ for Runge's Function becomes unbounded as $n \rightarrow \infty$.