Polynomial Interpolation

(Com S 477/577 Notes)

Yan-Bin Jia

Sep 21, 2017

1 Interpolation Problem

In practice, often we can measure a physical process or quantity (e.g., temperature) at a number of points (e.g., time instants for temperature), but we do not have an analytic expression for the process that would let us calculate its value at an arbitrary point. Interpolation provides a simple and good way of estimating the analytic expression, essentially a function, over the range spanned by the measured points.[1]

Consider a family of functions of a single variable x:

$$\Phi(x; a_0, a_1, \ldots, a_n),$$

where $a_0, \ldots, a_n$ are the parameters. The problem of *interpolation* for Φ can be stated as follows:

> *Given $n+1$ real or complex pairs of numbers (x_i, f_i), $i = 0, \ldots, n$, with $x_i \neq x_k$ for $i \neq k$, determine $a_0, \ldots, a_n$ such that $\Phi(x_i; a_0, \ldots, a_n) = f_i$, $i = 0, \ldots, n$.*

The above is a *linear interpolation* problem if Φ depends linearly on the parameters a_i:

$$\Phi(x; a_0, \ldots, a_n) = a_0\Phi_0(x) + a_1\Phi_1(x) + \cdots + a_n\Phi_n(x).$$

Typical problems in this class include *polynomial interpolation*

$$\Phi(x; a_0, \ldots, a_n) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n,$$

as well as *trigonometric interpolation*

$$\Phi(x; a_0, \ldots, a_n) = a_0 + a_1 e^{xi} + a_2 e^{2xi} + \cdots + a_n e^{nxi} \qquad (i^2 = -1).$$

Polynomial interpolation will be addressed shortly in length. Trigonometric interpolation is used extensively for the numerical Fourier analysis of time series and cyclic phenomena in general. We will discuss this further along with approximation in the future.

The class of linear interpolation also contains *spline interpolation*. In the special case of *cubic splines*, Φ is twice continuously differentiable on $[x_0, x_n]$ and coincident with some cubic polynomial on each $[x_i, x_{i+1}]$ for $i = 0, \ldots, n-1$. Spline interpolation is very useful for representing empirical

[1] If a point for evaluation is outside the range, then it is called *extrapolation*, which usually yields a value not as reliable as by interpolation.

curves (in graphics and drawing tools), and for dealing with ordinary differential equations (ODEs) and partial differential equations (PDEs). *Spline interpolation* is described in more details in the notes for optional reading.

The class of *nonlinear interpolation* includes two important schemes: *rational interpolation*,

$$\Phi(x; a_0, \ldots, a_n, b_0, \ldots, b_m) = \frac{a_0 + a_1 x + \ldots + a_n x^n}{b_0 + b_1 x + \ldots + b_m x^m},$$

and *exponential interpolation*,

$$\Phi(x; a_0, \ldots, a_n, \lambda_0, \ldots \lambda_n) = a_0 e^{\lambda_0 x} + a_1 e^{\lambda_1 x} + \cdots + a_n e^{\lambda_n x}.$$

The former scheme is often used for best approximating a given function by one easily computable. The latter scheme is used, for instance, in radioactive decay analysis.

2 Polynomial Interpolation

According to Weirstrass Approximation Theorem introduced earlier, any function continuous over a closed interval can be approximated arbitrarily well by polynomials. The theorem, nevertheless, does not tell us how to construct such a polynomial to satisfy a specified error range allowed for approximation.

2.1 Polynomial Forms

The most common form of a polynomial $p(x)$ is the *power form*:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n, \qquad a_n \neq 0.$$

Here n is the degree of the polynomial. The drawback with this form is that numerical round-offs can lead to a loss of significance for x not near zero. The *shifted power form* is based on Taylor expansion about some point c:

$$p(x) = a_0 + a_1 (x - c) + a_2 (x - c)^2 + \cdots + a_n (x - c)^n,$$

where $a_i = p^{(i)}(c)/i!$. The *Newton form* "expands" the polynomial about n points:

$$p(x) = a_0 + a_1 (x - c_1) + a_2 (x - c_1)(x - c_2) + \cdots + a_n (x - c_1)(x - c_2) \cdots (x - c_n).$$

An evaluation of the Newton form requires $n + \frac{n(n+1)}{2}$ additions and $\frac{n(n+1)}{2}$ multiplications. The *nested Newton form*

$$p(x) = a_0 + (x - c_1)\Big(a_1 + (x - c_2)\Big(a_2 + \cdots + (x - c_{n-1})(a_{n-1} + (x - c_n)) \cdots \Big)\Big)$$

permits Horner scheme evalulation that reduces to $2n$ additions and n multiplications.

2.2 Theoretical Foundation

Theorem 1 *Let $x_0, x_1, \ldots, x_n$ be $n+1$ distinct points in $[a, b]$. There exists a unique polynomial p of degree n or less that interpolates $f(x)$ at the points $\{x_i\}$, that is, $p(x_i) = f(x_i)$, for $0 \le i \le n$.*

To prove the uniqueness, we need to use the following celebrated result.

Theorem 2 (Fundamental Theorem of Algebra) *If $p(x)$ is a polynomial of degree $n \ge 1$ with real or complex coefficients, then there exists a complex number ξ such that $p(\xi) = 0$.*

Under the theorem, we can divide a polynomial $p(x)$ by $x - r_1$, where $p(r_1) = 0$ (the existence of r_1 is assured by the theorem), to obtain a quotient polynomial $p_1(x)$ with zero remainder. If the degree of $p_1(x)$ is still greater than or equal to one, we divide it by $x - r_2$, where $p_1(r_2) = 0$, and so on. The process continues until the degree of the quotient has decreased to zero. Thus, the theorem essentially states that a polynomial $p(x)$ of degree n and with real or complex coefficients can be factorized over the complex domain into a product $a_n(x - r_1)(x - r_2)\cdots(x - r_n)$, where a_n is the leading coefficient and $r_1, r_2, \ldots, r_n$ are all of its n complex roots.

Proof Theorem 1 (Uniqueness) Suppose p and q are each polynomials of degree at most n that agree at $\{x_i\}$, then $p - q$ is a polynomial of degree at most n that vanishes at $\{x_i\}$. Therefore, by the Fundamental Theorem of Algebra,

$$(p - q)(x) = c \prod_{i=0}^{n} (x - x_i),$$

where c is some real number. The left hand side has degree at most n, the right hand side has degree exactly $n + 1$ unless $c = 0$. Therefore $c = 0$ and $p = q$. Hence p is unique.

(Existence) It suffices to construct an interpolating polynomial. This can be done with the help of polynomials $l_0(x), l_1(x), \ldots, l_n(x)$ such that

$$l_i(x_k) = \begin{cases} 1 & \text{if } i = k; \\ 0 & \text{if } i \ne k. \end{cases} \qquad i, k = 0, \ldots, n.$$

In other words,

$$l_i(x) = \prod_{\substack{k=0 \\ k \ne i}}^{n} \frac{x - x_k}{x_i - x_k}.$$

The interpolating polynomial takes the *Lagrange form*:

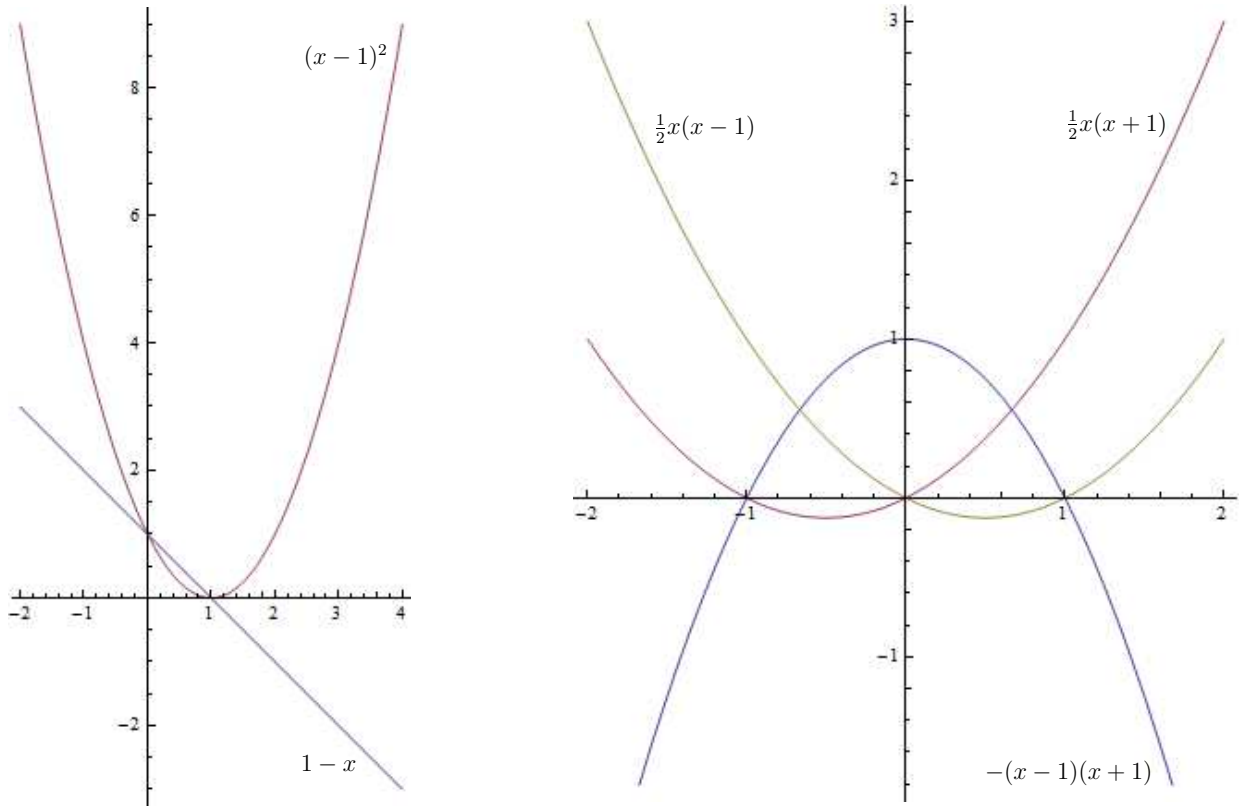
$$p_n(x) = f_0 \cdot l_0(x) + f_1 \cdot l_1(x) + \cdots + f_n \cdot l_n(x). \tag{1}$$

The polynomials $l_0, l_1, \ldots, l_n$ are called the *Lagrange polynomials*. □

The uniqueness of interpolation tells us that different ways of writing the interpolating polynomial $p(x)$ are really the same, and must therefore differ on numerical grounds (such as compactness, stabilities, efficiency, ...).

Example 1. Suppose we start with a polynomial, say,

$$f(x) = (x - 1)^2.$$



(a) (b)

Figure 1: Lagrange polynomials for interpolating $f(x) = (x-1)^2$ over two point sets: (a) $\{0, 1\}$ and (b) $\{-1, 0, 1\}$.

And suppose we have two values, at $x_0 = 0$ and $x_1 = 1$,

$$f(0) = 1 \qquad \text{and} \qquad f(1) = 0.$$

The Lagrange interpolation formula takes the form

$$\begin{aligned} p_1(x) &= f_0 l_0(x) + f_1 l_1(x) \\ &= l_0(x), \end{aligned}$$

since $f_0 = f(0) = 1$ and $f_1 = f(1) = 0$. Next, we have

$$l_0(x) = \frac{x - x_1}{x_0 - x_1} = \frac{x-1}{0-1} = 1 - x.$$

So $p_1(x) = 1 - x$, which is a line shown Figure 1(a), where we see that $p(x)$ does not match $f(x)$ well.

Now let us add the value of f at a third point, say $f(-1) = 4$. So we have

$$\begin{aligned} x_0 &= 0 \qquad & f_0 &= 1, \\ x_1 &= 1 \qquad & f_1 &= 0, \\ x_2 &= -1 \qquad & f_2 &= 4. \end{aligned}$$

The Lagrange polynomials can be computed as

$$\begin{aligned} l_0(x) &= \frac{x-x_1}{x_0-x_1}\cdot\frac{x-x_2}{x_0-x_2} &= -(x-1)(x+1), \\ l_1(x) &= \frac{x-x_0}{x_1-x_0}\cdot\frac{x-x_2}{x_1-x_2} &= \frac{1}{2}x(x+1), \\ l_2(x) &= \frac{x-x_0}{x_2-x_0}\cdot\frac{x-x_1}{x_2-x_1} &= \frac{1}{2}x(x-1). \end{aligned}$$

They are plotted in Figure 1(b). In fact, we do not need to compute $l_1(x)$ above since $f_1 = 0$. Now we have

$$\begin{aligned} p_2(x) &= f_0\cdot l_0(x) + f_1\cdot l_1(x) + f_2\cdot l_2(x) \\ &= -(x-1)(x+1) + 2x(x-1) \\ &= (x-1)(2x-x-1) \\ &= (x-1)^2. \end{aligned}$$

So, as one would expect, this approximation is exact.

The goodness of an approximation depends on the number of approximating points and also on their locations. One problem with the Lagrange interpolating polynomial is that we need n additions, $2n^2+2n$ subtractions, $2n^2+n-1$ multiplications, and $n+1$ divisions to evaluate $p(\xi)$ at a given point ξ. Even after all the denominators have been calculated once and for all and divided into the a_i values, we still need n additions, n^2+n subtractions, and n^2+n multiplications.

Another problem is that in practice, one may be uncertain as to how many interpolation points to use. So one may want to increase them over time and see whether the approximation gets better. In doing so, one would like to use the old approximation. It is not clear how to do that easily with the Lagrange form. In this sense, the Lagrange form is not incremental (plus it is also awkward to program).

2.3 Divided Differences

Instead of solving the interpolation problem all at once, one might consider solving it for smaller sets of support points first and then update these solutions to obtain the solution to the full interpolation problem. This leads to the idea of "divided differences". It is essentially a way of writing the interpolating polynomial in Newton form.

Let $p_n(x)$ denote the interpolating polynomial of degree n (or less) that interpolates $f(x)$ at $x_0,\ldots,x_n$. Suppose we write the polynomial as

$$p_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \cdots + a_n(x-x_0)\cdots(x-x_{n-1}). \tag{2}$$

Now notice that the last term on the right vanishes at $x_0,\ldots,x_{n-1}$. Therefore, the previous n terms must comprise the interpolating polynomial of degree $n-1$ (or less) that agrees with $f(x)$ at $x_0,\ldots,x_{n-1}$. In other words,

$$p_{n-1}(x) = a_0 + a_1(x-x_0) + \cdots + a_{n-1}(x-x_0)\cdots(x-x_{n-2})$$

with same $\{a_i\}$ coefficients as for $p_n(x)$. So, we can build $p_n(x)$ from $p_{n-1}(x)$, which we can build from $p_{n-2}(x)$, and so on.

Note that the evaluation of (2) for $x = \xi$ may be done recursively using Horner scheme:

$$p_n(\xi) = \Big(\dots \Big(a_n(\xi - x_{n-1}) + a_{n-1} \Big)(\xi - x_{n-2}) + \cdots a_1 \Big)(\xi - x_0) + a_0.$$

This requires n multiplications and $2n$ additions, assuming that we already have the coefficients $a_0, a_1, \dots, a_n$. Alternatively, we may evaluate each of the individual terms of (2) from left to right; with $2n-1$ multiplications and $2n$ additions we thereby calculate all of the values $p_0(\xi), p_1(\xi)$, $\dots, p_n(\xi)$. This indicates whether or not an interpolation process is converging.

It remains to determine the coefficients a_i. In principle, they can be calculated successively from

$$\begin{array}{rcccl} f_0 & = & p_n(x_0) & = & a_0, \\ f_1 & = & p_n(x_1) & = & a_0 + a_1(x_1 - x_0), \\ f_2 & = & p_n(x_2) & = & a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1), \\ & \vdots & & & \end{array}$$

This can be done with n divisions and n^2 multiplications. There is, however, a better way, which requires only $n(n+1)/2$ divisions and produces useful intermediate results.

Let us define $f[x_i, x_{i+1}, \dots, x_m]$ to be the leading coefficient of the polynomial of degree $m - i$ (or less) that agrees with $f(x)$ at the distinct points $x_i, \dots, x_m$. Then by construction, a_k, the leading coefficient of $p_k(x)$, is $f[x_0, \dots, x_k]$, called the k-th *divided difference* of f at the points $x_0, \dots, x_k$, and

$$\begin{array}{rcl} p_n(x) & = & f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ & & \quad + \cdots + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \cdots (x - x_{n-1}). \end{array}$$

Observe that

$$\begin{array}{rcl} f[x_0] & = & f(x_0), \\ f[x_0, x_1] & = & \dfrac{f(x_0) - f(x_1)}{x_0 - x_1}. \end{array}$$

More generally, we will see that

$$f[x_0, \dots, x_k] = \frac{f[x_0, \dots, x_{k-1}] - f[x_1, \dots, x_k]}{x_0 - x_k}. \tag{3}$$

To verify, first note that it holds for $k = 0$ or 1. Indeed for $k = 1$, the interpolating formula is

$$p_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0).$$

Now let us define two sets of polynomials:

$p_{k-1}(x)$ is the polynomial of degree $k-1$ (or less) which agrees with $f(x)$ at $x_0, \dots, x_{k-1}$.
$q_{k-1}(x)$ is the polynomial of degree $k-1$ (or less) which agrees with $f(x)$ at $x_1, \dots, x_k$.

The key recursive formula is

$$p_k(x) = \frac{x - x_k}{x_0 - x_k} p_{k-1}(x) + \frac{x_0 - x}{x_0 - x_k} q_{k-1}(x).$$

Now observe that

$$\begin{aligned} a_k &= f[x_0, \ldots, x_k] \\ &= \text{leading coefficient of } p_k(x) \\ &= \frac{\text{leading coeff of } p_{k-1}(x)}{x_0 - x_k} - \frac{\text{leading coeff of } q_{k-1}(x)}{x_0 - x_k} \\ &= \frac{f[x_0, \ldots, x_{k-1}] - f[x_1, \ldots, x_k]}{x_0 - x_k}. \end{aligned}$$

The calculation of divided differences can be expressed in the following tableau, called the *divided-difference scheme*:

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	$f(x_2)$		$f[x_1, x_2, x_3]$		$f[x_0, x_1, x_2, x_3, x_4]$
		$f[x_2, x_3]$		$f[x_1, x_2, x_3, x_4]$	
x_3	$f(x_3)$		$f[x_2, x_3, x_4]$		
		$f[x_3, x_4]$			
x_4	$f(x_4)$				

The procedural description is given below:

```
1  (α_0, α_1, ..., α_n) ← (f_0, f_1, ..., f_n)
2  for k = 1 to n
3      for j = n downto k
4          α_j ← (α_{j-1} − α_j) / (x_{j-k} − x_j)
```

This process requires $\frac{1}{2}(n^2 + n)$ divisions and $n^2 + n$ subtractions, so about three-fourths of the work implied in (1) has been saved.

EXAMPLE 2. $f(x) = (x - 1)^2$.

$x_0 = 0$	1		
		$\frac{1-0}{0-1} = -1$	
$x_1 = 1$	0		$\frac{-1-(-2)}{0-(-1)} = 1$
		$\frac{0-4}{1-(-1)} = -2$	
$x_2 = -1$	4		

Using only the points x_0 and x_1, we get

$$p_1(x) = f[x_0] + f[x_0, x_1](x - x_0)$$

$$\begin{aligned} &= 1 - 1 \cdot (x - 0) \\ &= 1 - x, \end{aligned}$$

as before. Using all three points x_0, x_1, x_2, we get

$$\begin{aligned} p_2(x) &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &= p_1(x) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &= 1 - x + 1 \cdot (x - 0)(x - 1) \\ &= (x - 1)^2. \end{aligned}$$

By applying fast Fourier transforms, it is possible to reduce the running time for interpolation to $O\left(n(\lg n)^2\right)$. We refer the reader to [1, pp. 298–300] for such a fast interpolation algorithm.

If we want to interpolate several polynomials at the same points $x_0, x_1, \ldots, x_n$ with varying values $f_0, f_1, \ldots, f_n$, it is desirable to rewrite the Lagrange form (1) as the following:

$$p_n(x) = \left(\frac{f_0 w_0}{x - x_0} + \cdots + \frac{f_n w_n}{x - x_n}\right) \Bigg/ \left(\frac{w_0}{x - x_0} + \cdots + \frac{w_n}{x - x_n}\right),$$

when $x \notin \{x_0, x_1, \ldots, x_n\}$, where

$$w_k = 1/(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n).$$

This form is also recommended for it numerical stability.

An important and somewhat surprising application of polynomial interpolation was discovered by Adi Shamir [5], who observed that polynomials mod p, where p is some chosen polynomial, can be used to "share a secret." This means that we can design a system of secret keys or passwords such that the knowledge of any $n + 1$ of the keys enables efficient calculation of a magic number N that unlocks a "door", but the knowledge of any n of the keys gives no information whatsoever about N. Shamir's amazingly simple solution to this problem is to choose a random polynomial

$$p(x) = a_n x^n + \cdots + a_1 x + a_0,$$

where $0 \le a_i < q$ for some large prime number q. Each part of the secret is an integer x such that $0 < x < q$, together with the value of $u(x) \bmod q$; and the supersecret number $N = a_0$. Given $n + 1$ values $p(x_i)$, we can deduce N by interpolation. But if only n values of $p(x_i)$ are given, there is a unique polynomial $p(x)$ for every given constant term that has the same values at $x_1, \ldots, x_n$; thus the n values do not make one particular N more likely than any other.

3 Error Estimation

Let $p_n(x)$ be the interpolating polynomial of degree at most n that interpolates function $f(x)$ at $x_0, x_1, \ldots, x_n$. The interpolation error of $p_n(x)$ is given by

$$e_n(x) = f(x) - p_n(x).$$

Let $\bar{x}$ be a point distinct from the above points, and let $p_{n+1}(x)$ interpolate $f(x)$ at $x_0, x_1, \ldots, x_n, \bar{x}$. It then follows that

$$p_{n+1}(x) = p_n(x) + f[x_0, \ldots, x_n, \bar{x}] \prod_{i=0}^{n} (x - x_i). \tag{4}$$

Therefore the interpolation error is

$$e_n(\bar{x}) = f[x_0, \ldots, x_n, \bar{x}] \prod_{i=0}^{n} (\bar{x} - x_i),$$

for all $\bar{x} \notin \{x_0, \ldots, x_n\}$. Namely, the error is the "next" term in the Newton form.

Of course, without knowing $f(\bar{x})$ we cannot evaluate $f[x_0, \ldots, x_n, \bar{x}]$, and therefore we do not know how big the error is. Nevertheless, the next theorem tells us that the divided difference $f[x_0, \ldots, x_k]$ in fact depends on the kth derivative of $f(x)$. This will allow us to estimate the interpolation error in certain cases.

Lemma 3 *Let $f(x)$ be a real-valued function that is k times differentiable in $[a, b]$. If $x_0, \ldots, x_k$ are $k+1$ distinct points in $[a, b]$, then there exists $\xi \in (a, b)$ such that*

$$f[x_0, \ldots, x_k] = \frac{f^{(k)}(\xi)}{k!}.$$

From Lemma 3 and equation (4) we immediately obtain a closed form of the interpolation error.

Theorem 4 *Let a real-valued function $f(x)$ be $n+1$ times differentiable in $[a, b]$. If the polynomial $p_n(x)$ of degree $\leq n$ interpolates $f(x)$ at distinct points $x_0, \ldots, x_n$ in $[a, b]$, then for all $\bar{x} \in [a, b]$, there exists $\xi \in (a, b)$ such that*

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

So if we have a bound on the $(n+1)$-st derivative of f, then we can estimate e_n accordingly.

Conversely, Theorem 4 also implies Lemma 3. To see this, we need only compare the Newton form

$$f(x_k) - p_{k-1}(x_k) = f[x_0, \ldots, x_k](x_k - x_0) \cdots (x_k - x_{k-1}).$$

with the error expression in Theorem 4

$$e_{k-1}(x_k) = f(x_k) - p_{k-1}(x_k) = \frac{f^{(k)}(\xi)}{k!} \prod_{i=0}^{k-1} (x_k - x_i),$$

where $\xi \in (a, b)$.

EXAMPLE 1. Determine the spacing h in a table of evenly spaced values of the function $f(x) = \sqrt{x}$ between 1 and 2, so that interpolation with a second-degree polynomial in this table will yield a desired accuracy.

Such a table contains the values $f(x_i)$, $i = 0, \ldots, n = \frac{1}{h}$, at points $x_i = 1 + ih$. If $\bar{x} \in [x_{i-1}, x_{i+1}]$, then we approximate $f(\bar{x})$ with $p_2(\bar{x})$, where $p_2(x)$ is the polynomial of degree 2 that interpolates f at x_{i-1}, x_i, x_{i+1}. By Theorem 4, the error is

$$e_2(\bar{x}) = \frac{f'''(\xi)}{3!}(\bar{x} - x_{i-1})(\bar{x} - x_i)(\bar{x} - x_{i+1}),$$

where ξ depends on $\bar{x}$. Though ξ is unknown to us, we can derive the following upper bounds:

$$\begin{aligned} f'''(\xi) &\leq \max_{1 \leq x \leq 2} |f'''(x)| \\ &= \max_{1 \leq x \leq 2} \frac{3}{8} x^{-\frac{5}{2}} \\ &= \frac{3}{8}, \end{aligned}$$

and for any $\bar{x} \in [x_{i-1}, x_{i+1}]$,

$$\begin{aligned} \Big|(\bar{x} - x_{i-1})(\bar{x} - x_i)(\bar{x} - x_{i+1})\Big| &\leq \max_{y \in [-h,h]} \Big|(y-h)y(y+h)\Big| \\ &= \frac{2}{3\sqrt{3}} h^3. \end{aligned}$$

In the last step, the maximum absolute value of $g(y) = (y-h)y(y+h)$ over $[-h, h]$ is obtained as follows. Since

$$g(h) = g(-h) = g(0) = 0,$$

$g(y)$ achieves its maximum (or minimum) in the interior of the interval. We have

$$g'(y) = 3y^2 - h^2.$$

That $g'(y) = 0$ yields $y = \pm\frac{h}{\sqrt{3}}$, where the maximum of $|g(y)|$ is attained.

Thus we have derived a bound on the interpolation error:

$$\begin{aligned} |e_2(\bar{x})| &\leq \frac{1}{3!} \cdot \frac{3}{8} \cdot \frac{2}{3\sqrt{3}} h^3 \\ &= \frac{h^3}{24\sqrt{3}}. \end{aligned}$$

Suppose we want the accuracy of at least 7 places after zero. We should choose h such that

$$\frac{h^3}{24\sqrt{3}} < 5 \cdot 10^{-8}.$$

This yields $h \approx 0.0127619$. And the number of entries is about 79.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] S. D. Conte and Carl de Boor. *Elementary Numerical Analysis*. McGraw-Hill, Inc., 2nd edition, 1972.

[3] M. Erdmann. Lecture notes for *16-811 Mathematical Fundamentals for Robotics*. The Robotics Institute, Carnegie Mellon University, 1998.

[4] D. E. Knuth. *Seminumerical Algorithms*, vol. 2 of *The Art of Computer Programming*, 3rd edition. Addison-Wesley, 1998.

[5] A. Shamir How to share a secret. *Communications of the ACM*, 22:612-613, 1979

[6] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer-Verlag New York, Inc., 2nd edition, 1993.