SIAM J. SCI. COMPUT.
Vol. 35, No. 3, pp. A1254–A1270

BARYCENTRIC HERMITE INTERPOLATION*

BURHAN SADIQ† AND DIVAKAR VISWANATH†

Abstract. Let $z_1, \ldots, z_K$ be distinct grid points. If $f_{k,0}$ is the prescribed value of a function at the grid point z_k and $f_{k,r}$ the prescribed value of the rth derivative, for $1 \le r \le n_k - 1$, the Hermite interpolant is the unique polynomial of degree $N-1$ ($N = n_1 + \cdots + n_K$) which interpolates the prescribed function values and function derivatives. We obtain another derivation of a method for Hermite interpolation recently proposed by Butcher et al. [*Numer. Algorithms,* 56 (2011), pp. 319–347]. One advantage of our derivation is that it leads to an efficient method for updating the barycentric weights. If an additional derivative is prescribed at one of the interpolation points, we show how to update the barycentric coefficients using only $\mathcal{O}(N)$ operations. Even in the context of confluent Newton series, a comparably efficient and general method to update the coefficients appears not to be known. If the method is properly implemented, it computes the barycentric weights with fewer operations than other methods and has very good numerical stability even when derivatives of high order are involved. We give a partial explanation of its numerical stability.

Key words. Hermite, barycentric, interpolation

AMS subject classifications. 65D05, 65D25

DOI. 10.1137/110833221

1. Introduction.

An example of a Hermite interpolant in barycentric form is the unique polynomial $\pi(z)$ of degree 3 such that $\pi(-1) = f_{-1}$, $\pi'(-1) = f'_{-1}$, $\pi(1) = f_1$, and $\pi'(1) = f'_1$ as given by

$$\pi(z) = (z-1)^2(z+1)^2\left(f_{-1}\left(\frac{1}{4(z+1)^2} + \frac{1}{4(z+1)}\right) + \frac{f'_{-1}}{4(z+1)} + f_1\left(\frac{1}{4(z-1)^2} - \frac{1}{4(z-1)}\right) + \frac{f'_1}{4(z-1)}\right). \tag{1.1}$$

In general, Hermite interpolation, the function value, and its first $n_k - 1$ derivatives are prescribed as

$$f_{k,0}, \ldots, f_{k,n_k-1}$$

at the interpolation point or grid point z_k for each z_k from the list $z_1, \ldots, z_K$. The problem is to find a polynomial $\pi(z)$ of degree $N-1$, where $N = n_1 + \cdots + n_K$, such that

$$\left.\frac{d^r\pi(z)}{dz^r}\right|_{z=z_k} = f_{k,r} \quad \text{for} \quad r = 0, \ldots, n_k - 1 \tag{1.2}$$

at each grid point z_k. We shall always assume the grid points z_k to be distinct. For an elegant proof of the existence and uniqueness of the Hermite interpolant, see [6].

*Submitted to the journal's Methods and Algorithms for Scientific Computing section May 9, 2011; accepted for publication (in revised form) October 2, 2012; published electronically May 2, 2013. This work was supported by NSF grants DMS-0715510, DMS-1115277, and SCREMS-1026317.
http://www.siam.org/journals/sisc/35-3/83322.html

†Department of Mathematics, University of Michigan, Ann Arbor, MI 48109 (bsadiq@umich.edu, divakar@umich.edu).

The polynomial $\pi(z)$ can be represented in either the Newton form or the barycentric form. In the Newton form, the grid points z_k must be ordered in some way [4]. If the grid points are not carefully ordered, the Newton form is susceptible to catastrophic numerical instability [7, 17]. In contrast, the barycentric form does not require the grid points to be ordered and treats all the grid points equally.

Barycentric form. The barycentric form of the interpolant is

$$\pi(z) = \pi^*(z) \sum_{k=1}^{K} \frac{f_{k,n_k-1}}{(n_k-1)!} \left(\frac{w_{k,0}}{(z-z_k)} \right) + \frac{f_{k,n_k-2}}{(n_k-2)!} \left(\frac{w_{k,0}}{(z-z_k)^2} + \frac{w_{k,1}}{(z-z_k)} \right) + \cdots + f_{k,0} \left(\frac{w_{k,0}}{(z-z_k)^{n_k}} + \cdots + \frac{w_{k,n_k-1}}{(z-z_k)} \right), \tag{1.3}$$

where $\pi^*(z)$ is defined as $\prod_{k=1}^{K}(z-z_k)^{n_k}$. It is evident from inspection that $\pi(z)$, as represented in (1.3), is a polynomial of degree $N-1$. For a unique choice of weights $w_{k,r}$, $\pi(z)$ will satisfy the interpolation conditions (1.2). Determining and updating the weights (or coefficients) $w_{k,r}$ of the barycentric form is the topic of this paper.

The representation of $\pi(z)$ shown in (1.3) is usually termed the first barycentric form. For the second barycentric form, take $f_{k,0} = 1$ and $f_{k,r} = 0$ for $r \geq 1$ corresponding to the function $f(z) = 1$. By the existence and uniqueness of the Hermite interpolant, we have $1 = \pi^*(z) \sum_{k=1}^{K} \sum_{r=0}^{n_k-1} w_{k,r}(z-z_k)^{n_k-r}$. Dividing (1.3) by the barycentric representation of 1, we have

$$\pi(z) = \frac{\sum_{k=1}^{K} \sum_{s=0}^{n_k-1} \frac{f_{k,s}}{s!} \sum_{r=0}^{n_k-s-1} w_{k,r}(z-z_k)^{n_k-r-s}}{\sum_{k=1}^{K} \sum_{r=0}^{n_k-1} w_{k,r}(z-z_k)^{n_k-r}}. \tag{1.4}$$

This is the second barycentric form. The second barycentric form has a useful property that we now state. If the first barycentric form (1.3) satisfies the interpolation conditions (1.2), so does the second barycentric form (1.4). Less obviously, for any choice of the weights $w_{k,r}$ with $w_{k,0} \neq 0$ for $k = 1, \ldots, K$, the second barycentric form is a rational function of z which satisfies the interpolation conditions (1.2) (see [13] and [1]). The second barycentric form is more robust in the presence of rounding errors in the weights $w_{k,r}$, as one may expect and as we will demonstrate in section 4.

For deriving the weights $w_{k,r}$, we shall work with the first barycentric form (1.3). It is obvious from inspection that either of the two forms can be used to evaluate the interpolant $\pi(z)$ at a given point z using $\mathcal{O}(N)$ arithmetic operations, although the second barycentric form is more robust in the presence of rounding errors.

Calculating barycentric weights using divided differences. One of the methods for finding the weights $w_{k,r}$ in $\mathcal{O}(N^2)$ operations is due to Schneider and Werner [13]. Two ideas go into that method. We briefly describe the two ideas in the simpler context of Lagrange interpolation (see Werner [20] for this special case), where the problem is to find a polynomial $p(z)$ of degree $K-1$ such that $p(z_k) = f_{k,0}$ holds at each grid point z_k.

The Newton representation of $p(z)$ is $a_0 + a_1(z-z_1) + a_2(z-z_1)(z-z_2) + \cdots + a_{K-1}\prod_{k<K}(z-z_k)$ for some coefficients a_k. The Lagrange representation takes the form $\sum_{k=1}^{K} b_k L_k(z)$, where $L_k(z)$ is the unnormalized Lagrange cardinal function $\prod_{j\neq k}(z-z_j)$. The first idea is to note that the coefficients a_k and the coefficients b_k are connected by a triangular matrix [10]. The second idea is to note that if $p(z) \equiv 1$, then $a_0 = 1$ and $a_k = 0$ for $k \geq 1$ and the coefficients b_k are equal to the Lagrange

weights w_k. The triangular relationship between the two sets of coefficients is inverted using divided differences to compute the Lagrange weights. The method of Schneider and Werner for computing the barycentric weights $w_{k,r}$ of the Hermite interpolant follows the same logic.

We make two comments about the method of Schneider and Werner. First, because the Newton expansion is used implicitly, the method depends upon divided differences and finding a good ordering of the grid points z_k. Even if the grid points are carefully ordered and scaled using the logarithmic capacity of the interval of interpolation, it is unclear from the extant literature if the method is numerically stable in the presence of high order derivatives. As our discussion in sections 4 and 5 will indicate, numerical stability is a more delicate issue when high order derivatives occur in the Hermite data.

Second, the method for finding barycentric weights described in section 2 typically has a lower operation count than the method of divided differences. However, Schneider and Werner [13] allow a general denominator in the barycentric form while we specialize the denominator to be 1. Even though it could be possible to specialize the method of divided differences to lower the operation count, that is unlikely to make it a better method for computing the barycentric weights. The argument for the method of section 2 is its simplicity, directness, and numerical stability.

The method of Butcher et al. [3] and its extension. In section 2, we give another derivation of the method of Butcher et al. for computing the barycentric weights. The method was derived by Butcher et al. using contour integrals and the manipulation of infinite series. Our derivation is more direct. The formalism of Butcher et al. extends to Birkhoff interpolation (or Hermite interpolation with incomplete data), a problem which is not discussed here.

To explain the basic idea we use in section 2 for computing the barycentric weights, we go back to the Hermite interpolant shown in (1.1). We want to compute a cubic polynomial whose Taylor expansion to two terms with center $z = 1$ is equal to $f_1 + f_1^{'}(z-1)$ and with center $z = -1$ is equal to $f_{-1} + f_{-1}^{'}(z+1)$. In the prefactor $(z-1)^2(z+1)^2$, $(z-1)^2$ has an effect on the Taylor expansion at $z = 1$ that is easily neutralized by dividing by $(z-1)^2$. The Taylor expansion of $(z+1)^{-2}$ to two terms about $z = 1$ is $1/4-(z-1)/4$. Therefore if $(z+1)^2$ is multiplied with $1/4-(z-1)/4$, the product is $1 + \mathcal{O}\left((z-1)^2\right)$. Thus the product

$$(z+1)^2(z-1)^2 \times \frac{1}{(z-1)^2} \times \left(\frac{1}{4} - \frac{z-1}{4}\right) \times (f_1 + f_1'(z-1))$$
$$= (z+1)^2(z-1)^2 \left(f_1\left(\frac{1}{4(z-1)^2} - \frac{1}{4(z-1)}\right) + \frac{f_1^{'}}{4(z-1)}\right) + \mathcal{O}\left((z-1)^2\right)$$

is equal to $f_1 + f_1^{'}(z-1) + \mathcal{O}\left((z-1)^2\right)$. This observation along with the fact that the expression above is evidently $\mathcal{O}\left((z+1)^2\right)$ explains part of the barycentric representation shown in (1.1).

In general, computing barycentric weights comes down to finding the coefficients in the Taylor polynomial of expressions of the form $\prod_{j\neq k}(z+z_k-z_j)^{-n_j}$. The coefficients of such expressions can be found efficiently and with good numerical stability using a standard identity from symmetric function theory.

In section 3, we show how to update the barycentric weights when an additional data item is introduced using only $\mathcal{O}(N)$ operations. Such an efficient update is not available for the Newton representation of the Hermite interpolant [4]. If the

Newton representation orders the grid points as $z_1, \ldots, z_K$, it can be easily updated if a function value is introduced at a new grid point z_{K+1} or if an additional derivative is introduced at z_K. However, introducing an additional derivative at one of the points $z_1, \ldots, z_{K-1}$ will require a lot of the divided differences table to be recomputed. Indeed, if the additional derivative is introduced at z_1, the entire divided differences table has to be recomputed. The method in section 3 handles all these cases using only $\mathcal{O}(N)$ operations.

Numerical stability. The second barycentric interpolant (1.4) of the Runge function $1/(1+z^2)$ in the interval $-1 \le z \le 1$ using $K = 512$ grid points and $n-1 = 47$ derivatives has a numerical error of about 10^{-15} in the ∞-norm and in double precision arithmetic. If the first barycentric form is used, the error is less than 10^{-12} everywhere except very close to the endpoints. The first barycentric form is very sensitive near the endpoints $z = \pm 1$. At the endpoints, the error in the first barycentric form is as large as 10^{-7}, a phenomenon that is discussed in section 4. Here we mention that the endpoints ± 1 are not included among the 512 grid points.

It is probably unwise to restrict the discussion of numerical stability to the interpolation error $||\pi(z) - f(z)||_\infty$. This is particularly true for the second barycentric form which is quite good at producing an accurate interpolant even if the barycentric weights $w_{k,r}$ themselves are inaccurate. In section 4, we use extended precision computations to check the relative errors in the barycentric weights. We find that the barycentric weights are computed with good accuracy.

The key to the accuracy of the barycentric weights appears to be the behavior of a triangular system that appears in the method of section 2 (see Lemma 2). In section 5, we prove that the barycentric weights are computed with small relative error in a few limited situations.

Barycentric Lagrange interpolation. Berrut and Trefethen [2] have given a new exposition of the barycentric form of Lagrange interpolation, emphasizing the usefulness of separating the computation of the Lagrange weights from the evaluation of the interpolant. They have pointed out that the emphasis on Newton interpolation found in almost every textbook on numerical analysis is misplaced. The barycentric form is as efficient as the Newton form and has better numerical stability. For an application to the computation of finite difference weights, see [12].

Both the first barycentric form (1.3) and second barycentric form (1.4) of the Hermite interpolant separate the computation of the weights $w_{k,r}$ from the evaluation of the interpolant. There are significant differences from the Lagrange setting, however. First, the computation of the barycentric weights is a great deal more complicated than in the Lagrange setting. The efficient updating of barycentric weights when new items of data are introduced is even more involved. Finally, the second barycentric form is more accurate than the first barycentric form in the Hermite setting as we show in section 4 (see Figure 4.3). In contrast, in the Lagrange setting, the first barycentric form is backward stable, whereas the second barycentric from has a more limited type of forward stability [9].

2. Barycentric weights. The Hermite interpolant $\pi(z)$ is the unique polynomial of degree $N-1$ which satisfies the $N-1$ interpolation conditions (1.2). The number of interpolation conditions at the grid point z_k, $1 \le k \le K$, is n_k and $N = n_1 + \cdots + n_K$. The polynomial $\pi^*(z)$ defined as $\prod_{k=1}^{K}(z - z_k)^{n_k}$ is of degree N. The Hermite interpolation conditions can be reworded as requiring the Taylor expansion of $\pi(z)$ centered at $z = z_k$ to be equal to

$$f_{k,0} + f_{k,1}(z - z_k) + \cdots + \frac{f_{k,n_k-1}}{(n_k-1)!}(z - z_k)^{n_k-1} + \mathcal{O}\left((z - z_k)^{n_k}\right) \tag{2.1}$$

at each grid point z_k.

Define $\pi_k(z) = \pi^*(z)(z - z_k)^{-n_k}$ so that $\pi_k(z)$ is a polynomial of degree $N - n_k$. The polynomial

$$W_k(z) = \sum_{r=0}^{r=n_k-1} w_{k,r}(z - z_k)^r \tag{2.2}$$

is defined by the requirement $\pi_k(z)W_k(z) = 1 + \mathcal{O}\left((z - z_k)^{n_k}\right)$. In other words, $W_k(z)$ is equal to the Taylor series of $\pi_k(z)^{-1}$ with center $z = z_k$ and truncated at the n_kth power.

The polynomial

$$\pi_k(z)W_k(z)\left(f_{k,0} + f_{k,1}(z - z_k) + \cdots + \frac{f_{k,n_k-1}}{(n_k-1)!}(z - z_k)^{n_k-1}\right)$$

has a Taylor expansion at $z = z_k$ which is identical to (2.1) because $\pi_k(z)W_k(z) = 1 + \mathcal{O}\left((z - z_k)^{n_k}\right)$ in the limit $z \to z_k$. In addition, for $j \neq k$ the polynomial is $\mathcal{O}\left((z - z_j)^{n_j}\right)$ in the limit $z \to z_j$ because $\pi_k(z)$ is $\mathcal{O}\left((z - z_j)^{n_j}\right)$ in that limit. However, the degree of the polynomial is $N + n_k - 2$, which is more than $N - 1$ if $n_k > 1$. This difficulty is easily fixed. We simply need to multiply $W_k(z)$, which is thought of as a polynomial in $(z - z_k)$, and $\sum_{r=0}^{n_k-1} \frac{f_{k,r}}{r!}(z - z_k)^r$ and drop all terms of order $(z - z_k)^{n_k}$ or higher. The polynomial

$$\begin{aligned}\pi_k(z)\Bigg(&f_{k,0}\left(w_{k,0} + \cdots + w_{k,n_k-1}(z - z_k)^{n_k-1}\right) + \cdots + \frac{f_{k,n_k-2}(z - z_k)^{n_k-2}}{(n_k-2)!} \\ &\times (w_{k,0} + w_{k,1}(z - z_k)) + \frac{f_{k,n_k-1}(z - z_k)^{n_k-1}}{(n_k-1)!}w_{k,0}\Bigg)\end{aligned} \tag{2.3}$$

is of degree $N - 1$. For $j \neq k$, it is $\mathcal{O}\left((z - z_j)^{n_j}\right)$ near z_j because $\pi_k(z)$ is divisible by $(z - z_j)^{n_j}$. At $z = z_k$, it satisfies the interpolation condition (2.1). We have the following lemma.

LEMMA 1. *The barycentric weights $w_{k,r}$, $r = 0, 1, \ldots, n_k - 1$, are the coefficients of the unique polynomial $W_k(z)$ of degree $n_k - 1$, shown in* (2.2)*, which satisfies $\pi_k(z)W_k(z) = 1 + \mathcal{O}\left((z - z_k)^{n_k}\right)$ in the limit $z \to z_k$.*

To calculate the barycentric weights, it is useful to look at the following equation:

$$\frac{1}{\left(1 - \frac{z}{\alpha_1}\right)\left(1 - \frac{z}{\alpha_2}\right)\ldots\left(1 - \frac{z}{\alpha_N}\right)} = 1 + \mathcal{I}_1 z + \mathcal{I}_2 z^2 + \cdots . \tag{2.4}$$

We assume $\alpha_k \neq 0$ and $\mathcal{I}_1, \mathcal{I}_2, \ldots$ on the right-hand side of (2.4) are defined by expanding the left-hand side in powers of z.

To find an efficient method to compute the $\mathcal{I}_r$, we define $\mathcal{P}_r = \sum_{k=1}^N \alpha_k^{-r}$ and differentiate (2.4) with respect to z to get

$$\frac{1}{\prod_{k=1}^N \left(1 - \frac{z}{\alpha_k}\right)} \sum_{k=1}^N \frac{1}{\alpha_k}\left(1 - \frac{z}{\alpha_k}\right)^{-1} = \mathcal{I}_1 + 2\mathcal{I}_2 z + 3\mathcal{I}_3 z^2 + \cdots ,$$

$$\left(1 + \mathcal{I}_1 z + \mathcal{I}_2 z^2 + \cdots\right)\left(\mathcal{P}_1 + \mathcal{P}_2 z + \cdots\right) = \mathcal{I}_1 + 2\mathcal{I}_2 z + \cdots .$$

Equating coefficients of z, we have the following lemma.

ALGORITHM 1 (Computing barycentric weights).

function FINDPI($\zeta_1, \ldots, \zeta_K$, $n_1, \ldots, n_K$)
  return $\zeta_1^{n_1} \ldots \zeta_K^{n_K}$
end function
function NINVERTLIST($z_1, \ldots, z_K$, $\zeta_1, \ldots, \zeta_K$)
  $\zeta_k = -1/z_k$ for $k = 1, \ldots, K$
end function
function POWERSUMS($\zeta_1, \ldots, \zeta_K$, $n_1, \ldots, n_K$, $\mathcal{P}_1, \ldots, \mathcal{P}_{n-1}$)
  Temporaries: $t_1, \ldots, t_K$
  $t_k = n_k$ for $1 \le k \le K$
  for $r = 1, \ldots, n-1$ **do**
    $t_k = \zeta_k t_k$ for $1 \le k \le K$
    $\mathcal{P}_r = t_1 + \cdots + t_K$
  end for
end function
function INVERSEPOLY($\mathcal{P}_1, \ldots, \mathcal{P}_{n-1}$, $\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_{n-1}$)
  $\mathcal{I}_0 = 1$
  for $r = 1, \ldots, n-1$ **do**
    $\mathcal{I}_r = \sum_{s=1}^{r} \mathcal{P}_s \mathcal{I}_{r-s}/r$
  end for
end function
Comment: The grid points z_k must be distinct.
function HERMITEWEIGHTS($z_1, \ldots, z_K, n_1, \ldots, n_K$, $w_{k,r}$ with $1 \le k \le K$ and $0 \le r < n_k$)
  for $k = 1, \ldots, K$ **do**
    Temporaries: $z'_1, \ldots, z'_{K-1}$, $n'_1, \ldots, n'_{K-1}, n$, C_k, $\zeta_1, \ldots, \zeta_{K-1}$
    $n = n_k$
    Temporaries: $\mathcal{P}_1, \ldots, \mathcal{P}_{n-1}$, $\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}$
    $z'_j = z_k - z_j$ for $1 \le j < k$ and $z'_j = z_k - z_{j+1}$ for $k \le j \le K-1$
    $n'_j = n_j$ for $1 \le j < k$ and $n'_j = n_{j+1}$ for $k \le j \le K-1$
    NINVERTLIST($z'_1, \ldots, z'_{K-1}, \zeta_1, \ldots, \zeta_{K-1}$)
    POWERSUMS($\zeta_1, \ldots, \zeta_{K-1}, n'_1, \ldots, n'_{K-1}$, $\mathcal{P}_1, \ldots, \mathcal{P}_{n-1}$)
    INVERSEPOLY($\mathcal{P}_1, \ldots, \mathcal{P}_{n-1}$, $\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_{n-1}$)
    $C_k = 1/$FINDPI($z'_1, \ldots, z'_{K-1}, n'_1, \ldots, n'_{K-1}$)
    $w_{k,r} = C_k \mathcal{I}_r$ for $r = 0, 1, \ldots, n-1$
  end for
end function

LEMMA 2. *The quantities $\mathcal{I}_r$ defined by* (2.4) *are related to the inverse power sums $\mathcal{P}_r = \sum_{k=1}^{N} \alpha_k^{-r}$ by*

$$\begin{aligned} \mathcal{I}_1 &= \mathcal{P}_1, \\ 2\mathcal{I}_2 &= \mathcal{P}_2 + \mathcal{I}_1\mathcal{P}_1, \\ 3\mathcal{I}_3 &= \mathcal{P}_3 + \mathcal{I}_1\mathcal{P}_2 + \mathcal{I}_2\mathcal{P}_1, \\ 4\mathcal{I}_4 &= \mathcal{P}_4 + \mathcal{I}_1\mathcal{P}_3 + \mathcal{I}_2\mathcal{P}_2 + \mathcal{I}_3\mathcal{P}_1, \end{aligned}$$

and so on.

Coincidentally, Lemma 2 appears as Lemma 2 in [3] and in [16], but with different proofs. It has been well known in symmetric function theory for a very long time.

Remark. This lemma appears in passing in modern books on combinatorics. Stanley [14, p. 396] refers to the review article by Vahlen [18, 1904] for information about the sources of such identities. The identities appear to be due to Crocchi [5].

If we note that

$$\pi_k(z)^{-1} = \prod_{j\neq k}(z_k - z_j)^{-n_j} \prod_{j\neq k}\left(1 - \frac{z - z_k}{z_j - z_k}\right)^{-n_j},$$

Lemmas 1 and 2 imply an algorithm for computing $w_{k,r}$. The algorithm begins by computing $C_k = \prod_{j\neq k}(z_k - z_j)^{-n_j}$ and defines $\mathcal{P}_r = \sum_{j\neq k} n_j(z_j - z_k)^{-r}$. The quantities $\mathcal{I}_r$ are obtained from Lemma 2 and $w_{k,r} = C_k\mathcal{I}_r$. The grid points z_k are assumed to be distinct.

Pseudocode derived from a C++ implementation is displayed as Algorithm 1. For another derivation of the same algorithm, see Butcher et al. [3].

The functions `PowerSums` and `InversePoly` defined on lines 7 and 15 of Algorithm 1 perform $2nK$ and n^2 arithmetic operations, respectively. Summing the cost of invoking those functions from lines 30 and 31, we find that the number of arithmetic operations performed by Algorithm 1 is $2NK + \sum_{k=1}^{K} n_k^2$ to leading order. Roughly half these operations are additions or subtractions and roughly half are multiplications. On line 29, a total of K^2 divisions are performed. That count can be easily reduced to $K^2/2$ because the differences $z_k - z_j$ that needed to be inverted are only $K^2/2$ in number. The arithmetic operations on lines 32 and 33 do not affect the leading order of the total count.

The method of divided differences for computing the barycentric weights $w_{k,r}$ requires $K(K-1)/2$ divisions and $\frac{N^2-\sum n_k^2}{2}$ multiplications [13]. The number of subtractions or additions is about the same as the number of multiplications. To compare the operation counts of the two methods, consider the situation in which $n_1 = \cdots = n_K = n$. In that situation, the number of multiplications used by the method of divided differences is $\left(n^2K^2 - Kn^2\right)/2$ as against $nK^2 + n^2K$ used by the method of Newton identities. The ratio of the two quantities is $(K-1)/(2K/n+1)$. The same ratio applies to the number of subtractions or additions. The two methods have similar operation counts if $n = 2$, but for $n > 2$ Algorithm 1 uses fewer arithmetic operations by a factor of approximately $n/2$.

The argument for Algorithm 1 over the method of divided differences is its simplicity and numerical stability, which will be illustrated in sections 4 and 5. In addition, Algorithm 1 has a favorable operation count.

3. Updating the barycentric weights. Suppose the barycentric weights $w_{k,r}$ have been computed assuming that the function value and $n_k - 1$ derivatives are prescribed at grid points z_k for $1 \le k \le K$. Suppose a new item of data is introduced at the grid point ζ. If ζ is a new grid point, the new item of data is the function value at the new grid point. If $\zeta = z_k$ for some $1 \le k \le K$, the new item of data must be the n_kth derivative at z_k. In either case, one new barycentric weight must be computed and all the other barycentric weights must be updated. We show how to do that in $\mathcal{O}(N)$ operations, where $N = n_1 + \cdots + n_K$.

Suppose new data is introduced at ζ and $z_k \neq \zeta$. Updating the weights $w_{k,r}$, $0 \le r < n_k$, is the easy part. The polynomial $\pi_k(z)$ must be changed to $\pi_k(z)(z-\zeta)$. The weights $w_{k,r}$ were determined such that $\pi_k(z)W_k(z) = 1 + \mathcal{O}\left((z-z_k)^{n_k}\right)$ for $z \to z_k$ with $W_k(z) = \sum_{r=0}^{n_k-1} w_{k,r}(z-z_k)^r$. If the new weights are $w'_{k,r}$ and the corresponding polynomial is $W'_k(z)$, then we require

$$\pi_k(z)(z-\zeta)W_k'(z) = 1 + \mathcal{O}\left((z-z_k)^{n_k}\right).$$

To ensure this condition, it is enough if we find $W_k'(z)$ of degree $n_k - 1$ such that

$$(z-\zeta)W_k'(z) = \big((z-z_k)+(z_k-\zeta)\big)W_k'(z) = W_k(z) + \mathcal{O}\left((z-z_k)^{n_k}\right)$$

in the limit $z \to z_k$. This gives the equations

$$\begin{aligned} (z_k-\zeta)w_{k,0}' &= w_{k,0}, \\ (z_k-\zeta)w_{k,r}' + w_{k,r-1}' &= w_{k,r} \quad \text{for} \quad 1 \le r < n_k, \end{aligned} \tag{3.1}$$

which are solved to deduce the updated barycentric weights $w_{k,r}'$. It takes two operations to update each barycentric weight $w_{k,r}$ for k with $z_k \neq \zeta$.

If $z_k \neq \zeta$ for $1 \le k \le K$, then ζ is the new grid point z_{K+1}. The weights $w_{k,r}$ are updated using (3.1) for $1 \le k \le K$ as already described. We need to compute the new weight $w_{K+1,0}$. This weight is computed using the formula $w_{K+1,0} = \prod_{j=1}^{K}(\zeta - z_k)^{-n_k}$. If ζ is a new grid point, namely, z_{K+1}, the total cost for updating all the weights and computing the new weight is $\mathcal{O}(N)$.

If $\zeta = z_\kappa$ for some κ, $1 \le \kappa \le K$, the weights $w_{k,r}$ are updated as already described if $k \neq \kappa$. We are left with the case of updating the weights $w_{\kappa,r}$ with $\zeta = z_\kappa$ for some κ, $1 \le \kappa \le K$. In this case, the new data item is another derivative at z_κ. Efficient updating of the weights $w_{\kappa,r}$ causes complications that we now turn to.

Define

$$\begin{aligned} A_k &= \left\{ 1/(z_j - z_k) \text{ repeated } n_j \text{ times} \;\middle|\; j \neq k \text{ and } 1 \le j \le K \right\}, \\ C_k &= \prod_{j \neq k} (z_k - z_j)^{-n_j}. \end{aligned}$$

Here A_k is a multiset. Define $\mathcal{P}_r(A_k)$ as the sum of the rth powers of the elements of A_k. Let $\mathcal{I}_r(A_k)$ be related to $\mathcal{P}_r(A_k)$ by the triangular identities of Lemma 2. According to Algorithm 1, the weight $w_{k,r}$ is given by $w_{k,r} = C_k \mathcal{I}_r(A_k)$ for $0 \le r < n_k$ and $1 \le k \le K$. When computing the weights $w_{k,0}, \ldots, w_{k,n_k-1}$, the $2n_k - 1$ intermediate quantities

$$C_k, \mathcal{P}_1(A_k), \ldots, \mathcal{P}_{n_k-1}(A_k), \mathcal{I}_1(A_k), \ldots, \mathcal{I}_{n_k-1}(A_k) \tag{3.2}$$

arise. We assume that these intermediate quantities are stored for each k. The total number of intermediate quantities stored is $2N - K$.

The reason for storing these intermediate quantities is as follows. Suppose a new data entry is introduced at the grid point $\zeta = z_\kappa$. Then a new weight w_{κ,n_κ} needs to be generated. To calculate that weight, $\mathcal{I}_{n_\kappa}(A_\kappa)$ will be generated using the triangular identities of Lemma 2 (the update rules of (3.1) are of no use here). The intermediate quantities are used to calculate $\mathcal{I}_{n_\kappa}(A_\kappa)$.

When new data is introduced at the point ζ, the intermediate quantities are updated at all grid points z_k regardless of whether $z_k \neq \zeta$ or $z_k = \zeta$ (so than we can handle a new data item introduced at z_k later on). If $z_k \neq \zeta$, some of the intermediate quantities in the list (3.2) are updated as follows:

$$\begin{aligned} C_k' &\leftarrow C_k/(z_k - \zeta), \\ \mathcal{P}_r(A_k') &\leftarrow \mathcal{P}_r(A_k) + (\zeta - z_k)^{-r} \quad \text{for} \quad 1 \le r < n_k, \end{aligned} \tag{3.3}$$

where the primes denote updated quantities with $A'_k = A_k \cup \{1/(\zeta - z_k)\}$. (This is a multiset union, not a set union.) To update $\mathcal{I}_r$, we go back to (2.4), which defines $\mathcal{I}_1, \mathcal{I}_2, \ldots$, and deduce the following:

$$\mathcal{I}_r(A'_k) - \frac{\mathcal{I}_{r-1}(A'_k)}{\zeta - z_k} = \mathcal{I}_r(A_k) \quad \text{for} \quad r = 1, \ldots, n_k - 1, \tag{3.4}$$

where it is assumed that $\mathcal{I}_0 = 1$. Using (3.3) and (3.4), it costs two operations to update each intermediate quantity in the list (3.2). Since the total number of intermediate quantities is $\mathcal{O}(N)$, the cost for updating the intermediate quantities for $k \neq \kappa$ is also $\mathcal{O}(N)$.

In $\zeta = z_\kappa$, A_κ does not change and none of the intermediate quantities in (3.2) with $k = \kappa$ needs to be updated. However, we need to generate $\mathcal{P}_{n_\kappa}(A_\kappa)$ and $\mathcal{I}_{n_\kappa}(A_\kappa)$. These are generated using the formulas

$$\mathcal{P}_{n_\kappa}(A_\kappa) = \sum_{j \neq \kappa} n_j \, (z_j - z_\kappa)^{-n_\kappa}, \quad \mathcal{I}_{n_\kappa}(A_\kappa) = (1/n_\kappa) \sum_{s=1}^{n_\kappa} \mathcal{P}(A_\kappa) \mathcal{I}_{n_\kappa - s}(A_\kappa),$$

the first of which is by definition and the second is a triangular identity of Lemma 2. If the temporary t_j that arise on line 8 of Algorithm 1 are preserved, $\mathcal{P}_{n_\kappa}$ can be computed using only $2n_\kappa - 1$ operations. The barycentric weights $w_{\kappa,0}, \ldots, w_{\kappa,n_\kappa - 1}$ do not change. The weight w_{κ,n_κ} is computed as $C_\kappa \mathcal{I}_{n_\kappa}$. We have thus completed describing an $\mathcal{O}(N)$ method for updating all the barycentric weights and generating a new weight when a new data item is introduced.

If the barycentric weights are generated by repeatedly adding new items of data at grid points, the grid points must be ordered in some fashion. As we noted in the introductory section, ordering the grid points may make the method vulnerable to numerical instability. If the weights need to be updated a small number of times to accommodate new items of data, the method described in this section will be numerically stable.

A more general updating problem is to update the existing barycentric weights and generate new ones, when n'_k new data items are introduced at the grid point z_k for $1 \leq k \leq K$. Algorithm 1 and the updating algorithm of this section can be combined to deduce an efficient and numerically stable solution to this more general updating problem.

4. Illustration of numerical stability. If the grid points z_k, $1 \leq k \leq K$, and the number of derivatives at grid points n_k, $1 \leq k \leq K$, are chosen without care, the barycentric interpolation of $f(z)$ using the Hermite interpolant $\pi(z)$ can be very badly conditioned. To test the barycentric formulas (1.3) and (1.4), we first determine a choice of z_k, n_k which leads to a well-conditioned interpolation problem.

Let $L_{k,r}(z)$ denote the polynomial of degree $N - 1$ ($N = n_1 + \cdots + n_K$) such that

$$\frac{d^s L_{k,r}(z)}{dz^s}\Big|_{z=z_\ell} = \delta_{r,s}\delta_{k,\ell} \quad \text{for} \quad 1 \leq \ell \leq K, \; 0 \leq s < n_l.$$

The fundamental polynomials (or cardinal functions) of Hermite interpolation $L_{k,r}(z)$ have a prescribed behavior at the grid points. However, they may become very large between the grid points, leading to a poorly conditioned interpolation problem.

If the grid points are chosen as $z_k = \cos\left((2k-1)\pi/2K\right)$, $1 \leq k \leq K$ (these are the Chebyshev points), and $n_1 = \cdots = n_K = n$, Szabados [16] has proved upper bounds on the fundamental polynomials in the interval $[-1, 1]$. In particular,

$$\sup_{-1\le z\le 1}\sum_{k=1}^{K}|L_{k,r}(z)| \le \begin{cases} \mathcal{O}\left(\dfrac{\log K}{K^r}\right) & \text{if } n-r \text{ is odd,} \\ \mathcal{O}\left(\dfrac{1}{K^r}\right) & \text{if } n-r \text{ is even} \end{cases}$$

for $r = 0, \ldots, n-1$. These bounds imply that a small change in the interpolation data $f_{k,r}$ will result in a small change in the interpolant $\pi(z)$ (defined by (1.3) or (1.4)), thus showing the Hermite interpolation problem to be well-conditioned.

In all our computations, the Chebyshev points z_k are replaced by $2z_k$ and correspondingly the data $f_{k,r}$ are replaced by $f_{k,r}/2^r$. If $z \in [-1, 1]$, we have

$$\lim_{K\to\infty}\left(\prod_{k=1}^{K}(z - z_k)\right)^{1/K} = \frac{1}{2}$$

because the capacity of an interval is a quarter of its length [6, p. 83]. So we may expect the prefactor $\pi^*(z)$ in the first barycentric form (1.3) as well as the quantity C_k, which occurs on line 32 of Algorithm 1, to be roughly of the order $1/2^{nK}$. For reasonably large K and n, C_k and $\pi^*(z)$ underflow in IEEE double precision arithmetic. Replacing z_k by $2z_k$ and taking $z \in [-2, 2]$ changes the limit above to 1, implying that the quantities such as C_k that occur in the algorithm are better scaled.

Barycentric Hermite interpolation problem is highly susceptible to overflows and underflows. Scaling the Chebyshev points so that the capacity of the interval is 1 as in the previous paragraph is only a partial cure. Even if the z_k are scaled as indicated, a product evaluated in the order $(2z - 2z_1)$, $(2z - 2z_1)(2z - 2z_2)$, $(2z - 2z_1)(2z - 2z_2)(2z - 2z_3)$, and so on may overflow or underflow in its intermediate stages even though the final result can be represented in machine arithmetic. To decrease the chances of such a thing happening, we reorder the Chebyshev points z_k in the Leja ordering [11] in addition to multiplying them by 2.

In our implementation of the barycentric formulas (1.3) and (1.4), the function values input to the interpolation procedures are $f_{n,r}/r!$, not $f_{n,r}$. This eliminates the need to first multiply and then divide a quantity by $r!$ (the computation of which is obviously prone to overflows) for certain functions such as the Runge function discussed below.

We now turn to the Runge function $f(z) = 1/(1 + z^2)$, which is the subject of computations shown in Figure 4.1. A numerically stable method to calculate higher order derivatives of the Runge function for $z \in [-1, 1]$ is implied by the following calculation:

$$f(z) = \frac{1}{2i}\left(\frac{1}{z-i} - \frac{1}{z+i}\right),$$
$$\frac{d^r f(z)}{dz^r} = (-1)^r \frac{r!}{2i}\left(\frac{1}{(z-i)^{r+1}} - \frac{1}{(z+i)^{r+1}}\right).$$

If $z - i = R\exp(i\theta)$, then we have $f^{(r)}(z)/r! = (-1)^{r-1}R^{r+1}\sin(r+1)\theta$. This formula has excellent numerical stability for $z \in [-1, 1]$.

Figure 4.1 shows that the error in interpolating the Runge function is $c_n \exp(-\alpha n K)$. The positive constant α appears to be independent of n. However, c_n increases with n. Therefore if we are allowed to use N items of information about

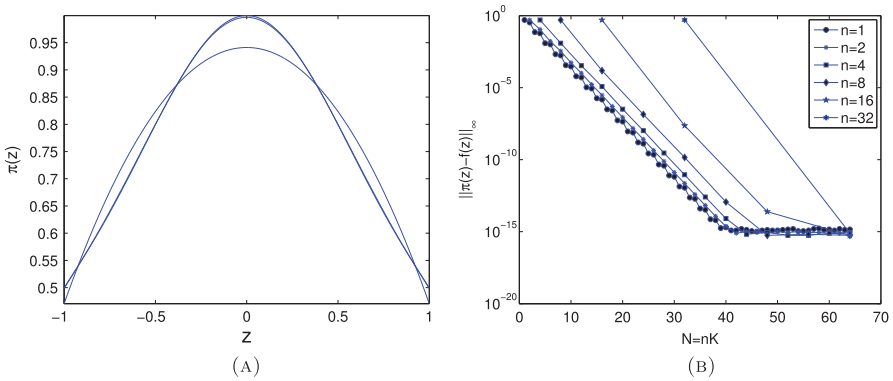

FIG. 4.1. (A) *Interpolants $\pi(z)$ to the Runge function $f(z) = 1/(1+z^2)$ with $K = 4$ and $n = 1, 2, 3$.* (B) *Dependence of interpolation error on the number of grid points K for fixed number of derivatives n at each grid point.*

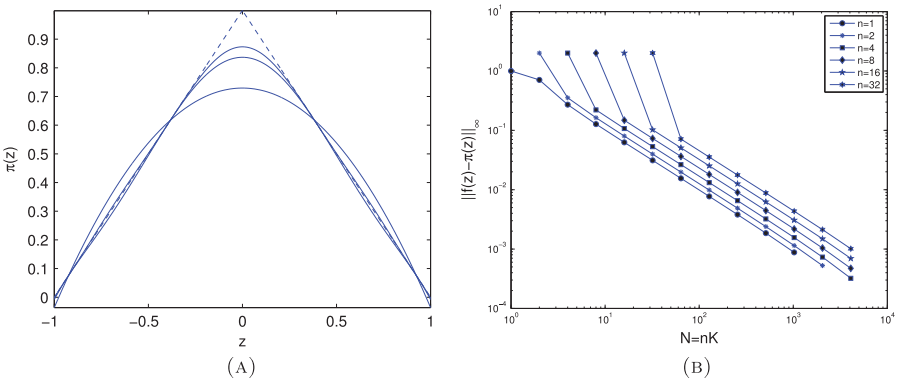

FIG. 4.2. (A) *Interpolants $\pi(z)$ to the hat function $f(z) = 1 - |z|$ with $K = 4$ and $n = 1, 2, 3$.* (B) *Dependence of interpolation error on the number of grid points K for fixed number of derivatives n at each grid point.*

the function to form an accurate interpolant, the best choice for this example is to take them all to be function values.

For the hat function $f(z) = 1 - |z|$, Figure 4.2 shows that the error is proportional to $1/K$. If the error is represented in the form c_n/nK, c_n once again increases with n.

The Runge function is analytic in an open set around $[-1, 1]$, while the hat function is of bounded variation but not even continuously differentiable. In the case $n = 1$, which corresponds to ordinary polynomial interpolation, the differing rates of convergence are easily explained by the difference in smoothness and analyticity [6]. That connection likely persists for $n > 1$, which corresponds to Hermite interpolation.

The interpolations discussed so far used the second barycentric form. Figure 4.3 compares the first barycentric form (1.3) to the second (1.4). Even with $K = 512$ grid points and $n = 48$ derivatives there is no sign of numerical instability at all. The first barycentric form is less accurate but has small errors for $z \in (-1, 1)$. At $z = \pm 1$, the first barycentric form produces much larger relative errors of about 10^{-7}. It is well known that the Chebyshev polynomials are bounded by 1 for $z \in [-1, 1]$ but increase rapidly outside the interval. That phenomenon is magnified by several orders for the fundamental polynomials of Hermite interpolation. Thus we should

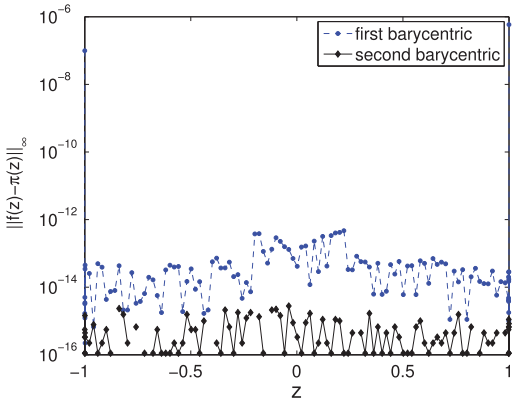

FIG. 4.3. *Interpolation errors to the Runge function* $f(z) = 1/(1+z^2)$ *with* $K = 512$ *grid points and* $n = 48$ *derivatives at each grid point. The first barycentric form is notably inaccurate at* $z = \pm 1$.

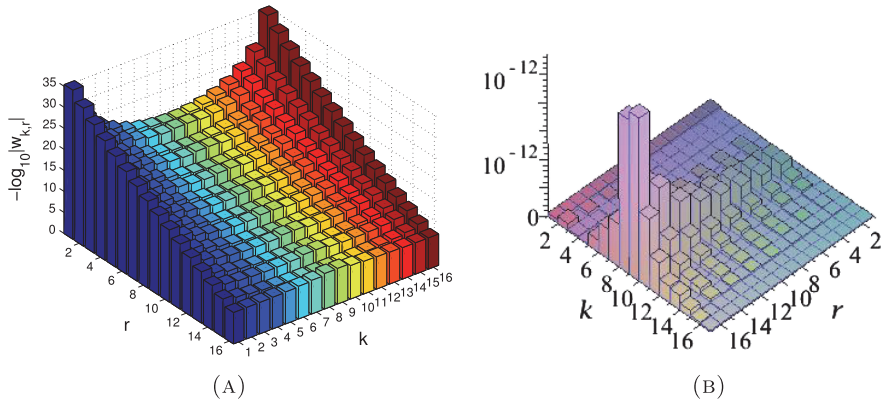

FIG. 4.4. (A) *Magnitude of the barycentric weights* $w_{k,r}$ *and* (B) *relative error in the weights. (The maximum relative error is* 2.86×10^{-12}*.) Both plots use* $K = 16$ *grid points and* $n = 16$ *derivatives at each grid point.*

expect the Hermite interpolant with $n = 48$ to be very sensitive near the endpoints of the interval.

Another phenomenon noticeable from Figure 4.3 is that the errors are somewhat elevated near the center of the interval. The second barycentric form exhibits neither elevated error near the middle of the interval nor sensitivity at the endpoints. It has excellent numerical stability.

In Figure 4.4, we directly examine the weights used for barycentric interpolation. These weights are computed using Algorithm 1. Figure 4.4 uses $K = 16$ and $n = 16$. The Chebyshev points z_k were multiplied by 2 but the display in the figure does not use Leja ordering. From part (A) of the figure, we see that the weights $w_{k,r}$ are quite small, especially when $r = 0$. Thus in spite of using an interval of capacity 1, the C_k that occur on line 32 of Algorithm 1 are getting quite small. Using an interval of capacity 1 still allows considerable fluctuations in $\prod_{k=1}^{K}(z - z_k)$ around 1 because it is only the $1/K$th power which is guaranteed to converge to 1. In computing the prefactor C_k, these fluctuations are raised to the power n with n being 16 in this instance.

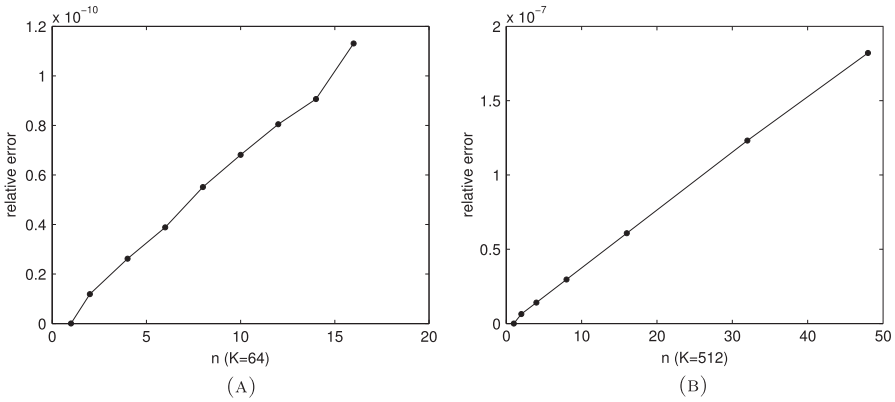

FIG. 4.5. *Maximum relative error in the barycentric weights $w_{k,r}$ as a function of the number of derivatives n at each grid point.*

From part (B) of Figure 4.4, we see that the relative errors in the weights $w_{k,r}$ are quite small. However, the relative errors are the highest for $r = n$ and z_k near the middle of the interval. These phenomena will be partially explained in the next section.

In both Figures 4.4 and 4.5, the relative errors in the barycentric weights were obtained by comparing double precision results of a C++ program with extended precision computations in Maple. Figure 4.5 shows the increase in the maximum relative error in the barycentric weights $w_{k,r}$ with n, which is the number of derivatives used at each grid point. The increase is mild. Indeed, in part (B) of the figure, which uses $K = 512$, the increase is only linear. Algorithm 1 for computing barycentric weights appears to have very good numerical stability.

5. Partial explanation of numerical stability. On line 32 of Algorithm 1, the weight $w_{k,r}$ is computed using $w_{k,r} = C_k \mathcal{I}_r$. The C_k are merely products of differences and will be computed with excellent relative accuracy. We will restrict ourselves to the error accumulation in the computation of $\mathcal{I}_r$.

We assume that $\{\alpha_1, \ldots, \alpha_N\}$ is a multiset of numbers and that $\mathcal{P}_r = \sum_{i=1}^{N} \alpha_i^{-r}$. The quantities $\mathcal{I}_r$ are defined by (2.4) and computed using $\mathcal{P}_r$ and the triangular identities of Lemma 2. For computing $w_{k,r}$, the multiset is taken to be $z_1 - z_k$ repeated n_1 times, $z_2 - z_k$ repeated n_2 times, and so on—see line 27 of Algorithm 1.

If u is the unit round-off ($u = 2^{-53}$ for IEEE double precision arithmetic), each arithmetic operation of the type $a + b$ will evaluate to $(a + b)(1 + \delta)$ with $|\delta| < u$ in floating point arithmetic. If $1 + \theta_n = \prod_{i=1}^{n}(1 + \delta_i)^{\rho_i}$, where $|\delta_i| \le u$ and $\rho_i = \pm 1$, we have $\theta_n \le \gamma_n$, where γ_n is defined as $nu/(1 - nu)$. (This is Lemma 3.1 of [8], for whose validity it is assumed that $nu < 1$.) This lemma is convenient for accumulating the effect of several rounding operations.

Another convenience is the counter notation [8, 15]. In this notation, $\langle n \rangle$ stands for a product of the form $\prod_{i=1}^{n}(1 + \delta_i)^{\rho_i}$, where $|\delta_i| \le u$ and $\rho_i = \pm 1$. Evidently, if $\langle n \rangle = 1 + \delta$, then $|\delta| \le \gamma_n$ (assuming $nu < 1$). In floating point arithmetic, $(a + b) + c$ evaluates to $a\,\langle 2 \rangle + b\,\langle 2 \rangle + c\,\langle 1 \rangle$.

If $\mathcal{P}_1 = 1/\alpha_1 + \cdots + 1/\alpha_N$ is computed in the order it is written, it evaluates to

$$1/\alpha_1\,\langle N \rangle + 1/\alpha_2\,\langle N \rangle + 1/\alpha_3\,\langle N - 1 \rangle \cdots + 1/\alpha_N\,\langle 2 \rangle\,.$$

If α_i is perturbed to $\alpha_i \langle N - i + 2 \rangle$, the computed result is the exact result. Thus the computation of $\mathcal{P}_1$ is backward stable. However, the computation of $\mathcal{P}_1$ can still have large relative errors if there are near cancellations. In Figure 4.5(A), the relative error in $\mathcal{P}_1$ (the maximum over all z_k which occurs for z_k near the middle of the interval) is 1.18×10^{-11}. In part (B), which corresponds to $K = 512$, the relative error in $\mathcal{P}_1$ is 6.42×10^{-9}. If there is a point at 0 and the positive and negative grid points are symmetric, the exact value of $\mathcal{P}_r$ is zero for r an odd number and the relative error is infinite. However, in this case, $\mathcal{P}_r$ will have small relative error for even r and the only power sums that influence the $\mathcal{I}_r$ correspond to even powers. If r is even, $\mathcal{P}_r$ is always computed with excellent relative accuracy as there can be no cancellations.

Since the error analysis of $\mathcal{P}_r$ is relatively tractable and in view of the comments made in the previous paragraph, we will temporarily assume all the power sums $\mathcal{P}_r$ to be computed with small relative errors. The more complex issue is the manner in which the errors are amplified when the quantities $\mathcal{I}_r$ are computed using the triangular identities of Lemma 2. Figure 4.5 shows that the errors grow only very mildly when going from $\mathcal{P}_r$ to $\mathcal{I}_r$.

To understand the error growth, we recast the triangular identities of Lemma 2 into matrix form:

$$\begin{pmatrix} 1 & & & & & \\ -\mathcal{P}_1 & 1 & & & & \\ -\mathcal{P}_2 & -\mathcal{P}_1 & 2 & & & \\ -\mathcal{P}_3 & -\mathcal{P}_2 & -\mathcal{P}_1 & 3 & & \\ \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} 1 \\ \mathcal{I}_1 \\ \mathcal{I}_2 \\ \mathcal{I}_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}. \tag{5.1}$$

If the condition number of the matrix in (5.1) is κ, the normwise relative error in the $\mathcal{I}_r$ is bounded by $\frac{2\epsilon\kappa}{1-\epsilon\kappa}$, where ϵ is of the order of u, the unit round-off. Unfortunately, this bound is useless because κ can be very large. If a single α_i is less than 1, which is always the case, the $\mathcal{P}_r$ will increase exponentially with r. The condition number of the system (5.1), retaining only $n+1$ rows in that system, will increase exponentially with n.

Although κ is quite bad, there is a simple way to get a matrix system equivalent to (5.1) with a small condition number. Suppose the α_i are all multiplied by a scaling factor s and changed to $s\alpha_i$. Then $\mathcal{P}_r$ and $\mathcal{I}_r$ are transformed to $P_r = s^{-r}\mathcal{P}_r$ and $I_r = s^{-r}\mathcal{I}_r$, respectively. If s is sufficiently large, then $|P_r| \le 1$ for $1 \le r \le n$.

The $(n+1) \times (n+1)$ matrix obtained by replacing the power sums $\mathcal{P}_r$ by their scaled version P_r has condition number bounded by $n+1$. In general, triangular systems have condition numbers that increase exponentially even if all the off-diagonal entries are bounded by 1 in magnitude [19]. Here the situation is different because the diagonal entries increase in the order $1, 1, 2, 3, \ldots$. We have the bound

$$|I_r| = \frac{|P_r + I_1 P_{r-1} + \cdots + I_{r-1} P_1|}{r} \le \frac{1 + |I_1| + \cdots + |I_{r-1}|}{r} \le 1,$$

where the last inequality is by induction. Thus all entries in the inverse will be bounded by 1.

If a large scaling factor s is used to control the condition number of the matrix, the difficulty comes in when we use $\mathcal{I}_n = s^n I_n$ to estimate the relative error in $\mathcal{I}_n$. The absolute errors in I_r can be bounded by a quantity of the form $n^{5/2}u$, or even nu if the scaling factor s is chosen to make the power sums small enough, where u is

the unit round-off. However, if I_n is much smaller than 1 in magnitude, the normwise bounds will imply a large relative error in I_n and in $\mathcal{I}_n$. This difficulty can be removed if a scaling factor s can be found such that $|P_r| \leq 1$ for $1 \leq r \leq n$ and $|I_n|$ is $\mathcal{O}(1)$. The linear growth in relative errors in Figure 4.5 suggests that such a scale might exist. Proving the existence of such a scale will require estimates of the quantities $\mathcal{P}_r$ and $\mathcal{I}_r$.

The discussion in this section partially backs up the finding of Figure 4.5 that the growth of relative errors in the barycentric weights with the order of the derivative is mild. The contribution to the errors made by $\mathcal{P}_r$ is therefore quite significant. Even though the computation of each power sum $\mathcal{P}_r$ is backward stable, the sum itself could be ill-conditioned. In fact, by looking up $n = 1$ in Figure 4.5, we see that the relative error in $\mathcal{P}_1$ is not so small. We implemented compensated summation in x86 assembly language and verified that it does not reduce forward errors.

The following proposition allows us to verify the accuracy of some of the barycentric weights.

PROPOSITION 3. *If the α_i are all positive, the relative error in $\mathcal{P}_r$ is bounded by γ_{r+N} and the relative errors in $\mathcal{I}_r$ are bounded by γ_{3rN} for sufficiently small unit round-off and $r \leq N$.*

Proof. The computed quantity $\mathcal{P}_r$ can be represented as

$$\alpha_1^{-r} \langle N + r \rangle + \alpha_2^{-r} \langle N + r - 1 \rangle + \cdots + \alpha_N^{-r} \langle r + 1 \rangle.$$

Because each α_i is positive, the computed quantity can be represented as $\mathcal{P}_r \langle N + r \rangle$, thus proving one half of the proposition.

For the other half, we note that $\mathcal{I}_r = (\mathcal{P}_r + \mathcal{I}_1 \mathcal{P}_{r-1} + \cdots + \mathcal{I}_{r-1} \mathcal{P}_1)/r$. By induction, the quantity computed for $\mathcal{I}_r$ can be represented as

$$\frac{\mathcal{P}_r \langle N + r \rangle \langle r \rangle + \cdots + \mathcal{I}_{r-1} \mathcal{P}_1 \langle 3(r-1)N \rangle \langle N + 1 \rangle \langle r \rangle}{r}.$$

In this expression the $\langle r \rangle$ factors that accompany each term in the numerator are caused by the arithmetic operations for computing $\mathcal{I}_r$ using $\mathcal{I}_s$ for $1 \leq s \leq r - 1$. The factors that accompany each term in the numerator can be multiplied and replaced by $\langle 3Nr \rangle$. Since each term is positive, it follows that the computed quantity can be represented as $\mathcal{I}_r \langle 3Nr \rangle$. ∎

This proposition implies that the weights $w_{k,r}$ are computed with excellent accuracy when z_k is either the least or the greatest of the grid points. Indeed, we see from Figure 4.4(B) that the relative errors near the endpoints of the interval are very small. If z_k is the least of the grid points, each z'_j , $1 \leq j \leq K - 1$, that occurs on line 27 of Algorithm 1 is negative and nonzero. Since $-1/z'_j$ corresponds to α_j in Proposition 3, the proposition is immediately applicable. If z_k is the greatest of the grid points, each $-1/z'_j$ is negative and the proposition is not immediately applicable. However, it is evident from inspection that $\mathcal{P}_r$ and $\mathcal{I}_r$ are negative or positive according as r is odd or even. Therefore every sum in the triangular identities of Lemma 2 has terms that are all positive or all negative. Thus the bounds on relative errors proved in the proposition will apply.

We have candidly highlighted the inadequacy, in situations not covered by Proposition 3, of the bounds that can be derived on the rounding errors in the computed barycentric weights. Thus the title of this section only claims a partial explanation. Existing results in rounding error analysis are known to us give bounds which are highly pessimistic.

Numerical stability of barycentric Lagrange interpolation has been studied by Higham [9]. It may be suspected that the methods of that paper can be used for the numerical stability analysis of barycentric Hermite interpolation. However, that is far from being the case. The definition of the condition number used by Higham perturbs the function values but not the grid points. Such a definition is useful in the context of Lagrange interpolation because the Lagrange weights are obtained from products of the type $\prod_{j\neq k}(z_j - z_k)$. The basic rules of double precision arithmetic imply right away that each Lagrange weight is computed, with a very small relative forward error which can be easily bounded. In the case of Hermite interpolation, the computation of the weights is a great deal more complicated, as we have emphasized. In particular, the computation involves back substitution or the inversion of the triangular system displayed explicitly in (5.1).

If good bounds on the forward errors of the barycentric weights $w_{k,r}$ can be obtained, the rounding error analysis simplifies greatly. The analysis of the first barycentric form (1.3) closely parallels the rounding error analysis of inner products of the type $\sum_{i=1}^{n} a_i b_i$. For the second barycentric form, one needs to tackle expressions of the form $A_1/A_2 - A_3/A_4$, where each A_i has the structure of an inner product. While this form is slightly more complicated algebraically, it poses none of the difficulties associated with triangular inversion or back substitution.

Higham's treatise [8] on numerical algorithms discusses componentwise analysis of linear systems in Chapter 7 and triangular inversion in Chapter 8. We now explain why none of the results in those two chapters help explain the excellent numerical stability of barycentric Hermite interpolation illustrated in Figure 4.5. In the componentwise bound of Higham's Theorem 7.4, it is crucial to note the $|A^{-1}|$ that appears in the numerator. Viswanath and Trefethen have shown [19] that the inverses of triangular matrices generically have norms that are exponential in the dimension. Therefore such bounds are highly pessimistic for the situation illustrated in Figure 4.5. As we described in detail in the arguments leading up to Proposition 3, an attempt at mitigating the norm of the inverse by scaling the grid points is ultimately of no value. Exactly the same comments apply to Higham's Lemma 7.9. The difficulty that one faces is more explicit in Higham's Lemma 8.6, Theorem 8.7, and Theorem 8.14. In each of those results, the exponential dependence of the worst-case bounds on the dimension of the matrix system is explicit. If those bounds are applied to (5.1), the resulting bounds on the errors in the barycentric weights, corresponding to the two plots in Figure 4.3, would be off by factors of 2^{64} and 2^{512}, respectively.

The worst-case bounds on rounding errors are exponential in the dimension of the linear system. The results of Viswanath and Trefethen [19] show that the same is true generically, thus suggesting strongly that the worst-case bounds will be difficult to improve. It is noteworthy that Higham [8] begins the chapter on triangular systems by quoting authorities who realized decades ago that the bounds that can be derived on the error are usually pessimistic, while in practice many triangular systems can be solved quite accurately.

We conclude by recommending the second barycentric form (1.4) with weights computed using the method of Butcher et al. [3], as presented in Algorithm 1, as the standard algorithm for Hermite interpolation. The reasons for this recommendation are the same as those given by Berrut and Trefethen [2] for barycentric Lagrange interpolation, namely, conceptual simplicity, excellent numerical stability, and flexibility in incorporating new data points. To those reasons, we may add the finding of section 2 that barycentric Hermite interpolation has a lower operation count than Hermite interpolation using divided differences.

Acknowledgments. We thank Prof. Sergey Fomin for references that led us to attribute Lemma 2 to Crocchi [5, 1880]. We thank the referees for useful suggestions. In particular, one of the referees told us about the ℓ macro in TeX and urged us to track down the reference to Crocchi, thus improving this paper.

REFERENCES

[1] J.-P. Berrut, R. Baltensperger, and H. Mittelmann, *Recent developments in barycentric rational interpolation*, in Trends and Applications in Constructive Approximation, M. G. de Bruin, D. H. Mache, and J. Szabados, eds., Internat. Ser. Numer. Math. 151, Birkhäuser, Basel, 2005, pp. 27–51.
[2] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Rev., 46 (2004), pp. 501–517.
[3] J. C. Butcher, R. M. Corless, L. Gonzalez-Vega, and A. Shakoori, *Polynomial algebra for Birkhoff interpolants*, Numer. Algorithms, 56 (2011), pp. 319–347.
[4] S. D. Conte and C. de Boor, *Elementary Numerical Analysis*, McGraw-Hill, New York, 1980.
[5] L. Crocchi, *Una relazione fra le funzioni simmetriche semplici e le funzioni simmetriche complete*, Giornale di Matematiche, 18 (1880), pp. 377–380.
[6] P. J. Davis, *Interpolation and Approximation*, Dover Publications, New York, 1975.
[7] B. Fischer and L. Reichel, *Newton interpolation in Fejér and Chebyshev points*, Math. Comp., 53 (1989), pp. 265–278.
[8] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphian, 2002.
[9] N. J. Higham, *The numerical stability of barycentric Lagrange interpolation*, IMA J. Numer. Anal., 24 (2004), pp. 547–556.
[10] L. M. Milne-Thomson, *The Calculus of Finite Differences*, Macmillan, New York, 1933.
[11] L. Reichel, *Newton interpolation at Leja points*, BIT, 30 (1990), pp. 332–346.
[12] B. Sadiq and D. Viswanath, *Finite Difference Weights Using the Modified Lagrange Interpolant*, preprint, arXiv:1102.3203, 2011.
[13] C. Schneider and W. Werner, *Hermite interpolation: The barycentric approach*, Computing, 46 (1991), pp. 35–51.
[14] R. P. Stanley, *Enumerative Combinatorics*, vol. 2, Cambridge University Press, Cambridge, UK, 1999.
[15] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.
[16] J. Szabados, *On the order of magnitude of fundamental polynomials of Hermite interpolation*, Acta Math. Hungar., 61 (1993), pp. 357–368.
[17] H. Tal-Ezer, *High degree polynomial interpolation in Newton form*, SIAM J. Sci. Statisti. Comput., 12 (1991), pp. 648–667.
[18] K. T. Vahlen, *Rationale funktionen der wurzeln; symmetrische und affektfunktionen*, Encyklopädie der Mathematischen Wissenschaften, 1 (1904), pp. 449–479.
[19] D. Viswanath and L. N. Trefethen, *Condition numbers of random triangular matrices*, SIAM J. Sci. Statist. Comput., 19 (1998), pp. 564–581.
[20] W. Werner, *Polynomial interpolation: Lagrange versus Newton*, Math. Comp., 43 (1984), pp. 205–217.