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,

$$\begin{aligned}\psi_n'(x_i) &= (x_i-x_n)\psi_{n-1}'(x_i),\ i=0,\dots,n-1,\\ \psi_n'(x_n) &= \psi_{n-1}(x_n).\end{aligned} \tag{1.1}$$

With these new notations, we can write

$$\begin{aligned}L_{n-1}f(x) &= \sum_{i=0}^{n-1}\frac{\psi_{n-1}(x)}{(x-x_i)\psi_{n-1}'(x_i)}f_i,\\ L_nf(x) &= \sum_{i=0}^{n}\frac{\psi_n(x)}{(x-x_i)\psi_n'(x_i)}f_i.\end{aligned} \tag{1.2}$$

Let us also recall one of the properties of divided differences (Theorem 5.1 a), in Lecture 2):

$$f[x_0,\dots,x_n] = \sum_{i=0}^{n}\frac{1}{\psi_n'(x_i)}f_i. \tag{1.3}$$

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

$$Q(x) \;=\; L_nf(x) - L_{n-1}f(x). \tag{1.4}$$

Obviously, Q is a polynomial of degree (at most) n and for $i = 0, \dots, n-1$,

$$Q(x_i) \;=\; f(x_i) - f(x_i) \;=\; 0,$$

so its n roots are precisely the nodes $x_0, \dots, x_{n-1}$. Then Q is of the form

$$\begin{aligned} Q(x) &= a_n(x - x_0)\dots(x - x_{n-1}) \;=\; a_n\psi_{n-1}(x), \\ Q(x_n) &= a_n\psi_{n-1}(x_n), \end{aligned} \tag{1.5}$$

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

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

$$Q(x_n) \;=\; f_n - L_{n-1}f(x_n). \tag{1.6}$$

By (1.5)–(1.6), it follows that

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

Now using (1.1)–(1.3), we get:

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

Thus,

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

So, by (1.4)–(1.6), we have the following recurrence relation for the Lagrange polynomial:

$$L_nf(x) \;=\; L_{n-1}f(x) + f[x_0,\dots,x_n]\psi_{n-1}(x),\; n \geq 1. \tag{1.7}$$

Iteratively, we get

$$\begin{aligned} L_0f(x) &= f(x_0), \\ L_1f(x) &= f(x_0) + f[x_0,x_1](x-x_0), \\ L_2f(x) &= f(x_0) + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1), \\ \dots \\ L_nf(x) &= f(x_0) + f[x_0,x_1](x-x_0) + \dots + f[x_0,\dots,x_n](x-x_0)\dots(x-x_{n-1}), \end{aligned} \tag{1.8}$$

The expression on the right-hand-side of (1.8) 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

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

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

$$\begin{aligned} N_nf(x) &= D_0 + (x-x_0)D_1 + (x-x_0)(x-x_1)D_2 + \dots + (x-x_0)\dots(x-x_{n-1})D_n \\ &= D_0 + (x-x_0)\Bigg[D_1 + (x-x_1)\Bigg[D_2 + \dots \\ &+ (x-x_{n-2})\Big[D_{n-1} + (x-x_{n-1})D_n\Big]\dots\Bigg]\Bigg]. \end{aligned} \tag{1.9}$$

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:

$$\begin{aligned}
f[x, x_0] &= \frac{f(x) - f(x_0)}{x - x_0} \\
f[x, x_0, x_1] &= \frac{f[x, x_0] - f[x_0, x_1]}{x - x_1} \\
f[x, x_0, x_1, x_2] &= \frac{f[x, x_0, x_1] - f[x_0, x_1, x_2]}{x - x_2} \\
\dots & \quad \dots \\
f[x, x_0, \dots, x_{n-1}] &= \frac{f[x, x_0, \dots, x_{n-2}] - f[x_0, x_1, \dots, x_{n-1}]}{x - x_{n-1}} \\
f[x, x_0, x_1, \dots, x_n] &= \frac{f[x, x_0, \dots, x_{n-1}] - f[x_0, x_1, \dots, x_n]}{x - x_n}.
\end{aligned} \tag{1.10}$$

Multiplying the first equation in (1.10) 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 $(x - x_0)(x - x_1)\dots(x - x_n)$, writing the right-hand-side first, we get

$$\begin{aligned}
f(x) - f(x_0) &= (x - x_0) f[x, x_0] \\
(x - x_0)\Big(f[x, x_0] - f[x_0, x_1]\Big) &= (x - x_0)(x - x_1) f[x, x_0, x_1] \\
(x - x_0)(x - x_1)\Big(f[x, x_0, x_1] - f[x_0, x_1, x_2]\Big) &= (x - x_0)(x - x_1)(x - x_2) f[x, x_0, x_1, x_2] \\
\dots & \quad \dots \\
\prod_{i=0}^{n-2}(x - x_i)\Big(f[x, x_0, \dots, x_{n-2}] - f[x_0, x_1, \dots, x_{n-1}]\Big) &= \prod_{i=0}^{n-1}(x - x_i) f[x, x_0, \dots, x_{n-1}] \\
\prod_{i=0}^{n-1}(x - x_i)\Big(f[x, x_0, \dots, x_{n-1}] - f[x_0, x_1, \dots, x_n]\Big) &= \prod_{i=0}^{n}(x - x_i) f[x, x_0, x_1, \dots, x_n].
\end{aligned}$$

Now, adding all equations above, we obtain

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

from which we have

$$R_nf(x) \;=\; f[x,x_0,\dots,x_n](x-x_0)\dots(x-x_n). \tag{1.11}$$

By the mean value formula for divided differences (Theorem 5.1 e), in Lecture 2), we find the previous formula for the remainder:

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

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

i	x_i	$f(x_i)$
0	0.1	0.2
1	0.2	0.24
2	0.3	0.3

Solution. First, we compute the divided differences:

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

The linear interpolate at the nodes $x_0 = 0.1$ and $x_1 = 0.2$ is then

$$N_1f(x) \;=\; f(x_0) + f[x_0,x_1](x-x_0) \;=\; 0.2 + 0.4(x-0.1) \;=\; 0.4x + 0.16,$$

so we have the approximation

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

The error of this approximation is

$$\begin{aligned} R_1 f(0.15) &= \frac{(0.15-0.1)(0.15-0.2)}{2!} f''(\xi) \\ &= -1.25\cdot 10^{-3}\, f''(\xi),\ \xi \in (0.1, 0.3). \end{aligned}$$

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

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

with an error of

$$\begin{aligned} R_2 f(0.15) &= \frac{(0.15-0.1)(0.15-0.2)(0.15-0.3)}{3!} f'''(\eta) \\ &= 6.25\cdot 10^{-5}\, f'''(\eta),\ \eta \in (0.1, 0.3). \end{aligned}$$

■

Example 1.2. Find the polynomial of minimum degree that interpolates the data $f(-1), f(0)$ and $f(2)$, for some function $f \in C^3[-1,2]$. Determine and discuss the remainder.

Solution. We find the divided differences table:

$$\begin{tikzcd}
-1 & \boldsymbol{f(-1)} \arrow[r] & \boldsymbol{f(0)-f(-1)} \arrow[r] & \boldsymbol{\frac{1}{6}\Big(2f(-1)-3f(0)+f(2)\Big)} \\
0 & f(0) \arrow[r] \arrow[ur] & \frac{1}{2}\Big(f(2)-f(0)\Big) \arrow[ur] & \\
2 & f(2) \arrow[ur] & &
\end{tikzcd}$$

Then the interpolation polynomial is

$$L_2f(x) \;=\; f(-1)+\Big(f(0)-f(-1)\Big)(x+1)+\frac{1}{6}\Big(2f(-1)-3f(0)+f(2)\Big)x(x+1).$$

We can now write it as a linear combination of the given function values,

$$L_2f(x) \;=\; \frac{1}{3}x(x-2)f(-1)-\frac{1}{2}(x+1)(x-2)f(0)+\frac{1}{6}x(x+1)f(2).$$

When written this way, it is very easy to check that L_2f satisfies the interpolation conditions, i.e.

$$L_2f(-1) \;=\; f(-1),\;\; L_2f(0) \;=f(0),\;\; L_2f(2) \;=f(2).$$

The coefficients of $f(-1), f(0)$ and $f(2)$ above are *precisely* the basis polynomials $l_0(x), l_1(x)$ and $l_2(x)$, as they satisfy $l_i(x_j)=\delta_{ij}$, but the computations were much easier to do this way. Alternatively, we can write the polynomial in the usual form,

$$L_2f(x) \;=\; \frac{1}{6}\Big(2f(-1)-3f(0)+f(2)\Big)x^2+\frac{1}{6}\Big(-4f(-1)+3f(0)+f(2)\Big)x+f(0).$$

This second form will be more convenient when we also want to differentiate or integrate the polynomial.

Now, the remainder can be written as

$$R_2f(x) \;=\; \frac{u(x)}{3!}f'''(\xi) \;=\; \frac{x(x+1)(x-2)}{3!}f'''(\xi),\; \xi\in(-1,2).$$

If possible, we may try to find a bound for $|u(x)|$ on $[-1,2]$ (but this may require Matlab ...). In this case, we have

$$\begin{aligned} u(x) \;&=\; x(x+1)(x-2) \;=\; x^3-x^2-2x,\\ u'(x) \;&=\; 3x^2-2x-2. \end{aligned}$$

The zeros of the derivative are $\dfrac{1\pm\sqrt{7}}{3}$, the smaller being a point of local maximum and the larger, a point of local minimum for $u(x)$ on $[-1,2]$. For $|u(x)|$, we have:

$$\max_{x\in[0,2]}|u(x)| = \left|u\left(\frac{1+\sqrt{7}}{3}\right)\right| = \frac{2}{27}(10+7\sqrt{7}).$$

Thus, a bound for the error is

$$|R_2 f(x)| \;\leq\; \frac{2}{27 \cdot 3!}(10 + 7\sqrt{7})\,|f'''(\xi)| \;\approx\; 0.3521\,|f'''(\xi)|,\ \xi \in (-1, 2).$$

■

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

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

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 5.6 in Lecture 2)

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

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

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

Using the notation

$$\left(\begin{array}{c} s \\ k \end{array} \right) = \frac{s(s-1)\cdots(s-k+1)}{k!}, \; s \in \mathbb{R}, k \in \mathbb{N}$$

(the generalized binomial coefficient), we find *Newton's forward difference form* of the interpolation polynomial.

$$\begin{aligned} L_n f(x) &= \sum_{i=0}^{n} \left(\begin{array}{c} s \\ i \end{array} \right) \Delta^i f_0 \\ &= f_0 + \left(\begin{array}{c} s \\ 1 \end{array} \right) \Delta f_0 + \left(\begin{array}{c} s \\ 2 \end{array} \right) \Delta^2 f_0 + \cdots + \left(\begin{array}{c} s \\ n \end{array} \right) \Delta^n f_0, \end{aligned} \tag{1.12}$$

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

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

$$f(x) - L_n f(x) = h^{n+1} \left(\begin{array}{c} s \\ n+1 \end{array} \right) f^{(n+1)}(\xi_x), \tag{1.13}$$

where ξ_x lies in the smallest interval containing $x_0, \ldots, x_n$ 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 $t = (x - x_n)/h$, we obtain

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

which can be written as

$$L_n f(x) = f_n + \left(\begin{array}{c} t \\ 1 \end{array} \right) \nabla f_n + \left(\begin{array}{c} t+1 \\ 2 \end{array} \right) \nabla^2 f_n + \cdots + \left(\begin{array}{c} t+n-1 \\ n \end{array} \right) \nabla^n f_n \tag{1.14}$$

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

In this case, the interpolation error is

$$f(x) - L_n f(x) \;=\; h^{n+1}\binom{t+n}{n+1} f^{(n+1)}(\eta_x), \tag{1.15}$$

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

Example 1.3. 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 (1.12), 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:

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

So,

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

(as before) and

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

Using backward differences, by (1.14), we get

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

with $t = (x - x_2)/h, \ h = 0.1$.
The backward differences are found in the table

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

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 1.4. 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

$$\begin{aligned}\overline{P}_{01}(x) &= l_0(x)f_0 + l_1(x)f_1 \\ &= \frac{x-x_1}{x_0-x_1}f_0 + \frac{x-x_0}{x_1-x_0}f_1 \\ &= \frac{(x-x_0)f_1-(x-x_1)f_0}{x_1-x_0} \\ &= \frac{(x-x_0)\overline{P}_1(x)-(x-x_1)\overline{P}_0(x)}{x_1-x_0},\end{aligned}$$

where $\overline{P}_0$ denotes the polynomial that interpolates f at the node x_0 (a polynomial of degree 0, hence, a constant, f_0), $\overline{P}_1$, the polynomial of degree 0 that interpolates f at the node x_1 (identically equal to f_1), and $\overline{P}_{01}$ 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

$$\overline{P}_{12}(x) = \frac{(x-x_1)\overline{P}_2(x)-(x-x_2)\overline{P}_1(x)}{x_2-x_1},$$

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

$$\overline{P}_{012}(x) = \frac{(x-x_0)\overline{P}_{12}(x)-(x-x_2)\overline{P}_{01}(x)}{x_2-x_0}. \tag{1.16}$$

Let us compute its values at the nodes.

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

Since $\overline{P}_{012}$ is a polynomial of degree 2 and it interpolates f at the nodes x_0, x_1, x_2, it follows by the uniqueness of the Lagrange interpolation polynomial, that $\overline{P}_{012} = L_2 f$.

In a similar fashion, we can construct recursively the polynomials

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

Proposition 1.5. *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*

$$\begin{aligned}
\overline{P}_{01\dots k}(x) &= \frac{1}{x_k-x_0}\begin{vmatrix} x-x_0 & \overline{P}_{01\dots k-1}(x) \\ x-x_k & \overline{P}_{12\dots k}(x) \end{vmatrix}\\
&= \frac{(x-x_0)\overline{P}_{12\dots k}(x)-(x-x_k)\overline{P}_{01\dots k-1}(x)}{x_k-x_0}.
\end{aligned} \tag{1.17}$$

Proof. Obviously, by its construction, the polynomial in (1.17) has degree k. Its values at the nodes are

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

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

□

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.

$$\begin{array}{lllll}
x_0 & \overline{P}_0 & & & \\
x_1 & \overline{P}_1 & \overline{P}_{01} & & \\
x_2 & \overline{P}_2 & \overline{P}_{12} & \overline{P}_{012} & \\
x_3 & \overline{P}_3 & \overline{P}_{23} & \overline{P}_{123} & \overline{P}_{0123}
\end{array}$$

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

$$x_4 \quad \overline{P}_4 \quad \overline{P}_{34} \quad \overline{P}_{234} \quad \overline{P}_{1234} \quad \overline{P}_{01234}$$

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. Notice that each polynomial $\overline{P}$ emphasizes which nodes it interpolates, listing *all* their indices. Then, why not specify the *starting* node and how many nodes *down* the polynomial interpolates? So, let

$$\begin{array}{lcl}
P_{i,0} & = & \overline{P}_i \\
P_{i,1} & = & \overline{P}_{i-1,i} \\
P_{i,2} & = & \overline{P}_{i-2,i-1,i} \\
\ldots & & \\
P_{i,i} & = & \overline{P}_{0,1,\ldots,i}.
\end{array}$$

In other words, $P_{i,j}$ is the polynomial of degree j that interpolates the nodes $x_i, x_{i-1}, \ldots, x_{i-j}$, for $j = 0, 1, \ldots, i$. So, define the new polynomials P as follows:

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

i.e., recursively,

$$\begin{aligned} P_{i,0} &:= f(x_i), \quad i = \overline{0,n}, \\ P_{i,j} &:= \frac{(x - x_{i-j})P_{i,j-1} - (x - x_i)P_{i-1,j-1}}{x_i - x_{i-j}} \\ &= \frac{1}{x_i - x_{i-j}} \begin{vmatrix} x - x_{i-j} & P_{i-1,j-1} \\ x - x_i & P_{i,j-1} \end{vmatrix}, \quad i \geq j > 0. \end{aligned} \tag{1.18}$$

Then, 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 $\{P_{ii}\}_{i \geq 0}$ 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

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

defining recursively

$$\begin{aligned} Q_{i,0} &:= f(x_i), \quad i = \overline{0,n}, \\ Q_{i,j+1} &:= \frac{1}{x_i - x_j} \begin{vmatrix} x - x_j & Q_{j,j} \\ x - x_i & Q_{i,j} \end{vmatrix} = \frac{(x - x_j)Q_{i,j} - (x - x_i)Q_{j,j}}{x_i - x_j}, \quad i > j \geq 0. \end{aligned} \tag{1.19}$$

Again, the Lagrange polynomial will be the one on the diagonal $L_n f = Q_{nn}$.

Example 1.6. 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

$$\sqrt{2} \approx 45/32 = 1.4063,$$

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

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

With Aitken's algorithm, (1.19), we construct the table

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

where

$$\begin{aligned}
Q_{21} &= \frac{(x - x_0)Q_{20} - (x - x_2)Q_{00}}{x_2 - x_0} = \frac{(1/2 - (-1)) \cdot 2 - (1/2 - 1) \cdot 1/2}{1 - (-1)} = 13/8, \\
Q_{22} &= \frac{(x - x_1)Q_{21} - (x - x_2)Q_{11}}{x_2 - x_1} = \frac{(1/2 - 0) \cdot 13/8 - (1/2 - 1) \cdot 5/4}{1 - 0} = 23/16,
\end{aligned}$$

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

The two algorithms are very similar and they actually yield the same values on the main diagonal of the table (the values $P_{nn} = Q_{nn} = L_n f$), so the errors of linear/quadratic interpolates are the same as before.

■