

PART II
Lecture Notes on
Interpolation
MATH 435

Professor Biswa Nath Datta
Department of Mathematical Sciences
Northern Illinois University
DeKalb, IL. 60115 USA

E-mail: dattab@math.niu.edu

URL: www.math.niu.edu/~dattab



PART II.
Interpolation

2 Interpolation

2.1 Problem Statement and Applications

Consider the following table:

x_0	f_0
x_1	f_1
x_2	f_2
\vdots	\vdots
x_k	f_k
\vdots	\vdots
x_n	f_n

In the above table, $f_k, k = 0, \dots, n$ are assumed to be the values of a certain function $f(x)$, evaluated at $x_k, k = 0, \dots, n$ in an interval containing these points. **Note that only the functional values are known, not the function $f(x)$ itself.** The problem is to find f_u corresponding to a nontabulated intermediate value $x = u$.

Such a problem is called an **Interpolation Problem**. The numbers x_0, x_1, \dots, x_n are called the **nodes**.

Interpolation Problem
Given $(n + 1)$ points: $(x_0, f_0), (x_1, f_1), \dots, (x_n, f_n)$, find f_u
corresponding to x_u , where $x_0 < x_u < x_n$; assum- ing that
f_0, f_1, \dots, f_n are the values of a certain function $f(x)$
at $x = x_0, x_1, \dots, x_n$, respectively.

The Interpolation problem is also a classical problem and dates back to the time of **Newton** and **Kepler**, who needed to solve such a problem in analyzing data on the positions of stars and planets. It is also of interest in numerous other practical applications. Here is an example.

2.2 Existence and Uniqueness

It is well-known that a continuous function $f(x)$ on $[a, b]$ can be approximated as close as possible by means of a polynomial. Specifically, for each $\epsilon > 0$, there exists a polynomial $P(x)$ such that $|f(x) - P(x)| < \epsilon$ for all x in $[a, b]$. This is a classical result, known as **Weierstrass Approximation Theorem**.

Knowing that $f_k, k = 0, \dots, n$ are the values of a certain function at x_k , the most obvious thing then to do is to construct a polynomial $P_n(x)$ of degree at most n that passes through the $(n + 1)$ points: $(x_0, f_0), (x_1, f_1), \dots, (x_n, f_n)$.

Indeed, if the nodes x_0, x_1, \dots, x_n are assumed to be distinct, then such a polynomial always does exist and is unique, as can be seen from the following.

Let $P_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ be a polynomial of degree at most n . If $P_n(x)$ interpolates at x_0, x_1, \dots, x_n , we must have, by definition

$$\begin{aligned} P_n(x_0) &= f_0 = a_0 + a_1x_0 + a_2x_0^2 + \dots + a_nx_0^n \\ P_n(x_1) &= f_1 = a_0 + a_1x_1 + a_2x_1^2 + \dots + a_nx_1^n \\ &\vdots \\ P_n(x_n) &= f_n = a_0 + a_1x_n + a_2x_n^2 + \dots + a_nx_n^n \end{aligned} \tag{2.1}$$

These equations can be written in matrix form:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix} \tag{2.2}$$

Because x_0, x_1, \dots, x_n are distinct, it can be shown [**Exercise**] that the matrix of the above system is nonsingular. Thus, the linear system for the unknowns a_0, a_1, \dots, a_n has a unique solution, in view of the following well-known result, available in any linear algebra text book.

The $n \times n$ algebraic linear system $Ax = b$ has a unique solution for every b if and only if A is nonsingular.

This means that $P_n(x)$ exists and is unique.

Theorem 2.1 (Existence and Uniqueness Theorem for Polynomial Interpolation)

Given $(n + 1)$ distinct points x_0, x_1, \dots, x_n and the associated values f_0, f_1, \dots, f_n of a function $f(x)$ at these points (that is, $f(x_i) = f_i, i = 0, 1, \dots, n$), there is a **unique polynomial** $P_n(x)$ of degree at most n such that $P_n(x_i) = f_i, i = 0, 1, \dots, n$. The coefficients of this polynomial can be obtained by solving the $(n + 1) \times (n + 1)$ linear system (2.2).

Definition: The polynomial $P_n(x)$ in Theorem 2.1 is called the **interpolating polynomial**.

2.3 The Lagrange Interpolation

Once we know that the interpolating polynomial exists and is unique, the problem then becomes how to construct an interpolating polynomial; that is, how to construct a polynomial $P_n(x)$ of degree at most n , such that

$$P_n(x_i) = f_i, i = 0, 1, \dots, n.$$

It is natural to obtain the polynomial by solving the linear system (2.1) in the previous section. Unfortunately, the matrix of this linear system, known as the **Vandermonde Matrix**, is usually **highly ill-conditioned**, and the **solution of such an ill-conditioned system, even by the use of a stable method, may not be accurate**. There are, however, several other ways to construct such a polynomial, that do not require solution of a Vandermonde system. We describe one such in the following:

Suppose $n = 1$, that is, suppose that we have only two points $(x_0, f_0), (x_1, f_1)$, then it is easy to see that the linear polynomial

$$P_1(x) = \frac{x - x_1}{(x_0 - x_1)}f_0 + \frac{(x - x_0)}{(x_1 - x_0)}f_1$$

is an interpolating polynomial, because

$$P_1(x_0) = f_0, P_1(x_1) = f_1.$$

For convenience, we shall write the polynomial $P_1(x)$ in the form

$$P_1(x) = L_0(x)f_0 + L_1(x)f_1,$$

where, $L_0(x) = \frac{x - x_1}{x_0 - x_1}$, and $L_1(x) = \frac{x - x_0}{x_1 - x_0}$.

Note that both the polynomials $L_0(x)$ and $L_1(x)$ are polynomials of degree 1.

The concept can be generalized easily for polynomials of higher degrees.

To generate polynomials of higher degrees, let's define the set of polynomials $\{L_k(x)\}$ recursively, as follows:

$$L_k(x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_0)(x_k - x_1) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}, \quad k = 0, 1, 2, \dots, n. \quad (2.3)$$

We will now show that the polynomial $P_n(x)$ defined by

$$P_n(x) = L_0(x)f_0 + L_1(x)f_1 + \cdots + L_n(x)f_n \quad (2.4)$$

is an interpolating polynomial.

To see this, note that

$$\begin{aligned} L_0(x) &= \frac{(x - x_1) \cdots (x - x_n)}{(x_0 - x_1) \cdots (x_0 - x_n)} \\ L_1(x) &= \frac{(x - x_0)(x - x_2) \cdots (x - x_n)}{(x_1 - x_0)(x_1 - x_2) \cdots (x_1 - x_n)} \\ &\vdots \\ L_n(x) &= \frac{(x - x_0)(x - x_1)(x - x_2) \cdots (x - x_{n-1})}{(x_n - x_0)(x_n - x_1)(x_n - x_2) \cdots (x_n - x_{n-1})} \end{aligned}$$

Also, note that

$$L_0(x_0) = 1, \quad L_0(x_1) = L_0(x_2) = \cdots = L_0(x_n) = 0$$

$$L_1(x_1) = 1, \quad L_1(x_0) = L_1(x_2) = \cdots = L_1(x_n) = 0$$

In general

$L_k(x_k) = 1$ and $L_k(x_i) = 0, \quad i \neq k.$
--

Thus

$$P_n(x_0) = L_0(x_0)f_0 + L_1(x_0)f_1 + \cdots + L_n(x_0)f_n = f_0$$

$$P_n(x_1) = L_0(x_1)f_0 + L_1(x_1)f_1 + \cdots + L_n(x_1)f_n = 0 + f_1 + \cdots + 0 = f_1$$

\vdots

$$P_n(x_n) = L_0(x_n)f_0 + L_1(x_n)f_1 + \cdots + L_n(x_n)f_n = 0 + 0 + \cdots + 0 + f_n = f_n$$

That is, the polynomial $P_n(x)$ has the property that $P_n(x_k) = f_k, \quad k = 0, 1, \dots, n.$

The polynomial $P_n(x)$ defined by (2.3) is known as the **Lagrange Interpolating Polynomial**.

Example 2.1 Interpolate $f(3)$ from the following table using Lagrangian interpolation:

0	7
1	13
2	21
4	43

$$L_0(x) = \frac{(x-1)(x-2)(x-4)}{(-1)(-2)(-4)}$$

$$L_1(x) = \frac{(x-0)(x-2)(x-4)}{1 \cdot (-1)(-3)}$$

$$L_2(x) = \frac{(x-0)(x-1)(x-4)}{2 \cdot 1 \cdot (-2)}$$

$$L_3(x) = \frac{(x-0)(x-1)(x-2)}{4 \cdot 3 \cdot 2}$$

From (2.4) with $n = 3$, we have

$$P_3(x) = 7L_0(x) + 13L_1(x) + 21L_2(x) + 43L_3(x)$$

$$\text{Now, } L_0(3) = \frac{1}{4}, \quad L_1(3) = -1, \quad L_2(3) = \frac{3}{2}, \quad L_3(3) = \frac{1}{4}.$$

$$\text{So, } P_3(3) = 7L_0(3) + 13L_1(3) + 21L_2(3) + 43L_3(3) = 31.$$

Verify: Note that $f(x)$ in this case is $f(x) = x^2 + 5x + 7$, and the exact value of $f(x)$ at $x = 3$ is 31.

Example 2.2 *Given*

i	x_i	$f(x_i)$
0	2	$\frac{1}{2}$
1	2.5	$\frac{1}{2.5}$
2	4	$\frac{1}{4}$

Interpolate $f(x)$ at $x = 3$.

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x - 2.5)(x - 4)}{(-0.5)(-2)} = (x - 2.5)(x - 4)$$

$$L_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(x - 2)(x - 4)}{(0.5)(-1.5)} = -\frac{1}{0.75}(x - 2)(x - 4)$$

$$L_2(x) = \frac{(x - x_1)(x - x_0)}{(x_2 - x_1)(x_2 - x_0)} = \frac{1}{3}(x - 2.5)(x - 2)$$

$$\text{So, } P_2(x) = f(x_0)L_0(x) + f(x_1)L_1(x) + f(x_2)L_2(x) = \frac{1}{2}L_0(x) + \frac{1}{2.5}L_1(x) + \frac{1}{4}L_2(x)$$

$$\text{Thus, } P_2(3) = \frac{1}{2}L_0(3) + \frac{1}{2.5}L_1(3) + \frac{1}{4}L_2(3) = \frac{1}{2}(-0.5) + \frac{1}{2.5} \left(\frac{1}{0.75} \right) + \frac{1}{4} \left(\frac{.5}{3} \right) = 0.3250$$

Verify: (The value of $f(x)$ at $x = 3$ is $\frac{1}{3} \approx 0.3333$).

2.4 Error in Interpolation

If $f(x)$ is approximated by an interpolating polynomial $P_n(x)$, we would like to obtain an expression for the error of interpolation at a give intermediate point, say, \bar{x} .

That is, we would like to calculate $E(\bar{x}) = f(\bar{x}) - P_n(\bar{x})$.

Note that, since $P_n(x_i) = f(x_i)$, $E(x_i) = 0$, $i = 0, 1, 2, \dots, n$, that is, **there are no errors of interpolating at a tabulated point.**

Here is a result for the error expression $E(\bar{x})$.

For proof of Theorem 2.2 we need the following result from Calculus:

Generalized Rolle's Theorem

Suppose that $f(x)$ is continuous on $[a, b]$ and n times differentiable on (a, b) . If $f(x)$ has n distinct zeros in $[a, b]$, then $f^{(n+1)}(c) = 0$ where $a < c < b$.

Theorem 2.2 (Interpolation-Error Theorem) *Let $P_n(x)$ be the interpolating polynomial that interpolates at $(n + 1)$ distinct numbers in $[a, b]$, and let $f(x)$ be $(n + 1)$ times continuously differentiable on $[a, b]$. Then for every \bar{x} in $[a, b]$, there exists a number $\xi = \xi(\bar{x})$ in (a, b) (depending on \bar{x}) such that*

$$E_n(\bar{x}) = f(\bar{x}) - P_n(\bar{x}) = \frac{f^{(n+1)}(\xi(\bar{x}))}{(n + 1)!} \prod_{i=0}^n (\bar{x} - x_i). \quad (2.5)$$

Proof: If \bar{x} is one of the numbers x_0, x_1, \dots, x_n : then the result follows trivially. Because, the error in this case is zero, and the result will hold for any arbitrary ξ .

Next, assume that \bar{x} is not one of the numbers x_0, x_1, \dots, x_n .

Define a function $g(t)$ in variable t in $[a, b]$:

$$g(t) = f(t) - P_n(t) - [f(\bar{x}) - P_n(\bar{x})] * \left[\frac{(t - x_0)(t - x_1) \cdots (t - x_n)}{(\bar{x} - x_0)(\bar{x} - x_1) \cdots (\bar{x} - x_n)} \right]. \quad (2.6)$$

Then, noting that $f(x_k) = P_n(x_k)$, for $k = 0, 1, 2, \dots, n$, we have

$$\begin{aligned} g(x_k) &= f(x_k) - P_n(x_k) - [f(\bar{x}) - P_n(\bar{x})] \left[\frac{(x_k - x_0) \cdots (x_k - x_n)}{(\bar{x} - x_0) \cdots (\bar{x} - x_n)} \right] \\ &= P_n(x_k) - P_n(x_k) - [f(\bar{x}) - P_n(\bar{x})] \times 0 \\ &= 0 \end{aligned} \quad (2.7)$$

(Note that the numerator of the fraction appearing above contains the term $(x_k - x_k) = 0$). Furthermore,

$$\begin{aligned} g(\bar{x}) &= f(\bar{x}) - P_n(\bar{x}) - [f(\bar{x}) - P_n(\bar{x})] * \left[\frac{(\bar{x} - x_0) \cdots (\bar{x} - x_n)}{(\bar{x} - x_0) \cdots (\bar{x} - x_n)} \right] \\ &= f(\bar{x}) - P_n(\bar{x}) - f(\bar{x}) + P_n(\bar{x}) \\ &= 0 \end{aligned} \quad (2.8)$$

Thus, $g(t)$ becomes identically zero at $(n + 2)$ distinct points: x_0, x_1, \dots, x_n , and \bar{x} . Furthermore, $g(t)$ is $(n + 1)$ times continuously differentiable, since $f(x)$ is so.

Therefore, by **generalized Rolle's theorem** there exists a number $\xi(\bar{x})$ in (a, b) such that $g^{(n+1)}(\xi) = 0$.

Let's compute $g^{(n+1)}(t)$ now. From (2.5) we have

$$g^{(n+1)}(t) = f^{(n+1)}(t) - P_n^{(n+1)}(t) - [f(\bar{x}) - P_n(\bar{x})] \frac{d^{n+1}}{dt^{n+1}} \left[\frac{(t - x_0)(t - x_1) \cdots (t - x_n)}{((\bar{x} - x_0)(\bar{x} - x_1) \cdots (\bar{x} - x_n))} \right] \quad (2.9)$$

Then [**Exercise**]:

$$\begin{aligned} &\frac{d^{n+1}}{dt^{n+1}} \left[\frac{(t - x_0)(t - x_1) \cdots (t - x_n)}{(\bar{x} - x_0)(\bar{x} - x_1) \cdots (\bar{x} - x_n)} \right] \\ &= \frac{1}{(\bar{x} - x_0)(\bar{x} - x_1) \cdots (\bar{x} - x_n)} \cdot \frac{d^{n+1}}{dt^{n+1}} [(t - x_0)(t - x_1) \cdots (t - x_n)] \\ &= \frac{1}{(\bar{x} - x_0)(\bar{x} - x_1) \cdots (\bar{x} - x_n)} (n + 1)! \end{aligned}$$

(note that the expression within [] in (2.9) is a polynomial of degree $(n + 1)$).

Also, $P_n^{(n+1)}(t) = 0$, because P_n is a polynomial of degree at most n . Thus, $P_n^{(n+1)}(\xi) = 0$.

So,

$$g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - \frac{(f(\bar{x}) - P_n(\bar{x}))}{(\bar{x} - x_0) \cdots (\bar{x} - x_n)} (n+1)! \quad (2.10)$$

Since $g^{(n+1)}(\xi) = 0$, from (2.10), we have $E_n(\bar{x}) = f(\bar{x}) - P_n(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (\bar{x} - x_0) \cdots (\bar{x} - x_n)$.

■

Remark: To obtain the error of interpolation using the above theorem, we need to know the $(n+1)$ th derivative of the $f(x)$ or its absolute maximum value on the interval $[a, b]$. Since in practice this value is hardly known, this error formula is of limited use only.

Example 2.3 Let's compute the maximum absolute error for Example 2.2. Here $n = 2$.

$$\begin{aligned} E_2(\bar{x}) &= f(\bar{x}) - P_2(\bar{x}) \\ &= \frac{f^{(3)}(\xi)}{3!} (\bar{x} - x_0)(\bar{x} - x_1)(\bar{x} - x_2) \end{aligned}$$

To know the maximum value of $E_2(\bar{x})$, we need to know $f^{(3)}(x)$.

Let's compute this now:

$$f(x) = \frac{1}{x}, \quad f'(x) = -\frac{1}{x^2}, \quad f''(x) = \frac{2}{x^3}, \quad f^{(3)}(x) = -\frac{6}{x^4}.$$

So, $|f^{(3)}(\xi)| < \frac{6}{2^4} = \frac{6}{16}$ for $2 < x \leq 4$.

Since $\bar{x} = 3$, $x_0 = 2$, $x_1 = 2.5$, $x_2 = 4$, we have

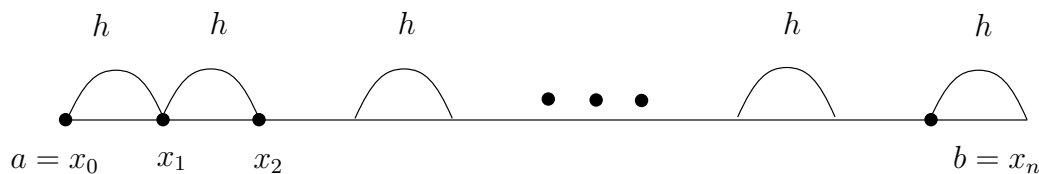
$$|E_2(\bar{x})| \leq \left| \frac{6}{16} \times \frac{1}{6} (3-2)(3-2.5)(3-4) \right| = 0.03125.$$

Note that in four-digit arithmetic, the difference between the value obtained by interpolation and the exact value is $0.3333 - 0.3250 = 0.0083$.

2.5 Simplification of the Error Bound for Equidistant Nodes

The error formula in Theorem 2.2 can be simplified in case the tabulated points (nodes) are equally spaced; because, in this case it is possible to obtain a nice bound for the expression: $(\bar{x} - x_0)(\bar{x} - x_1) \cdots (\bar{x} - x_n)$.

Suppose the nodes are equally spaced with spacing h ; that is $x_{i+1} - x_i = h$.



Then it can be shown [**Exercise**] that

$$|(\bar{x} - x_0)(\bar{x} - x_1) \cdots (\bar{x} - x_n)| \leq \frac{h^{n+1}}{4} n!$$

If we also assume that $|f^{(n+1)}(x)| \leq M$, then we have

$$|E_n(\bar{x})| = |f(\bar{x}) - P_n(\bar{x})| \leq \frac{M}{(n+1)!} \frac{h^{n+1}}{4} n! = \frac{Mh^{n+1}}{4(n+1)}. \quad (2.11)$$

Example 2.4 Suppose a table of values for $f(x) = \cos x$ has to be prepared in $[0, 2\pi]$ with nodes of spacing h , using **linear interpolation**, with an error of interpolation of at most 5×10^{-7} . How small should h be?

Here $n = 1$.

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

$$\max |f^{(2)}(x)| = 1, \quad \text{for } 0 \leq x \leq 2\pi$$

Thus $M = 1$.

So, by (2.11) above we have

$$|E_1(\bar{x})| = |f(\bar{x}) - P_1(\bar{x})| \leq \frac{h^2}{8}.$$

Since the maximum error has to be 5×10^{-7} , we must have:

$$\frac{h^2}{8} \leq 5 \times 10^{-7} = \frac{1}{2} \times 10^{-6}. \quad \text{That is, } h \leq 6.3246 \times 10^{-4}.$$

Example 2.5 Suppose a table is to be prepared for the function $f(x) = \sqrt{x}$ on $[1, 2]$. Determine the spacing h in a table such that the interpolation with a polynomial of degree 2 will give accuracy $\epsilon = 5 \times 10^{-8}$.

We first compute the maximum absolute error.

$$\text{Since } f^{(3)}(x) = \frac{3}{8}x^{-\frac{5}{2}},$$

$$M = |f^{(3)}(x)| \leq \frac{3}{8} \quad \text{for } 1 \leq x \leq 2.$$

Thus, taking $n = 2$ in (2.11) the maximum (absolute) error is $\frac{3}{8} \times \frac{h^3}{12} = \frac{1}{32}h^3$.

Thus, to have an accuracy of $\epsilon = 5 \times 10^{-8}$, we must have $\frac{1}{32}h^3 < 5 \times 10^{-8}$ or $h^3 < 160 \times 10^{-8}$.

This means that a spacing h of about $h = \sqrt[3]{160 \times 10^{-8}} = 0.0117$ will be needed in the Table to guarantee the accuracy of 5×10^{-8} .

2.6 Divided Differences and the Newton-Interpolation Formula

A major difficulty with the Lagrange Interpolation is that one is not sure about the degree of interpolating polynomial needed to achieve a certain accuracy. Thus, if the accuracy is not good enough with polynomial of a certain degree, one needs to increase the degree of the polynomial, and **computations need to be started all over again.**

Furthermore, computing various Lagrangian polynomials is an expensive procedure. **It is, indeed, desirable to have a formula which makes use of $P_{k-1}(x)$ in computing $P_k(x)$.**

The following form of interpolation, known as **Newton's interpolation** allows us to do so. The idea is to obtain the interpolating polynomial $P_n(x)$ in the following form:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}) \quad (2.12)$$

The constants a_0 through a_n can be determined as follows:

$$\text{For } x = x_0, \quad P_n(x_0) = a_0 = f_0$$

$$\text{For } x = x_1, \quad P_n(x_1) = a_0 + a_1(x_1 - x_0) = f_1,$$

which gives

$$a_1 = \frac{f_1 - a_0}{x_1 - x_0} = \frac{f_1 - f_0}{x_1 - x_0}.$$

The other numbers $a_i, i = 2, \dots, n$ can similarly be obtained.

It is convenient to introduce the following notation, because we will show how the numbers a_0, \dots, a_n can be obtained using these notations.

$$f[x_i] = f(x_i) \text{ and } f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$$

Similarly,

$$f[x_i, x_{i+1}, \dots, x_{i+k-1}, x_{i+k}] = \frac{f[x_{i+1}, \dots, x_{i+k-1}, x_{i+k}] - f[x_i, x_{i+1}, \dots, x_{i+k-1}]}{x_{i+k} - x_i}$$

With these notations, we then have

$$\begin{aligned} a_0 &= f_0 = f(x_0) = f[x_0] \\ a_1 &= \frac{f_1 - f_0}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = f[x_0, x_1]. \end{aligned}$$

Continuing this, it can be shown that **[Exercise]**

$$a_k = f[x_0, x_1, \dots, x_k]. \quad (2.13)$$

The number $f[x_0, x_1, \dots, x_k]$ is called the k -th **divided difference**.

Substituting these expressions of a_k in (2.13), the interpolating polynomial $P_n(x)$ now can be written in terms of the **divided differences**:

$$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \cdots (x - x_{n-1}). \quad (2.14)$$

Notes:

- (i) Each divided difference can be obtained from two previous ones of lower orders.
 For example, $f[x_0, x_1, x_2]$ can be computed from $f[x_0, x_1]$, and $f[x_1, x_2]$, and so on.
 Indeed, they can be arranged in form of a table as shown below:

Table of Divided Differences ($n = 4$)

x	$f(x)$	1 st Divided Difference	2 nd Divided Difference	3 rd Divided Difference	4 th Divided Difference
x_0	f_0				
		$f[x_0, x_1]$			
x_1	f_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_2		$f[x_1, x_2, x_3]$		
		$f[x_2, x_3]$		$f[x_1, x_2, x_3, x_4]$	$f[x_0, x_1, x_2, x_3, x_4]$
x_3	f_3		$f[x_2, x_3, x_4]$		
		$f[x_3, x_4]$			
x_4	f_4				

- (ii) Note that in computing $P_n(x)$ we need only the diagonal entries of the above table; that is, we need only $f[x_0], f[x_0, x_1], \dots, f[x_0, x_1, \dots, x_n]$.
- (iii) Since the divided differences are generated recursively, the interpolating polynomials of successively higher degrees can also be generated recursively. **Thus the work done previously can be used gainfully.**

For example,

$$\begin{aligned}
 P_1(x) &= f[x_0] + f[x_0, x_1](x - x_0) \\
 P_2(x) &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\
 &= P_1(x) + f[x_0, x_1, x_2](x - x_0)(x - x_1).
 \end{aligned}$$

$$\begin{aligned}
 \text{Similarly, } P_3(x) &= P_2(x) + f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2) \\
 P_4(x) &= P_3(x) + f[x_0, x_1, x_2, x_3, x_4](x - x_0)(x - x_1)(x - x_2)(x - x_3)
 \end{aligned}$$

and so on.

Thus, in computing $P_2(x)$, $P_1(x)$ has been gainfully used; in computing $P_3(x)$, $P_2(x)$ has been gainfully used, etc.

Example 2.6 Interpolate at $x = 2.5$ using the following Table, with polynomials of degree 3 and 4.

i	x_i	f_i	1st diff.	2nd diff.	3rd diff.	4th diff.	5th diff.
0	1.0	0					
			0.3522				
1	1.5	0.17609		-0.1023			
			0.2499		0.0265534		
2	2.0	0.30103		-0.0491933		-0.006409	
			0.1761		0.01053		-0.001413
3	3.0	0.47712		-0.0281333		-0.002169	
			0.1339		0.005107		
4	3.5	0.54407		-0.01792			
			0.11598				
5	4.0	0.60206					

From (2.14), with $n = 3$, we have

$$\begin{aligned}
 P_3(2.5) &= 0 + (2.5 - 1.0)(0.35222) + (2.5 - 1.0)(2.5 - 1.5)(-0.1023) \\
 &\quad + (2.5 - 1.0)(2.5 - 1.5)(2.5 - 2.0)(.0265534) \\
 &= .52827 - .15345 + .019915 \\
 &= \boxed{0.394795}
 \end{aligned}$$

Similarly, with $n = 4$, we have

$$\begin{aligned}
 P_4(2.5) &= P_3(2.5) + (2.5 - 1.0)(2.5 - 1.5)(2.5 - 2.0)(2.5 - 3.0)(-.006409) \\
 &= 0.394795 + .002403 = \boxed{0.397198}.
 \end{aligned}$$

Note that $P_4(2.5) - P_3(2.5) = \boxed{0.002403}$.

Verification. The above is a table for $\log(x)$.

The exact value of $\log(2.5)$ (correct up to 5 decimal places) is 0.39794.

(Note that in computing $P_4(2.5)$, $P_3(2.5)$ computed previously has been gainfully used).

If we use the notation $D_{ij} = f[x_i, \dots, x_{i+j}]$. Then

Algorithm 2.1 *Algorithm for Generating Divided Differences***Inputs:**

The definite numbers x_0, x_1, \dots, x_n and the values f_0, f_1, \dots, f_n .

Outputs:

The Divided Difference $D_{00}, D_{11}, \dots, D_{nn}$.

Step 1: (Initialization). Set

$$D_{i,0} = f_i, \quad i = 0, 1, 2, \dots, n.$$

Step 2: For $i = 1, 2, \dots, n$ do

$$\begin{array}{l}
 j = 1, 2, \dots, i \text{ do} \\
 D_{ij} = \frac{D_{i,j-1} - D_{i-1,j-1}}{x_i - x_{i-j}}
 \end{array}$$

End

A Relationship Between n th Divided Difference and the n^{th} Derivative

The following theorem shows how the n th derivative of a function $f(x)$ is related to the n th divided difference. The proof is omitted. It can be found in any advanced numerical analysis text book (e.g., Atkins on (1978), p. 144).

Theorem 2.3 *Suppose f is n times continuously differentiable and x_0, x_1, \dots, x_n are $(n+1)$ distinct numbers in $[a, b]$. Then there exists a number ξ in (a, b) such that*

$$f[x_0, x_1, \dots, x_n] = \frac{f^{(n)}(\xi)}{n!}$$

The Newton Interpolation with Equally Spaced Nodes

Suppose again that x_0, \dots, x_n are equally spaced with spacing h ;

that is $x_{i+1} - x_i = h, \quad i = 0, 1, 2, \dots, n-1$. Let $x - x_0 = sh$.

Then $x - x_0 = sh$

$$x - x_1 = x - x_0 + x_0 - x_1 = (x - x_0) - (x_1 - x_0) = sh - h = (s-1)h$$

In general, $x - x_i = (s-i)h$.

So,

$$\begin{aligned}
 P_n(x) &= P_n(x_0 + sh) = f[x_0] + f[x_0, x_1](x - x_0) + \dots + f[x_0, x_1, \dots, x_n](x - x_0) \dots (x - x_{n-1}) \\
 &= f[x_0] + sh f[x_0, x_1] + s(s-1)h^2 f[x_0, x_1, x_2] + \dots + s(s-1) \dots (s-h+1)h^n f[x_0, \dots, x_n] \\
 &= \sum_{k=0}^n s(s-1) \dots (s-k+1)h^k f[x_0, \dots, x_k].
 \end{aligned}
 \tag{2.15}$$

Invoke now the notation:

$$\binom{s}{k} = \frac{s(s-1) \dots (s-k+1)}{k!}$$

We can then write

$$P_n(x) = P_n(x_0 + sh) = \sum_{k=0}^n \binom{s}{k} h^k k! f[x_0, \dots, x_n] \quad (2.16)$$

The Newton Forward-Difference Formula

Let's introduce the notations:

$$\Delta f_i = f(x_{i+1}) - f(x_i).$$

$$\text{Then, } \Delta f_0 = f(x_1) - f(x_0)$$

So,

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{\Delta f_0}{h}. \quad (2.17)$$

Also, note that

$$\begin{aligned} \Delta^2 f_0 &= \Delta(\Delta f_0) = \Delta(f(x_1) - f(x_0)) \\ &= \Delta f(x_1) - \Delta f(x_0) \\ &= f(x_2) - f(x_1) - f(x_1) + f(x_0) \\ &= f(x_2) - 2f(x_1) + f(x_0) \end{aligned} \quad (2.18)$$

So,

$$\begin{aligned} f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{(x_2 - x_0)} \\ &= \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{(x_2 - x_0)} \\ &= \frac{f(x_2) - 2f(x_1) + f(x_0)}{h \times 2h} = \frac{\Delta^2 f_0}{2h^2} \end{aligned} \quad (2.19)$$

In general, we have

Theorem 2.4

$$\begin{aligned} f[x_0, x_1, \dots, x_k] \\ = \frac{1}{k! h^k} \Delta^k f_0. \end{aligned} \quad (2.20)$$

Proof is by induction on k .

For $k = 0$, the result is trivially true. We have also proved the result for $k = 1$, and $k = 2$. Assume now that the result is true $k = m$. Then we need to show that the result is also true for $k = m + 1$.

$$\text{Now, } f[x_0, \dots, x_{m+1}] = \frac{f[x_1, \dots, x_{m+1}] - f[x_0, \dots, x_m]}{x_{m+1} - x_0} = \frac{\left(\frac{\Delta^m f_1}{m!h^m} - \frac{\Delta^m f_0}{m!h^m}\right)}{(m+1)h} = \frac{\Delta^m(f_1 - f_0)}{m!(m+1)h^{m+1}} =$$

$$\frac{\Delta^{m+1} f_0}{(m+1)!h^{m+1}}$$

The numbers $\Delta^k f_i$ are called k^{th} **order forward differences** of f at $x = i$.

We now show how the interpolating polynomial $P_n(x)$ given by (2.16) can be written using forward differences.

$$\begin{aligned} P_n(x) = P_n(x_0 + sh) &= f[x_0] + shf[x_0, x_1] + s(s-1)h^2f[x_0, x_1, x_2] \\ &+ \dots + s(s-1)\dots(s-n+1)h^n f[x_0, x_1, \dots, x_n] \\ &= \sum_{k=0}^n s(s-1)\dots(s-k+1)h^k f[x_0, x_1, \dots, x_k] \\ &= \sum_{k=0}^n \binom{s}{k} k!h^k f[x_0, x_1, \dots, x_k] = \sum_{k=0}^n \binom{s}{k} \Delta^k f_0 \text{ using (2.20)}. \end{aligned}$$

Newton's Forward-Difference Interpolation Formula

Let x_0, x_1, \dots, x_n be $(n+1)$ equidistant points with distance h ; that is $x_{i+1} - x_i = h$. Then Newton's interpolating polynomial $P_n(x)$ of degree at most n , using forward-differences $\Delta^k f_0$, $k = 0, 1, 2, \dots, n$ is given by

$$P_n(x) = \sum_{k=0}^n \binom{s}{k} \Delta^k f_0,$$

where $s = \frac{x - x_0}{h}$.

Illustration: The Forward Difference Table with $n = 4$

x	$f(x)$	Δf	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$
x_0	f_0				
		Δf_0			
x_1	f_1		$\Delta^2 f_0$		
		Δf_1		$\Delta^3 f_0$	
x_2	f_2		$\Delta^2 f_1$		$\Delta^4 f_0$
		Δf_2		$\Delta^3 f_1$	
x_3	f_3		$\Delta^2 f_2$		
		Δf_3			
x_4	f_4				

Example 2.7 Let $f(x) = e^x$.

x	f	Δf	$\Delta^2 f$	$\Delta^3 f$
0	1			
		1.7183		
1	2.7183		2.9525	
		4.6708		5.0731
2	7.3891		8.0256	8.7176
		12.6964		13.7907
3	20.0855		21.8163	
		34.5127		
4	54.5982			

Let $x = 1.5$

$$\text{Then } s = \frac{x - x_0}{h} = \frac{1.5 - 0}{1} = 1.5$$

$$P_4(1.5) = \binom{1.5}{0} 1 + \binom{1.5}{1} 1.7183 + \binom{1.5}{2} 2.9525 + \binom{1.5}{3} 5.0731 + \binom{1.5}{4} 8.7176 = 1 + 1.5(1.7183) + 0.3750(2.9525) + (-0.0625)(5.0731) + (0.0234)(8.7176) = 4.5716$$

The correct answer up to 4 decimal digits is 4.4817.

Interpolation using Newton-Backward Differences

While interpolating at some value of x near the end of the difference table, it is logical to reorder the nodes so that the end-differences can be used in computation. The backward differences allow us to do so.

The backward differences are defined by

$$\nabla f_n = f_n - f_{n-1}, n = 1, 2, 3, \dots,$$

and

$$\nabla^k f_n = \nabla(\nabla^{k-1} f_n), k = 2, 3, \dots$$

Thus, $\nabla^2 f_n = \nabla(\nabla f_n) = \nabla(f_n - f_{n-1})$
 $= f_n - f_{n-1} - (f_{n-1} - f_{n-2})$
 $= f_n - 2f_{n-1} + f_{n-2},$

and so on.

The following a relationship between the backward-differences and the divided differences can be obtained.

$$f[x_{n-k}, \dots, x_{n-1}, x_n] = \frac{1}{k!h^k} \nabla^k f_n.$$

Using these backward-differences, we can write the Newton interpolation formula as:

$$P_n(x) = f_n + s\nabla f_n + \frac{s(s+1)}{2!} \nabla^2 f_n + \dots + \frac{s(s+1)\dots(s+n-1)}{n!} \nabla^n f_n.$$

Again, using the notation $\binom{-s}{k} = \frac{(-s)(-s-1)\dots(-s-k+1)}{k!}$ we can rewrite the above formula as:

Newton's Backward-Difference Interpolations Formula

$$P_n(x) = \sum_{k=0}^n (-1)^k \binom{-s}{k} \nabla^k f_n.$$

2.7 Spline-Interpolation

So far we have been considering interpolation by means of a single polynomial in the entire range.

Let's now consider interpolation using different polynomials (but of the same degree) at different intervals. Let the function $f(x)$ be defined at the nodes $a = x_0, x_1, x_2, \dots, x_n = b$. The problem now is to construct piecewise polynomials $S_j(x)$ on each interval $[x_j, x_{j+1}]$, $j = 0, 1, 2, \dots, n-1$, so that the resulting function $S(x)$ is an interpolant for the function $f(x)$.

The simplest such polynomials are of course, linear polynomials (straight lines). The interpolating polynomial in this case is called **linear spline**. The two biggest disadvantages of a linear spline are (i) the *convergence is rather slow*, and (ii) *not suitable for applications demanding smooth approximations*, since these splines have corner at the knots.

“Just imagine a cross section of an airplane wing in the form of a linear spline and you quickly decide to go by rail. (Stewart (1998), p. 93).”

Likewise the quadratic splines have also certain disadvantages.

The most common and widely used splines are **cubic splines**. Assume that the cubic polynomial $S_j(x)$, has the following form:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \quad j = 0, 1, \dots, n - 1.$$

Since $S_j(x)$ contains four unknowns, to construct n cubic polynomials $S_0(x), S_1(x), \dots, S_{n-1}(x)$, we need $4n$ conditions. To have these $4n$ conditions, a cubic spline $S(x)$ for the function $f(x)$ can be conveniently defined as follows:

Cubic Spline Interpolant

A function $S(x)$, denoted by $S_j(x)$, over the interval $[x_j, x_{j+1}]$, $j = 0, 1, \dots, n - 1$ is called a cubic spline interpolant if the following conditions hold:

- (i) $S_j(x_j) = f_j, \quad j = 0, 1, 2, \dots, n.$
- (ii) $S_{j+1}(x_{j+1}) = S_j(x_{j+1}), \quad j = 0, 1, 2, \dots, n - 2.$
- (iii) $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1}), \quad j = 0, 1, 2, \dots, n - 2.$
- (iv) $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1}), \quad j = 0, 1, 2, \dots, n - 2.$

Then, since for j there are four unknowns: a_j, b_j, c_j , and d_j and there are n such polynomials to be determined, all together there are $4n$ unknowns. However, conditions (i)-(iv) above give only $(4n - 2)$ equations: (i) gives $n + 1$, and each of (ii)-(iv) gives $n - 1$. So, *to completely determine the cubic spline, we must need two more equations*. To obtain these two additional equations, we can invoke *boundary conditions*.

- If the second derivative of $S(x)$ can be approximated, than we can take: $S''(x_0) = S''(x_n) = 0$ (**free or natural boundary**).
- On the other hand, if only the first derivative can be estimated, we can use the following boundary condition: $S'(x_0) = f'(x_0)$, and $S'(x_n) = f'(x_n)$ (**clamped boundary**).

We will discuss only the **clamped cubic spline** here.

From the condition (i), it immediately follows that

$$a_j = f_j, \quad j = 0, 1, 2, \dots, n - 1.$$

Set $h_j = x_{j+1} - x_j$. With some mathematical manipulations, it can then be shown (see for e.g., Burden and Faires, pp. 144-145), that once a_j 's are known, the coefficients c_j , $j = 0, 1, 2, \dots, n$ can be found by solving the following linear algebraic system of equations:

$$Ax = r,$$

where

A =

$$\begin{bmatrix} 2h_0 & h_0 & 0 & & & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \ddots & & \\ 0 & h_1 & 2(h_1 + h_2) & h_2 & \ddots & \\ & \ddots & \ddots & \ddots & \ddots & 0 \\ & & \ddots & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & & & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix}$$

$$x = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix}, r = \begin{pmatrix} \frac{3}{h_0}(a_1 - a_0) - 3f'(a) \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{pmatrix}.$$

Once c_0, c_1, \dots, c_n are known by solving the above system of equations, the quantities b_j and d_j can be computed as follows:

$$b_j = \frac{(a_{j+1} - a_j)}{h_j} - \frac{h_j}{3}(c_{j+1} + 2c_j), j = n - 1, n - 2, \dots, 0$$

$$d_j = \frac{c_{j+1} - c_j}{3h_j}, j = n - 1, n - 2, \dots, 0.$$

Definition: A matrix $A = (a_{ij})$ is **strictly column diagonally dominant** if

$$\begin{aligned} |a_{11}| &> |a_{21}| + |a_{31}| + \dots + |a_{n1}| \\ |a_{22}| &> |a_{12}| + |a_{32}| + \dots + |a_{n2}| \\ &\vdots \\ |a_{nn}| &> |a_{1n}| + |a_{2n}| + \dots + |a_{n-1,n}| \end{aligned}$$

A **row diagonally dominant** matrix can be similarly defined.

An important property of a strictly column diagonally dominant matrix is:

Solving $Ax = b$ with A Strictly Column Diagonally Dominant.

If A is strictly column diagonally dominant, then

- Solution of $Ax = b$ using Gaussian elimination does not need pivoting
- Gaussian elimination process is numerically stable.

For details, see “*Numerical Linear Algebra and Applications*” by B. N. Datta, Brooks/Cole Publishing Co., California, 1995 (*Second edition is to be published by SIAM in 2009*).

Algorithm 2.2 Computing Clamped Cubic Spline

Inputs:

(i) The nodes x_0, x_1, \dots, x_n .

(ii) The functional values:

$$f(x_i) = f_i, \quad i = 0, 1, \dots, n.$$

(Note that $a = x_0$ and $b = x_n$).

(iii) $f'(x_0) = f'(a)$ and $f'(x_n) = f'(b)$.

Outputs: The coefficients a_0, \dots, a_n ; b_0, \dots, b_n ; c_0, c_1, \dots, c_n , and d_0, d_1, \dots, d_n of the n polynomials $S_0(x), S_1(x), \dots, S_{n-1}(x)$ of which the cubic interpolant $S(x)$ is composed.

Step 1. For $i = 0, 1, \dots, n - 1$ do

Set $h_j = x_{j+1} - x_j$

Step 2. Compute a_0, a_1, \dots, a_n :

For $i = 0, 1, \dots, n - 1$ do

$$a_i = f(x_i), \quad i = 0, 1, \dots, n.$$

End

Step 3. Compute the coefficients c_0, c_1, \dots, c_n by solving the system $Ax = r$, where A , x , and r are as defined above.

Step 4. Compute the coefficients b_0, \dots, b_n and d_0, d_1, \dots, d_n as given in the box above.

Exercises

1. Prove that the Vandermonde matrix of system 2.1 is nonsingular if x_k , $k = 0, 1, \dots, n$ are all distinct.

2. Prove by induction that

$$\frac{d^{n+1}}{dt^{n+1}} \frac{\prod_{i=0}^n (t - x_i)}{\prod_{i=0}^n (\bar{x} - x_i)} = \frac{(n+1)!}{\prod_{i=0}^n (\bar{x} - x_i)}$$

3. Consider the following table

x	$f(x)$
1	0
1.01	0.01
1.02	0.0198
1.03	0.0296

- (a) Find an approximation of $f(1.025)$ using
- i. The associated Vandermonde linear system
 - ii. Lagrangian interpolating polynomial of degree 3
 - iii. Newton interpolating polynomial of degree 3
- (b) Compute the maximum error in interpolation using the interpolating error formula.
- (c) Compare the error made in each case (i)-(iii) with the maximum error.
4. Suppose that a table is to be prepared for the function $f(x) = \ln x$ on $[1, 3.5]$ with equal spacing nodes such that the interpolation with third degree polynomial will give an accuracy of $\epsilon = 5 \times 10^{-8}$. Determine how small h has to be to guarantee the above accuracy.

5. Prove by induction that

(a) $f[x_0, x_1, \dots, x_k] = \frac{\Delta^k f_0}{k!h^k}$.

(b) $a_k = f[x_0, x_1, \dots, x_k]$, where a_k 's are the coefficients of the Newton interpolating polynomial.

6. (a) Find an approximate value of $\log_{10}(5)$ using Newton's forward difference formula with $x_0 = 1$, $x_1 = 2.5$, $x_2 = 4$, and $x_3 = 5.5$.

(b) Repeat part (a) using Newton's backward differences.

(c) Compare the results.

7. Using the data of problem 2, estimate $f(1.025)$ with linear and quadratic splines and compare your results with those obtained there.

8. Show that $f[x_0, x_1, \dots, x_n]$ does not change if the nodes are rearranged.

9. Let $\psi_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$. Let $x_{i+1} - x_i = h, i = 0, 1, \dots, n - 1$. Then prove that

$$(a) \max_{x_0 \leq x \leq x_1} |\psi_1(x)| = \frac{h^2}{4}$$

$$(b) \max_{x_0 \leq x \leq x_2} |\psi_2(x)| = \frac{2\sqrt{3}}{9}h^3$$

[**Hint:** To bound $|\psi_2(x)|$, shift this polynomial along the x -axis: $\hat{\psi}_2(x) = (x + h)x(x - h)$ and obtain the bound of $\hat{\psi}_2(x)$].

$$(c) \max_{x_0 \leq x \leq x_3} |\psi_3(x)| = h^4$$

$$\max_{x_1 \leq x \leq x_2} |\psi_3(x)| = \frac{9}{16}h^4.$$

(d) Using the results of (a), (b), and (c), obtain the maximum absolute error in each case for the functions:

$$f(x) = e^x, \quad 0 \leq x \leq 2$$

$$f(x) = \sin x \quad 0 \leq x \leq \frac{\Pi}{2}$$

(e) The interpretation of the result of Part (c) is that if the interpolation nodes are chosen so that the interpolation point x is chosen close to the midpoint of $[x_0, x_3]$, then the interpolation error is smaller than if n were chosen anywhere between x_0 and x_3 . Verify this statement with the data of Problem 2.