MATHEMATICS OF COMPUTATION
VOLUME 43, NUMBER 168
OCTOBER 1984, PAGES 501-528

Accurate Computation of Divided Differences of the Exponential Function

By A. McCurdy, K. C. Ng and B. N. Parlett[1]

Abstract. The traditional recurrence for the computation of exponential divided differences, along with a new method based on the properties of the exponential function, are studied in detail in this paper. Our results show that it is possible to combine these two methods to compute exponential divided differences accurately. A hybrid algorithm is presented for which our error bound grows quite slowly with the order of the divided difference.

Introduction. We need accurate divided differences for computing certain functions of matrices $f(A)$ by means of the Newton interpolating polynomial (cf. Section 6):

$$f(A) = \Delta_1^0 f \cdot I + \sum_{k=1}^{n-1} \Delta_1^k f \cdot \prod_{j=1}^{k} (A - x_j I),$$

where Δ_1^k stand for the divided differences of f on the eigenvalues of A. One can evaluate $f(A)$ by computing first the divided differences and then accumulating the polynomial. The divided differences must be of high relative accuracy because they are the coefficients of products of matrices which, in some cases, have very large norms. What makes such accuracy possible is that the divided differences are not for arbitrary smooth functions f but for well-known analytic functions such as exp, sin and cos. Thus we can exploit their properties in the computation.

In this paper we restrict our attention to exponential divided differences. A new technique, namely argument reduction for matrix exponentials, makes it realistic to consider data sets with imaginary parts bounded by π in magnitude. Based on this an algorithm is presented for which our error bound grows quite slowly with the order of the divided difference.

We begin by collecting together a considerable amount of information on divided differences and we hope that there will be other applications for accurate divided differences of well-known functions.

1. Basic Notation and Theorems.

1.1. *Definition of Divided Difference.* Following McCurdy [7], we will use an uncommon but compact notation for divided difference. For completeness and simplicity we use the contour integral representation to define the divided differences. Our attention will be on the basic properties (1.2.1), (1.2.2) and (1.2.3) given in Subsection 1.2.

Received September 14, 1983; revised January 30, 1984.

1980 *Mathematics Subject Classification.* Primary 39-04; Secondary 33A10, 65D20.

[1] The authors gratefully acknowledge support by the Office of Naval Research Contract N00014-76-C-0013.

Let f be a holomorphic function defined inside and on a simple closed contour C enclosing the sequence $Z = [\zeta_1, \zeta_2, \ldots, \zeta_n, \ldots]$ of complex numbers. Z denotes the abscissae (or, for those who do not like Latin, data points or nodes, or even knots). We use $\Delta^k_i f$ to denote the kth order divided difference of f on $\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k}$. For any integer $i > 0$, the kth order divided difference $\Delta^k_i f$ on Z is defined (following Gel'fand) to be

$$(1.1.1) \qquad \Delta^k_i f = \Delta^k_i(Z)f \equiv \frac{1}{2\pi i}\int_C \frac{f(\omega)\,d\omega}{(\omega - \zeta_i)(\omega - \zeta_{i+1})\cdots(\omega - \zeta_{i+k})}.$$

The superscript of $\Delta^k_i f$ denotes the order and the subscript denotes the starting point in Z. Reference to the abscissae Z is usually suppressed.

Remark 1. An alternative, and more elementary definition (used in Conte and de Boor [2, cf. p. 40] designates $\Delta^k_i f$ as the coefficient of x^k in the unique polynomial of minimal degree which interpolates f at $\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k}$.

Remark 2. Milne-Thomson [13] writes $\Delta^k_i f$ as $[\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k}]$, suppressing the function while de Boor considers $[\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k}]$ as a linear functional whose value on f is written $[\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k}]f$. Davis [3] uses $f^{[k]}(\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k})$; some others like Atkinson [1] use $f[\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k}]$ while Kahan and Farkas [6] and Gabel [4] use $\Delta f(\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k})$, which suggested the compact notation used here. Much of this introductory section is taken from the thesis of McCurdy [7].

1.2. *Basic Properties of Divided Differences.* Let $f^{(k)}$ denote the kth derivative of f. From basic complex analysis one can deduce from (1.1.1) that

(1.2.1) $\Delta^k_i f$ does not depend on the order of $\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k}$ in Z,

(1.2.2) if $\zeta_i \neq \zeta_{i+k}$, then $\Delta^k_i f = (\Delta^{k-1}_{i+1} f - \Delta^{k-1}_i f)/(\zeta_{i+k} - \zeta_i)$,

(1.2.3) if $\zeta_i = \zeta_{i+1} = \cdots = \zeta_{i+k}$, then $\Delta^k_i f = f^{(k)}(\zeta_i)/k!$, in particular $\Delta^0_i f = f(\zeta_i)$.

Most definitions for divided difference are based on (1.2.1), (1.2.2) and (1.2.3). Thus our definition agrees with them when the function is holomorphic. In this paper f will be holomorphic.

1.3. *Integral Representation.*

THEOREM (HERMITE-GENOCCHI).

$$(1.3.1) \quad \Delta^k_i f = \int_0^1 \int_0^{\nu_1} \cdots \int_0^{\nu_{k-1}} f^{(k)}[\zeta_i + (\zeta_{i+1} - \zeta_i)\nu_1 + \cdots + (\zeta_{i+k} - \zeta_{i+k-1})\nu_k]\, d\nu_k \cdots d\nu_2\, d\nu_1.$$

Proof. See Gel'fand [5].

COROLLARY.

$$(1.3.2) \qquad |\Delta^k_i f| \leqslant \frac{1}{k!}\max_{\xi \in \overline{\Omega}} |f^{(k)}(\xi)|,$$

where $\overline{\Omega}$ is the convex hull of $\zeta_i, \ldots, \zeta_{i+k}$.

1.4. *Mean Value Representation.* For real abscissae, (1.3.1) implies that there exists some $\eta \in \overline{\Omega}$ such that

$$(1.4.1) \qquad \Delta^k_i f = \frac{1}{k!} f^{(k)}(\eta).$$

One might hope to generalize this representation for complex abscissae by requiring η to lie in the convex hull of the abscissae, but this will not suffice, as is easily seen by the following example:

Example I. $\zeta_1 = 1, \zeta_2 = 2, f(\zeta) \equiv \exp(2\pi i \zeta)$,

$$\Delta^1_1 f = \frac{e^{4\pi i} - e^{2\pi i}}{2-1} = 0 \neq f^{(1)}(\eta) \tag{1.4.2}$$

for any finite η.

In the above example, if we require both abscissae to lie in f*'s fundamental domain* $\{\zeta\colon \mathrm{Re}(\zeta) \in [\frac{-1}{2}, \frac{1}{2})\}$ (note that $f(\zeta + n) = f(\zeta)$ for any integer n), then the best we can have is that there is some η close to their convex hull for which (1.4.1) holds. The next example illustrates this property.

Example II. $\zeta_1 = t$, $\zeta_2 = -t$, t is a small nonzero real number,

$$\Delta^1_1 f = \frac{e^{2\pi i t} - e^{-2\pi i t}}{2t} = i\frac{\sin(2\pi t)}{t} \neq f^{(1)}(\eta) \tag{1.4.3}$$

for any *real* η.

1.5. *Matrix Representation*. The traditional way of computing $\Delta^k_i f$ uses the divided difference table. Each divided difference is computed from its two immediate neighbors in the column to its left (use (1.2.3) for coincident abscissae and (1.2.2) for the rest).

$$\begin{array}{ccccc}
\zeta_1 & f(\zeta_1) & & & \\
 & & \Delta^1_1 f & & \\
\zeta_2 & f(\zeta_2) & & \cdot & \\
 & & \Delta^1_2 f & \cdot & \cdot \\
\cdot & \cdot & \vdots & & \Delta^{n-1}_1 f \\
\cdot & \cdot & \cdot & & \\
\cdot & \cdot & & & \\
 & & \Delta^1_{n-1} f & & \\
\zeta_n & f(\zeta_n) & & &
\end{array}$$

For our purposes it is more helpful to arrange the table as an upper triangular matrix, for example

$$\Delta f \equiv \begin{bmatrix} f(\zeta_1) & \Delta^1_1 f & \cdots & \Delta^{n-1}_1 f \\ & f(\zeta_2) & \cdots & \Delta^{n-2}_2 f \\ & & \ddots & \vdots \\ & & & f(\zeta_n) \end{bmatrix}. \tag{1.5.1}$$

The symbol $\Delta f = \Delta(Z)f$, without the superscript and subscript, is used here to represent a matrix, not a scalar. Let Z_n be the special $n \times n$ bidiagonal matrix associated with the ordered set Z

$$Z_n \equiv \begin{bmatrix} \zeta_1 & 1 & & \\ & \zeta_2 & 1 & \\ & & \ddots & \ddots \\ & & & 1 \\ & & & \zeta_n \end{bmatrix}. \tag{1.5.2}$$

THEOREM (OPITZ). *The divided difference table is a matrix function*

$$\Delta f = f(Z_n). \tag{1.5.3}$$

Proof. See McCurdy [7] or Opitz [10].

Remark. Opitz [10] first obtained the result but his paper is little known in the U.S.A. and is in German. McCurdy rediscovered it in 1979 when working on his thesis.

1.6. *Our Objective.* Given any $Z = [\zeta_1, \zeta_2, \dots, \zeta_n]$, can we compute $\Delta_1^k \exp$ for $k = 0, 1, \dots, n-1$ with guaranteed high relative accuracy? Using the matrix representation, it is equivalent to ask "Can we compute the first row of $\Delta\exp$, or $\exp(Z_n)$, accurately?" The answer is affirmative if the abscissae are close to the real line.

In the next two sections, we discuss some basic and hybrid methods for computing $\Delta\exp$. In Section 4 we give the results of McCurdy [7] for real abscissae Z, which show that one can compute $\Delta\exp$ accurately in all circumstances. We turn to the complex case in Section 5 and show that in certain cases the problem is "difficult" (to be precise, certain sets Z give unexpectedly small values for $\Delta_i^k \exp$, and we call them "difficult"). For difficult Z, we cannot expect high relative accuracy; the situation is like approximating zero by some nonzero number. Finally, in Section 6, we discuss the application of the divided differences to matrix exponentials.

2. Basic Methods for Computing Exponential Divided Differences.

2.1. *Standard Recurrence.* When all ζ's in Z are distinct, we can use the well-known recurrence scheme (1.2.2) to compute the divided differences table Δf: $\Delta_i^k f$ for $i > 0$, $k \geqslant 0$ and $k + i \leqslant n$.

SR (Standard Recurrence scheme).[2]

$$\Delta_i^k f = \frac{\Delta_{i+1}^{k-1} f - \Delta_i^{k-1} f}{\zeta_{i+k} - \zeta_i} \tag{2.1.1}$$

for each $k = 1, 2, \dots, n$ and $i = 1, 2, \dots, n-k$, where $\Delta_i^0 f \equiv f(\zeta_i)$. □

SR is probably the simplest algorithm. It takes only $n^2/2 + O(n)$ arithmetic operations to fill up the whole of Δf when all data in Z are distinct. However, when some $f(\zeta_i)$ are close together and given to limited precision, it may produce enormous relative error. For example, consider the exponential function on data $[1, 1.0001]$. Assume function values given to 8 decimal digits, then

$$\Delta_1^1 \exp = \frac{2.7185537 - 2.7182818}{1.0001 - 1} = 2.7190000 \; (\mathit{Ans.}\, 2.718417747\dots).$$

Four digits have been lost during the subtraction (which is performed exactly!). Notice that the loss does not depend on the number of digits carried by the function values. The first four digits of the function values agree, therefore four digits will be lost no matter how many digits are given. Since the higher order differences of exp behave like exp (because the derivative of exp is exp), we would expect $\Delta_1^n \exp$ to

[2] Parlett's Recurrence for computing $f(Z_n)$ (Parlett [11]) is identical to the standard iterative scheme for computing Δf. The technique is based on the commutativity of Z_n and $f(Z_n)$: $Z_n \cdot f(Z_n) = f(Z_n) \cdot Z_n$, cf. Parlett [11].

lose $4n$ digits if the data are as close together as in the example. Consequently, when only 12 or 16 decimals are available it is quite possible to lose them all for higher divided differences!

If the tabular values are the only data then there is no simple escape from this loss of information. That is why divided differences have a bad name in practice. However, in a number of applications the functional form of f is known (e.g. exp) and can be exploited to obtain accurate values in this situation. This is the essential point of our paper.

We shall suppress the reference to exp or Z in the exponential divided differences when it can be done without ambiguity. Thus Δ^k_i, $\Delta^k_i(Z)$ and $\Delta^k_i\exp$ may all mean $\Delta^k_i(Z)\exp$.

2.2. *Special Formula for The First Divided Difference.* If the sine function for *complex* arguments is available and fully accurate then we have a reliable formula for the first divided difference. Let $\omega \equiv (\zeta_{i+1} + \zeta_i)/2$ and $\psi \equiv (\zeta_{i+1} - \zeta_i)/2$, then

$$\Delta^1_i = \frac{e^{\zeta_{i+1}} - e^{\zeta_i}}{\zeta_{i+1} - \zeta_i} = e^{\omega} \cdot \frac{e^{\psi} - e^{-\psi}}{\psi - (-\psi)} = e^{\omega} \cdot \frac{\sinh(\psi)}{\psi} = e^{\omega} \cdot \frac{\sin(i\psi)}{i\psi}.$$

If $\psi = 0$, we set $\Delta^1_i = e^{\zeta_i}$.

Function FDD(x, y) (First Divided Difference). Given complex data x, y, FDD will return the value of $\Delta^1_1([x, y])$.

```
1. $\omega = (y + x)/2$,
2. $\psi = \omega - x$,
3. if $\psi = 0$ then FDD $= e^x$,
4. if $\psi \neq 0$ then FDD $= e^{\omega} \cdot \sin(i\psi)/(i\psi)$.
5. Return. □
```

2.3. *Taylor Series.* Another simple way to compute Δ is by its Taylor series

$$\Delta \equiv \Delta\exp = \exp(Z_n) = I + Z_n + Z_n^2/2! + \cdots.$$

Because of the special structure of Z_n, there is an extremely elegant algorithm for the first row of the matrix Δ. Explanation is given in Appendix A of [8]. This approach does not apply when f is known only by its values on Z.

Algorithm TS (Taylor Series). Given Z as in Subsection 2.1, this algorithm computes $[d(1), d(2), \ldots, d(n)] := [\Delta^0_1, \ldots, \Delta^{n-1}_1]$ by Taylor series. In what follows, k indicates the current loop number, and $s(i)$ stores the $(1, i)$th element of matrix $(Z_n)^{k+i-1}/(k+i-1)!$

```
TS 1. [Initialize]. Set $d(i) = s(i) = 1/(i-1)!$ for $i = 1, 2, \ldots, n$.
TS 2. [Loop.] For $k = 1, 2, \ldots$ until convergence do
      TS 2.1  s(1) $\leftarrow \zeta_1 \cdot s(1)/k$
      TS 2.2  For $i = 2, 3, \ldots, n$ do
                  $s(i) \leftarrow [\zeta_i \cdot s(i) + s(i-1)]/(k+i-1)$,
                  $d(i) \leftarrow d(i) + s(i)$.
TS 3. Set $d(1) = \exp(\zeta_1)$ and the algorithm terminates. □
```

Algorithm TS computes only the first row of Δ. If one wants the whole divided differences table, one has to use the following TS(II), which essentially computes the whole Δ by repeating TS on the submatrices in Z_n.

Algorithm TS(II) (Taylor Series Algorithm (II)). Given Z and a matrix array F, this algorithm computes $F = \Delta$ by Taylor series. In what follows, k indicates the current loop number, and $F(i, m)$, $i > m$, stores the (i, m)th element of matrix $(Z_n)^{k+i-m}/(k + i - m)!$.

```
TS (II) 1. [Initialize.] Set $F(i, m) = F(m, i) = 1/(i - m)!$ for $1 \leqslant m \leqslant i \leqslant n$.
TS (II) 2. [Loop.] For $k = 1, 2, \ldots$ until convergence do
        TS (II) 2.1. For $m = 1, 2, \ldots, n - 1$ do
                $F(m, m) \leftarrow \zeta_m \cdot F(m, m)/k$,
                for $i = m + 1, \ldots, n$ do
                    $F(i, m) \leftarrow [\zeta_i \cdot F(i, m) + F(i - 1, m)]/(k + i - m)$,
                    $F(m, i) \leftarrow F(m, i) + F(i, m)$.
TS (II) 3. For $m = 1, 2, \ldots, n$ set $F(m, m) = \exp(\zeta_m)$ and restore zero to the
lower parts of $F$, i.e., $F(i, m) \leftarrow 0$ for $0 < m < i$, and the algorithm terminates. □
```

Accuracy. TS method is fast and accurate only when all ζ_i are close to zero. Let $\gamma \equiv \max_{\zeta \in Z}|\zeta_i|$ and call it the "radius" of Z. Numerical examples show that when the radius is bigger than 2 or 3, TS may not be reliable. The situation is like computing $e^{-\gamma}$ by its Taylor series, i.e., by $1 - \gamma + \gamma^2/2! + \cdots$. In finite precision arithmetic, when γ is large, $e^{-\gamma}$ is small and the roundoff error from the intermediate term $\gamma^k/k!$ (which is large) could impair the accuracy of the series. If one wants the roundoff of the intermediate terms to have no serious effect on $e^{-\gamma}$, say, confined to the last binary digit of $e^{-\gamma}$, then γ must be small enough so that $e^{-\gamma} \geqslant 2^{-1} \cdot \max_k(\gamma^k/k!)$, which implies $\gamma \leqslant \ln 2 \approx 0.7$. It seems reasonable to require $\gamma < 0.7$ if one wants TS to yield accurate answers.

Criterion. Use TS when γ is less than 0.7. This criterion will be used throughout our paper, for we need TS to yield accurate answers in the Scaling and Squaring method in Subsection 2.4. One may relax the constant 0.7 a little bit but we will stick to this value. Our examples (cf. Table 2.3.3) show that the error grows rapidly with γ and it becomes unbearable when γ is bigger than 2 or 3.

Remark. The number of terms l needed in the series depends on the radius γ and the machine precison ε. In Appendix A of [8] we show that in the presence of roundoff it is sufficient to choose l such that

$$(2.3.1) \qquad \sum_{j=l+1}^{\infty} \frac{\gamma^j}{j!} \leqslant \varepsilon.$$

For example, if $\gamma = 0.7$, then for $\varepsilon = 2^{-24}$, $l = 9$ and for $\varepsilon = 2^{-56}$, $l = 16$.

Operation Count and Storage. The operation count is $2\,ln$ for TS and ln^2 for TS(II), where l is the number of terms needed. Two working n-vectors are required for storing d and s in TS while a whole matrix is needed in TS(II).

Numerical Example. Let $u = \cos(0.1) + i\sin(0.1)$. We ran TS on a Vax 11/780[3] using single precision ($\varepsilon = 2^{-24}$) on the set

$$(2.3.2) \qquad Z =^4 [-\gamma u, -\gamma u, \ldots, -\gamma u, \gamma u, \ldots, \gamma u]$$

[3] Vax is a trademark of the Digital Equipment Corporation.

[4] To be precise, if Z has n points, then the first $[(n + 1)/2]$ are $-\gamma u$ and the others are γu.

TABLE (2.3.3)

Max relative error coefficient in Z (*cf.* 2.3.2) *with different* n *and* γ.

	$n = 5$	$n = 11$	$n = 17$	$n = 23$	$n = 29$
$\gamma = 0.7$	1.3	2.3	1.9	4.0	2.7
$\gamma = 1.2$	1.9	4.9	8.8	8.8	8.8
$\gamma = 1.7$	3.8	7.9	14.5	10.6	30.7
$\gamma = 2.2$	38.7	38.7	38.7	38.7	38.7
$\gamma = 2.7$	39.0	59.7	122.0	122.0	122.0
$\gamma = 3.2$	26.7	96.1	133.0	159.0	159.0
$\gamma = 3.7$	291.0	321.0	321.0	484.0	484.0
$\gamma = 4.2$	704.0	1820.0	1820.0	1970.0	1970.0
$\gamma = 4.7$	2250.0	2250.0	2300.0	2300.0	2980.0
$\gamma = 5.2$	2980.0	5530.0	8240.0	8240.0	8240.0

with different values of γ and n, where n is the number of points in Z. Here γ is also the radius of the data because $|u| = 1$. Our results are summarized in the above table. Each entry in Table (2.3.3) is the maximum magnitude of the relative errors in $\Delta(Z)$ as a multiple of ε.[5] Note the rapid growth of the error as γ increases.

2.4. *Scaling and Squaring*. Scaling and squaring is a general technique for computing the matrix exponential (see Ward [12]). It makes use of the functional properties of the exponential function. Since the divided difference table can be regarded as the exponential of Z_n, we can apply this technique to Z_n in order to compute Δ.

2.4.1. *SS* (*Scaling and Squaring*) *Method*. When the abscissae are not close enough for TS, we can shift and scale down the size of Z_n by, for example, setting

$$Y_n = \frac{1}{2^k}(Z_n - \eta I),$$

where k and η are chosen so that Y_n has small diagonal elements. Since exp has the following properties:

(i) $\exp(A + xI) = e^x \cdot \exp(A)$,

(ii) $\exp(A/2^k)^{2^k} = \exp(A)$,

we can recover $\exp(Z_n)$ from $F = \exp(Y_n)$ by $\exp(Z_n) = e^\eta \cdot [\exp(Y_n)]^{2^k}$. The matrix power F^{2^k} can be computed by repeated squaring of F (i.e., $F \leftarrow F^2$) k times.

Four Major Steps for SS:

Step 1. Determine[6] η and k so that $Y_n = (Z_n - \eta I)/2^k$ has radius $\leqslant 0.7$.[7]

Step 2. Compute $F = \exp(Y_n)$ by Taylor series.

Step 3. ($F \leftarrow F^2$) k times.

Step 4. Shift back F: $F \leftarrow e^\eta \cdot F$.

The squaring in Step 3 normally requires $kn^3/6 + n^2/2 + n/3$ operations (F is triangular) and a matrix storage for F; this is quite expensive when n is large.

[5] Thus the number 8240 corresponding to $n = 29$ and $\gamma = 5.2$ means that the maximum relative error in $\Delta(Z)$ is 8240ε.

[6] We usually use the arithmetic mean of the data as the shift.

[7] The number 0.7 comes from the criterion in Subsection 2.3. It is proved to be almost the best for SS in McCurdy [7] when Z is real.

However, there is an alternative method which requires only $kn^2 + O(1)$ operations: with some modification of Steps 2 and 3, one can replace every "intermediate" F by some divided differences table. (Notice that in Step 2

$$Y_n = \begin{bmatrix} 2^{-k}\zeta_1 & 2^{-k} & \\ & \cdots & 2^{-k} \\ & & 2^{-k}\zeta_n \end{bmatrix}$$

and does not generate a divided differences table.) Consequently with the backfilling technique in Subsection 2.4.2, one can generate the whole matrix F from its first row and therefore only the first row is needed in the squaring, thus reducing the operations and storage required. This method does sacrifice some accuracy, however. Before presenting the algorithms (in Subsection 2.4.4), we describe the backfilling technique and discuss a subtle modification of Steps 2 and 3. In general we cannot avoid using a 2-dimensional array to form F^2 unless F has some special structure.

2.4.2. *Backfilling the Divided Difference Table.* Consider again the divided difference table Δf

$$\Delta f \equiv \begin{bmatrix} f(\zeta_1) & \Delta_1^1 f & \cdots & \Delta_1^{n-1} f \\ & f(\zeta_2) & \cdots & \Delta_2^{n-2} f \\ & & \ddots & \vdots \\ & & & f(\zeta_n) \end{bmatrix}.$$

Algorithm SR shows that Δf can be generated from its diagonal elements. However, it is also true that Δf can be generated from its first row by the formula (1.2.2): given $\Delta_1^0, \Delta_1^1, \dots, \Delta_1^{n-1}$,

$$\Delta_i^k f = (\zeta_{i+k} - \zeta_{i-1}) \cdot \Delta_{i-1}^{k+1} f + \Delta_{i-1}^k f \tag{2.4.2.1}$$

for $i = 2, 3, \dots, n$ and $k = 0, 1, 2, \dots, n - i$.

The only worry in using formula (2.4.2.1) is the propagation of the error in $\Delta_i^k f$, which may be serious, especially when the ζ's are far apart. When $f = \exp$ and Z is real and in natural order ($\zeta_i < \zeta_j$ for $i < j$), (2.4.2.1) is reliable because all $\Delta_i^k \exp$ and $(\zeta_{i+k} - \zeta_{i-1})$ are positive, and summing positive numbers is quite stable. Thus bad situations occur only when Z has large variation in the imaginary parts. In this case the backfilling step frequently exhibits instability. The following is a typical example.

Numerical Example. Let $Z = [-24i, -21i, -18i, \dots, 18i, 21i, 24i]$. We compute the last column of $\Delta(Z)$: Δ_k^{n-k} for $k = 1, 2, \dots, 18$ by backfilling and compare it with the correct answer (in Table 2.4.2.2). The last column of the table denotes the magnitude of the relative errors in the corresponding divided difference Δ_k^{n-k}.

Table (2.4.2.2)

Backfilling yields enormous error for Z with a large variation in the imaginary parts.

k	Correct values to six digits		Backfilling		rel. error
1	(.699024e − 16	.000000e + 00)	(.699024e − 16	.000000e + 00)	.21e − 07
2	(.118971e − 15	.167766e − 14)	(.118971e − 15	.167766e − 14)	.11e − 07
3	(−.375574e − 13	.535368e − 14)	(−.375574e − 13	.535368e − 14)	.21e − 07
4	(−.168358e − 12	−.780734e − 12)	(−.168358e − 12	−780734e − 12)	.15e − 06
5	(.149915e − 10	−.436262e − 11)	(.149915e − 10	−.436262e − 11)	.52e − 06
6	(.976633e − 10	.264278e − 09)	(.976634e − 10	.264279*e* − 09)	.20e − 05
7	(−.424631e − 08	.192067e − 08)	(−.424635e − 08	.192068e − 08)	.72e − 05
8	(−.333270e − 07	−.616516e − 07)	(−.333270e − 07	−.616534e − 07)	.26e − 04
9	(.800392e − 06	−.508937e − 06)	(.800473e − 06	−.508930e − 06)	.86e − 04
10	(.678836e − 05	.917160e − 05)	(.678798e − 05	.917475e − 05)	.28e − 03
11	(−.912472e − 04	.781070e − 04)	(−.913509e − 04	.780928e − 04)	.87e − 03
12	(−.761200e − 03	−.771374e − 03)	(−.760814e − 03	−.774224e − 03)	.27e − 02
13	(.538045e − 02	−.611926e − 02)	(.544411e − 02	−.611179e − 02)	.79e − 02
14	(.390049e − 01	.296790e − 01)	(.389112e − 01	.307910e − 01)	.23e − 01
15	(−.121109	.184993)	(−.135378	.184486)	.65e − 01
16	(−580745	−.323969)	(−.584346	−.443728)	.18

2.4.3. *Modification of Step* 2 *and Step* 3. We may assume the data have been shifted to have mean 0. For $0 \leqslant i \leqslant k$, define the bidiagonal matrix $Z_n^{(i)}$ to be

$$Z_n^{(i)} \equiv \begin{bmatrix} 2^{-i}\zeta_1 & 1 & & \\ & 2^{-i}\zeta_2 & 1 & \\ & & \ddots & 1 \\ & & & 2^{-i}\zeta_n \end{bmatrix}.$$

Also let the diagonal matrix R be

$$R = \begin{bmatrix} 1 & & & & \\ & 2 & & & \\ & & 2^2 & & \\ & & & \ddots & \\ & & & & 2^{n-1} \end{bmatrix}.$$

Our objective here is to replace every intermediate "F" in Steps 2 and 3 by $\exp(Z_n^{(i)})$, so that we can apply the backfilling technique and avoid the storage for a whole matrix.

Modified Step 2. Compute $F_0 = \exp(Z_n^{(k)})$ by TS.

Modified Step 3. Compute $F_i = RF_{i-1}^2R^{-1}$ for $i = 1, 2, \ldots, k$.

Lemma. *$F_i = \exp(Z_n^{(k-i)})$ for $0 \leqslant i \leqslant k$, in particular, $F_k = \exp(Z_n^{(0)}) = \exp(Z_n)$.*

Proof. Assume $F_l = \exp(Z_n^{(k-l)})$ for some $l \geqslant 0$, then

$$\begin{aligned} F_{l+1} &= RF_l^2R^{-1} = R\left[\exp(Z_n^{(k-l)})\right]^2R^{-1} = R \cdot \exp\left(2Z_n^{(k-l)}\right) \cdot R^{-1} \\ &= \exp\left(2RZ_n^{(k-l)}R^{-1}\right). \end{aligned}$$

From the definition, it may be verified that $Z_n^{(j)} = 2RZ_n^{(j+1)}R^{-1}$ for $j \geqslant 0$. Hence, $F_{l+1} = \exp(Z_n^{(k-l-1)})$. The lemma holds when $l = 0$. By induction, we have $F_i = \exp(Z_n^{(k-i)})$ for $i \geqslant 0$. □

Since every intermediate "F" is of form $\exp(Z_n^{(i)})$, each of them is a divided difference table (with different scaled abscissae). By the previous section, F can be generated from its first row. Hence it is possible to do the squaring (for the first row) without keeping the whole matrix.

2.4.4. *Algorithm for SS.*

Algorithm SS (Scaling and Squaring). Given Z as in Subsection 2.2, this algorithm computes $[d(1),\dots,d(n)] := [\Delta_1^0, \Delta_1^1, \dots, \Delta_1^{n-1}]$ by scaling and squaring. In what follows, vector s stores the current column of F and vector r stores the first row of the current F.

```
SS1. [$n = 1$?] If $n = 1$, return $d(1) = e^{\zeta_1}$ and the algorithm terminates.
SS2. [Shifting.] Set $\eta = (\sum \zeta_i)/n$ and replace $\zeta_i$ by $\zeta_i - \eta$.
SS3. [Scaling.] Determine the least integer $k \geqslant 0$ such that $2^{-k}\max_i|\zeta_i| \leqslant 0.7$, then
     replace $\zeta_i$ by $2^{-k}\zeta_i$ for all $i$.
SS4. [TS.] Call TS with $Z$ equal to the current $\zeta_i$'s, result goes to $d$.
SS5. [Squaring.] For $kk = 1, 2, \dots, k$ do
     SS5.1 Set $s(1) = d(1)$, for $i = 2, 3, \dots, n$ do
           SS5.1.1 [Backfill the $i$th column of $F$ in $s$.]
                   $x = s(1)$
                   $s(1) = d(i)$
                   For $j = 2, \dots, i - 1$ do
                       $y = s(j)$
                       $s(j) = x + (\zeta_i - \zeta_{j-1})s(j - 1)$
                       $x = y$
                       next $j$
                   $s(i) = \exp(\zeta_i)$.
           SS5.1.2 [form the $(1, i)$th element of $RF^2R^{-1}$.]
                   $r(i) = 2^{-(i-1)}\sum_{j=1}^{i} d(j)s(j)$.

     SS5.2. [Update $d$ and $\zeta_i$'s.]
            $d(i) \leftarrow r(i)$ for $i = 2, \dots, n$,
            $\zeta_i \leftarrow 2\zeta_i$ for $i = 1, 2, \dots, n$,
            $d(1) = \exp(\zeta_1)$.

SS6. [Backfill the last column of $F$.] Set $s(1) = d(1)$, for $i = 2, 3, \dots, n$ do
     $x = s(1)$
     $s(1) = d(i)$
     For $j = 2, \dots, i - 1$ do
         $y = s(j)$
         $s(j) = x + (\zeta_i - \zeta_{j-1})s(j - 1)$
         $x = j$
         next $j$
     next $i$.
```

```
SS7. [Shift back and stop.]
        $\zeta_i \leftarrow \zeta_i + \eta$, $d(i) \leftarrow e^{\eta} \cdot d(i)$, $s(i-1) \leftarrow e^{\eta} \cdot s(i-1)$ for $i = 2,\ldots,n$, set $d(1) = \exp(\zeta_1)$ and $s(n) = \exp(\zeta_n)$ and the algorithm terminates. □
```

Remarks. (1) If the function FDD (cf. Subsection 2.2) for the first divided difference is available, one can improve the accuracy of SS by using FDD whenever the first divided difference is wanted.

(2) SS6 is necessary for the Simple Hybrid Algorithm in the next section, ortherwise it is not needed.

The backfilling step may not always be reliable: when Z has a large variation in its imaginary parts it is likely that formula (2.4.2.1) will magnify inherited errors. In that case straightforward squaring is needed. Here is the algorithm.

Algorithm SS(II) (Scaling and Squaring Algorithm (II)). Given Z and matrix F, this algorithm computes $F = \Delta$ by scaling and squaring. For the R in step 5.1, cf. Subsection 2.4.3.

```
SS(II) 1. [$n = 1$?] If $n = 1$, return $F(1,1) = e^{\zeta_1}$ and the algorithm terminates.
SS(II) 2. [Shifting.] Set $\eta = (\Sigma\, \zeta_i)/n$ and replace $Z$ by $Z - \eta$.
SS(II) 3. [Scaling.] Determine the least integer $k \geqslant 0$ such that $2^{-k}\max_i|\zeta_i| \leqslant 0.7$,
          then replace $Z$ by $2^{-k}Z$.
SS(II) 4. [TS(II).] Call TS(II) with data $Z$, result goes to $F$.
SS(II) 5. [Squaring.] For $kk = 1,2,\ldots,k$ do
              SS(II) 5.1. [Update $F$.] $F = R \cdot F^2 \cdot R^{-1}$.
              SS(II) 5.2. [Update $\zeta_i$'s.] $Z \leftarrow 2 \cdot Z$.
              SS(II) 5.3. [Update $F(i,i)$.] $F(i,i) = \exp(\zeta_i)$ for $i = 1,2,\ldots,n$.
SS(II)6.  [Shift back and stop.] $Z \leftarrow Z + \eta$, $F \leftarrow e^{\eta} \cdot F$ and the algorithm
          terminates. □
```

Operation Count and Storage. The major part of this computation is the squaring step, which is repeated k times. The operation count for each squaring is $n^2 + O(1)$ in SS5 and $n^3/6 - n^2/2 + O(n)$ in SS(II) 5 (with n function call on exp). Hence the total operations needed are $\approx kn^2$ in SS and $\approx kn^3/6$ in SS(II), where k is the least nonnegative integer such that $2^{-k}\gamma \leqslant 0.7$.[8] Therefore when $\gamma > 0.7$, $k = [\log_2(\gamma/0.7) + 1] \approx \log_2 \gamma + 1.5$.[9] Four n-vectors are needed for storing d, s, r and Z in SS while a whole matrix is needed in SS(II).

Accuracy. Both SS and SS(II) may be viewed as extensions of TS (Taylor Series). They can accept moderately spread data without suffering as much as TS (cf. Subsection 4.3). The choice between SS and SS(II) is discussed in Subsection 5.4. The following example illustrates the big difference between TS and SS.

Numerical Example. Let $Z = [-16,-12,-8,-4,0,4,8,12,16]$. We compare TS and SS in the computation of $\Delta_1^k(Z)$ for $k = 1,2,\ldots,8$. Results are summarized in Table (2.4.4.1). The values in the last two columns are the magnitude of the relative error in the corresponding divided differences; notice the enormous error in the first few $\Delta_1^k(Z)$ for TS.

[8] Here $\gamma = \gamma(Z) = \max_i|\zeta_i - (\Sigma_i\, \zeta_i)/n|$ is the "radius" of Z (after it has been shifted).

[9] Here $[x]$ denotes the greatest integer less than x.

TABLE (2.4.4.1)

Divided differences on Z, TS *vs* SS.

	correct values to 6 digits	TS op: 645 (÷ or *)	SS op: 1018	relative error (SS)	relative error (TS)
Δ^1_1	.150792e − 05	.281912e − 01	.150792e − 05	.12e − 06	.4
Δ^2_1	.101027e − 04	.281155e − 03	.101027e − 04	.20e − 06	.83
Δ^3_1	.451239e − 04	.242131e − 03	.451239e − 04	.36e − 06	.37
Δ^4_1	.151160e − 03	.154376e − 03	.151160e − 03	.60e − 06	.21e − 01
Δ^5_1	.405094e − 03	.405302e − 03	.405095e − 03	.88e − 06	.51e − 03
Δ^6_1	.904679e − 03	.904669e − 03	.904680e − 03	.12e − 05	.11e − 04
Δ^7_1	.173175e − 02	.173175e − 02	.173176e − 02	.14e − 05	.48e − 05
Δ^8_1	.290059e − 02	.290059e − 02	.290059e − 02	.17e − 05	.73e − 06

3. Hybrid Methods.

3.1. *Example.* Our discussion so far suggests that it may be possible to compute Δ accurately by combining the two methods (SR and SS) of Section 2. Let us consider the following task:

"Given $Z = [50i, 10^{-5} + 50i, -10^{-5} - 50i, -50i]$, compute $\Delta = \Delta\exp$."

In addition to SR and SS, we can compute Δ by the following "mixed" method. Decompose Δ into a 2×2 block matrix and name the blocks I, II and III,

$$\Delta = \left[\begin{array}{cc|cc} \Delta^0_1 & \Delta^1_1 & \Delta^2_1 & \Delta^3_1 \\ & \Delta^0_2 & \Delta^1_2 & \Delta^2_2 \\ \hline & & \Delta^0_3 & \Delta^1_3 \\ & & & \Delta^0_4 \end{array}\right] = \left[\begin{array}{c|c} \mathrm{I} & \mathrm{III} \\ \hline & \mathrm{II} \end{array}\right].$$

Since ζ_1 and ζ_2 are close together (also ζ_3 and ζ_4), SS is right for them and we use SS to compute I and II. Then we use SR to fill up III.

In order to compare this mixed approach with SS and SR, we ran these three algorithms in 24-binary digit (~ 7 decimal) arithmetic. The results are summarized in the following table. For simplicity we only compare Δ^2_1 and Δ^3_1. The symbol μ in the last column stands for the multiplication or division; thus 6μ, 4 exp means six multiplication/divisions and four calls to exp are needed.

Method	Δ^2_1	Δ^3_1	Op. count
SR	(−.262260e − 02 −.970843e − 02)	(−.193573e − 03 −.558794e − 10)	6μ, 4 exp
SS	(−.262376e − 02 −.970219e − 02)	(−.194077e − 03 −.295204e − 07)	196μ, 10 exp
Mixed	(−.262376e − 02 −.970218e − 02)	(−.194043e − 03 .207219e − 09)	26μ, 10 exp
Exact	(−.262376d − 02 −.970218d − 02)	(−.194043d − 03 .204162d − 09)	

The following should be noticed:

(1) SR gives poor results on Δ^2_1 and Δ^3_1.

(2) The answers of SS are not bad. This shows that SS can indeed accept moderately spread data, but the price is high.

(3) The mixed method gives the most accurate answer.

3.2. *Simple Hybrid Method.* The example in Subsection 3.1 shows that when one can group the data into clusters (allow overlap)

$$Z = [\underbrace{\zeta_1, \zeta_2, \ldots, \zeta_i, \zeta_k}_{\text{I}}, \ldots, \zeta_l, \underbrace{\zeta_{l+1}, \ldots, \zeta_n}_{\text{III}}], \qquad \underbrace{\zeta_i, \zeta_k, \ldots, \zeta_l}_{\text{II}}$$

then one can compute $\Delta(Z)$ by

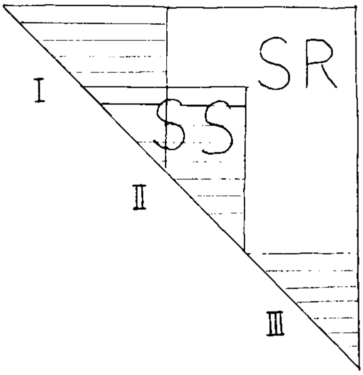


FIGURE 3.2.1

This clustering should satisfy

(1) within each diagonal block of Z_n the data are close enough together so that SS may be used for the corresponding block in Δ,

(2) data belonging to different blocks should be sufficiently separated so that SR can be used to fill up the rest of Δ.

This mixed approach, which we call the *simple hybrid method* (SH), demands a suitable ordering on the data Z. Such an ordering brings together all close abscissae and we may call it a nested ordering (to be defined precisely in Subsection 3.3). Under a nested ordering, the radius[10] of each $[\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k}]$ is close to the distance between the endpoints. In other words, if ζ_i and ζ_{i+k} are close together, then all ζ_i, $\zeta_{i+1}, \ldots, \zeta_{i+k}$ are close together. In that case, we can group the abscissae as follows. The data ζ_i, $\zeta_{i+1}, \ldots, \zeta_{i+k}$ will be in the same cluster if $|\zeta_{i+k} - \zeta_i|$ is less than some value g. This g depends on k (the number of points in the data set) only and we will discuss the value of $g \equiv g_k$ for each k in Subsection 4.4. For the time being, assume g_k is given; we are ready to describe the simple hybrid method.

Method SH.

[1] Determine the clustering.

[2] Compute the clustered block (shaded area of Figure 3.2.1) by SS. Notice that we only need SS to return the first row and the last column of each block.

[3] Fill up the rest to the first row by SR.

[10] The *radius* of Z is defined to be $\gamma(Z) \equiv \max_{1 \leqslant i \leqslant n} |\zeta_i - \eta|$ where $\eta = (\sum_i \zeta_i)/n$.

In practice, [1], [2], and [3] are always combined for each cluster. Here is an implementation.

Algorithm SH (Simple Hybrid Algorithm). Given Z and the decision function G, this algorithm computes $[d(1),\dots,d(n)] := [\Delta_1^0, \Delta_1^1, \dots, \Delta_1^{n-1}]$ by the simple hybrid method. In what follows, vector s will store the last column of the current cluster; vector d will store the first row of the current cluster; μ will be the currently computed row number (of Δ); and ν, j will be the first and last index of the next cluster.

```
SH1. [$n = 1$?] If yes, set $d(1) = \exp(\zeta_1)$ and the algorithm terminates.
SH2. [Initialize.] Set $\mu = \min_{i \leqslant n}\{i\colon |\zeta_i - \zeta_n| \leqslant g_{n-i}\}$ and compute the $\mu$th row of
     $\Delta$ by calling SS; result goes in $d(\mu),\dots,d(n)$. Set $j = n$.
SH3. [$\mu = 1$?] If yes, the algorithm terminates.
SH4. [Loop.]
     SH4.1. [Find the next cluster.] Find cluster $[\nu, j]$, $\nu \leqslant j$.
        (a) $j = j - 1$
        (b) $\nu = \min\{i\colon |\zeta_i - \zeta_j| \leqslant g_{j-i}$ with $i < j\}$
        (c) if $\mu \leqslant \nu$ then go back to (a) else SH4.2.
     SH4.2. [Update $d$ from $\nu$ to $j$.]
        SH4.2.1. [call SS on $[\zeta_\nu,\dots,\zeta_j]$.]
           Results go to $d(\nu),\dots,d(j)$ and $s(1),\dots,s(j - \nu + 1)$
           $s$ is the last column of the cluster.
        SH4.2.2. [Fill up $d(j + 1),\dots,d(n)$ by SR.]
           For $k = \mu - \nu, \mu - \nu - 1,\dots,1$ do
                $d(j) = s(k)$
                for $i = j + 1, j + 2,\dots,n$ do
                      $d(i) = [d(i) - d(i - 1)]/[\zeta_i - \zeta_{\nu+k-1}]$,
                next $i$
           next $k$
SH5. [Update $\mu$.] Set $\mu = \nu$ and $j = j - 1$. Go back to SH3. □
```

Operation Count and Storage. The total number of operations depends on the clustering. The worst case might take $O(n^3)$ but it would be very rare, e.g., if $Z = [1, 2, 3,\dots, 2n]$ and $g_j = n$ for any j, then there will be exactly n clusters and each cluster has n data points, which means $n \cdot O(n^2) = O(n^3)$ operations are needed (cf. Figure 3.2.2). Such a situation is very unlikely to happen for a realistic set of g_k, $k = 1, 2, \dots$. For our decision constants (which will be discussed later), the operator count is usually $O(n^2)$. Storage requirements will be the same as SS.

3.3. *Ordering Problem.* When Z is not nested, one may not be able to group the data to have properties (1) and (2) in Subsection 3.2. In that case, a much more sophisticated combination of SS and SR, a *recursive hybrid method*, may be needed. Let us consider a different example $Z = [-50, 50, 50, -50]$. Since the first and the last elements are equal, we cannot use SR for Δ_1^3 and hence the whole of Z should be treated as one block. But then SS is not that suitable because the radius of Z is large. However, instead of the whole Z, we consider the subset $[-50, 50, 50]$ (which can be grouped into two clusters) and obtain the first three divided differences Δ_1^0, Δ_1^1, Δ_1^2.

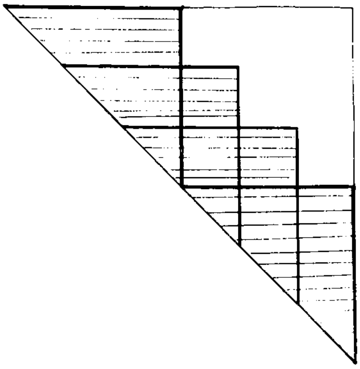


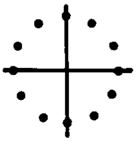
FIGURE 3.2.2

As for the last one, we make use of the fact that it does not depend on the ordering of Z, and thus compute Δ_1^3 by considering the reordered data set $[-50, -50, 50, 50]$. Notice that both $[-50, 50, 50]$ and $[-50, -50, 50, 50]$ can be clustered for SH.

The disadvantage of the above method is that in some sense the first three divided differences have been computed twice. Had we known *in advance* that the reordering would be necessary, we could have avoided the repetition; *for in our application the abscissae ζ_i can be arranged in any order to give a Z but then it is $\Delta(Z)$ which must be computed.* It thus raises the question:

Does there exist a nested ordering for any given Z?

The answer is yes when Z is real (the natural increasing ordering) but not always in general, e.g., consider data that form a circle in the complex plane.



Data that form a circle in the complex plane cannot be nested.

Before we discuss the details of the recursive hybrid methods, we mention the **decision function** G and the decision constants g_k, $k = 1, 2, \ldots$. Given any abscissae W with k points, $G(W)$ yields a pair of points (ω_i, ω_j), ω_i, $\omega_j \in W$ such that $|\omega_i - \omega_j|$ is an approximation of the radius of W. As in Subsection 3.2, the decision whether we should apply SS on the whole of W becomes the test $|\omega_i - \omega_j| \leqslant g_k$, where g_k depends on k. Examples for $G(W) = (\omega_\mu, \omega_\nu)$, ω_μ, $\omega_\nu \in W$ are

(3.3.1) $|\omega_\mu - \omega_\nu| = \operatorname{diam}(W)$.

(3.3.2) $\operatorname{Re}(\omega_\mu - \omega_\nu) = \operatorname{diam}(\operatorname{Re}(W))$.

(3.3.3) $|\omega_\mu - \omega_\nu| = \max_{\omega_i \in \Lambda; \omega_j \in W} |\omega_i - \omega_j|$, $\Lambda \equiv \{\omega_i \in W\colon \operatorname{Re}(\omega_i) = \max_j \operatorname{Re}(\omega_j)\}$.

We will discuss G in Sections 4 and 5. Now assume that G is given and use it to define a nested ordering:

Definition 3.3.4. Z is nested (with respect to G) if

$$G([\zeta_i, \zeta_{i+1}, \ldots, \zeta_{i+k}]) = (\zeta_{i+k}, \zeta_i) \quad \text{for any } 1 \leqslant i, 1 \leqslant k, i + k \leqslant n. \square$$

It is easy to verify that if G is one of (3.3.1)–(3.3.3), and if Z is real, then an arrangement of ζ_i in increasing order gives a nested ordering.

3.4. *Recursive Hybrid Method.* Every divided difference can be computed by $\Delta_i^k = \mathrm{RH}([\zeta_i, \dots, \zeta_{k+1}])$, where the function RH is defined below.

Recursive Function RH(Z). This function computes the highest order divided difference on the given data Z. Let k denote the number of points in Z, then RH return $\Delta_1^{k-1}(Z)\exp$.

[1] If $k = 1$ return $(\exp(\zeta_1))$.

[2] Compute $G(Z) = (\zeta_\mu, \zeta_\nu)$.

[3] If $|\zeta_\mu - \zeta_\nu| \leqslant g_k$ call SS and return $(d(k))$ else return the following

$$\frac{\mathrm{RH}(Z_{(\mu)}) - \mathrm{RH}(Z_{(\nu)})}{\zeta_\mu - \zeta_\nu}, \tag{3.4.1}$$

where $Z_{(i)} \equiv [\zeta_1, \zeta_2, \dots, \zeta_{i-1}, \zeta_{i+1}, \dots, \zeta_n]$. □

We leave the details of the proof that RH does return the highest divided difference to the reader. Notice that when Z is nested, $G([\zeta_i, \dots, \zeta_{i+k}]) = (\zeta_{i+k}, \zeta_i)$ and the above decision (step [3]) means that $\Delta_1^k[\zeta_i, \dots, \zeta_{k+i}]$ should be computed by SS if $|\zeta_{i+k} - \zeta_i| \leqslant g_k$, which is exactly what SH did. Thus

RH reduces to SH if the abscissae are nested. □

Since the operation count of RH could be enormous, like $O(2^n)$, one would hope to find a nested ordering for the ζ_i's to determine Z and then apply SH on it. A practical modification is to attempt to nest the abscissae (according to G) before steps [2] and [3]. If it can be done, then SH can be applied to the rearranged Z (recall that the divided difference does not depend on the ordering of the data). Later on we will see that the abscissae can always be taken close to real (cf. Subsection 5.3) and consequently ordering according to the real part gives an almost nested ordering, see Subsection 5.4.

Our purpose in introducing RH is to show that, in principle, $\Delta_i^k(Z)\exp$ can be computed accurately using fixed precision arithmetic.

4. Real Exponential Divided Differences. Exponential divided differences for real abscissae are positive and increasing functions of their abscissae. These properties permit derivation of bounds on the error growth in SR (Standard Recurrence) and SS (Scaling and Squaring). For future use, we consider the more general function $\exp_\tau$ with scaling parameter τ, that is $\exp_\tau(\xi) \equiv e^{\tau\xi}$. For simplicity, we write

$$\exp_\tau^{(n)}(\xi) = \frac{d^n}{d\xi^n}(\exp_\tau)(\xi).$$

In the rest of this section, we consider exclusively divided differences on real abscissae $X = [\xi_1, \xi_2, \dots, \xi_n]$, even if some of the properties hold for general complex abscissae.

4.1. *Basic Theorems and Properties.*

Translation and Scaling Invariance Property. Let U be the constant vector $[1, 1, \dots, 1]$. Then for any constants τ, α,

$$\Delta_1^{n-1}(X + \alpha U)\exp_\tau = e^{\tau\alpha} \cdot \Delta_1^{n-1}(X)\exp_\tau, \tag{4.1.1}$$

and

$$\Delta_1^{n-1}(X)\exp_\tau = \tau^{n-1}\Delta_1^{n-1}(\tau X)\exp. \tag{4.1.2}$$

Proof. (4.1.1) follows easily from the matrix equation $\exp(A + \alpha I) = e^{\alpha} \cdot \exp(A)$ (using (1.5.3)), and (4.1.2) follows from (1.3.1) directly.

Recursive Integral Formula. For given X and any $\tau \geqslant 0$, $i = 1, 2, \ldots, n$, we have

$$\Delta_1^{n-1} \exp_\tau = e^{\tau\xi_i} \int_0^\tau e^{-\sigma\xi_i} \cdot \Delta_{(i)}^{n-2} \exp_\sigma d\sigma, \tag{4.1.3}$$

where

$$\Delta_{(i)}^{n-2} f \equiv \Delta^{n-2}(X_{(i)}) f, \qquad X_{(i)} = X \backslash \{\xi_i\}. \tag{4.1.4}$$

Proof. From the Hermite-Genocchi integral representation formula (1.3.1), we have

$$\begin{aligned}\Delta_1^{n-1} \exp_\tau &= \int_0^1 \int_0^{\nu_1} \cdots \int_0^{\nu_{n-2}} \\ &\qquad \exp_\tau^{(n-1)}[\xi_1 + (\xi_2 - \xi_1)\nu_1 + \cdots + (\xi_n - \xi_{n-1})\nu_{n-1}]\, d\nu_{n-1} \cdots d\nu_1 \\ &= \int_0^1 \int_0^{\nu_1} \cdots \int_0^{\nu_{n-2}} \\ &\qquad \tau^{n-1} \exp[\tau\xi_1 + (\xi_2 - \xi_1)\tau\nu_1 + \cdots + (\xi_n - \xi_{n-1})\tau\nu_{n-1}]\, d\nu_{n-1} \cdots d\nu_1\end{aligned}$$

by the definition of $\exp_\tau$. The change of variables $\sigma_j = \tau\nu_j$ for $j = 1, 2, \ldots, n-1$ yields the alternative expression

$$\begin{aligned}\Delta_1^{n-1} \exp_\tau &= \int_0^\tau \int_0^{\sigma_1} \cdots \int_0^{\sigma_{n-2}} \\ &\qquad \exp[\tau\xi_1 + (\xi_2 - \xi_1)\sigma_1 + \cdots + (\xi_n - \xi_{n-1})\sigma_{n-1}]\, d\sigma_{n-1} \cdots d\sigma.\end{aligned} \tag{4.1.5}$$

We recognize that this is a recurrence for $\Delta_1^{n-1} \exp_\tau$, namely

$$\Delta_1^{n-1} \exp_\tau = e^{\tau\xi_1} \int_0^\tau e^{-\sigma\xi_1} \cdot \Delta_2^{n-2} \exp_\sigma d\sigma$$

where $\sigma \equiv \sigma_1$. By the symmetry property (1.2.1), the ordering of the abscissae is arbitrary; we may replace ξ_1 by any ξ_i, $1 \leqslant i \leqslant n$, hence establishing the formula. □

THEOREM 1. *For all $\tau > 0$ and $k \geqslant 0$, $\Delta_1^k \exp_\tau$ is*

(i) *positive.*

(ii) *strictly increasing in each abscissa ξ_i, for $i = 1, \ldots, n$.*

Proof. (i) follows from the mean value representation (1.4.1). For (ii),

$$\frac{\partial}{\partial\xi_i} \Delta_1^{n-1} \exp_\tau = \int_0^\tau (\tau - \sigma) e^{(\tau-\sigma)\xi_i} \cdot \Delta_{(i)}^{n-2} \exp_\sigma d\sigma > 0,$$

since the integrand is positive.

THEOREM 2. *Suppose $\beta \leqslant \xi_i \leqslant \gamma$ for each abscissa ξ_i, $1 \leqslant i \leqslant n$. Then for each i there exists a $\xi \in [\beta, \gamma]$ such that*

$$\Delta_{(i)}^{n-2} \exp_\tau = \left(\xi - \xi_i + \frac{n-1}{\tau}\right) \cdot \Delta_1^{n-1} \exp_\tau. \tag{4.1.6}$$

Proof. By (4.1.1) and (4.1.2),

$$\begin{aligned}\Delta_1^{n-1}(\tau(X - \xi U)) \exp &= \tau^{-(n-1)} e^{-\tau\xi} \cdot \Delta_1^{n-1}(X) \exp_\tau \\ &= \tau^{-(n-1)} e^{\tau(\xi_i - \xi)} \int_0^\tau e^{-\sigma\xi_i} \cdot \Delta_{(i)}^{n-2}(X) \exp_\sigma d\sigma\end{aligned}$$

for any $i = 1, 2, \ldots, n$ and ξ. Differentiating with respect to τ yields

$$\text{(4.1.7)}\quad \begin{aligned} &\frac{d}{d\tau}\Delta_1^{n-1}(\tau(X - \xi U))\exp \\ &\quad = \tau^{-(n-1)}e^{-\tau\xi}\left[\left(\xi_i - \xi - \frac{n-1}{\tau}\right)\cdot \Delta_1^{n-1}(X)\exp_\tau + \Delta_{(i)}^{n-2}(X)\exp_\tau\right]. \end{aligned}$$

Every element of the vector $X - \beta U$ is nonnegative, and so $\Delta_1^{n-1}(\tau(X - \beta U))\exp$ is increasing in τ. Similarly, every element of $X - \gamma U$ is nonpositive and $\Delta_1^{n-1}(\tau(X - \gamma U))\exp$ is decreasing in τ. Hence

$$\frac{d}{d\tau}\Delta_1^{n-1}(\tau(X - \beta U))\exp \geqslant 0 \geqslant \frac{d}{d\tau}\Delta_1^{n-1}(\tau(X - \gamma U))\exp$$

so for some $\xi \in [\beta, \gamma]$, the derivative is zero. The result then follows from (4.1.7). □

COROLLARY 1. *Lower bound on $\Delta_1^{n-1}\exp_\tau$. If $\xi_n \geqslant \xi_i$ for each $i = 1, 2, \ldots, n$, then*

$$\text{(4.1.8)}\qquad \Delta_1^{n-1}\exp_\tau \geqslant \frac{\tau}{n-1}\cdot \Delta_1^{n-2}\exp_\tau.$$

Proof. Choose $i = n, \gamma = \xi_n$ in (4.1.6), and note that $\xi - \xi_n \leqslant 0$. □

COROLLARY 2. *Upper bound on $\Delta_1^{n-1}\exp_\tau$. If $\xi_1 \leqslant \xi_i$ for each $i = 1, 2, \ldots, n$, then*

$$\text{(4.1.9)}\qquad \Delta_1^{n-1}\exp_\tau \leqslant \frac{\tau}{n-1}\cdot \Delta_2^{n-2}\exp_\tau. \quad □$$

4.2. *Error Growth in Standard Recurrence.* We now examine the error growth of one step of SR when X is in increasing order. Equation (4.1.8) leads directly to a bound on the relative error growth in one step of SR. Let ε_i^k be the relative error in $\Delta_i^k = \Delta_i^k\exp$, i.e., $fl(\Delta_i^k) = (1 + \varepsilon_i^k)\cdot\Delta_i^k$, where $fl(\Delta_i^k)$ is computed by SR. For simplicity, let us first assume that the recurrence step (2.1.1) is done exactly, in which case ε_i^k may be regarded as *the inherited uncertainty of* Δ_i^k. We have

$$fl(\Delta_i^k) = \frac{fl(\Delta_{i+1}^{k-1}) - fl(\Delta_i^{k-1})}{\xi_{i+k} - \xi_i} = \frac{(1 + \varepsilon_{i+1}^{k-1})\Delta_{i+1}^{k-1} - (1 + \varepsilon_i^{k-1})\Delta_i^{k-1}}{\xi_{i+k} - \xi_i}.$$

After some algebraic manipulation, one obtains

$$|fl(\Delta_i^k) - \Delta_i^k| \leqslant \left[\Delta_i^k + \frac{2}{\xi_{i+k} - \xi_i}\Delta_i^{k-1}\right]\cdot\max\left\{|\varepsilon_{i+1}^{k-1}|, |\varepsilon_i^{k-1}|\right\}.$$

By (4.1.8), since $\xi_{i+k} \geqslant \xi_j$ for $i \leqslant j \leqslant i + k$.

$$|\varepsilon_i^k| \equiv \frac{|fl(\Delta_i^k) - \Delta_i^k|}{\Delta_i^k} \leqslant \left[1 + \frac{2k}{\xi_{i+k} - \xi_i}\right]\cdot\max\left\{|\varepsilon_{i+1}^{k-1}|, |\varepsilon_i^{k-1}|\right\}.$$

Therefore, we have

Uncertainty growth[11] of one step of SR (with X an increasing order)

$$\text{(4.2.1)}\qquad |\varepsilon_i^k| \leqslant \left[1 + \frac{2k}{\xi_{i+k} - \xi_i}\right]\cdot\max\left\{|\varepsilon_{i+1}^{k-1}|, |\varepsilon_i^{k-1}|\right\}. \,□$$

This bound is quite realistic. Take the example in Subsection 2.1: $Z = [1, 1.0001]$. Both $\Delta_1^0 = e^1$ and $\Delta_2^0 = e^{1.0001}$ can be computed accurately with $|\varepsilon_1^0|, |\varepsilon_2^0| \leqslant \varepsilon$, so Eq.

[11] This is the growth of the uncertainties in the data. As long as there are uncertainties in the input, SR will propagate them even if the arithmetic of each step is done exactly.

(4.2.1) predicts $|\varepsilon_1^1| \leqslant 20001\varepsilon$. In Subsection 2.1, with $\varepsilon = 5 \cdot 10^{-8}$, we have $\varepsilon_1^1 = (2.719 - 2.7184 \cdots)/(2.718 \cdots) \approx 0.0002142 \approx 4284\varepsilon$.

In finite arithmetic, the execution of SR may introduce some roundoff error to Δ_i^k. An error analysis in Appendix B of [8] shows that only a small modification of (4.2.1) is needed to incorporate the effects of roundoff into the propagation of uncertainty.

Error Growth of One Step of SR. Provided that ε_i^{k-1} and ε_{i+1}^{k-1} are small,[12] we have

$$(4.2.2) \qquad |\varepsilon_i^k| \leqslant 4\varepsilon + \left[1 + \frac{2k}{\xi_{i+k} - \xi_i}\right] \cdot \max\left\{|\varepsilon_{i+1}^{k-1}|, |\varepsilon_i^{k-1}|\right\}. \square$$

Proof. See Appendix B of [8]. □

4.3. *Error Bounds on SS*. Based on the positivity of Δ_i^k (Theorem 1 in 4.1), we can apply standard error analysis to obtain relative error bounds on SS. For example, in the squaring step, each entry of the matrix is positive and therefore no cancellation occurs and we have for each F

$$|fl(F^2) - F^2| \leqslant n\varepsilon \cdot (F^2),$$

where $|E|$ denotes the matrix all of whose elements are the absolute values of the elements of E and our notation $A \leqslant B$ means that $a_{i,j} \leqslant b_{i,j}$ for every i and j.

A detailed error analysis of Algorithm SS(II) is presented in Appendix B of [8]. As a direct corollary of Eq. (B.6) in [8], we have the following bound:

Scaling and Squaring Error Bounds. Given real abscissae X in increasing order, denote the relative error of $\Delta^j{}_i(X)$ by ε_i^j as in the previous section, and recall $\gamma \equiv \max_i |\xi_i - \eta|$ where η is the arithmetic mean of ξ_i. For convenience set $\gamma' \equiv \max(\gamma, 0.7)$. We have

$$(4.3.1) \qquad |\varepsilon_i^k| \leqslant (C_0 k + C_1 k \ln \gamma' + C_2 \gamma' - 2) \cdot \varepsilon,$$

where $C_0 = 13.68$ and $C_1 = 1.4427$. C_2 depends on ε; in particular, $C_2 = 134.4$ when $\varepsilon = 2^{-24}$, and $C_2 = 215.426$ when $\varepsilon = 2^{-56}$.

Proof. See Appendix B of [8]. □

Remark. The bound on ε_i^k is quite pessimistic. Numerical results show that most of the time the constants C_i should be reduced to 0.01 times their values given above (cf. the remark in Appendix B of [8]).

4.4. *Decision Criteria for the Hybrid Methods*. Using the bounds in the previous section we demonstrate that one can determine G and g_i so that the recursive function RH (for the highest divided difference) always yields a result with bounded error. For convenience, we write $X^{(n)} \equiv X$ to indicate that X has n abscissae. The function RH$(X^{(n)})$ is:

(1) RH$(X^{(1)}) = \exp(\xi_1)$.

(2) Compute $G(X^{(n)}) = (\xi_\mu, \xi_\nu)$, where $\xi_\mu \equiv \max_i \xi_i$ and $\xi_\nu \equiv \min_i \xi_i$.

(3) If $|\xi_\mu - \xi_\nu| \leqslant g_{n-1}$ call SS(II) and RH$(X^{(n)}) := (d(1, n))$ else

$$(4.4.1) \qquad \mathrm{RH}(X^{(n)}) := \frac{\mathrm{RH}\left(X_{(\mu)}^{(n-1)}\right) - \mathrm{RH}\left(X_{(\nu)}^{(n-1)}\right)}{\xi_\mu - \xi_\nu},$$

where $X_{(i)}^{(n-1)} \equiv [\xi_i, \xi_2, \ldots, \xi_{i-1}, \xi_{i+1}, \ldots, \xi_n]$.

[12] See Appendix B of [8] for details. In general, it suffices to require them $\leqslant \frac{1}{4} \cdot (1 + 2k/|\zeta_{i+k} - \zeta_i|)^{-1}$.

Based on the bounds (4.3.1) and (4.2.2), we are going to show by induction that

THEOREM. *For a given precision ε, there exist some constants g_j and $\varepsilon^{(j)}$, where $j = 1, 2, \ldots$ such that for any n and $X^{(n)}$ the relative error*

$$|\varepsilon_1^{n-1}| = \left| \frac{fl(\mathrm{RH}(X^{(n)})) - \Delta_1^{n-1}(X^{(n)})}{\Delta_1^{n-1}(X^{(n)})} \right| \leqslant \varepsilon^{(n-1)}.$$

It can be shown that $\varepsilon^{(k)} = O(k^2) \cdot \varepsilon$ as $k \to \infty$ (see Remark 1).

Proof. Step 0. When $n = 1$, $\Delta_1^0(X) = \mathrm{RH}(X) = \exp(\xi_1)$. Therefore $\varepsilon^{(0)}$ can be set equal to ε (we assume function exp can be evaluated accurately, i.e., $|\varepsilon_1^0| \leqslant \varepsilon$).

Step 1. When $n = 2$, assume $\xi_1 < \xi_2$, then $G(X) = (\xi_2, \xi_1)$. Let $\vartheta = \xi_2 - \xi_1$. To compute RH($X$), SR yields (cf. 4.2.2)

$$(4.4.2) \qquad |\varepsilon_1^1| \leqslant 4\varepsilon + (1 + 4/\vartheta) \cdot \varepsilon^{(0)} \leqslant (5 + 4/\vartheta)\varepsilon$$

and SS yields (cf. 4.3.1)

$$(4.4.3) \qquad |\varepsilon_1^1| < [2C_0 + 2C_1 \log \gamma' + C_2\gamma' - 2] \cdot \varepsilon.$$

Since $\gamma' \equiv \max(\gamma, 0.7) < \max(\vartheta, 0.7)$ (4.4.3) becomes

$$(4.4.4) \quad |\varepsilon_1^1| < \left[2C_0 + 2C_1 \log(\max(\vartheta, 0.7)) + C_2(\max(\vartheta, 0.7)) - 2\right] \cdot \varepsilon.$$

Notice that the bound in (4.4.2) is monotonic decreasing in ϑ and the one in (4.4.4) is monotonic nondecreasing so they have only one intersection. Let it occur at $\vartheta = g_1$. It means that ε_1^1 will always be bounded by $\varepsilon^{(1)} \equiv (5 + 4/g_1)\varepsilon$ if one computes RH(X) by SS when $G(X) < g_1$ and SR (i.e., by (4.4.1)) otherwise.

Step 2. Assume that for $1 < n$ the assertion is true, i.e., ε_1^{n-1} in RH($X^{(n)}$) is bounded by some constant $\varepsilon^{(n-1)}$ for any $X = X^{(n)}$. Consider $X = X^{(n+1)}$. Let ϑ denote $|\xi_\nu - \xi_\mu|$, where $G(X^{(n+1)}) = (\xi_\nu, \xi_\mu)$. To compute RH($X$), SR, or Eq. (4.4.1) yields

$$(4.4.5) \qquad |\varepsilon_1^n| \leqslant 4\varepsilon + (1 + 2n/\vartheta) \cdot \varepsilon^{(n-1)}$$

and SS yields

$$(4.4.6) \quad |\varepsilon_1^n| < \left[nC_0 + nC_1 \log(\max(\vartheta, 0.7)) + C_2(\max(\vartheta, 0.7)) - 2\right] \cdot \varepsilon.$$

Again the bound in (4.4.5) is monotonic decreasing on ϑ and (4.4.6) is monotonic nondecreasing on ϑ, so they have only one intersection and let it occur at $\vartheta = g_n$. Therefore ε_1^n will always be bounded by $\varepsilon^{(n)} \equiv 4\varepsilon + (1 + 2n/g_{n-1}) \cdot \varepsilon^{(n-1)}$ if one computes RH(X) by SS when $|\xi_\nu - \xi_\mu| < g_n$, and by (4.4.1) otherwise.

By induction, our assertion is true for all n. □

One can generate those g_j, $\varepsilon^{(j)}$ recursively by equating the bounds in (4.4.5) and (4.4.6) and solve it for $j = 1, 2, \ldots$ with the initial value $\varepsilon^{(0)} = \varepsilon$:

$$\begin{aligned}(4.4.7\text{a}) \quad & 4\varepsilon + (1 + 2j/\vartheta) \cdot \varepsilon^{(j-1)} \\ & \qquad = \left[jC_0 + jC_1 \log(\max(\vartheta, 0.7)) + C_2(\max(\vartheta, 0.7)) - 2\right] \cdot \varepsilon\end{aligned}$$

with

$$(4.4.7\text{b}) \quad \varepsilon^{(j)} = \left[jC_0 + jC_1 \log(\max(\vartheta, 0.7)) + C_2(\max(\vartheta, 0.7)) - 2\right] \cdot \varepsilon.$$

For $\varepsilon = 2^{-24}$, we compute some of the g_j and $\varepsilon^{(j)}$ according to (4.4.7) and list them in Table (4.4.8). Therefore, we have shown

COROLLARY. *When* $\varepsilon = 2^{-24}$, *we have*

$$\left|\frac{fl(\mathrm{RH}(X^{(n)})) - \Delta_1^{n-1}(X^{(n)})}{\Delta_1^{n-1}(X^{(n)})}\right| \leqslant \left(\frac{\varepsilon^{(n-1)}}{\varepsilon}\right)\cdot\varepsilon, \tag{4.4.8}$$

where the values of g_j *for* RH *and the values of* $(\varepsilon^{(j)}/\varepsilon)$ *are given in Table* (4.4.8). □

TABLE (4.4.8)

Single precision decision criteria $(\varepsilon = 2^{-24})$ *and error bounds for the hybrid algorithm.*

j	g_j	Error bound $\varepsilon^{(j)}/\varepsilon$	digits lost $(\log_{10}(\varepsilon^{(j)}/\varepsilon))$
1	0.02	0.105e + 03	2.02
2	2.100	0.310e + 03	2.49
3	4.846	0.697e + 03	2.84
4	9.227	0.131e + 04	3.12
5	15.41	0.216e + 04	3.33
6	23.48	0.326e + 04	3.51
7	33.50	0.463e + 04	3.67
8	45.48	0.626e + 04	3.80
9	59.46	0.816e + 04	3.91
10	75.42	0.103e + 05	4.01
20	345.4	0.469e + 05	4.67
40	1487.	0.201e + 06	5.30
60	3430.	0.462e + 06	5.67
80	6173.	0.832e + 06	5.92
100	9716.	0.131e + 07	6.12

Remark 1. The asymptotic value of g_i is $i^2 + O(i)$, as can be seen from the equation $\varepsilon^{(k)} = C_2 g_k \cdot \varepsilon$ and $C_2 g_{k+1} = (1 + 2k/g_{k+1}) \cdot C_2 g_k$ obtained by omitting the lower order terms in (4.4.7). One can verify by induction that $k^2 - 3k < g_k < k^2$ and consequently the error bound $\varepsilon^{(k)} = (C_2 k^2 + O(k))\varepsilon$.

Remark 2. Although the error bounds in Table (4.4.8) are not ridiculous, they are quite pessimistic. Also, the value of g_j in the above table is too large to be useful. For example, when $n = 20$, $g_{20} = 345.4$ and it means that Δ_1^{20} is computed by $(\Delta_2^{19} - \Delta_1^{19})/(\xi_{21} - \xi_1)$ only if $\xi_{21} - \xi_1 > 345.4$! Experience shows that as long as $\xi_{n+1} - \xi_1 > 25$ or 26, SR always yields satisfactory answers. Since SR is much faster than SS, one prefers SR to SS whenever SR yields satisfactory results. So we would like a set of values for g_j and $\varepsilon^{(j)}$ which is more *realistic*. After numerous numerical experiments we obtained the following experimental formula for g_j and $\varepsilon^{(j)}$ (for any precision ε).

Experimental Formula.

$$g_j \equiv \left(1 + \frac{\ln j}{10}\right)\cdot j, \qquad \varepsilon(j) = 5g_j \cdot \varepsilon. \tag{4.4.9}$$

The practical value for g_j is much smaller than the one in Table (4.4.8)—it is like $j + 0.1 j \ln j$ versus j^2. For comparison, take $j = 40$, (4.4.9) yields 54.76 while (4.4.8) yields 1487! We ran our SH (with g_j in (4.4.9)) on a Z that has 20 data points distributed irregularly from −27 to 25. The results are summarized in Table (4.4.10). The last column "digit lost" is $\log_{10}$ (relative error). It is most satisfactory.

TABLE (4.4.10)

Test example for Simple Hybrid Method.

n	ζ_n	Correct Δ_1^{n-1} to 7 digits	SH Δ_1^{n-1}	digits lost
1	-27.0	0.1879529e − 11	0.1879529e − 11	0.
2	-26.0	0.3229560e − 11	0.3229560e − 11	0.
3	-15.0	0.2317134e − 08	0.2317134e − 08	0.09
4	-14.0	0.3012897e − 08	0.3012897e − 08	0.
5	-12.0	0.2983682e − 08	0.2983682e − 08	0.
6	-10.0	0.2246401e − 08	0.2246401e − 08	0.
7	-8.0	0.1353474e − 08	0.1353474e − 08	0.50
8	-7.9	0.4257157e − 09	0.4257158e − 09	0.56
9	-7.8	0.9465834e − 10	0.9465836e − 10	0.61
10	-2.7	0.4272183e − 10	0.4272186e − 10	0.96
11	1.0	0.2364207e − 10	0.2364209e − 10	1.10
12	1.1	0.6378568e − 11	0.6378574e − 11	1.22
13	1.2	0.1208801e − 11	0.1208802e − 11	1.33
14	1.3	0.1806541e − 12	0.1806544e − 12	1.45
15	3.0	0.2706591e − 13	0.2706596e − 13	1.49
16	7.0	0.5415335e − 14	0.5415346e − 14	1.53
17	9.0	0.1022545e − 14	0.1022547e − 14	1.58
18	13.0	0.2453144e − 15	0.2453151e − 15	1.65
19	24.0	0.3804000e − 15	0.3803999e − 15	0.72
20	25.0	0.1456325e − 15	0.1456325e − 15	0.54

5. Complex Exponential Divided Differences.

5.1. *Can we Have High Relative Accuracy*? As we have seen in Section 4, the real exponential divided differences can be computed with high relative accuracy. What makes it possible is that $\Delta^k_j(X) = \Delta^k_j(X)\exp$ is positive for real X. This property fails for complex data Z, for $\Delta^k_j(Z)$ can take on any complex value. However, one can still say something about the error in $\Delta^k_j(Z)$. In order to do that some extra notation is needed. Let X and Y be the real and imaginary part of Z, i.e., if $Z = [\zeta_1, \zeta_2, \ldots, \zeta_n]$, then $X = [\xi_1, \xi_2, \ldots, \xi_n]$ and $Y = [\eta_1, \eta_2, \ldots, \eta_n]$ so that $\zeta_k = \xi_k + i\eta_k$ for $k = 1, 2, \ldots, n$. Also let $\Delta^k_j(W)$ denote the exponential divided differences on the abscissae W. Our treatment of error in the complex case is based on the following inequality.

LEMMA. *With the notation given above*

$$|\Delta^k_i(Z)| \leqslant \Delta^k_i(X). \tag{5.1.1}$$

Proof. Use the Hermite-Genocchi expression (1.3.1) for $\Delta^k_i(Z)$ and note that

$$\left|\exp\left[\zeta_i + (\zeta_{i+1} - \zeta_i)\nu_1 + \cdots\right]\right| = \exp\left[\xi_i + (\xi_{i+1} - \xi_i)\nu_1 + \cdots\right]. \square$$

Inequality (5.1.1) enables us to bound the error in the computed $\Delta^k_i(Z)$ in terms of $\Delta^k_i(X)$. The bounds are similar to those in Section 4. We summarize the results below, and leave the details to Appendix B in [8]. Let ε be the unit roundoff and e^k_i be the *absolute error* of $\Delta^k_i(Z)$, i.e., $fl(\Delta^k_i(Z)) = \Delta^k_i(Z) + e^k_i$. Define ε^k_i, the *pseudo relative error* in $\Delta^k_i(Z)$, to be $\varepsilon^k_i \equiv e^k_i/\Delta^k_i(X)$.

(1) *Error Growth of SR* (*Standard Recurrence*). Suppose that $\Delta^k_i(Z)$ is computed by SR, and also $\mathrm{Re}(\zeta_{i+k}) \geqslant \mathrm{Re}(\zeta_j)$ for $i \leqslant j < i + k$. Then, to first order in ε, the

pseudo relative error ε_i^k satisfies:

$$(5.1.2) \qquad |\varepsilon_i^k| \leqslant 4\varepsilon + \left[1 + \frac{2k}{|\zeta_{i+k} - \zeta_i|}\right] \cdot \max\left\{|\varepsilon_{i+1}^{k-1}|, |\varepsilon_i^{k-1}|\right\}. \square$$

Proof. See Appendix B of [8]. □

(2) *Error Bounds of SS* (*Scaling and Squaring*). Let the radius γ be defined as in Subsection 2.4. Suppose that $\Delta(Z)$ is computed by SS(II). Then to first order in ε we have

Error bound

$$(5.1.3) \qquad |\varepsilon_i^k| \leqslant \left[C_0 k + C_1 k \ln(\max(\gamma, 0.7)) + C_2 \cdot \max(\gamma, 0.7) - 2\right] \cdot \varepsilon,$$

where C_i, $i = 0, 1, 2$ take the same values as in (4.3.2).

Proof. See Appendix B (Corollary (B.6)) of [8]. □

The above bounds for complex abscissae Z are similar to those for the real ones in Section 4, except that the meaning of the error ε_i^k is different: here ε_i^k is the error in $\Delta_i^k(Z)$ relative to $\Delta_i^k(X)$. The same analysis as in Subsection 4.4 shows that the hybrid methods yield small ε_i^k like $O(k^2)\varepsilon$, i.e., yields $\Delta_i^k(Z)$ with small absolute error *compared to* $\Delta_i^k(X)$, provided that the decision function G satisfies:

(1) $G(Z) = (\zeta_\mu, \zeta_\nu)$ and $|\zeta_\mu - \zeta_\nu| \approx \gamma$,

(2) $\mathrm{Re}(\zeta_\mu) \geqslant \mathrm{Re}(\zeta_i)$ for any $\zeta_i \in Z$.

It leads to the definition (3.3.4) for G, i.e., $G(Z) \equiv (\zeta_\mu, \zeta_\nu)$ such that

$$(5.1.4) \qquad |\zeta_\mu - \zeta_\nu| = \max_{\substack{\zeta_i \in \Lambda \\ \zeta_j \in Z}} |\zeta_i - \zeta_j| \quad \text{where } \Lambda \equiv \left\{\zeta_i \in Z\colon \mathrm{Re}(\zeta_i) = \max_j \mathrm{Re}(\zeta_j)\right\}.$$

For this G, we always have $2G(Z) > \gamma(Z)$ (usually $G(Z) > \gamma(Z)$ except in some rare situations). Therefore, with the above G, we have

(3) *Error Bound of* RH(X). There exist constants g_j for the RH and constants $\varepsilon^{(j)}$, $j = 1, 2, \ldots$ such that for any $X = X^{(k)}$,

$$|\varepsilon_1^k| \leqslant \varepsilon^{(k-1)}.$$

Proof. The proof is similar to the one in Subsection 4.4. □

If one assumes $G(Z) > \gamma$, then when $\varepsilon = 2^{-24}$, the values of g_i are the same as those in Table (4.4.8).

Remark. In the implementation of the hybrid methods, one can avoid using RH in the real case because one can always order the data so as to be nested. In the complex case there may not exist such an ordering and RH seems unavoidable in order to secure good relative accuracy in the most general case (e.g., 500 points on a circle of radius 500 in the complex plane). However, in Subsection 5.3 we will show how to salvage SH when the data are complex.

5.2. *Computational Difficulty.* Let X, Y and Z be as defined in 5.1. The error bound for RH(X) in Subsection 5.1 implies that if we compute $\Delta_1^{n-1}(Z)$ in RH in finite precision arithmetic (with precision ε) then the relative error in $fl(\Delta_1^{n-1}(Z))$ will be bounded by

$$(5.2.1) \qquad \left|\frac{fl(\Delta_1^{n-1}(Z)) - \Delta_1^{n-1}(Z)}{\Delta_1^{n-1}(Z)}\right| \leqslant \left|\frac{\Delta_1^{n-1}(X)}{\Delta_1^{n-1}(Z)}\right| \cdot \varepsilon^{(n-1)},$$

where $\varepsilon^{(k)} \leqslant C_2 k^2 \varepsilon$ (see Remark 1 of Subsection 4.4). Let ρ denote the coefficient of $\varepsilon^{(n-1)}$, i.e., $\rho = \rho(Z) \equiv \Delta_1^{n-1}(X)/|\Delta_1^{n-1}(Z)|$. When $\rho \gg 1$, (5.2.1) implies that it is difficult to obtain $\Delta_i^k(Z)$ with high relative accuracy. This difficulty is intrinsic to the RH method. One way to overcome it is to increase the precision of the arithmetic operations and of the variables. Another possible approach is to find special formulae which build up $\Delta_i^k(Z)$ from even tinier quantities, e.g., FDD, the formula for Δ_1^1 exp given in Subsection 2.2. Unfortunately for $n > 2$, we do not know if any such formulae exist.

We call $\rho(Z)$ the difficulty of Z for computing the exponential divided difference $\Delta_1^{n-1}(Z)$. It indicates whether $\Delta_1^{n-1}(Z)$ can be computed with high relative accuracy. If $\rho(Z)$ is large then we call the data Z difficult for the computation of $\Delta_1^{n-1}(Z)$.

We have used the word difficulty instead of the common term condition number for several good reasons. By tradition, a condition number should measure the sensitivity of an output to small perturbations in the input. Our quantity ρ is merely a bound on the most accurate method we have for computing $\Delta_1^{n-1}(Z)$exp. More important is the fact that we want to compute these numbers with a small relative error, not absolute. Z is difficult precisely when the number we wish to compute is close to zero. Indeed this number may not be sensitive (absolutely) to small changes in Z.

From (5.1.1), we always have $\rho \geqslant 1$. In general, ρ does not have an upper bound[13] (e.g., $\rho([0, 2\pi i]) = \infty$). However, when Z is real, $\rho = 1$. Also Z is not difficult if all its elements are close to the real axis.

THEOREM. *Given* $Z = ([\zeta_1, \ldots, \zeta_n]$, *let* $\eta_j = \operatorname{Im} \zeta_j$, $j = 1, \ldots, n$. *If* $\max_j |\eta_j| \leqslant \gamma < \pi/2$, *then*

$$\rho(Z) \leqslant \sec(\gamma).$$

Proof. Our proof follows from the following mean value representation of $\Delta_j^k(Z)$: there exist real μ, ν with $\min_{j \leqslant l \leqslant j+k} \eta_l \leqslant \mu, \nu \leqslant \max_{j \leqslant l \leqslant j+k} \eta_l$, such that

$$\Delta_j^k(Z) = \Delta_j^k(Z)(\cos(\nu) + i \sin(\mu)). \tag{5.2.2}$$

From the identity $\exp(\xi + i\eta) = \exp(\xi)(\cos(\eta) + i\sin(\eta))$ and the Hermite-Genocchi representation (1.3.1), we have

$$\begin{aligned}\Delta_j^k(Z) = \int_0^1 \int_0^{\nu_1} \cdots \int_0^{\nu_{k-1}} & \exp\left[\xi_j + \cdots + (\xi_{j+k} - \xi_{j+k-1})\nu_k\right] \\ & \times \cos\left[\eta_j + \cdots + (\eta_{j+k} - \eta_{j+k-1})\nu_k\right] d\nu_k \cdots d\nu_2 \, d\nu_1 \\ + i \cdot \int_0^1 \int_0^{\nu_1} \cdots \int_0^{\nu_{k-1}} & \exp\left[\xi_j + \cdots + (\xi_{j+k} - \xi_{j+k-1})\nu_k\right] \\ & \times \sin\left[\eta_j + \cdots + (\eta_{j+k} - \eta_{j+k-1})\nu_k\right] d\nu_k \cdots d\nu_2 \, d\nu_1.\end{aligned}$$

Since exp is positive on real values, Eq. (5.2.2) follows from the integral mean value theorem applied to each part. Now with the condition $\max_j |\eta_j| \leqslant \gamma < \pi/2$, (5.2.2)

[13] We conjecture that if the imaginary parts of the data are restricted in $(0, 2\pi)$, then any divided difference Δ_i^k never vanishes (and hence $\rho \neq \infty$). It can be proved for $k = 1$; but we do not know any proof when k is bigger than 1.

gives a lower bound for $|\Delta^k_j(Z)|$:

$$(5.2.3) \qquad |\Delta^k_j(Z)| \geqslant \cos(\gamma)\cdot\Delta^k_j(X) > 0.$$

The theorem follows from the definition of ρ. □

Note that when $|\mathrm{Im}(\zeta_i)| \leqslant 0.45\pi$, $i = 1,\dots,n$, then $\rho \leqslant 6.41$. Examples of difficult Z are: those abscissae close to $[0, 2\pi i, 4\pi i,\dots, 2k\pi i]$ (any divided difference on these abscissae vanishes, i.e., $\rho = \infty$). A surprising example is $Z = [0, i, 2.04254 + 7.97730i]$. For this Z, we computed $\Delta(Z)$ with approximately 7 decimal precision and give the corresponding ρ in the table below. Notice that SS lost 6 digits in the last divided difference, which has $\rho \approx 10^6$.

TABLE 5.2.3

Divided differences on $Z = [0, i, 2.04254 + 7.97730i]$.

	Correct values to 3 digits	SS	ρ
Δ^1_1	(0.841 0.460)	(0.841 0.460)	1.05
Δ^2_1	(−0.144e − 06 −0.731e − 07)	(−0.161e − 06 −0.800e − 07)	7.14e06

Remark 1. In application to the matrix exponential the need for high relative accuracy in $\Delta^k_i(Z)$ decreases with $|\Delta^k_i(Z)|$. When it is satisfactory to compare the error in $|\Delta^k_i(Z)|$ with $\Delta^k_i(X)$ then the difficulty evaporates.

Remark 2. In general $\rho(Z)$ increases with the spread of the imaginary parts. For example,

$$\rho([\zeta_1,\zeta_2]) = \frac{\Delta^1_1([\mathrm{Re}(\zeta_1),\mathrm{Re}(\zeta_2)])}{|\Delta^1_1([\zeta_1,\zeta_2])|} = \frac{|\zeta_1-\zeta_2|}{|\mathrm{Re}(\zeta_1-\zeta_2)|}\cdot\frac{|e^{\mathrm{Re}(\zeta_1)}-e^{\mathrm{Re}(\zeta_2)}|}{|e^{(\zeta_1)}-e^{(\zeta_2)}|}$$
$$\leqslant \frac{|\zeta_1-\zeta_2|}{|\mathrm{Re}(\zeta_1-\zeta_2)|}.$$

So the bigger the difference of the imaginary parts the larger is the difficulty. As a point of interest we also compute the difficulties on circles with various radii and number of points. The results are summarized in the following table. Each entry is the difficulty of abscissae distributed uniformly on a circle with radius γ.

TABLE 5.2.4

Difficulty of circles for the highest divided difference.

	$\gamma = 5$	$\gamma = 10$	$\gamma = 15$	$\gamma = 20$	$\gamma = 25$
$n = 5$	2.2	3.1	3.2	3.2	3.2
$n = 10$	1.7	7.9	28.9	35.9	45.1
$n = 15$	1.5	4.5	25.0	173.6	301.7
$n = 20$	1.3	3.2	12.6	77.5	651.8

5.3. *Ordering and Matrix Argument Reduction.* A nested ordering may not exist for general complex data Z. However, if the imaginary parts of the data are bounded by π, then one can order the data according to their real parts and get an almost nested ordering. In this section, which is based on the period $2\pi i$ of exp, we indicate briefly a way to transform the data to values that have bounded imaginary parts. We refer the reader to Ng [9] for details.

Definition of The Reduction Function Mod(A). Since exp has period $2\pi i$ the strip $-\pi < \mathrm{Im}(\zeta) \leqslant \pi$ is representative. Let us define the argument reduction function for exp as follows:

$$\mathrm{Mod}(\zeta) \equiv \zeta - 2k\pi i \quad \text{if } (2k-1)\pi < \mathrm{Im}(\zeta) \leqslant (2k+1)\pi.$$

We have $\exp(\zeta) = \exp(\mathrm{Mod}(\zeta))$. Now we are going to extend the function Mod to matrices. Let J be the Jordan normal form of A, i.e., $A = P^{-1}JP$, and $J = \mathrm{diag}(J_{i_1}, \ldots, J_{i_l})$ where J_m is the Jordan block with diagonal equal to eigenvalue λ_m of A. Let k_m be the integer such that

$$(2k_m - 1)\pi < \mathrm{Im}(\lambda_m) \leqslant (2k_m + 1)\pi.$$

Define

(1) $\mathrm{Mod}(J_m) \equiv J_m - 2k_m \pi i I$;

(2) $\mathrm{Mod}(J) \equiv \mathrm{diag}(\mathrm{Mod}(J_{i_1}), \ldots, \mathrm{Mod}(J_{i_l}))$;

(3) $\mathrm{Mod}(A) \equiv P^{-1}\mathrm{Mod}(J)P$.

It is not difficult to prove that $\exp(A) = \exp(\mathrm{Mod}(A))$ according to (1), (2), and (3). Thus Mod generalizes argument reduction to matrices and yields a matrix that has eigenvalues with bounded imaginary parts.

As we have mentioned in the introduction, the application behind the computation of $\Delta^k{}_j\exp$ is matrix exponentials. If one applies the matrix argument reduction **before** computing the exponential, then all the eigenvalues of the matrix would have bounded imaginary parts, thus solving the ordering problem in the computation of the divided differences.

Remark 1. There is another way to reduce the imaginary parts of the data: since $\Delta\exp = \exp(Z_n)$, we may apply argument reduction directly on Z_n and compute $\exp(\mathrm{Mod}(Z_n))$. However, the bidiagonal structure of Z_n will be destroyed by the reduction and therefore some modifications of the algorithm TS are needed. The work for the whole compuatation increases significantly.

Remark 2. For the computation of Mod(A), there is a stable method which avoids using the Jordan decomposition of a matrix. When A is triangular the work needed is approximately $n^3/3$ operations which is quite practical. An algorithm for argument reduction can be found in Ng [9].

5.4. *Conclusion*: *SH for Data With Restricted Imaginary Parts*. Although RH gives the divided differences with guaranteed accuracy, it is impractical to implement it unless the order of the divided differences is very small like 3 or 4, because the number of operations grows like 2^n. Subsection 5.3 shows that (assuming one has the matrix function Mod(A)) one can consider matrices with eigenvalues close to the real line, so there is no loss of generality in considering Z with imaginary parts bounded by π. There are two advantages to small imaginary parts. The first is that we can order the abscissae according to their real parts and obtain an almost nested ordering (according to the G defined in Subsection 5.1). Thus one can apply SH (Simple Hybrid method) instead of RH (Recursive Hybrid function). The second is that the backfilling step in SS is stable, which implies that one can replace SS(II) by SS with very slight sacrifice in accuracy. But the trade-off is significant, since SS takes $O(n^2)$ operations and requires only a few vectors for storage while SS(II) take $O(n^3)$ and requires a matrix storage. We conclude this section by proposing the following.

Computation of $\Delta(Z)$. Given Z with $\mathrm{Re}(Z)$ in increasing order and $|\mathrm{Im}(Z)| \leqslant \pi$. Use algorithm SH with the following G to compute $\Delta(Z)$.

Decision Function G for SH on Z. The function G on $Z = [\zeta_1, \ldots, \zeta_n]$ is defined to be $G(Z) = (\zeta_n, \zeta_1)$,[14] and the decision is, for $i < j$,

$$\zeta_i,\ \zeta_j \text{ belong to the same cluster if } \mathrm{Re}\big(\zeta_j - \zeta_i\big) < g_{j-i}$$

where the values of g_l, $l = 1, 2, \ldots,$ can be those in (4.4.9). □

Numerical Results. We ran the SH algorithm on Z that has the same real parts as in (4.4.10) but with the imaginary parts $= \pm\pi$. The results are summarized in the following table.

TABLE (5.4.1)

Test example for SH on complex data, $\Delta_1^{n-1}(Z)\exp$.

n	ζ_n	Correct values to 7 digits	SH	digits lost
1	$-27.0 + \pi i$	(−0.1879529d − 11 −0.1643136d − 18)	(−0.1879529e − 11 −0.1643136e − 18)	0.
2	$-26.0 - \pi i$	(−0.7978484d − 13 −0.5013023d − 12)	(−0.7978483e − 13 −0.5013023e − 12)	0.
3	$-15.0 + \pi i$	(−0.1747305d − 08 0.9981036d − 09)	(−0.1747305e − 08 0.9981036e − 09)	0.
4	$-14.0 - \pi i$	(0.4155101d − 09 −0.4757347d − 09)	(0.4155102e − 09 −0.4757347e − 09)	0.
5	$-12.0 + \pi i$	(0.1814275d − 09 0.1323147d − 08)	(0.1814274e − 09 0.1323147e − 08)	0.43
6	$-10 - \pi i$	(0.7591439d − 09 −0.5204460d − 09)	(0.7591440e − 09 −0.5204460e − 09)	0.32
7	$-8.0 + \pi i$	(0.4204520d − 09 0.5119519d − 09)	(0.4204520e − 09 0.5119519e − 09)	0.
8	$-7.9 - \pi i$	(0.2091884d − 09 −0.3885102d − 10)	(0.2091884e − 09 −0.3885013e − 10)	0.20
9	$-7.8 + \pi i$	(0.4721783d − 10 0.2416627d − 10)	(0.4721784e − 10 0.2416627e − 10)	0.48
10	$-2.7 - \pi i$	(0.2229288d − 10 −0.1255209d − 10)	(0.2229289e − 10 −0.1255209e − 10)	0.90
11	$1.0 + \pi i$	(0.1147709d − 10 0.8703504d − 11)	(0.1147709e − 10 0.8703510e − 11)	0.99
12	$1.1 - \pi i$	(0.3820952d − 11 −0.7453152d − 12)	(0.3820956e − 11 −0.7453158e − 12)	1.18
13	$1.2 + \pi i$	(0.7360573d − 12 0.2693091d − 12)	(0.7360580e − 12 0.2693094e − 12)	1.25
14	$1.3 - \pi i$	(0.1204098d − 12 −0.1441720d − 13)	(0.1204099e − 12 −0.1441722e − 13)	1.29
15	$3.0 + \pi i$	(0.1798853d − 13 0.5672050d − 14)	(0.1798856e − 13 0.5672058e − 14)	1.42
16	$7.0 - \pi i$	(0.3734263d − 14 −0.8906243d − 15)	(0.3734269e − 14 −0.8906257e − 15)	1.43
17	$9.0 + \pi i$	(0.7041432d − 15 0.2104887d − 15)	(0.7041446e − 15 0.2104891e − 15)	1.51
18	$13.0 - \pi i$	(0.1705491d − 15 −0.5255464d − 16)	(0.1705495e − 15 −0.5255475e − 16)	1.51
19	$24.0 + \pi i$	(0.1783471d − 15 0.2243819d − 15)	(0.1783471e − 15 0.2243819e − 15)	0.27
20	$25.0 - \pi i$	(0.9540051d − 16 −0.1855942d − 16)	(0.9540053e − 16 −0.1855942e − 16)	0.60

6. Application to Computing Matrix Exponentials.

6.1. *Repesentation of $f(A)$ by the Newton Interpolating Polynomial.* Let A be $n \times n$ and let f be any scalar function with at least n continuous derivatives at the eigenvalues $\zeta_1, \ldots, \zeta_n$ of A. Associated with f is the unique polynomial of degree $n - 1$ which interpolates f at the ζ_i. A convenient representation of this polynomial was given by Newton,

$$p_{n-1}(t) = f(\zeta_1) + \sum_{k=1}^{n-1} \Delta_1^k f \cdot \prod_{j=1}^{k} (t - \zeta_j).$$

Here $\Delta_1^k f$ denotes the kth order divided difference of f at the abscissae $\zeta_1, \ldots, \zeta_{k+1}$.

A fundamental result in matrix theory is that

$$f(A) = p_{n-1}(A). \tag{6.1.1}$$

[14] For such Z and G, one can show that $2(G(Z) + \pi) > \gamma(Z)$.

That is,

Newton Interpolating Polynomial of $f(A)$.

$$f(A) = \Delta_1^0 f \cdot I + \sum_{k=1}^{n-1} \Delta_1^k f \cdot \prod_{j=1}^{k} (A - \zeta_j I). \tag{6.1.2}$$

In our applications, A is in triangular form. Therefore the eigenvalues are just the diagonal elements of the matrix and the matrix products can be formed efficiently.

6.2. *Matrix Exponentials.* Let A be triangular. Since exp is periodic on the imaginary axis with period 2π, we can use argument reduction in marix form (cf. Subsection 5.4) on A, replace A by another triangular A' such that $\exp(A) = \exp(A')$ and $|\mathrm{Im}(a'_{i,i})| \leqslant \pi$. There is no loss of generality in assuming that argument reduction has been done and therefore the imaginary parts of the eigenvalues of A are bounded. Now we can apply SH on the eigenvalues to obtain the divided differences and compute $\exp(A)$ by (6.1.2).

Department of Mathematics
University of California
Berkeley, California 94720

1. K. E. Atkinson, *An Introduction to Numerical Analysis*, Wiley, New York, 1978.

2. S. D. Conte and C. de Boor, *Elementary Numerical Analysis*, 3rd ed., McGraw-Hill, New York, 1980.

3. C. Davis, "Explicit functional calculus," *Linear Algebra Appl.*, v. 6, 1973, pp. 193–199.

4. G. F. Gabel, *A Predictor-Corrector Method Using Divided Differences*, Technical Report No. 5, Dept. of Computer Science, Univ. of Toronto, Oct. 1968.

5. A. O. Gel'fand, *Calculus of Finite Differences*, Hindustan, India, 1971.

6. W. Kahan & I. Farkas, "Algorithm 167—Calculation of confluent divided differences," *Comm. ACM*, v. 6, 1963, pp. 164–165.

7. A. C. McCurdy, *Accurate Computation of Divided Differences*, UCB/ERL M80/28, Univ. of California, Berkeley, 1980.

8. A. McCurdy, K. C. Ng & B. N. Parlett, *Accurate Computation of Divided Differences of the Exponential Function*, CPAM-160, Univ. of California, Berkeley, June 1983.

9. K. C. Ng, *The Computation of the Matrix Exponential*, Thesis, Univ. of California, Berkeley, December 1983.

10. G. Opitz, "Steigungsmatrizen," *Z. Angew. Math. Mech.*, v. 44, 1964, pp. T52–T54.

11. B. N. Parlett, "A recurrence among the elements of functions of triangular matrices," *Linear Algebra Appl.*, v. 14, 1976, pp. 117–121.

12. R. C. Ward, "Numerical computation of the matrix exponential with accuracy estimate," *SIAM J. Numer. Anal.*, v. 14, 1977, pp. 600–610.

13. L. M. Milne-Thomson, *The Calculus of Finite Differences*, Macmillan, London, 1933.