1.2 Efficient Computation of Interpolation Polynomials

Recall the Lagrange interpolation polynomial of a function f, at the distinct nodes $(x_i, f_i), f_i = f(x_i),\ i = \overline{0,n}$:

$$L_n f(x) \;=\; \sum_{i=0}^{n} l_i(x) f_i, \tag{1.1}$$

where

$$\begin{aligned} l_i(x) &= \prod_{\substack{j=0\\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j} \;=\; \frac{u_i(x)}{u_i(x_i)} \;=\; \frac{u_i(x)}{u'(x_i)}, \\ u(x) &= \prod_{j=0}^{n}(x-x_j),\;\; u_j(x) \;=\; \frac{u(x)}{x-x_j},\; j=0,1,\dots,n. \end{aligned}$$

This formula is well-suited for many theoretical uses of interpolation, but it is less desirable for practical computations. Among its shortcomings: each evaluation of $L_n f(x)$ requires $O(n^2)$ flops (additions and multiplications); adding a new node (x_{n+1}, f_{n+1}) requires a new computation from scratch and knowing $L_n f(x)$ does not lead to a less expensive way to evaluate $L_{n+1} f(x)$. For these reasons, we need alternative and more easily computable formulations and expressions for interpolation polynomials.

1.2.1 Barycentric interpolation

The Lagrange formula (1.1) can be rewritten in such a way that it can be evaluated and updated in $O(n)$ flops.

$$L_n f(x) \;=\; \sum_{i=0}^{n} \frac{u_i(x)}{u'(x_i)} f_i \;=\sum_{i=0}^{n} \frac{u(x)}{(x-x_i)u'(x_i)} f_i \;=\; u(x)\sum_{i=0}^{n} \frac{\dfrac{1}{u'(x_i)}}{x-x_i} f_i.$$

Let

$$w_i \;=\; \frac{1}{u'(x_i)} \;=\; \frac{1}{\displaystyle\prod_{\substack{j=0\\ j\neq i}}^{n}(x_i-x_j)},\; i=0,1,\dots,n. \tag{1.2}$$

These are called **barycentric weights**. With these, the Lagrange interpolation polynomial can be written as

$$L_n f(x) = u(x) \sum_{i=0}^{n} \frac{w_i}{x - x_i} f_i. \tag{1.3}$$

Formula (1.3) is called the *first barycentric formula*.

Now, Lagrange interpolation is a formula requiring $O(n^2)$ operations for calculating some quantities independent of x, the weights w_i, followed by $O(n)$ flops for evaluating $L_n f(x)$, once these numbers are known. Incorporating a new node x_{n+1} entails two calculations:
- dividing each $w_i, i = 0, \ldots, n$ by $x_i - x_{n+1}$, for a cost of $2n + 2$ flops,
- computing a new weight w_{n+1} using formula (1.2), for another $n + 1$ flops.

Formula (1.3) can be improved even further. Notice that for the constant function $f \equiv 1$, the Lagrange polynomial is f itself

$$L_n f \equiv 1,$$

by the uniqueness of the interpolation polynomial. Substituting in (1.3), we find

$$1 = u(x) \sum_{i=0}^{n} \frac{w_i}{x - x_i}$$

and further

$$u(x) = \frac{1}{\displaystyle\sum_{i=0}^{n} \frac{w_i}{x - x_i}}.$$

Then the Lagrange polynomial can be written as

$$L_n f(x) = \frac{\displaystyle\sum_{i=0}^{n} \frac{w_i}{x - x_i} f_i}{\displaystyle\sum_{i=0}^{n} \frac{w_i}{x - x_i}}, \tag{1.4}$$

called the *second barycentric formula*.

1.2.2 Newton-type methods

The next procedures gives an alternative form for the interpolation polynomial, as well as for the remainder.

Newton's divided difference formula

To better understand (and write) the transition from n to $n+1$ nodes, we slightly change the notations. For the monic polynomial of the nodes ("monic" means the leading coefficient is 1), previously denoted by "$u(x)$", we introduce a new notation, one that also emphasizes the *number of nodes* that it refers to. So, let

$$\begin{aligned} \psi_n(x) &= (x-x_0)\dots(x-x_{n-1})(x-x_n), \\ \psi_{n-1}(x) &= (x-x_0)\dots(x-x_{n-1}). \end{aligned}$$

Then we have

$$\begin{aligned} \psi_n(x) &= (x-x_n)\psi_{n-1}(x), \\ \psi_n'(x) &= \psi_{n-1}(x) + (x-x_n)\psi_{n-1}'(x). \end{aligned}$$

Hence,

$$\begin{aligned} \psi_n'(x_i) &= (x_i-x_n)\psi_{n-1}'(x_i),\ i=0,\dots,n-1, \\ \psi_n'(x_n) &= \psi_{n-1}(x_n). \end{aligned} \tag{1.5}$$

With these new notations, we can write

$$\begin{aligned} L_{n-1}f(x) &= \sum_{i=0}^{n-1} \frac{\psi_{n-1}(x)}{(x-x_i)\psi_{n-1}'(x_i)} f_i, \\ L_n f(x) &= \sum_{i=0}^{n} \frac{\psi_n(x)}{(x-x_i)\psi_n'(x_i)} f_i. \end{aligned} \tag{1.6}$$

Let us also recall one of the properties of divided differences (Theorem 4.7 a), in Lecture 2):

$$f[x_0,\dots,x_n] = \sum_{i=0}^{n} \frac{1}{\psi_n'(x_i)} f_i. \tag{1.7}$$

We want to derive a simple recursive formula from $L_{n-1}f$ to $L_n f$, when adding a new node x_n. Let

$$Q(x) \;=\; L_n f(x) - L_{n-1} f(x). \tag{1.8}$$

Obviously, Q is a polynomial of degree n and $Q(x_i) = 0, i = 0, \ldots, n-1$, so its n roots are precisely the nodes $x_0, \ldots, x_{n-1}$. Then Q is of the form

$$\begin{aligned} Q(x) &= a_n(x - x_0)\ldots(x - x_{n-1}) \;=\; a_n\psi_{n-1}(x), \\ Q(x_n) &= a_n\psi_{n-1}(x_n), \end{aligned} \tag{1.9}$$

for some constant $a_n \in \mathbb{R}$ that we want to find.

On the other hand, the polynomial $L_n f$ also interpolates f at the node x_n, so $L_n f(x_n) = f(x_n) = f_n$ and, thus,

$$Q(x_n) \;=\; f_n - L_{n-1} f(x_n). \tag{1.10}$$

By (1.9)–(1.10), it follows that

$$Q(x) \;=\; \frac{f_n - L_{n-1}f(x_n)}{\psi_{n-1}(x_n)}.$$

Now using (1.5)–(1.7), we get:

$$\begin{aligned} a_n &= \frac{f_n}{\psi_{n-1}(x_n)} - \frac{1}{\psi_{n-1}(x_n)} \sum_{i=0}^{n-1} \frac{\psi_{n-1}(x_n)}{(x_n - x_i)\psi'_{n-1}(x_i)} f_i \\ &= \frac{f_n}{\psi_{n-1}(x_n)} + \sum_{i=0}^{n-1} \frac{f_i}{(x_i - x_n)\psi'_{n-1}(x_i)} \\ &= \frac{f_n}{\psi'_n(x_n)} + \sum_{i=0}^{n-1} \frac{f_i}{\psi'_n(x_i)} \\ &= \sum_{i=0}^{n} \frac{f_i}{\psi'_n(x_i)} \;=\; f[x_0, \ldots, x_n]. \end{aligned}$$

Thus,

$$Q(x) \;=\; f[x_0, \ldots, x_n]\psi_{n-1}(x).$$

So, by (1.8)–(1.10), we have the following recurrence relation for the Lagrange polynomial:

$$L_nf(x) = L_{n-1}f(x) + f[x_0,\dots,x_n]\psi_{n-1}(x),\ n \ge 1. \tag{1.11}$$

Iteratively, we get

$$\begin{aligned}
L_0f(x) &= f(x_0),\\
L_1f(x) &= f(x_0) + f[x_0,x_1](x-x_0),\\
L_2f(x) &= f(x_0) + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1),\\
\dots&\\
L_nf(x) &= f(x_0) + f[x_0,x_1](x-x_0) + \dots + f[x_0,\dots,x_n](x-x_0)\dots(x-x_{n-1}),
\end{aligned} \tag{1.12}$$

The expression on the right-hand-side of (1.12) is called **Newton's divided difference form** of the interpolation polynomial, or **Newton's interpolation polynomial** and it is denoted by $N_nf(x)$. To be clear, by the uniqueness of the interpolation polynomial at $n+1$ distinct nodes, the two polynomials *coincide*, $L_nf(x) = N_nf(x)$, they are just expressed (written) in different forms.

If we denote by

$$D_i = f[x_0,\dots,x_i],\ i \ge 0,$$

Newton's polynomial N_nf can be written in the *nested form*

$$\begin{aligned}
N_nf(x) &= D_0 + (x-x_0)D_1 + (x-x_0)(x-x_1)D_2 + \dots + (x-x_0)\dots(x-x_{n-1})D_n\\
&= D_0 + (x-x_0)\Big[D_1 + (x-x_1)\Big[D_2 + \dots\\
&+ (x-x_{n-2})\Big[D_{n-1} + (x-x_{n-1})D_n\Big]\dots\Big]\Big].
\end{aligned} \tag{1.13}$$

Writing it this way, we can see that the evaluation of $N_nf(x)$ requires only n multiplications and n additions (once the divided differences have been computed), so this is a more computationally efficient formula for the interpolation polynomial.

Next, we also want to express the remainder in a new form. Let $[a,b]$ denote the smallest interval containing the distinct nodes $\{x_0,\dots,x_n\}$ and let $x \in [a,b]$ be fixed. We write recursively:

$$\begin{aligned}
f\,[x, x_0] &= \frac{f(x) - f(x_0)}{x - x_0} \\
f\,[x, x_0, x_1] &= \frac{f\,[x, x_0] - f\,[x_0, x_1]}{x - x_1} \\
f\,[x, x_0, x_1, x_2] &= \frac{f\,[x, x_0, x_1] - f\,[x_0, x_1, x_2]}{x - x_2} \qquad (1.14)\\
\dots \quad & \quad \dots \\
f\,[x, x_0, \dots, x_{n-1}] &= \frac{f\,[x, x_0, \dots, x_{n-2}] - f\,[x_0, x_1, \dots, x_{n-1}]}{x - x_{n-1}} \\
f\,[x, x_0, x_1, \dots, x_n] &= \frac{f\,[x, x_0, \dots, x_{n-1}] - f\,[x_0, x_1, \dots, x_n]}{x - x_n}.
\end{aligned}$$

Multiplying the first equation in (1.14) by $(x - x_0)$, the second by $(x - x_0)(x - x_1)$, the third by $(x - x_0)(x - x_1)(x - x_2)$, ..., the next to last by $(x - x_0)(x - x_1) \dots (x - x_{n-1})$ and the last one by $(x - x_0)(x - x_1) \dots (x - x_n)$ and adding them term by term, we obtain

$$\begin{aligned}
f(x) &= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \dots \\
&+ f[x_0, \dots, x_n](x - x_0) \dots (x - x_{n-1}) + f[x, x_0, \dots, x_n]\psi_n(x) \\
&= N_n f(x) + f[x, x_0, \dots, x_n]\psi_n(x),
\end{aligned}$$

from which we have

$$R_n f(x) = f[x, x_0, \dots, x_n](x - x_0) \dots (x - x_n). \qquad (1.15)$$

By the mean value formula for divided differences (Theorem 4.7 e), in Lecture 2), we find the previous formula for the remainder:

$$R_n f(x) = \frac{(x - x_0) \dots (x - x_n)}{(n+1)!} f^{(n+1)}(\xi),\ \xi \in (a, b).$$

Example 1.1. Given the data below, find $N_1(0.15)$ and $N_2(0.15)$, the linear and quadratic interpolates evaluated at $x = 0.15$.

n	x_n	$f(x_n)$
0	0.1	0.2
1	0.2	0.24
2	0.3	0.3

Solution First, we compute the divided differences:

$$\begin{tikzcd}
x_0 = 0.1 \quad f[x_0] = 0.2 \arrow[r] & f[x_0, x_1] = \dfrac{0.24 - 0.2}{0.2 - 0.1} = 0.4 \arrow[r] & f[x_0, x_1, x_2] = \dfrac{0.6 - 0.4}{0.3 - 0.1} = 1 \\
x_1 = 0.2 \quad f[x_1] = 0.24 \arrow[r] \arrow[ur] & f[x_1, x_2] = \dfrac{0.3 - 0.24}{0.3 - 0.2} = 0.6 \arrow[ur] & \\
x_2 = 0.3 \quad f[x_2] = 0.3 \arrow[ur] & &
\end{tikzcd}$$

The linear interpolate is then

$$N_1 f(x) \;=\; f(x_0) + f[x_0, x_1](x - x_0) \;=\; 0.2 + 0.4(x - 0.1) \;=\; 0.4x + 0.16,$$

so we have the approximation

$$f(0.15) \;\approx\; N_1(0.15) \;=\; 0.22.$$

Using all three nodes, we find the quadratic interpolate

$$\begin{aligned}
N_2 f(x) &= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\
&= 0.2 + 0.4(x - 0.1) + 1 \cdot (x - 0.1)(x - 0.2) \\
&= x^2 + 0.1x + 0.18,
\end{aligned}$$

which yields the approximation

$$f(0.15) \;\approx\; N_2(0.15) \;=\; 0.2175.$$

■

Newton's forward and backward difference formula

In the case where the interpolating nodes x_i are not equally spaced, we use Newton's divided difference formula presented above; however, when the nodes are equidistant, we can construct simpler and less expensive algorithms, using finite differences. Historically, these algorithms were of great importance in interpolating functions whose values were given in tables, but the availability of more powerful computers diminished their relevance. However, with new processors (fpu's), they have made a comeback.

Assume the values of a function f are known at the h-step equidistant nodes

$$x_i = x_0 + ih, i = 0, 1, \dots$$

Recall the forward differences of the function f

$$\begin{aligned} \Delta^1 f(x_i) &= f(x_i+h) - f(x_i) = f_{i+1} - f_i, \\ \Delta^k f(x_i) &= \Delta^{k-1} f(x_i+h) - \Delta^{k-1} f(x_i) = \Delta^{k-1} f_{i+1} - \Delta^{k-1} f_i \end{aligned}$$

and the property (Proposition 4.12 in Lecture 2)

$$f[x_0, x_0+h, \dots, x_0+ih] = \frac{1}{i!h^i}\Delta^i f_0.$$

The Newton form of the nth degree polynomial $L_n f$ of f at the nodes $x_i = x_0 + ih, i = 0, 1, \dots, n$, can be simplified. Denote by $s = (x - x_0)/h$. Then

$$\begin{aligned} (x-x_0)\dots(x-x_{i-1})f[x_0,\dots,x_0+ih] &= \big(sh\big)\cdot\big((s-1)h\big)\dots\big((s-i+1)h\big)\frac{1}{i!h^i}\Delta^i f_0 \\ &= \frac{s(s-1)\dots(s-i+1)}{i!}\Delta^i f_0. \end{aligned}$$

Using the notation

$$\binom{s}{k} = \frac{s(s-1)\cdots(s-k+1)}{k!}, \; s \in \mathbb{R}, k \in \mathbb{N}$$

(the generalized binomial coefficient), we find *Newton's forward difference formula.*

$$L_n f(x) = f_0 + \binom{s}{1}\Delta f_0 + \binom{s}{2}\Delta^2 f_0 + \dots + \binom{s}{n}\Delta^n f_0, \tag{1.16}$$

with $s = (x - x_0)/h$.

The error after n iterations, for $x = x_0 + sh$, is given by

$$f(x) - L_n f(x) = h^{n+1} \binom{s}{n+1} f^{(n+1)}(\xi_x), \tag{1.17}$$

where ξ_x lies in the smallest interval containing $x_0, \dots, x_n$ and x.

Similarly, using backward differences ∇

$$\begin{aligned} \nabla^0 f_i &= f_i, \\ \nabla^1 f_i &= f_i - f_{i-1}, \\ \nabla^k f_i &= \nabla^{k-1} f_i - \nabla^{k-1} f_{i-1} \end{aligned}$$

and the change of variables $s = (x - x_n)/h$, we obtain

$$L_n f(x) = f_n + \frac{s}{1!} \nabla f_n + \frac{s(s+1)}{2!} \nabla^2 f_n + \cdots + \frac{s(s+1)\dots(s+n-1)}{n!} \nabla^n f_n,$$

which can be written as

$$L_n f(x) = f_n + \binom{s}{1} \nabla f_n + \binom{s+1}{2} \nabla^2 f_n + \cdots + \binom{s+n-1}{n} \nabla^n f_n \tag{1.18}$$

This is called *Newton's backward difference formula*.

In this case, the interpolation error is

$$f(x) - L_n f(x) = h^{n+1} \binom{s+n}{n+1} f^{(n+1)}(\xi_x), \tag{1.19}$$

where ξ_x lies in the smallest interval containing $x_0, \dots, x_n$ and x.

Example 1.2. Consider again the data in Example 1.1.

n	x_n	f_n
0	0.1	0.2
1	0.2	0.24
2	0.3	0.3

Let us find $L_2 f(0.15)$ using finite differences.

Solution By (1.16), we have

$$\begin{aligned}
L_2 f(x) &= f_0 + \binom{s}{1}\Delta f_0 + \binom{s}{2}\Delta^2 f_0 \\
&= f_0 + \frac{s}{1!}\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 \\
&= f_0 + \frac{x-x_0}{h}\Delta f_0 + \frac{(x-x_0)(x-x_0-h)}{2h^2}\Delta^2 f_0,
\end{aligned}$$

where $s = (x - x_0)/h,\ h = 0.1.$
We compute the forward differences:

$$\begin{tikzcd}
x_0 = 0.1 & f_0 = 0.2 \arrow[r] & \Delta f_0 = 0.24 - 0.2 = 0.04 \arrow[r] & \Delta^2 f_0 = 0.06 - 0.04 = 0.02 \\
x_1 = 0.2 & f_1 = 0.24 \arrow[r] \arrow[ur] & \Delta f_1 = 0.3 - 0.24 = 0.06 \arrow[ur] & \\
x_2 = 0.3 & f_2 = 0.3 \arrow[ur] & &
\end{tikzcd}$$

So,

$$\begin{aligned}
L_2 f(x) &= 0.2 + \frac{x-0.1}{0.1}\cdot 0.04 + \frac{(x-0.1)(x-0.2)}{0.02}\cdot 0.02 \\
&= x^2 + 0.1x + 0.18
\end{aligned}$$

and

$$L_2 f(0.15) = 0.2175.$$

Using backward differences, by (1.18), we get

$$\begin{aligned}
L_2 f(x) &= f_2 + \binom{s}{1}\nabla f_2 + \binom{s+1}{2}\nabla^2 f_2 \\
&= f_2 + \frac{s}{1!}\nabla f_2 + \frac{(s+1)s}{2!}\nabla^2 f_2 \\
&= f_2 + \frac{x-x_2}{h}\nabla f_2 + \frac{(x-x_2+h)(x-x_2)}{2h^2}\nabla^2 f_2
\end{aligned}$$

with $s = (x - x_2)/h,\ h = 0.1.$

The backward differences are found in the table

$$\begin{tikzcd}
x_0 = 0.1 & f_0 = 0.2 \arrow[dr] & & \\
x_1 = 0.2 & f_1 = 0.24 \arrow[r] \arrow[dr] & \nabla f_1 = 0.24 - 0.2 = 0.04 \arrow[dr] & \\
x_2 = 0.3 & f_2 = 0.3 \arrow[r] & \nabla f_2 = 0.3 - 0.24 = 0.06 \arrow[r] & \nabla^2 f_2 = 0.06 - 0.04 = 0.02
\end{tikzcd}$$

Hence,

$$\begin{aligned}
L_2 f(x) &= 0.3 + \frac{x - 0.3}{0.1} \cdot 0.06 + \frac{(x-0.2)(x-0.3)}{0.02} \cdot 0.02 \\
&= x^2 + 0.1x + 0.18
\end{aligned}$$

and

$$L_2 f(0.15) = 0.2175.$$

■

Remark 1.3. Interpolation algorithms can be classified according to the “step” of the grid (the distance between two consecutive nodes, when sorted in increasing order). There are *variable step* methods (the Lagrange form with fundamental polynomials, the barycentric formulas, Newton’s divided difference formula) and *constant step* algorithms (Newton’s forward and backward formulas). For variable step methods the precision is the same at any intermediate value in the interval covered by the data (x_i, f_i). So these methods do not have so-called *preferential precision zones*. In contrast, Newton’s forward formula is particularly useful (i.e. it has higher precision) for interpolating the values of $f(x)$ near the beginning of the set of values (closer to the first node, (x_0, f_0)), whereas the backward formula is preferred when the value of $f(x)$ is required near the end of the table (in the vicinity of the last node, (x_n, f_n)).

1.2.3 Aitken-type methods

These are variable step iterative methods and they highlight another important aspect: in many cases, the degree required to attain a certain desired accuracy in polynomial interpolation is *not known*. It can be obtained from the remainder, but that assumes knowledge (or at least knowing a bound) of $||f^{(n+1)}||_\infty$.

The idea behind these methods is to write an interpolation polynomial of degree n, iteratively, in terms of two interpolation polynomials of degree $n-1$, that only use a part of the $n+1$ nodes. Let us illustrate the idea for a simple case. For two nodes, x_0 and x_1, the polynomial of degree 1 interpolating these data, can be written successively (using Lagrange basis polynomials) as

$$\begin{aligned} P_{01}(x) &= l_0(x)f_0 + l_1(x)f_1 \\ &= \frac{x-x_1}{x_0-x_1}f_0 + \frac{x-x_0}{x_1-x_0}f_1 \\ &= \frac{(x-x_0)f_1-(x-x_1)f_0}{x_1-x_0} \\ &= \frac{(x-x_0)P_1(x)-(x-x_1)P_0(x)}{x_1-x_0}, \end{aligned}$$

where P_0 denotes the polynomial that interpolates f at the node x_0 (a polynomial of degree 0, hence, a constant, f_0), P_1, the polynomial of degree 0 that interpolates f at the node x_1 (identically equal to f_1), and P_{01} the polynomial of degree 1 that interpolates f at the nodes x_0, x_1. Similarly, if we add another node, x_2, we can define

$$P_{12}(x) = \frac{(x-x_1)P_2(x)-(x-x_2)P_1(x)}{x_2-x_1},$$

which is the polynomial of degree 1 that interpolates f at the nodes x_1, x_2. We proceed further and define

$$P_{012}(x) = \frac{(x-x_0)P_{12}(x)-(x-x_2)P_{01}(x)}{x_2-x_0}. \tag{1.20}$$

Let us compute its values at the nodes.

$$\begin{aligned} P_{012}(x_0) &= \frac{0-(x_0-x_2)P_{01}(x_0)}{x_2-x_0} = P_{01}(x_0) = f_0, \\ P_{012}(x_1) &= \frac{(x_1-x_0)P_{12}(x_1)-(x_1-x_2)P_{01}(x_1)}{x_2-x_0} = \frac{(x_1-x_0)f_1-(x_1-x_2)f_1}{x_2-x_0} = f_1, \\ P_{012}(x_2) &= \frac{(x_2-x_0)P_{12}(x_2)-0}{x_2-x_0} = P_{12}(x_2) = f_2. \end{aligned}$$

Since P_{012} is a polynomial of degree 2, by the uniqueness of the Lagrange interpolation polynomial, it follows that $P_{012} = L_2 f$.

In a similar fashion, we can construct recursively the polynomials

$$\begin{aligned} P_{123}(x) &= \frac{(x-x_1)P_{23}(x)-(x-x_3)P_{12}(x)}{x_3-x_1}, \\ P_{0123}(x) &= \frac{(x-x_0)P_{123}(x)-(x-x_3)P_{012}(x)}{x_3-x_0} \\ \dots \end{aligned}$$

Proposition 1.4. *Let $x_0,\dots,x_k$ be distinct nodes and let $f_i, i=0,\dots,k$, be the values of a function f at the nodes. Then the Lagrange polynomial interpolating f at these nodes is given by*

$$\begin{aligned} P_{01\dots k}(x) &= \frac{1}{x_k-x_0}\begin{vmatrix} x-x_0 & P_{01\dots k-1}(x) \\ x-x_k & P_{12\dots k}(x) \end{vmatrix} \\ &= \frac{(x-x_0)P_{12\dots k}(x)-(x-x_k)P_{01\dots k-1}(x)}{x_k-x_0}. \end{aligned} \tag{1.21}$$

Proof. Obviously, by its construction, the polynomial in (1.21) has degree k. Its values at the nodes are

$$\begin{aligned} P_{01\dots k}(x_0) &= \frac{-(x_0-x_k)P_{01\dots k-1}(x_0)}{x_k-x_0} = P_{01\dots k-1}(x_0) = f_0, \\ P_{01\dots k}(x_j) &= \frac{(x_j-x_0)P_{12\dots k}(x_j)-(x_j-x_k)P_{01\dots k-1}(x_j)}{x_k-x_0} = f_j,\ j=\overline{1,k-1}, \\ P_{01\dots k}(x_k) &= \frac{(x_k-x_0)P_{12\dots k}(x_k)}{x_k-x_0} = P_{12\dots k}(x_k) = f_k. \end{aligned}$$

Hence, by the uniqueness of the Lagrange interpolation polynomial, it follows that $P_{01\dots k} = L_k f$. □

Thus, we established a recurrence relation between a Lagrange interpolation polynomial of degree k and two Lagrange interpolation polynomials of degree $k-1$. The computations can be organized in a table, illustrated below for 4 nodes.

$$\begin{array}{lllll} x_0 & P_0 & & & \\ x_1 & P_1 & P_{01} & & \\ x_2 & P_2 & P_{12} & P_{012} & \\ x_3 & P_3 & P_{23} & P_{123} & P_{0123} \end{array}$$

Now, if, for instance, P_{0123} does not provide a desired approximation precision, we can consider a new node and add a new line to the table:

$$x_4 \quad P_4 \quad P_{34} \quad P_{234} \quad P_{1234} \quad P_{01234}$$

and we can compare neighboring elements on a row, column or diagonal to check if the desired accuracy has been achieved.

The method described above is called *Neville's method.*

The notations can be simplified. We denote now the polynomials above by $\tilde{P}$ (instead of P) and define the new polynomials P as follows:

$$P_{i,j} \;=\; \tilde{P}_{i-j,i-j+1,\ldots,i-1,i},\; j=i,i-1,\ldots 0,$$

i.e., recursively,

$$\begin{aligned}
&P_{i,0} := f(x_i), \quad i=\overline{0,n},\\
&P_{i,j} := \frac{(x-x_{i-j})P_{i,j-1}-(x-x_i)P_{i-1,j-1}}{x_i-x_{i-j}}\\
&= \frac{1}{x_i-x_{i-j}}\begin{vmatrix} x-x_{i-j} & P_{i-1,j-1}\\ x-x_i & P_{i,j-1}\end{vmatrix}, \quad i\ge j>0.
\end{aligned} \tag{1.22}$$

We get a new table

$$\begin{array}{lllll}
x_0 & P_{00} & & & \\
x_1 & P_{10} & P_{11} & & \\
x_2 & P_{20} & P_{21} & P_{22} & \\
x_3 & P_{30} & P_{31} & P_{32} & P_{33}
\end{array}$$

and the Lagrange polynomial will be the one on the diagonal $L_n f = P_{nn}$.

If the interpolation converges, then the sequence $\{P_{ii}\}_{i\ge 0}$ also converges and we can use the stopping criterion

$$|P_{ii}-P_{i-1,i-1}| \;<\; \varepsilon.$$

Aitken's method is similar to Neville's method. We construct the table

$$\begin{array}{lllll}
x_0 & P_{00} & & & \\
x_1 & P_{10} & P_{11} & & \\
x_2 & P_{20} & P_{21} & P_{22} & \\
x_3 & P_{30} & P_{31} & P_{32} & P_{33}
\end{array}$$

defining recursively

$$
\begin{aligned}
P_{i,0} &:= f(x_i), \quad i = \overline{0,n}, \\
P_{i,j+1} &:= \frac{1}{x_i - x_j} \begin{vmatrix} x - x_j & P_{j,j} \\ x - x_i & P_{i,j} \end{vmatrix} = \frac{(x - x_j)P_{i,j} - (x - x_i)P_{j,j}}{x_i - x_j}, \quad i > j \geq 0.
\end{aligned} \tag{1.23}
$$

Example 1.5. Approximate $\sqrt{2}$ interpolating the function $f(x) = 2^x$ at the nodes $-1, 0, 1$, and then at the nodes $-1, 0, 1, 2$.

Solution
With Neville's method, we have the table

$$
\begin{array}{llll}
x_0 = -1 & P_{00} = 1/2 & & \\
x_1 = 0 & P_{10} = 1 & P_{11} = 5/4 & \\
x_2 = 1 & P_{20} = 2 & P_{21} = 3/2 & P_{22} = 23/16,
\end{array}
$$

where, for $x = 1/2$,

$$
\begin{aligned}
P_{11} &= \frac{(x - x_0)P_{10} - (x - x_1)P_{00}}{x_1 - x_0} = \frac{(1/2 - (-1)) \cdot 1 - (1/2 - 0) \cdot 1/2}{0 - (-1)} = 5/4, \\
P_{21} &= \frac{(x - x_1)P_{20} - (x - x_2)P_{10}}{x_2 - x_1} = \frac{(1/2 - 0) \cdot 2 - (1/2 - 1) \cdot 1}{1 - 0} = 3/2, \\
P_{22} &= \frac{(x - x_0)P_{21} - (x - x_2)P_{11}}{x_2 - x_0} = \frac{(1/2 - (-1)) \cdot 3/2 - (1/2 - 1) \cdot 5/4}{1 - (-1)} = 23/16.
\end{aligned}
$$

Thus, with linear interpolation, we get the approximation

$$\sqrt{2} \approx 23/16 = 1.4375$$

and

$$|P_{22} - P_{11}| = 3/16 = 0.1875.$$

We add a new node $x_3 = 2$ and a new line to the table, to get

$$
\begin{array}{lllll}
x_0 = -1 & P_{00} = 1/2 & & & \\
x_1 = 0 & P_{10} = 1 & P_{11} = 5/4 & & \\
x_2 = 1 & P_{20} = 2 & P_{21} = 3/2 & P_{22} = 23/16 & \\
x_3 = 2 & P_{30} = 4 & P_{31} = 1 & P_{32} = 11/8 & P_{33} = 45/32,
\end{array}
$$

with

$$\begin{aligned}
P_{31} &= \frac{(x-x_2)P_{30}-(x-x_3)P_{20}}{x_3-x_2} = \frac{(1/2-1)\cdot 4-(1/2-2)\cdot 2}{2-1} = 1,\\
P_{32} &= \frac{(x-x_1)P_{31}-(x-x_3)P_{21}}{x_3-x_1} = \frac{(1/2-0)\cdot 1-(1/2-2)\cdot 3/2}{2-0} = 11/8\\
P_{33} &= \frac{(x-x_0)P_{32}-(x-x_3)P_{22}}{x_3-x_0} = \frac{(1/2-(-1))\cdot 11/8-(1/2-2)\cdot 23/16}{2-(-1)} = 45/32.
\end{aligned}$$

The new approximation (using quadratic interpolation) is

$$\sqrt{2} \approx 45/32 = 1.4063, \text{ with } |P_{33}-P_{22}| = 1/32 = 0.0313.$$

Let us note that the exact value of $\sqrt{2}$ rounded to 4 correct decimals is 1.4142, so the actual errors of the two approximations are

$$|\sqrt{2}-P_{22}| = 0.0233 \text{ şi } |\sqrt{2}-P_{33}| = 0.0079.$$

With Aitken's algorithm, (1.23), we construct the table

$$\begin{array}{lllll}
x_0=-1 & P_{00}=1/2 & & & \\
x_1=0 & P_{10}=1 & P_{11}=5/4 & & \\
x_2=1 & P_{20}=2 & P_{21}=13/8 & P_{22}=23/16 & \\
x_3=2 & P_{30}=4 & P_{31}=9/4 & P_{32}=3/2 & P_{33}=45/32,
\end{array}$$

where

$$\begin{aligned}
P_{21} &= \frac{(x-x_0)P_{20}-(x-x_2)P_{00}}{x_2-x_0} = \frac{(1/2-(-1))\cdot 2-(1/2-1)\cdot 1/2}{1-0} = 13/8,\\
P_{22} &= \frac{(x-x_1)P_{21}-(x-x_2)P_{11}}{x_2-x_1} = \frac{(1/2-0)\cdot 13/8-(1/2-1)\cdot 5/4}{1-0} = 23/16\\
P_{31} &= \frac{(x-x_0)P_{30}-(x-x_3)P_{00}}{x_3-x_0} = \frac{(1/2-(-1))\cdot 4-(1/2-2)\cdot 1/2}{2-(-1)} = 9/4,\\
P_{32} &= \frac{(x-x_1)P_{31}-(x-x_3)P_{11}}{x_3-x_1} = \frac{(1/2-0)\cdot 9/4-(1/2-2)\cdot 5/4}{2-0} = 3/2,\\
P_{33} &= \frac{(x-x_2)P_{32}-(x-x_3)P_{22}}{x_3-x_2} = \frac{(1/2-1)\cdot 3/2-(1/2-2)\cdot 23/16}{2-1} = 45/32.
\end{aligned}$$

■