SIAM REVIEW
Vol. 46, No. 3, pp. 501–517

Barycentric Lagrange Interpolation*

Jean-Paul Berrut†
Lloyd N. Trefethen‡

Dedicated to the memory of Peter Henrici (1923–1987)

Abstract. Barycentric interpolation is a variant of Lagrange polynomial interpolation that is fast and stable. It deserves to be known as the standard method of polynomial interpolation.

Key words. barycentric formula, interpolation

AMS subject classifications. 65D05, 65D25

DOI. 10.1137/S0036144502417715

1. Introduction. "Lagrangian interpolation is praised for analytic utility and beauty but deplored for numerical practice." This heading, from the extended table of contents of one of the most enjoyable textbooks of numerical analysis [1], expresses a widespread view.

In the present work we shall show that, on the contrary, the Lagrange approach is in most cases the method of choice for dealing with polynomial interpolants. The key is that the Lagrange polynomial must be manipulated through the formulas of *barycentric interpolation*. Barycentric interpolation is not new, but most students, most mathematical scientists, and even many numerical analysts do not know about it. This simple and powerful idea deserves a place at the heart of introductory courses and textbooks in numerical analysis.[1]

As always with polynomial interpolation, unless the degree of the polynomial is low, it is usually not a good idea to use uniformly spaced interpolation points. Instead one should use Chebyshev points or other systems of points clustered at the boundary of the interval of approximation.

2. Lagrange and Newton Interpolation. Let $n+1$ distinct interpolation points (nodes) x_j, $j = 0, \ldots, n$, be given, together with corresponding numbers f_j, which may or may not be samples of a function f. Unless stated otherwise, we assume that the nodes are real, although most of our results and comments generalize to the

*Received by the editors November 8, 2002; accepted for publication (in revised form) December 24, 2003; published electronically July 30, 2004.
http://www.siam.org/journals/sirev/46-3/41771.html

†Département de Mathématiques, Université de Fribourg, 1700 Fribourg/Pérolles, Switzerland (Jean-Paul.Berrut@unifr.ch). The work of this author was supported by the Swiss National Science Foundation, grant 21–59272.99.

‡Computing Laboratory, Oxford University, Oxford, UK (LNT@comlab.ox.ac.uk).

[1]We are speaking here of a method of interpolating data in one space dimension by a polynomial. The "barycentric coordinates" popular in computational geometry for representing data on triangles and tetrahedra are related, but different.

complex plane. Let Π_n denote the vector space of all polynomials of degree at most n. The classical problem addressed here is that of finding the polynomial $p \in \Pi_n$ that interpolates f at the points x_j, i.e.,

$$p(x_j) = f_j, \qquad j = 0, \ldots, n.$$

The problem is well-posed; i.e., it has a unique solution that depends continuously on the data. Moreover, as explained in virtually every introductory numerical analysis text, the solution can be written in *Lagrange form* [44]:

$$p(x) = \sum_{j=0}^{n} f_j \ell_j(x), \qquad \ell_j(x) = \frac{\prod_{k=0,\, k\neq j}^{n} (x - x_k)}{\prod_{k=0,\, k\neq j}^{n} (x_j - x_k)}. \tag{2.1}$$

The *Lagrange polynomial* ℓ_j corresponding to the node x_j has the property

$$\ell_j(x_k) = \begin{cases} 1, & j = k, \\ 0, & \text{otherwise}, \end{cases} \qquad j, k = 0, \ldots, n. \tag{2.2}$$

At this point, many authors specialize Lagrange's formula (2.1) for small numbers n of nodes, before asserting that certain shortcomings make it a bad choice for practical computations. Among the shortcomings sometimes claimed are these:

1. Each evaluation of $p(x)$ requires $O(n^2)$ additions and multiplications.
2. Adding a new data pair (x_{n+1}, f_{n+1}) requires a new computation from scratch.
3. The computation is numerically unstable.

From here it is commonly concluded that the Lagrange form of p is mainly a theoretical tool for proving theorems. For computations, it is generally recommended that one should instead use Newton's formula, which requires only $O(n)$ flops for each evaluation of p once some numbers, which are independent of the evaluation point x, have been computed.[2]

Newton's approach consists of two steps. First, compute the *Newton tableau* of divided differences

$$\begin{array}{lllll}
f[x_0] & & & & \\
 & f[x_0, x_1] & & & \\
f[x_1] & & f[x_0, x_1, x_2] & & \\
 & f[x_1, x_2] & & f[x_0, x_1, x_2, x_3] & \ddots \\
f[x_2] & & f[x_1, x_2, x_3] & \vdots & \qquad f[x_0, x_1, x_2, \ldots, x_n] \\
 & f[x_2, x_3] & & & \iddots \\
f[x_3] & & \vdots & f[x_{n-3}, \ldots, x_n] & \\
\vdots & \vdots & f[x_{n-2}, x_{n-1}, x_n] & & \\
 & f[x_{n-1}, x_n] & & & \\
f[x_n] & & & &
\end{array}$$

[2]We define a *flop*—floating point operation—as a multiplication or division plus an addition or subtraction. This is quite a simplification of the reality of computer hardware, since divisions are much slower than multiplications on many processors, but this matter is not of primary importance for our discussion since we are mainly interested in the larger differences between $O(n)$ and $O(n^2)$ or $O(n^2)$ and $O(n^3)$. As usual, $O(n^p)$ means $\leq cn^p$ for some constant c.

by the recursive formula

$$f[x_j, x_{j+1}, \ldots, x_{k-1}, x_k] = \frac{f[x_{j+1}, \ldots, x_k] - f[x_j, \ldots, x_{k-1}]}{x_k - x_j} \tag{2.3}$$

with initial condition $f[x_j] = f_j$. This requires about n^2 subtractions and $n^2/2$ divisions. Second, for each x, evaluate $p(x)$ by the *Newton interpolation formula*

$$p(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots$$

$$+ f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1) \cdots (x - x_{n-1}). \tag{2.4}$$

This requires a mere n flops when carried out by nested multiplication, much less than the $O(n^2)$ required for direct application of (2.1).

3. An Improved Lagrange Formula. The first purpose of the present work is to emphasize that the Lagrange formula (2.1) can be rewritten in such a way that it too can be evaluated and updated in $O(n)$ operations, just like its Newton counterpart (2.4). To this end, it suffices to note that the numerator of ℓ_j in (2.1) can be written as the quantity

$$\ell(x) = (x - x_0)(x - x_1) \cdots (x - x_n) \tag{3.1}$$

divided by $x - x_j$. (ℓ is denoted by Ω or ω in many texts.) If we define the *barycentric weights* by[3]

$$w_j = \frac{1}{\prod_{k \neq j}(x_j - x_k)}, \qquad j = 0, \ldots, n, \tag{3.2}$$

that is, $w_j = 1/\ell'(x_j)$ [37, p. 243], we can thus write ℓ_j as

$$\ell_j(x) = \ell(x) \frac{w_j}{x - x_j}.$$

Now we note that all the terms of the sum in (2.1) contain the factor $\ell(x)$, which does not depend on j. This factor may therefore be brought in front of the sum to yield

$$p(x) = \ell(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j} f_j. \tag{3.3}$$

That's it! Now Lagrange interpolation too is a formula requiring $O(n^2)$ flops for calculating some quantities independent of x, the numbers w_j, followed by $O(n)$ flops for evaluating p once these numbers are known. Rutishauser [51] called (3.3) the "first form of the barycentric interpolation formula."

What about updating? A look at (3.2) shows that incorporating a new node x_{n+1} entails two calculations:

- Divide each w_j, $j = 0, \ldots, n$, by $x_j - x_{n+1}$ (one flop for each point), for a cost of $n + 1$ flops.
- Compute w_{n+1} with formula (3.2), for another $n + 1$ flops.

[3] In practice, in view of the remarks of the previous footnote, one would normally choose to implement this formula by n multiplications followed by a single division.

The Lagrange interpolant can thus also be updated with $O(n)$ flops! By applying this update recursively while minimizing the number of divisions, we get the following algorithm for computing $w_j = w_j^{(n)}$, $j = 0, \dots, n$:

$w_0^{(0)} = 1$
for $j = 1$ **to** n **do**
  for $k = 0$ **to** $j-1$ **do**
    $w_k^{(j)} = (x_k - x_j) w_k^{(j-1)}$
  end
  $w_j^{(j)} = \prod_{k=0}^{j-1} (x_j - x_k)$
end
for $j = 0$ **to** n **do**
  $w_j^{(j)} = 1/w_j^{(j)}$
end

This algorithm performs the same operations as (3.2), only in a different order.[4]

A remarkable advantage of Lagrange over Newton interpolation, rarely mentioned in the literature, is that *the quantities that have to be computed in $O(n^2)$ operations do not depend on the data f_j.* This feature permits the interpolation of as many functions as desired in $O(n)$ operations each once the weights w_j are known, whereas Newton interpolation requires the recomputation of the divided difference tableau for each new function.

Another advantage of the Lagrange formula is that it does not depend on the order in which the nodes are arranged. In the Newton formula, the divided differences do have such a dependence. This seems odd aesthetically, and it has a computational consequence: for larger values of n, many orderings lead to numerical instability. For stability, it is necessary to select the points in a *van der Corput* or *Leja sequence* or in another related sequence based on a certain equidistribution property [18, 23, 61, 67].

This is not to say that Newton interpolation has no advantages. One is that it leads to elegant methods for incorporating information on derivatives $f^{(p)}(x_j)$ when solving so-called Hermite interpolation problems. Another may be its ability to accommodate vector or matrix interpolation from scalar data [60]; using a representation like (3.3) would require costly matrix inversion. There are also some theoretical instances in which Newton interpolation is preferable, e.g., the construction of multistep formulas for the solution of ordinary differential equations [33, 45].

4. The Barycentric Formula. Equation (3.3) is not the end of the story: it can be modified to an even more elegant formula, the one that is often used in practice.

Suppose we interpolate, besides the data f_j, the constant function 1, whose interpolant is of course itself. Inserting into (3.3), we get

$$1 = \sum_{j=0}^{n} \ell_j(x) = \ell(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j}. \tag{4.1}$$

Dividing (3.3) by this expression and cancelling the common factor $\ell(x)$, we obtain

[4] This contrasts with a somewhat faster but unstable [28, p. 96] algorithm that takes advantage of the relation $\sum_{k=0}^{j} w_k^{(j)} = 0$ [67].

the *barycentric formula* for p:

$$p(x) = \frac{\displaystyle\sum_{j=0}^{n} \frac{w_j}{x - x_j} f_j}{\displaystyle\sum_{j=0}^{n} \frac{w_j}{x - x_j}} \qquad \text{(4.2)}$$

BARYCENTRIC FORMULA

where w_j is still defined by (3.2). Rutishauser [51] called (4.2) the "second (true) form of the barycentric formula." See p. 229 of [39] for comments on this terminology.

We see that the barycentric formula is a Lagrange formula, but one with a special and beautiful symmetry. The weights w_j appear in the denominator exactly as in the numerator, except without the data factors f_j. This means that any common factor in all the weights w_j may be cancelled without affecting the value of p_n, and in the discussion to follow we will use this freedom.

Like (3.3), (4.2) can also take advantage of the updating of the weights w_j in $O(n)$ flops to incorporate a new data pair (x_{n+1}, f_{n+1}).

5. Chebyshev and Other Point Distributions. For certain special sets of nodes x_j, one can give explicit formulas for the barycentric weights w_j. The obvious place to start is *equidistant nodes* with spacing $h = 2/n$ on the interval $[-1, 1]$. Here the weights can be directly calculated to be $w_j = (-1)^{n-j}\binom{n}{j}/(h^n n!)$ [55], which after cancelling the factors independent of j yields

$$w_j = (-1)^j \binom{n}{j}. \qquad \text{(5.1)}$$

For the interval $[a, b]$ we would multiply the original formula for w_j by $2^n(b-a)^{-n}$, but this constant factor too can be dropped, so we end up with (5.1) again, regardless of a and b.

A glance at (5.1) reveals that if n is large, the weights w_j for equispaced barycentric interpolation vary by exponentially large factors, of order approximately 2^n. This sounds dangerous, and it is. The effect will be that even small data near the center of the interval are associated with large oscillations in the interpolant, on the order of 2^n times bigger, near the edge of the interval [40, 65]. This so-called *Runge phenomenon* is not a problem with the barycentric formula, but is intrinsic in the underlying interpolation problem. Among other things it implies that polynomial interpolation in equally spaced points is highly *ill-conditioned:* small changes in the data may cause huge changes in the interpolant.

For polynomial interpolation to be a well-conditioned process, unless n is rather small, one must dispense with equally spaced points [59, Thm. 6.21.3]. As is well known in approximation theory, the right approach is to use point sets that are clustered at the endpoints of the interval with an asymptotic density proportional to $(1 - x^2)^{-1/2}$ as $n \to \infty$. Remarkably, this is the same asymptotic density one gets if $[-1, 1]$ is interpreted as a conducting wire and the points x_j are interpreted as point charges that repel one another with an inverse-linear force and that are allowed to move along the wire to find their equilibrium configuration [64, Chap. 5]. And it is precisely the same asymptotic density that is required to make the weights w_j of comparable scale in the sense that although they may not all be exactly equal, they do not vary by factors exponentially large in n.

The simplest examples of clustered point sets are the families of *Chebyshev points*, obtained by projecting equally spaced points on the unit circle down to the unit interval $[-1,1]$. Four standard varieties of such points have been defined, and for each, there is an explicit formula for ℓ in (3.1) that may be easily differentiated [9, 35, 47]. From the identity

$$w_j = \frac{1}{\ell'(x_j)} \tag{5.2}$$

mentioned earlier, we accordingly obtain explicit formulas for the weights w_j.

The *Chebyshev points of the first kind* are given by

$$x_j = \cos\frac{(2j+1)\pi}{2n+2}, \qquad j = 0,\dots,n.$$

In this case after cancelling factors independent of j we find [40, p. 249]

$$w_j = (-1)^j \sin\frac{(2j+1)\pi}{2n+2}. \tag{5.3}$$

Note that these numbers vary by factors $O(n)$, not exponentially, reflecting the good distribution of the points. The *Chebyshev points of the second kind* are given by

$$x_j = \cos\frac{j\pi}{n}, \qquad j = 0,\dots,n.$$

Here we find [52]

$$w_j = (-1)^j \delta_j, \qquad \delta_j = \begin{cases} 1/2, & j = 0 \text{ or } j = n, \\ 1, & \text{otherwise;} \end{cases} \tag{5.4}$$

all but two of the weights are exactly equal. Formulas for the Chebyshev points of the third and fourth kinds can be found in [9].

For all of these sets of Chebyshev points, if the interval $[-1,1]$ is linearly transformed to $[a,b]$, the weights as defined by (3.2) all get multiplied by $2^n(b-a)^{-n}$. However, as this factor cancels out in the barycentric formula, there is again no need to include it (and it is safer not to—see section 7).

One sees that, with equidistant or Chebyshev points, no expensive computations are needed to get the weights w_j, and thus only $O(n)$ operations are required for evaluating p_n. No other interpolation method seems to achieve this, for as mentioned in section 3, Newton interpolation always requires $O(n^2)$ operations for the divided differences.

Here is a MATLAB code segment that samples the function $f(x) = |x| + x/2 - x^2$ in 1001 Chebyshev points of the second kind and evaluates the barycentric interpolant in 5000 points. One would rarely use so many points in practice; we pick such large values just to illustrate the effectiveness of the formula. (At the end of section 7 we shall modify this program to avoid failure when $x = x_i$.)

```
n = 1000;
fun = inline('abs(x)+.5*x-x.^2');
x = cos(pi*(0:n)'/n);
f = fun(x);
c = [1/2; ones(n-1,1); 1/2].*(-1).^((0:n)');
```

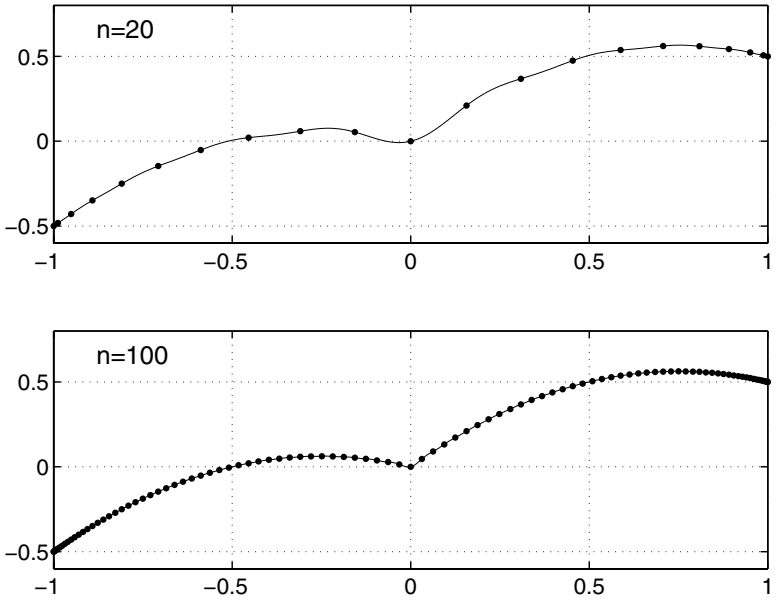

Fig. 5.1 *Barycentric interpolation of the function* $f(x) = |x| + x/2 - x^2$ *in* 21 *and* 101 *Chebyshev points of the second kind on* $[-1, 1]$. *The dots mark the interpolated values* f_j.

```
xx = linspace(-1,1,5000)';
numer = zeros(size(xx));
denom = zeros(size(xx));
for j = 1:n+1
   xdiff = xx-x(j);
   temp = c(j)./xdiff;
   numer = numer + temp*f(j);
   denom = denom + temp;
end
ff = numer./denom;
plot(x,f,'.',xx,ff,'-')
```

On one of our workstations this code runs in about 1 second, producing a plot like those shown in Figure 5.1 but for the case $n = 1000$. For further numerical examples of barycentric interpolation, see [6] and [40].

Other point sets with the asymptotic distribution $(1 - x^2)^{-1/2}$ also lead to well-conditioned polynomial approximation, notably the *Legendre points*—zeros or extrema of the Legendre polynomials. However, explicit formulas for the weights w_j are not known for Legendre points.

6. Convergence Rates for Smooth Functions. If a smooth function f defined on the interval $[-1, 1]$ is interpolated by polynomials in Chebyshev points or in any other system of points with the asymptotic density $(1 - x^2)^{-1/2}$, the rate of convergence of the interpolants to f as $n \to \infty$ is remarkably fast. In particular, suppose that f is a function that can be analytically continued to a function $f(z)$ that is analytic (holomorphic) in a neighborhood of $[-1, 1]$ in the complex plane. Then the

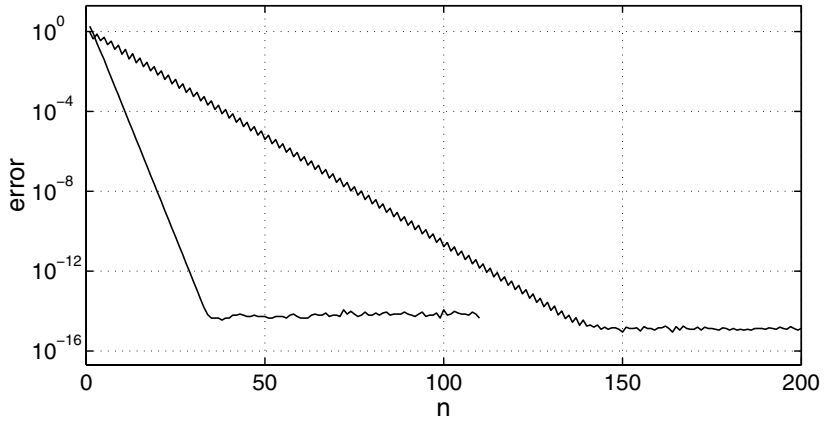

Fig. 6.1 *Convergence of the error* (6.1) *in polynomial interpolation of two smooth functions f in Chebyshev points of the second kind in $[-1,1]$. The lower curve corresponds to $f(x) = \exp(x)/\cos(x)$, and the upper curve to $f(x) = (1+16x^2)^{-1}$. In both cases we observe steady convergence down to the level of rounding errors. The wiggles in the second function arise from the missing central node for odd n.*

interpolants p_n satisfy an error estimate

$$\max_{x\in[-1,1]} |f(x) - p_n(x)| \le CK^{-n} \tag{6.1}$$

for some constants C and $K > 1$ [24, p. 173]. The bigger the region of analyticity, the bigger we may take K to be. To be precise, if f is analytic on and inside an ellipse in the complex plane with foci ± 1 and axis lengths $2L$ and 2ℓ, then we may take $K = L + \ell$ [64, Chap. 5].

Figure 6.1 illustrates this fast convergence for two smooth functions. The convergence rate for $f(x) = \exp(x)/\cos(x)$ is determined by the poles of $f(z)$ at $z = \pm\pi/2$; we have

$$K \;=\; \frac{\pi}{2} + \sqrt{\pi^2 - 1} \;\approx\; 2.7822.$$

The convergence rate for $f(x) = (1 + 16x^2)^{-1}$ is determined by the poles of this function at $z = \pm i/4$; we have

$$K \;=\; \frac{1}{4} + \sqrt{\frac{17}{16}} \approx\; 1.2808.$$

These facts about convergence pertain to polynomial interpolation in general, not barycentric interpolation per se, but the latter has a special feature that might be well suited to taking advantage of such convergence rates under certain circumstances. Suppose we apply the barycentric formula to Chebyshev points associated with values $n = 2, 4, 8, 16, \ldots$. Then it is possible to compute the interpolants recursively, reusing the previous function values f_j each time n is doubled [39]. If f is analytic in a neighborhood of $[-1,1]$, each doubling of n results approximately in a squaring of the error; one has *quadratic convergence* of the overall process.

A function f may be less smooth than the class we have considered (e.g., twice continuously differentiable, or infinitely differentiable); or it may be smoother (e.g.,

analytic in the entire complex plane). A great deal is known about exactly how the convergence rates of polynomial interpolants depend on the precise degree of smoothness of f. See Chapter 5 of [64] for an introduction and [58] for a more advanced treatment.

7. Numerical Stability. Polynomial interpolation is an area in which the errors introduced by computer arithmetic are often a problem. One of the major advantages of the barycentric formula is its good behavior in this respect. But this is a subject where confusion is widespread, and one must be sure to distinguish several different phenomena.

One phenomenon to be clear about is that for improperly distributed sets of nodes, as discussed in section 5, the underlying interpolation problem is ill-conditioned: small changes in the data may cause large changes in the interpolant, typically manifested as large oscillations near the endpoints. Unless the nodes converge to the $(1-x^2)^{-1/2}$ distribution, the condition numbers grow exponentially as $n \to \infty$. In such circumstances polynomial interpolation is usually not a good method for applications, regardless of whether one uses a Lagrange, Newton, or any other formulation. Even though the interpolants may converge to f in theory as $n \to \infty$ when f is sufficiently smooth, rounding errors will destroy the convergence in practice, since the rounding errors are not smooth and thus get multiplied by exponentially large factors.

Let us suppose, then, that we are working with a set of points that are clustered in the above sense, such as Chebyshev or Legendre points. The barycentric formula is now stable provided that two matters described below are attended to. For years this conclusion has been "folklore" in certain circles, and a rigorous analysis will be provided in a forthcoming paper by N. J. Higham that was motivated by reading an early draft of the present article [41]. (In a word, Higham shows that (3.3) is unconditionally stable and that (4.2) is stable too, provided one has a clustered set of interpolation points.)

The first matter is the question of underflow and overflow in the computation of the weights in cases where we are working from the general formula (3.2) rather than from well-behaved explicit expressions like (5.3) and (5.4). In most cases difficulties can be avoided as follows. Suppose we are working on an interval $[a, b]$ of length $4C$. (In the field of complex analysis, C is called the *capacity* of the interval. The following remarks generalize to approximation on more complicated sets in the complex plane besides intervals.) Then as $n \to \infty$, the scale of the weights w_j as defined by (3.2) will grow or decay exponentially at the rate C^{-n}, and the concern about under- or overflow corresponds to situations where n is large or C is far from 1. To minimize the risk, one can multiply each factor $x_j - x_k$ in (3.3) by C^{-1}. This has the effect of rescaling all the weights uniformly by a factor C^n. It is very likely that the resulting numbers will be bigger than 10^{-290} and smaller than 10^{290}—and since the under- and overflow thresholds in IEEE double precision arithmetic are 10^{-308} and 10^{308}, this will be enough to ensure that no digits of accuracy are lost. Exponent exception during computation is still possible in extreme cases, since the absolute values of the quantities $w_k^{(j)}$ in the algorithm in section 3 may grow significantly before decreasing again (or vice versa). In rare situations where n is so large that this is a concern, the problem can be addressed by taking the factors in a random or, better, van der Corput or Leja ordering [18].

The other matter is less trivial but also easily coped with: what if the value of x in (4.2) is very close to one of the interpolation points x_k or, in an extreme case, exactly equal? Consider first the case in which $x \approx x_k$ but $x \neq x_k$. The quotient

$w_k/(x - x_k)$ will be very large, and it would seem that there might be a risk of inaccuracy in this number associated with the subtraction of two nearby quantities in the denominator. However, as pointed out by Henrici [39], this is not in fact a problem. Loosely speaking, there is indeed inaccuracy of this kind, but the same inaccurate numbers appear in both the numerator and the denominator of (4.2), and these inaccuracies cancel out; the formula remains stable overall. Rigorous arguments that make this intuitive idea precise are provided by Higham [41].

If $x = x_k$ exactly, on the other hand, something must certainly be done: the barycentric formulas require a division by zero, and if the MATLAB code segment given earlier is run in standard IEEE arithmetic, the corresponding values of the interpolant `ff` will come out as `NaN`, i.e., not-a-number. One can solve the problem by adding three lines containing a new variable `exact` to the kernel of the MATLAB code, as follows:

```
numer = zeros(size(xx));
denom = zeros(size(xx));
exact = zeros(size(xx));
for j = 1:n+1
   xdiff = xx-x(j);
   temp = c(j)./xdiff;
   numer = numer + temp*f(j);
   denom = denom + temp;
   exact(xdiff==0) = 1;          This 1 should be a j
end
ff = numer./denom;
jj = find(exact); ff(jj) = f(exact(jj));
```

The above observations seem to make barycentric interpolation entirely reliable in practice.

The barycentric interpolation formula has a further kind of stability or robustness property that proves advantageous in some applications. Suppose (4.2) is applied with an arbitrary set of positive weights w_j, not necessarily those given by (3.2). For example, the weights might have been computed somehow with large errors. It is easily seen that the resulting function $p(x)$ still interpolates the data f, even though it is no longer in general a polynomial (see section 9.2).

8. Trigonometric, Sinc, and Laurent Interpolation. Polynomial interpolation in Chebyshev points on $[-1, 1]$ is equivalent to other familiar interpolation problems, and accordingly, these too have barycentric formulas. For example, one can transplant any function f defined on $[-1, 1]$ to a 2π-periodic, *even* function F defined on $[-\pi, \pi]$ by changing variables to $\theta = \cos^{-1} x$ and defining $F(\theta) = f(x) = f(\cos\theta)$ [58]. The barycentric formula becomes [9]

$$t_n(\theta) = \frac{\displaystyle\sum_{j=0}^{n} \frac{w_j}{\cos\theta - \cos\theta_j} F_j}{\displaystyle\sum_{j=0}^{n} \frac{w_j}{\cos\theta - \cos\theta_j}}, \qquad w_j = \frac{1}{\displaystyle\prod_{k\neq j} (\cos\theta_j - \cos\theta_k)}, \tag{8.1}$$

with $\theta_j = \cos^{-1} x_j$ and $F_j = F(\theta_j)$. Since $t_n(\theta) = p_n(x)$ is a polynomial of degree $\leq n$ in $\cos\theta$, it is a trigonometric polynomial of degree $\leq n$ in θ [49, p. 3], and (8.1) is the *trigonometric interpolant* of F. For Chebyshev points of the second kind $\{x_j\}$,

the transplanted nodes $\{\theta_j\}$ are equally spaced, and by (5.4), (8.1) becomes

$$t_n(\theta) = \sum_{j=0}^{n} \frac{(-1)^j \delta_j}{\cos\theta - \cos\theta_j} F_j \Bigg/ \sum_{j=0}^{n} \frac{(-1)^j \delta_j}{\cos\theta - \cos\theta_j} .$$

This formula is a special case of the trigonometric interpolant of degree $\le n$ of an arbitrary function F in an even number $2n$ of *equidistant* nodes $\theta_j = j\pi/n$, $j = 0, \ldots, 2n-1$, on the interval $[0, 2\pi]$ [39]:

$$t(\theta) = \sum_{j=0}^{2n-1} (-1)^j \cot\frac{\theta - \theta_j}{2} F(\theta_j) \Bigg/ \sum_{j=0}^{2n-1} (-1)^j \cot\frac{\theta - \theta_j}{2} . \tag{8.2}$$

The last expression can be further generalized. The function t is nothing but the so-called *cardinal* or *sinc interpolant* S_h of the 2π-periodically extended function F at the equidistant points $\theta_j = jh$, $h = \pi/n$, $j = -\infty, \ldots, \infty$ on the whole real line R [56]. After replacement of the cotangent with its Mittag–Leffler series, (8.2) becomes the barycentric formula for S_h [10],

$$S_h(\theta) = \sum_{j=-\infty}^{\infty} \frac{(-1)^j}{\theta - \theta_j} F(\theta_j) \Bigg/ \sum_{j=-\infty}^{\infty} \frac{(-1)^j}{\theta - \theta_j} .$$

In this way we recover the sinc interpolant for more general functions on R (i.e., nonperiodic). Gautschi has recently found another way of evaluating S_h that is more efficient in many cases [29].

Another closely related problem is that of polynomial interpolation in equally spaced points on the unit circle in the complex plane. For the n roots of unity $z_j = \exp(2\pi ij/n)$, $j = 0, \ldots, n-1$, one has $\ell(z) = \prod_j (z - z_j) = z^n - 1$, and the simplified weights are easily computed from (5.2) as $w_j = z_j$, yielding the barycentric formula [32, 40]

$$p_n(z) = \sum_{j=0}^{n-1} \frac{z_j}{z - z_j} f(z_j) \Bigg/ \sum_{j=0}^{n-1} \frac{z_j}{z - z_j} .$$

This is a starting point for all kinds of computations related to Cauchy integrals and Taylor and Laurent series in complex analysis [38].

9. Further Uses of the Lagrange Representation. Lagrange and barycentric formulas for polynomial interpolation have many other uses. Here are a few.

9.1. Estimation of the Lebesgue Constant. We mentioned in section 5 that some sets of interpolation points are better than others. Famous mathematicians, among them P. Erdős, have given much effort to studies of which point sets are good or best, and how such matters can be quantified. A crucial observation is that, for a fixed set of $n+1$ nodes x_j, the operator P_n that maps a function f to its interpolating polynomial p_n is a linear projection, with $P_n p_n = p_n$. A good measure of the behavior of the interpolation problem is the norm of P_n, known as the *Lebesgue constant*,

$$\Lambda_n = \|P_n\| = \sup_{f \in C[a,b]} \frac{\|P_n f\|}{\|f\|}, \tag{9.1}$$

where $\|f\| = \max_{x\in[a,b]} |f(x)|$ and $C[a,b]$ denotes the space of all continuous functions on $[a,b]$. This number can be shown to be a function of the Lagrange polynomials ℓ_j of (2.1) [48, 58]:

$$\Lambda_n = \max_{x\in[a,b]} \sum_{j=0}^{n} |\ell_j(x)| .$$

For some sets of interpolation points, the values of Λ_n have been determined or estimated theoretically; see [17], [28, p. 121], and [36]. For any nodes, we can use the computed weights (3.2) to yield a lower bound:

$$\Lambda_n \geq \frac{1}{2n^2} \frac{\max_{0\leq j\leq n} |w_j|}{\min_{0\leq j\leq n} |w_j|}. \tag{9.2}$$

This inequality can be derived by applying Markov's inequality on the size of derivatives of polynomials [12]. Notice that it quantifies the observation of section 5 that if the barycentric weights vary widely, the interpolation problem must be ill-conditioned. Other illustrations of the relevance of the quotient of the barycentric weights can be found in [27] and [67].

9.2. Rational Interpolation. It was mentioned in section 7 that if the barycentric formula (4.2) is applied with an arbitrary set of weights w_j, not necessarily those given by (3.2), then the resulting function $p(x)$ still interpolates the data f even though it is no longer in general a polynomial. Back-multiplication of the numerator and denominator by $\ell(x)$ shows that $p(x)$ is in fact the quotient of two polynomials of degree at most n, i.e., a rational interpolant of f. It is easy to see that, conversely, every rational function r that interpolates the data f_j in the points x_j can be written in the form (4.2) for some weights w_j [12, p. 79]. These observations lead to an efficient method for computing rational interpolants based on computing the kernel of a variant type of Vandermonde matrix [13]. The barycentric representation of rational interpolants has several advantages over other representations, besides those it shares with polynomial interpolation. In particular, it allows for an easier detection of so-called unattainable points and of poles in the interval of interpolation [11, 54].

9.3. Differentiation of Polynomial Interpolants. Suppose we have a function u represented by a polynomial interpolant in Lagrange form, $u(x) = \sum_{j=0}^{n} u_j \ell_j(x)$. Then the first and second derivatives of u are

$$u'(x) = \sum_{j=0}^{n} u_j \ell_j'(x), \qquad u''(x) = \sum_{j=0}^{n} u_j \ell_j''(x). \tag{9.3}$$

Now by (4.2), the barycentric representation of ℓ_j is

$$\ell_j(x) = \frac{w_j}{x - x_j} \Bigg/ \sum_{k=0}^{n} \frac{w_k}{x - x_k} .$$

Multiplying through, and then multiplying both sides by $x - x_i$ to render them differentiable at $x = x_i$, we have

$$\ell_j(x) \sum_{k=0}^{n} w_k \frac{x - x_i}{x - x_k} = w_j \frac{x - x_i}{x - x_j},$$

from which it follows with $s(x) = \sum_{k=0}^{n} w_k(x - x_i)/(x - x_k)$ that

$$\ell_j'(x)s(x) + \ell_j(x)s'(x) = w_j \left(\frac{x - x_i}{x - x_j} \right)'$$

and

$$\ell_j''(x)s(x) + 2\ell_j'(x)s'(x) + \ell_j(x)s''(x) = w_j \left(\frac{x - x_i}{x - x_j} \right)''.$$

General formulas for $\ell_j'(x)$ and $\ell_j''(x)$ can be derived from these expressions. At $x = x_i$, straightforward computations yield $s(x_i) = w_i$, $s'(x_i) = \sum_{k \neq i} w_k/(x_i - x_k)$, and $s''(x_i) = -2\sum_{k \neq i} w_k/(x_i - x_k)^2$, from which, together with $\ell_j(x_i) = 0$, we get for $i \neq j$

$$\ell_j'(x_i) = \frac{w_j/w_i}{x_i - x_j}, \qquad \ell_j''(x_i) = -2\frac{w_j/w_i}{x_i - x_j}\left[\sum_{k \neq i} \frac{w_k/w_i}{x_i - x_k} - \frac{1}{x_i - x_j} \right]. \tag{9.4}$$

As for the case $i = j$, (4.1) implies $\sum_{j=0}^{n} \ell_j^{(m)}(x) = 0$ for each differentiation order m and thus

$$\ell_j'(x_j) = -\sum_{i \neq j} \ell_j'(x_i), \qquad \ell_j''(x_j) = -\sum_{i \neq j} \ell_j''(x_i). \tag{9.5}$$

The advantages of (9.5) from the point of view of numerical stability are discussed in [2, 3, 5, 7].

What we have just achieved with the aid of Lagrange interpolation formulas is the computation of the entries of what are commonly known as first- and second-order *differentiation matrices* $D^{(1)}$ and $D^{(2)}$:

$$D_{ij}^{(1)} = \ell_j'(x_i), \qquad D_{ij}^{(2)} = \ell_j''(x_i). \tag{9.6}$$

These matrices have a simple interpretation. If f is a vector of function values associated with the grid $\{x_j\}$, then $D^{(1)}f$ is the vector obtained by interpolating these data, then differentiating the interpolant at the grid points—and similarly for $D^{(2)}$. These formulas hold virtually unchanged for rational interpolation [4]; see [54] for a more general differentiation formula.

9.4. Spectral Methods for Differential Equations. Differentiation of interpolants is the basis of one of the most important applications of polynomial interpolation: *spectral collocation methods* for the numerical solution of ordinary and partial differential equations. Suppose f is a function on $[-1, 1]$, for example, and we want to solve numerically the boundary value problem

$$\frac{d^2u}{dx^2} = f, \qquad -1 < x < 1, \tag{9.7}$$

together with boundary conditions $u(-1) = u(1) = 0$. One way to do this is to set up a Chebyshev grid and consider the $(n-1) \times (n-1)$ matrix problem

$$Dv = f,$$

where D is the matrix consisting of the interior rows and columns of the matrix $D^{(2)}$ of (9.6), v is an $(n-1)$-vector of unknown approximations to u at the interior grid

points, and f is now the $(n-1)$-vector of values of $f(x)$ sampled at these same grid points. If a continuous function $v(x)$ is constructed by barycentric interpolation from the solution vector v together with boundary values zero, then v may be an extremely good approximation to the exact solution u. If f is analytic in a neighborhood of $[-1,1]$, for example, the accuracy will improve geometrically at the rate discussed in section 6 as $n \to \infty$.

Spectral collocation methods have aroused great interest in recent decades and have given rise to a large body of literature, including the books [16, 24, 64] (practically oriented) and [19, 25, 66] (more advanced). These are the high-accuracy, "p versions" of the more general class of finite element (and spectral element) methods for the numerical solution of differential equations, a subject in which polynomials written in Lagrange form also play an important role [22].

9.5. Fast Multipole Methods. Finally, the barycentric formula has natural advantages for applications to fast multipole methods, which are fast algorithms invented by Rokhlin [50] and Greengard [31] for evaluating certain sums [15]. In [21], multipole methods are applied to the evaluation of interpolating polynomials of large degrees by means of the first form of the barycentric formula (3.3). Tests based on the second formula (4.2) are described in [14].

10. Historical Notes. Why is barycentric interpolation not better known?

It seems that the first appearance of the barycentric representation is in the article [62], in which W. Taylor restricted himself to equidistant points. The term "barycentric" seems to appear for the first time in [20] (although Hamming implicitly attributed it to Taylor in [34]). Did Dupuy know of Taylor's work? He did not cite it. Kuntzmann wrote the first book with extensive discussion of barycentric formulas [43], but his work did not have much impact in the USA.

In the first edition of his very influential text [34], Hamming presented the barycentric formula very nicely just after introducing "the Lagrange method of interpolation" and commented that the formula is "easier to use than the Lagrange formula." Why did he discard this paragraph in the second edition? His comment may show that he mainly had hand calculations in mind and believed the advent of the computer would render obsolete any better way of evaluating the formula. This very unfortunate omission would probably have been avoided had Winrich's 1969 paper [68], which clearly documented the superiority of the barycentric formula for every $n > 3$, appeared a little earlier. In [37], another influential book of the 1960s and 1970s, Henrici did not mention barycentric formulas either.

Barycentric formulas were certainly noticed by Rutishauser and Stiefel at the ETH in Zurich. Rutishauser coauthored the chapter on interpolation in [53] with Bulirsch, and he presented barycentric formulas in his classes at the ETH in the 1960s. However, his lecture notes were only published much later by M. Gutknecht [51]. Stiefel emphasized the barycentric formula in [57], as did Henrici in his later text [40]. However, the former is in German, and the second did not have the impact of [37] in the USA. Schwarz's book, first in German and then in an English translation, followed Stiefel's work and retained the barycentric formula [55]. Some French and German authors have described the barycentric representation [46, 63], and in the USA it sometimes appears as an exercise [42, 64]. A recent (transatlantic) text that treats barycentric formulas nicely is the textbook by Gautschi [28], who, as it happens, was the translator of Rutishauser's lecture notes into English [51].

The first publication mentioning that the Lagrange representation can be updated seems to be the important paper by Werner [67]. (If Werner's paper had attracted

more attention, there would have been less need for this one!) Werner's algorithm is in fact twice as fast as the one we give in section 3, but replaces the quotient defining $w_j^{(j)}$ with a long, unstable sum, a drawback also experienced by Gautschi in the related application of the construction of quadrature rules [27]. It should be noted that in 1973 Gander had already updated Lagrange and barycentric formulas for the special case of the extrapolation to the limit [26].

Schneider and Werner were the first to notice that the advantages of the barycentric representation carry over from polynomial to rational interpolation, as mentioned in section 9.2. The first publication of a differentiation matrix as in section 9.3 seems to have been by Bellman et al. [8] as a special case of the general method of *differential quadrature* introduced by these authors. In the context of spectral methods, the first appearance may have been in [30]. In general, however, the literature on spectral methods makes almost no mention of barycentric formulas.

If you look in the index of a book of numerical analysis, you probably won't find "barycentric." Let us hope it will be different a generation from now.

Acknowledgments. We are indebted to Walter Gautschi for reading and amending a first draft of this work, and to Nick Higham and Pete Stewart for detailed comments and suggestions that have led to many improvements.

REFERENCES

[1] F. S. Acton, *Numerical Methods That [Usually] Work*, AMS, Providence, RI, 1990.
[2] R. Baltensperger, *Improving the accuracy of the matrix differentiation method for arbitrary collocation points*, Appl. Numer. Math., 33 (2000), pp. 143–149.
[3] R. Baltensperger and J.-P. Berrut, *The errors in calculating the pseudospectral differentiation matrices for Čebyšev–Gauss–Lobatto points*, Comput. Math. Appl., 37 (1999), pp. 41–48 (errata: Comput. Math. Appl., 38 (1999), p. 119).
[4] R. Baltensperger and J.-P. Berrut, *The linear rational collocation method*, J. Comput. Appl. Math., 134 (2001), pp. 243–258.
[5] R. Baltensperger and M. R. Trummer, *Spectral differencing with a twist*, SIAM J. Sci. Comput., 24 (2003), pp. 1465–1487.
[6] Z. Battles and L. N. Trefethen, *An extension of MATLAB to continuous functions and operators*, SIAM J. Sci. Comput., to appear.
[7] A. Bayliss, A. Class, and B. Matkowsky, *Roundoff error in computing derivatives using the Chebyshev differentiation matrix*, J. Comput. Phys., 116 (1994), pp. 380–383.
[8] R. Bellman, B. G. Kashef, and J. Casti, *Differential quadrature: A technique for the rapid solution of nonlinear partial differential equations*, J. Comput. Phys., 10 (1972), pp. 40–52.
[9] J.-P. Berrut, *Baryzentrische Formeln zur trigonometrischen Interpolation* I, Z. Angew. Math. Phys. (ZAMP), 35 (1984), pp. 91–105.
[10] J.-P. Berrut, *Barycentric formulae for cardinal (SINC-)interpolants*, Numer. Math., 54 (1989), pp. 703–718 (erratum: Numer. Math, 55 (1989), p. 747).
[11] J.-P. Berrut, *Linear rational interpolation of continuous functions over an interval*, in Mathematics of Computation 1943–1993: A Half-Century of Computational Mathematics, Proceedings of Symposia in Applied Mathematics, W. Gautschi, ed., AMS, Providence, RI, 1994, pp. 261–264.
[12] J.-P. Berrut and H. Mittelmann, *Lebesgue constant minimizing linear rational interpolation of continuous functions over the interval*, Comput. Math. Appl., 33 (1997), pp. 77–86.
[13] J.-P. Berrut and H. Mittelmann, *Matrices for the direct determination of the barycentric weights of rational interpolation*, J. Comput. Appl. Math., 78 (1997), pp. 355–370.
[14] M. Blöchliger, *Ein Spezialfall der Multipolmethode für baryzentrische Formeln*, Diploma thesis, University of Fribourg, Switzerland, 1998.
[15] J. P. Boyd, *Multipole expansions and pseudospectral cardinal functions: A new generalization of the Fast Fourier Transform*, J. Comp. Phys., 102 (1992), pp. 184–186.
[16] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, 2nd ed., Dover, New York, 2001.

[17] L. Brutman, *Lebesgue functions for polynomial interpolation—a survey*, in The Heritage of P. L. Chebyshev: A Festschrift in Honor of the 70th Birthday of T. J. Rivlin, Ann. Numer. Math., 4 (1997), pp. 111–127.
[18] D. Calvetti and L. Reichel, *On the evaluation of polynomial coefficients*, Numer. Algebra, 33 (2003), pp. 153–161.
[19] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.
[20] M. Dupuy, *Le calcul numérique des fonctions par l'interpolation barycentrique*, C.R. Acad. Sci., 226 (1948), pp. 158–159.
[21] A. Dutt, M. Gu, and V. Rokhlin, *Fast algorithms for polynomial interpolation, integration, and differentiation*, SIAM J. Numer. Anal., 33 (1996), pp. 1689–1711.
[22] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Introduction to Computational Methods for Differential Equations*, Clarendon Press, Oxford, 1995.
[23] B. Fischer and L. Reichel, *Newton interpolation in Fejér and Chebyshev points*, Math. Comp., 53 (1989), pp. 265–278.
[24] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, Cambridge, UK, 1996.
[25] D. Funaro, *Polynomial Approximation of Differential Equations*, Springer-Verlag, Berlin, 1992.
[26] W. Gander, *Numerische Implementationen des Rombergschen Extrapolationsverfahrens, mit Anwendungen auf die Summation unendlicher Reihen*, ETH Doctoral Thesis 5172, Zürich, 1973.
[27] W. Gautschi, *Moments in quadrature problems*, Comput. Math. Appl., 33 (1997), pp. 105–118.
[28] W. Gautschi, *Numerical Analysis: An Introduction*, Birkhäuser, Boston, 1997.
[29] W. Gautschi, *Barycentric formulae for cardinal (SINC-)interpolants by Jean–Paul Berrut (Remark)*, Numer. Math., 87 (2001), pp. 791–792.
[30] D. Gottlieb, M. Y. Hussaini, and S. A. Orszag, *Theory and applications of spectral methods*, in Spectral Methods for Partial Differential Equations, R. J. Voigt, D. Gottlieb, and M. Y. Hussaini, eds., SIAM, Philadelphia, 1984, pp. 1–54.
[31] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.
[32] M. H. Gutknecht, *Numerical conformal mapping methods based on function conjugation*, J. Comput. Appl. Math., 14 (1986), pp. 31–78.
[33] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations* I. *Non-stiff Problems*, Springer-Verlag, Berlin, 1987.
[34] R. W. Hamming, *Numerical Methods for Scientists and Engineers*, 2nd ed., McGraw–Hill, New York, 1973.
[35] D. C. Handscomb and J. C. Mason, *Chebyshev Polynomials*, Chapman & Hall/CRC, Boca Raton, FL, 2003.
[36] W. Heinrichs, *Strong convergence estimates for pseudospectral methods*, Appl. Math., 37 (1992), pp. 401–417.
[37] P. Henrici, *Elements of Numerical Analysis*, Wiley, New York, 1962.
[38] P. Henrici, *Fast Fourier methods in computational complex analysis*, SIAM Rev., 21 (1979), pp. 481–527.
[39] P. Henrici, *Barycentric formulas for interpolating trigonometric polynomials and their conjugates*, Numer. Math., 33 (1979), pp. 225–234.
[40] P. Henrici, *Essentials of Numerical Analysis*, Wiley, New York, 1982.
[41] N. J. Higham, *The numerical stability of barycentric Lagrange interpolation*, IMA J. Numer. Anal., to appear.
[42] D. Kincaid and W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, Wadsworth, Belmont, CA, 1991.
[43] J. Kuntzmann, *Méthodes Numériques: Interpolation—Dérivées*, Dunod, Paris, 1959.
[44] J. L. Lagrange, *Leçons élémentaires sur les mathématiques, données à l'Ecole Normale en 1795*, in Oeuvres VII, Gauthier–Villars, Paris, 1877, pp. 183–287.
[45] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems*, Wiley, Chichester, UK, 1991.
[46] G. Maess, *Vorlesungen über numerische Mathematik,* II. *Analysis*, Birkhäuser, Basel, 1988.
[47] J. C. Mason, *Chebyshev polynomials of the second, third and fourth kinds in approximation, indefinite integration and integral transforms*, J. Comput. Appl. Math., 49 (1993), pp. 169–178.
[48] M. J. D. Powell, *Approximation Theory and Methods*, Cambridge University Press, Cambridge, UK, 1981.

[49] T. J. Rivlin, *The Chebyshev Polynomials*, Wiley, New York, 1974.
[50] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.
[51] H. Rutishauser, *Vorlesungen über numerische Mathematik*, Vol. 1, Birkhäuser, Basel, Stuttgart, 1976; English translation, *Lectures on Numerical Mathematics*, Walter Gautschi, ed., Birkhäuser, Boston, 1990.
[52] H. E. Salzer, *Lagrangian interpolation at the Chebyshev points* $x_{n,\nu}, = \cos(\nu\pi/n)$, $\nu = 0(1)n$; *some unnoted advantages*, Comput. J., 15 (1972), pp. 156–159.
[53] R. Sauer and I. Szabó, *Mathematische Hilfsmittel des Ingenieurs*, Springer-Verlag, Berlin, Heidelberg, 1968.
[54] C. Schneider and W. Werner, *Some new aspects of rational interpolation*, Math. Comp., 47 (1986), pp. 285–299.
[55] H. R. Schwarz, *Numerische Mathematik*, 4th ed., Teubner, Stuttgart, 1997; English translation of the 2nd edition, *Numerical Analysis: A Comprehensive Introduction*, Wiley, New York, 1989.
[56] F. Stenger, *Numerical Methods Based on Sinc and Analytic Functions*, Springer-Verlag, New York, 1993.
[57] E. Stiefel, *Einführung in die numerische Mathematik*, Teubner, Stuttgart, 1961.
[58] J. Szabados and P. Vértesi, *Interpolation of Functions*, World Scientific, Singapore, 1990.
[59] G. Szegő, *Orthogonal Polynomials*, AMS, Providence, RI, 1978.
[60] H. Tal-Ezer, *Polynomial approximation of functions of matrices and applications*, J. Sci. Comput., 4 (1989), pp. 25–60.
[61] H. Tal-Ezer, *High degree polynomial interpolation in Newton form*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 648–667.
[62] W. J. Taylor, *Method of Lagrangian curvilinear interpolation*, J. Res. Nat. Bur. Standards, 35 (1945), pp. 151–155.
[63] R. Théodor, *Initiation à l'Analyse Numérique*, Masson, Paris, 1989.
[64] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000.
[65] L. N. Trefethen and J. A. C. Weideman, *Two results on polynomial interpolation in equally spaced points*, J. Approx. Theory, 65 (1991), pp. 247–260.
[66] R. J. Voigt, D. Gottlieb, and M. Y. Hussaini, *Spectral Methods for Partial Differential Equations*, SIAM, Philadelphia, 1984.
[67] W. Werner, *Polynomial interpolation: Lagrange versus Newton*, Math. Comp., 43 (1984), pp. 205–217.
[68] L. B. Winrich, *Note on a comparison of evaluation schemes for the interpolating polynomial*, Comput. J., 12 (1969), pp. 154–155.