SIAM REVIEW
Vol. 45, No. 1, pp. 3–49

Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later*

Cleve Moler†
Charles Van Loan‡

Abstract. In principle, the exponential of a matrix could be computed in many ways. Methods involving approximation theory, differential equations, the matrix eigenvalues, and the matrix characteristic polynomial have been proposed. In practice, consideration of computational stability and efficiency indicates that some of the methods are preferable to others but that none are completely satisfactory.

Most of this paper was originally published in 1978. An update, with a separate bibliography, describes a few recent developments.

Key words. matrix, exponential, roundoff error, truncation error, condition

AMS subject classifications. 15A15, 65F15, 65F30, 65L99

PII. S0036144502418010

1. Introduction.

Mathematical models of many physical, biological, and economic processes involve systems of linear, constant coefficient ordinary differential equations

$$\dot{x}(t) = Ax(t).$$

Here A is a given, fixed, real or complex n-by-n matrix. A solution vector $x(t)$ is sought which satisfies an initial condition

$$x(0) = x_0.$$

In control theory, A is known as the state companion matrix and $x(t)$ is the system response.

In principle, the solution is given by $x(t) = e^{tA}x_0$, where e^{tA} can be formally defined by the convergent power series

$$e^{tA} = I + tA + \frac{t^2A^2}{2!} + \cdots .$$

The effective computation of this matrix function is the main topic of this survey.

*Published electronically February 3, 2003. A portion of this paper originally appeared in *SIAM Review*, Volume 20, Number 4, 1978, pages 801–836.
http://www.siam.org/journals/sirev/45-1/41801.html

†The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098 (moler@mathworks.com).
‡Department of Computer Science, Cornell University, 4130 Upson Hall, Ithaca, NY 14853-7501 (cv@cs.cornell.edu).

We will primarily be concerned with matrices whose order n is less than a few hundred, so that all the elements can be stored in the main memory of a contemporary computer. Our discussion will be less germane to the type of large, sparse matrices which occur in the method of lines for partial differential equations.

Dozens of methods for computing e^{tA} can be obtained from more or less classical results in analysis, approximation theory, and matrix theory. Some of the methods have been proposed as specific algorithms, while others are based on less constructive characterizations. Our bibliography concentrates on recent papers with strong algorithmic content, although we have included a fair number of references which possess historical or theoretical interest.

In this survey we try to describe all the methods that appear to be practical, classify them into five broad categories, and assess their relative effectiveness. Actually, each of the "methods" when completely implemented might lead to many different computer programs which differ in various details. Moreover, these details might have more influence on the actual performance than our gross assessment indicates. Thus, our comments may not directly apply to particular subroutines.

In assessing the effectiveness of various algorithms we will be concerned with the following attributes, listed in decreasing order of importance: generality, reliability, stability, accuracy, efficiency, storage requirements, ease of use, and simplicity. We would consider an algorithm completely satisfactory if it could be used as the basis for a general purpose subroutine which meets the standards of quality software now available for linear algebraic equations, matrix eigenvalues, and initial value problems for nonlinear ordinary differential equations. By these standards, none of the algorithms we know of are completely satisfactory, although some are much better than others.

Generality means that the method is applicable to wide classes of matrices. For example, a method which works only on matrices with distinct eigenvalues will not be highly regarded.

When defining terms like reliability, stability, and accuracy, it is important to distinguish between the inherent sensitivity of the underlying problem and the error properties of a particular algorithm for solving that problem. Trying to find the inverse of a nearly singular matrix, for example, is an inherently sensitive problem. Such problems are said to be poorly posed or badly conditioned. No algorithm working with finite precision arithmetic can be expected to obtain a computed inverse that is not contaminated by large errors.

An algorithm is said to be reliable if it gives some warning whenever it introduces excessive errors. For example, Gaussian elimination without some form of pivoting is an unreliable algorithm for inverting a matrix. Roundoff errors can be magnified by large multipliers to the point where they can make the computed result completely erroneous, but there is no indication of the difficulty.

An algorithm is stable if it does not introduce any more sensitivity to perturbation than is inherent in the underlying problem. A stable algorithm produces an answer which is exact for a problem close to the given one. A method can be stable and still not produce accurate results if small changes in the data cause large changes in the answer. A method can be unstable and still be reliable if the instability can be detected. For example, Gaussian elimination with either partial or complete pivoting must be regarded as a mildly unstable algorithm because there is a possibility that the matrix elements will grow during the elimination and the resulting roundoff errors will not be small when compared with the original data. In practice, however, such growth is rare and can be detected.

The accuracy of an algorithm refers primarily to the error introduced by truncating infinite series or terminating iterations. It is one component, but not the only component, of the accuracy of the computed answer. Often, using more computer time will increase accuracy provided the method is stable. For example, the accuracy of an iterative method for solving a system of equations can be controlled by changing the number of iterations.

Efficiency is measured by the amount of computer time required to solve a particular problem. There are several problems to distinguish. For example, computing only e^A is different from computing e^{tA} for several values of t. Methods which use some decomposition of A (independent of t) might be more efficient for the second problem. Other methods may be more efficient for computing $e^{tA}x_0$ for one or several values of t. We are primarily concerned with the order of magnitude of the work involved. In matrix eigenvalue computation, for example, a method which required $O(n^4)$ time would be considered grossly inefficient because the usual methods require only $O(n^3)$.

In estimating the time required by matrix computations it is traditional to estimate the number of multiplications and then employ some factor to account for the other operations. We suggest making this slightly more precise by defining a basic floating point operation, or "flop," to be the time required for a particular computer system to execute the FORTRAN statement

$$A(I,J) = A(I,J) + T^*A(I,K).$$

This involves one floating point multiplication, one floating point addition, a few subscript and index calculations, and a few storage references. We can then say, for example, that Gaussian elimination requires $n^3/3$ flops to solve an n-by-n linear system $Ax = b$.

The eigenvalues of A play a fundamental role in the study of e^{tA} even though they may not be involved in a specific algorithm. For example, if all the eigenvalues lie in the open left half plane, then $e^{tA} \to 0$ as $t \to \infty$. This property is often called "stability," but we will reserve the use of this term for describing numerical properties of algorithms.

Several particular classes of matrices lead to special algorithms. If A is symmetric, then methods based on eigenvalue decompositions are particularly effective. If the original problem involves a single, nth order differential equation which has been rewritten as a system of first order equations in the standard way, then A is a companion matrix and other special algorithms are appropriate.

The inherent difficulty of finding effective algorithms for the matrix exponential is based in part on the following dilemma. Attempts to exploit the special properties of the differential equation lead naturally to the eigenvalues λ_i and eigenvectors v_i of A and to the representation

$$x(t) = \sum_{i=1}^{n} \alpha_i e^{\lambda_i t} v_i.$$

However, it is not always possible to express $x(t)$ in this way. If there are confluent eigenvalues, then the coefficients α_i in the linear combination may have to be polynomials in t. In practical computation with inexact data and inexact arithmetic, the gray area where the eigenvalues are nearly confluent leads to loss of accuracy. On the other hand, algorithms which avoid use of the eigenvalues tend to require considerably

more computer time for any particular problem. They may also be adversely affected by roundoff error in problems where the matrix tA has large elements.

These difficulties can be illustrated by a simple 2-by-2 example,

$$A = \begin{bmatrix} \lambda & \alpha \\ 0 & \mu \end{bmatrix}.$$

The exponential of this matrix is

$$e^{tA} = \begin{bmatrix} e^{\lambda t} & \alpha \dfrac{e^{\lambda t} - e^{\mu t}}{\lambda - \mu} \\ 0 & e^{\mu t} \end{bmatrix}.$$

Of course, when $\lambda = \mu$, this representation must be replaced by

$$e^{tA} = \begin{bmatrix} e^{\lambda t} & \alpha t e^{\lambda t} \\ 0 & e^{\lambda t} \end{bmatrix}.$$

There is no serious difficulty when λ and μ are exactly equal, or even when their difference can be considered negligible. The degeneracy can be detected and the resulting special form of the solution invoked. The difficulty comes when $\lambda - \mu$ is small but not negligible. Then, if the divided difference

$$\frac{e^{\lambda t} - e^{\mu t}}{\lambda - \mu}$$

is computed in the most obvious way, a result with a large relative error is produced. When multiplied by α, the final computed answer may be very inaccurate. Of course, for this example, the formula for the off-diagonal element can be written in other ways which are more stable. However, when the same type of difficulty occurs in nontriangular problems, or in problems that are larger than 2-by-2, its detection and cure is by no means easy.

The example also illustrates another property of e^{tA} which must be faced by any successful algorithm. As t increases, the elements of e^{tA} may grow before they decay. If λ and μ are both negative and α is fairly large, the graph in Figure 1 is typical.

Several algorithms make direct or indirect use of the identity

$$e^{sA} = (e^{sA/m})^m.$$

The difficulty occurs when s/m is under the hump but s is beyond it, for then

$$\|e^{sA}\| \ll \|e^{sA/m}\|^m.$$

Unfortunately, the roundoff errors in the mth power of a matrix, say B^m, are usually small relative to $\|B\|^m$ rather than $\|B^m\|$. Consequently, any algorithm which tries to pass over the hump by repeated multiplications is in difficulty.

Finally, the example illustrates the special nature of symmetric matrices. A is symmetric if and only if $\alpha = 0$, and then the difficulties with multiple eigenvalues and the hump both disappear. We will find later that multiple eigenvalue and hump problems do not exist when A is a normal matrix.

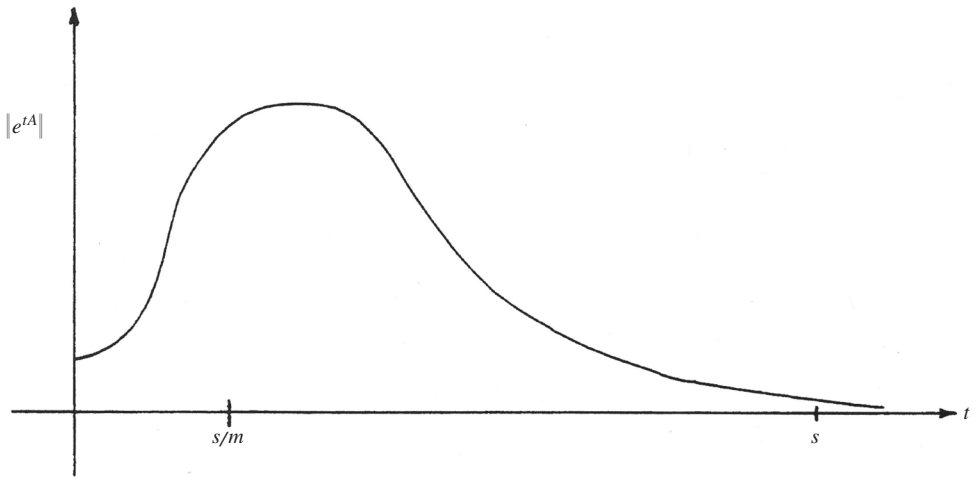

Fig. 1 *The "hump."*

It is convenient to review some conventions and definitions at this time. Unless otherwise stated, all matrices are n-by-n. If $A = (a_{ij})$, we have the notions of transpose, $A^T = (a_{ji})$, and conjugate transpose, $A^* = (\overline{a_{ji}})$. The following types of matrices have an important role to play:

$$\begin{aligned}
&A \text{ symmetric } \leftrightarrow A^T = A,\\
&A \text{ Hermitian } \leftrightarrow A^* = A,\\
&A \text{ normal } \leftrightarrow A^*A = AA^*,\\
&Q \text{ orthogonal } \leftrightarrow Q^TQ = I,\\
&Q \text{ unitary } \leftrightarrow Q^*Q = I,\\
&T \text{ triangular } \leftrightarrow t_{ij} = 0, \qquad i > j,\\
&D \text{ diagonal } \leftrightarrow d_{ij} = 0, \qquad i \neq j.
\end{aligned}$$

Because of the convenience of unitary invariance, we shall work exclusively with the 2-norm:

$$\|x\| = \left[\sum_{i=1}^{n} |x_i|^2\right]^{1/2}, \qquad \|A\| = \max_{\|x\|=1} \|Ax\|.$$

However, all our results apply with minor modification when other norms are used.

The condition of an invertible matrix A is denoted by $\text{cond}(A)$ where

$$\text{cond}(A) = \|A\|\|A^{-1}\|.$$

Should A be singular, we adopt the convention that it has infinite condition. The commutator of two matrices B and C is $[B, C] = BC - CB$.

Two matrix decompositions are of importance. The Schur decomposition states that for any matrix A, there exists a unitary Q and a triangular T, such that

$$Q^*AQ = T.$$

If $T = (t_{ij})$, then the eigenvalues of A are $t_{11}, \ldots, t_{nn}$.

The Jordan canonical form decomposition states that there exists an invertible P such that

$$P^{-1}AP = J,$$

where J is a direct sum, $J = J_1 \oplus \cdots \oplus J_k$, of Jordan blocks

$$J_i = \begin{bmatrix} \lambda_i & 1 & 0 & \cdots & 0 \\ 0 & \lambda_i & 1 & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ & & & & 1 \\ 0 & 0 & 0 & \cdots & \lambda_i \end{bmatrix} \qquad (m_i\text{-by-}m_i).$$

The λ_i are eigenvalues of A. If any of the m_i are greater than 1, A is said to be defective. This means that A does not have a full set of n linearly independent eigenvectors. A is derogatory if there is more than one Jordan block associated with a given eigenvalue.

2. The Sensitivity of the Problem. It is important to know how sensitive a quantity is before its computation is attempted. For the problem under consideration we are interested in the relative perturbation

$$\phi(t) = \frac{\|e^{t(A+E)} - e^{tA}\|}{\|e^{tA}\|}.$$

In the following three theorems we summarize some upper bounds for $\phi(t)$ which are derived in Van Loan [32].

THEOREM 1. *If* $\alpha(A) = \max\{\mathrm{Re}(\lambda) | \lambda$ *an eigenvalue of* $A\}$ *and* $\mu(A) = \max\{\mu | \mu$ *an eigenvalue of* $(A^* + A)/2\}$, *then*

$$\phi(t) \leqq t\|E\| \exp[\mu(A) - \alpha(A) + \|E\|]t \qquad (t \geqq 0).$$

The scalar $\mu(A)$ *is the "log norm" of* A *(associated with the* 2*-norm) and has many interesting properties* [35], [36], [37], [38], [39], [40], [41], [42]. *In particular,* $\mu(A) \geqq \alpha(A)$.

THEOREM 2. *If* $A = PJP^{-1}$ *is the Jordan decomposition of* A *and* m *is the dimension of the largest Jordan block in* J, *then*

$$\phi(t) \leqq t\|E\| M_J(t)^2 e^{M_J(t)\|E\|t} \qquad (t \geqq 0),$$

where

$$M_J(t) = m \ \mathrm{cond}(P) \max_{0 \leqq j \leqq m-1} t^j/j!.$$

THEOREM 3. *If* $A = Q(D + N)Q^*$ *is the Schur decomposition of* A *with* D *diagonal and* N *strictly upper triangular* $(n_{ij} = 0, i \geqq j)$, *then*

$$\phi(t) \leqq t\|E\| M_S(t)^2 e^{M_S(t)\|E\|t} \qquad (t \geqq 0),$$

where

$$M_S(t) = \sum_{k=0}^{n-1} (\|N\|t)^k/k!.$$

To obtain a corollary to any of these theorems one can show that if A is normal, then

$$\phi(t) \leqq t\|E\|e^{\|E\|t}.$$

This shows that the perturbation bounds on $\phi(t)$ for normal matrices are as small as can be expected. Furthermore, when A is normal, $\|e^{sA}\| = \|e^{sA/m}\|^m$ for all positive integers m implying that the "hump" phenomenon does not exist. These observations lead us to conclude that the e^A problem is "well conditioned" when A is normal.

It is rather more difficult to characterize those A for which e^{tA} is very sensitive to changes in A. The bound in Theorem 2 suggests that this might be the case when A has a poorly conditioned eigensystem as measured by $\text{cond}(P)$. This is related to a large $M_S(t)$ in Theorem 3 or a positive $\mu(A) - \alpha(A)$ in Theorem 1. It is unclear what the precise connection is between these situations and the hump phenomena we described in the introduction.

Some progress can be made in understanding the sensitivity of e^{tA} by defining the "matrix exponential condition number" $\nu(A,t)$:

$$\nu(A,t) = \max_{\|E\|=1} \left\| \int_0^t e^{(t-s)A} E e^{sA}\, ds \right\| \frac{\|A\|}{\|e^{tA}\|}.$$

A discussion of $\nu(A,t)$ can be found in [32]. One can show that there exists a perturbation E such that

$$\phi(t) \cong \frac{\|E\|}{\|A\|}\nu(A,t).$$

This indicates that if $\nu(A,t)$ is large, small changes in A can induce relatively large changes in e^{tA}. It is easy to verify that

$$\nu(A,t) \geqq t\|A\|,$$

with equality if and only if A is normal. When A is not normal, $\nu(A,t)$ can grow like a high degree polynomial in t.

3. Series Methods. The common theme of what we call series methods is the direct application to matrices of standard approximation techniques for the scalar function e^t. In these methods, neither the order of the matrix nor its eigenvalues plays a direct role in the actual computations.

METHOD 1. TAYLOR SERIES. The definition

$$e^A = I + A + A^2/2! + \cdots$$

is, of course, the basis for an algorithm. If we momentarily ignore efficiency, we can simply sum the series until adding another term does not alter the numbers stored in the computer. That is, if

$$T_k(A) = \sum_{j=0}^{k} A^j/j!$$

and $\text{fl}[T_k(A)]$ is the matrix of floating point numbers obtained by computing $T_k(A)$ in floating point arithmetic, then we find K so that $\text{fl}[T_K(A)] = \text{fl}[T_{K+1}(A)]$. We then take $T_K(A)$ as our approximation to e^A.

Such an algorithm is known to be unsatisfactory even in the scalar case [4] and our main reason for mentioning it is to set a clear lower bound on possible performance. To illustrate the most serious shortcoming, we implemented this algorithm on the IBM 370 using "short" arithmetic, which corresponds to a relative accuracy of $16^{-5} \cong 0.95 \cdot 10^{-6}$. We input

$$A = \begin{bmatrix} -49 & 24 \\ -64 & 31 \end{bmatrix}$$

and obtained the output

$$e^A \simeq \begin{bmatrix} -22.25880 & -1.432766 \\ -61.49931 & -3.474280 \end{bmatrix}.$$

A total of $K = 59$ terms were required to obtain convergence. There are several ways of obtaining the correct e^A for this example. The simplest is to be told how the example was constructed in the first place. We have

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & -17 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}^{-1},$$

and so

$$e^A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} e^{-1} & 0 \\ 0 & e^{-17} \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}^{-1},$$

which, to 6 decimal places, is

$$e^A \simeq \begin{bmatrix} -0.735759 & 0.551819 \\ -1.471518 & 1.103638 \end{bmatrix}.$$

The computed approximation even has the wrong sign in two components.

Of course, this example was constructed to make the method look bad. But it is important to understand the source of the error. By looking at intermediate results in the calculation we find that the two matrices $A^{16}/16!$ and $A^{17}/17!$ have elements between 10^6 and 10^7 in magnitude but of opposite signs. Because we are using a relative accuracy of only 10^{-5}, the elements of these intermediate results have absolute errors larger than the final result. So, we have an extreme example of "catastrophic cancellation" in floating point arithmetic. It should be emphasized that the difficulty is not the truncation of the series, but the truncation of the arithmetic. If we had used "long" arithmetic, which does not require significantly more time but which involves 16 digits of accuracy, then we would have obtained a result accurate to about nine decimal places.

Concern over where to truncate the series is important if efficiency is being considered. The example above required 59 terms, giving Method 1 low marks in this connection. Among several papers concerning the truncation error of Taylor series, the paper by Liou [52] is frequently cited. If δ is some prescribed error tolerance, Liou suggests choosing K large enough so that

$$\|T_K(A) - e^A\| \leqq \left(\frac{\|A\|^{K+1}}{(K+1)!}\right)\left(\frac{1}{1 - \|A\|/(K+2)}\right) \leqq \delta.$$

Moreover, when e^{tA} is desired for several different values of t, say $t = 1, \ldots, m$, he suggests an error checking procedure which involves choosing L from the same inequality with A replaced by mA and then comparing $[T_K(A)]^m x_0$ with $T_L(mA)x_0$. In related papers Everling [50] has sharpened the truncation error bound implemented by Liou, and Bickhart [46] has considered relative instead of absolute error. Unfortunately, all these approaches ignore the effects of roundoff error and so must fail in actual computation with certain matrices.

METHOD 2. PADÉ APPROXIMATION. The (p, q) Padé approximation to e^A is defined by

$$R_{pq}(A) = [D_{pq}(A)]^{-1} N_{pq}(A),$$

where

$$N_{pq}(A) = \sum_{j=0}^{p} \frac{(p+q-j)!p!}{(p+q)!j!(p-j)!} A^j$$

and

$$D_{pq}(A) = \sum_{j=0}^{q} \frac{(p+q-j)!q!}{(p+q)!j!(q-j)!} (-A)^j.$$

Nonsingularity of $D_{pq}(A)$ is assured if p and q are large enough or if the eigenvalues of A are negative. Zakian [76] and Wragg and Davies [75] considered the advantages of various representations of these rational approximations (e.g., partial fraction, continued fraction) as well as the choice of p and q to obtain prescribed accuracy.

Again, roundoff error makes Padé approximations unreliable. For large q, $D_{qq}(A)$ approaches the series for $e^{-A/2}$, whereas $N_{qq}(A)$ tends to the series for $e^{A/2}$. Hence, cancellation error can prevent the accurate determination of these matrices. Similar comments apply to general (p, q) approximants. In addition to the cancellation problem, the denominator matrix $D_{pq}(A)$ may be very poorly conditioned with respect to inversion. This is particularly true when A has widely spread eigenvalues. To see this again consider the (q, q) Padé approximants. It is not hard to show that for large enough q, we have

$$\text{cond}[D_{qq}(A)] \simeq \text{cond}(e^{-A/2}) \geqq e^{(\alpha_1 - \alpha_n)/2},$$

where $\alpha_1 \geqq \cdots \geqq \alpha_n$ are the real parts of the eigenvalues of A.

When the diagonal Padé approximants $R_{qq}(A)$ were computed for the same example used with the Taylor series and with the same single precision arithmetic, it was found that the most accurate was good to only three decimal places. This occurred with $q = 10$ and $\text{cond}[D_{qq}(A)]$ was greater than 10^4. All other values of q gave less accurate results.

Padé approximants can be used if $\|A\|$ is not too large. In this case, there are several reasons why the diagonal approximants $(p = q)$ are preferred over the off-diagonal approximants $(p \neq q)$. Suppose $p < q$. About qn^3 flops are required to evaluate $R_{pq}(A)$, an approximation which has order $p+q$. However, the same amount of work is needed to compute $R_{qq}(A)$ and this approximation has order $2q > p + q$. A similar argument can be applied to the superdiagonal approximants $(p > q)$.

There are other reasons for favoring the diagonal Padé approximants. If all the eigenvalues of A are in the left half plane, then the computed approximants with

$p > q$ tend to have larger rounding errors due to cancellation while the computed approximants with $p < q$ tend to have larger rounding errors due to badly conditioned denominator matrices $D_{pq}(A)$.

There are certain applications where the determination of p and q is based on the behavior of

$$\lim_{t\to\infty} R_{pq}(tA).$$

If all the eigenvalues of A are in the open left half plane, then $e^{tA} \to 0$ as $t \to \infty$, and the same is true for $R_{pq}(tA)$ when $q > p$. On the other hand, the Padé approximants with $q < p$, including $q = 0$, which is the Taylor series, are unbounded for large t. The diagonal approximants are bounded as $t \to \infty$.

Method 3. Scaling and squaring. The roundoff error difficulties and the computing costs of the Taylor and Padé approximants increases as $t\|A\|$ increases or as the spread of the eigenvalues of A increases. Both of these difficulties can be controlled by exploiting a fundamental property unique to the exponential function:

$$e^A = (e^{A/m})^m.$$

The idea is to choose m to be a power of two for which $e^{A/m}$ can be reliably and efficiently computed, and then to form the matrix $(e^{A/m})^m$ by repeated squaring. One commonly used criterion for choosing m is to make it the smallest power of two for which $\|A\|/m \leqq 1$. With this restriction, $e^{A/m}$ can be satisfactorily computed by either Taylor or Padé approximants. When properly implemented, the resulting algorithm is one of the most effective we know.

This approach has been suggested by many authors and we will not try to attribute it to any one of them. Among those who have provided some error analysis or suggested some refinements are Ward [72], Kammler [97], Kallstrom [116], Scraton [67], and Shah [56], [57].

If the exponential of the scaled matrix $e^{A/2^j}$ is to be approximated by $R_{qq}(A/2^j)$, then we have two parameters, q and j, to choose. In Appendix A we show that if $\|A\| \leqq 2^{j-1}$, then

$$[R_{qq}(A/2^j)]^{2^j} = e^{A+E},$$

where

$$\frac{\|E\|}{\|A\|} \leqq 8\left[\frac{\|A\|}{2^j}\right]^{2q}\left(\frac{(q!)^2}{(2q)!(2q+1)!}\right).$$

This "inverse error analysis" can be used to determine q and j in a number of ways. For example, if ε is any error tolerance, we can choose among the many (q, j) pairs for which the above inequality implies

$$\frac{\|E\|}{\|A\|} \leqq \varepsilon.$$

Since $[R_{qq}(A/2^j)]^{2^j}$ requires about $(q + j + \frac{1}{3})n^3$ flops to evaluate, it is sensible to choose the pair for which $q + j$ is minimum. Table 1 specifies these "optimum" pairs for various values of ε and $\|A\|$. By way of comparison, we have included the corresponding optimum (k, j) pairs associated with the approximant $[T_k(A/2^j)]^{2^j}$.

Table I *Optimum scaling and squaring parameters with diagonal Padé and Taylor series approximation.*

$\|A\|$ \ ε	10^{-3}	10^{-6}	10^{-9}	10^{-12}	10^{-15}
10^{-2}	$(1,0)$ $(1,0)$	$(1,0)$ $(2,1)$	$(2,0)$ $(3,1)$	$(3,0)$ $(4,1)$	$(3,0)$ $(5,1)$
10^{-1}	$(1,0)$ $(3,0)$	$(2,0)$ $(4,0)$	$(3,0)$ $(4,2)$	$(4,0)$ $(4,4)$	$(4,0)$ $(5,4)$
10^{0}	$(2,1)$ $(5,1)$	$(3,1)$ $(7,1)$	$(4,1)$ $(6,3)$	$(5,1)$ $(8,3)$	$(6,1)$ $(7,5)$
10^{1}	$(2,5)$ $(4,5)$	$(3,5)$ $(6,5)$	$(4,5)$ $(8,5)$	$(5,5)$ $(7,7)$	$(6,5)$ $(9,7)$
10^{2}	$(2,8)$ $(4,8)$	$(3,8)$ $(5,9)$	$(4,8)$ $(7,9)$	$(5,8)$ $(9,9)$	$(6,8)$ $(10,10)$
10^{3}	$(2,11)$ $(5,11)$	$(3,11)$ $(7,11)$	$(4,11)$ $(6,13)$	$(5,11)$ $(8,13)$	$(6,11)$ $(8,14)$

These pairs were determined from Corollary 1 in Appendix A and from the fact that about $(k+j-1)n^3$ flops are required to evaluate $[T_k(A/2^j)]^{2^j}$.

To read the table, for a given ε and $\|A\|$ the top ordered pair gives the optimum (q,j) associated with $[R_{qq}(A/2^j)]^{2^j}$ while the bottom ordered pair specifies the most efficient choice of (k,j) associated with $[T_k(A/2^j)]^{2^j}$.

On the basis of the table we find that Padé approximants are generally more efficient than Taylor approximants. When $\|A\|$ is small, the Padé approximant requires about one half as much work for the same accuracy. As $\|A\|$ grows, this advantage decreases because of the larger amount of scaling needed.

Relative error bounds can be derived from the above results. Noting from Appendix A that $AE = EA$, we have

$$\frac{\|[R_{qq}(A/2^j)]^{2^j} - e^A\|}{\|e^A\|} = \frac{\|e^A(e^E - I)\|}{\|e^A\|} \leqq \|E\|e^{\|E\|} \leqq \varepsilon\|A\|e^{\varepsilon\|A\|}.$$

A similar bound can be derived for the Taylor approximants.

The analysis and our table does *not* take roundoff error into account, although this is the method's weakest point. In general, the computed square of a matrix R can be severely affected by arithmetic cancellation since the rounding errors are small when compared to $\|R\|^2$ but not necessarily small when compared to $\|R^2\|$. Such cancellation can only happen when $\mathrm{cond}(R)$ is large because $R^{-1}R^2 = R$ implies

$$\mathrm{cond}(R) \geqq \frac{\|R\|^2}{\|R^2\|}.$$

The matrices which are repeatedly squared in this method can be badly conditioned. However, this does not necessarily imply that severe cancellation actually takes place. Moreover, it is possible that cancellation occurs only in problems which involve a large

hump. We regard it as an open question to analyze the roundoff error of the repeated squaring of $e^{A/m}$ and to relate the analysis to a realistic assessment of the sensitivity of e^A.

In his implementation of scaling and squaring Ward [72] is aware of the possibility of cancellation. He computes an a posteriori bound for the error, including the effects of both truncation and roundoff. This is certainly preferable to no error estimate at all, but it is not completely satisfactory. A large error estimate could be the result of any of three distinct difficulties:

(i) The error estimate is a severe overestimate of the true error, which is actually small. The algorithm is stable but the estimate is too pessimistic.

(ii) The true error is large because of cancellation in going over the hump, but the problem is not sensitive. The algorithm is unstable and another algorithm might produce a more accurate answer.

(iii) The underlying problem is inherently sensitive. No algorithm can be expected to produce a more accurate result.

Unfortunately, it is currently very difficult to distinguish among these three situations.

METHOD 4. CHEBYSHEV RATIONAL APPROXIMATION. Let $c_{qq}(x)$ be the ratio of two polynomials each of degree q and consider $\max_{0 \leqq x < \infty} |c_{qq}(x) - e^{-x}|$. For various values of q, Cody, Meinardus, and Varga [62] have determined the coefficients of the particular c_{qq} which minimizes this maximum. Their results can be directly translated into bounds for $\|c_{qq}(A) - e^A\|$ when A is Hermitian with eigenvalues on the negative real axis. The authors are interested in such matrices because of an application to partial differential equations. Their approach is particularly effective for the sparse matrices which occur in such applications.

For non-Hermitian (nonnormal) A, it is hard to determine how well $c_{qq}(A)$ approximates e^A. If A has an eigenvalue λ off the negative real axis, it is possible for $c_{qq}(\lambda)$ to be a poor approximation to e^λ. This would imply that $c_{qq}(A)$ is a poor approximation to e^A since

$$\|e^A - c_{qq}(A)\| \geqq |e^\lambda - c_{qq}(\lambda)|.$$

These remarks prompt us to emphasize an important facet about approximation of the matrix exponential, namely, there is more to approximating e^A than just approximating e^z at the eigenvalues of A. It is easy to illustrate this with Padé approximation. Suppose

$$A = \begin{bmatrix} 0 & 6 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Since all of the eigenvalues of A are zero, $R_{11}(z)$ is a perfect approximation to e^z at the eigenvalues. However,

$$R_{11}(A) = \begin{bmatrix} 1 & 6 & 18 & 54 \\ 0 & 1 & 6 & 18 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

whereas

$$e^A = \begin{bmatrix} 1 & 6 & 18 & 36 \\ 0 & 1 & 6 & 18 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and thus

$$\|e^A - R_{11}(A)\| = 18.$$

These discrepancies arise from the fact that A is not normal. The example illustrates that nonnormality exerts a subtle influence upon the methods of this section even though the eigensystem per se is not explicitly involved in any of the algorithms.

4. Ordinary Differential Equation Methods. Since e^{tA} and $e^{tA}x_0$ are solutions to ordinary differential equations (ODEs), it is natural to consider methods based on numerical integration. Very sophisticated and powerful methods for the numerical solution of general nonlinear differential equations have been developed in recent years. All worthwhile codes have automatic step size control and some of them automatically vary the order of approximation as well. Methods based on single step formulas, multistep formulas, and implicit multistep formulas each have certain advantages. When used to compute e^{tA} all these methods are easy to use and require very little additional programming or other thought. The primary disadvantage is a relatively high cost in computer time.

The ODE programs are designed to solve a single system

$$\dot{x} = f(x,t), \qquad x(0) = x_0$$

and to obtain the solution at many values of t. With $f(x,t) = Ax$ the kth column of e^{tA} can be obtained by setting x_0 to the kth column of the identity matrix. All the methods involve a sequence of values $0 = t_0, t_1, \ldots, t_j = t$ with either fixed or variable step size $h_i = t_{i+1} - t_i$. They all produce vectors x_i which approximate $x(t_i)$.

Method 5. General purpose ODE solver. Most computer center libraries contain programs for solving initial value problems in ODEs. Very few libraries contain programs that compute e^{tA}. Until the latter programs are more readily available, undoubtedly the easiest and, from the programmer's point of view, the quickest way to compute a matrix exponential is to call upon a general purpose ODE solver. This is obviously an expensive luxury since the ODE routine does not take advantage of the linear, constant coefficient nature of our special problem.

We have run a very small experiment in which we have used three recently developed ODE solvers to compute the exponentials of about a dozen matrices and have measured the amount of work required. The programs are as follows:

(1) RKF45. Written by Shampine and Watts [108], this program uses the Fehlberg formulas of the Runge–Kutta type. Six function evaluations are required per step. The resulting formula is fifth order with automatic step size control. (See also [4].)

(2) DE/STEP. Written by Shampine and Gordon [107], this program uses variable order, variable step Adams predictor-corrector formulas. Two function evaluations are required per step.

(3) IMPSUB. Written by Starner [109], this program is a modification of Gear's DIFSUB [106] and is based on implicit backward differentiation formulas intended for stiff differential equations. Starner's modifications add the ability to solve "infinitely

Table 2 *Work as a function of subroutine and local error tolerance.*

	10^{-6}	10^{-9}	10^{-12}
RKF45	217	832	3268
DE/STEP	118	160	211
IMPSUB	173	202	1510

stiff" problems in which the derivatives of some of the variables may be missing. Two function evaluations are usually required per step, but three or four may occasionally be used.

For RKF45 the output points are primarily determined by the step size selection in the program. For the other two routines, the output is produced at user-specified points by interpolation. For an n-by-n matrix A, the cost of one function evaluation is a matrix-vector multiplication or n^2 flops. The number of evaluations is determined by the length of the integration interval and the accuracy requested.

The relative performance of the three programs depends fairly strongly on the particular matrix. RKF45 often requires the most function evaluations, especially when high accuracy is sought, because its order is fixed. But it may well require the least actual computer time at modest accuracies because of its low overhead. DE/STEP indicates when it thinks a problem is stiff. If it doesn't give this indication, it usually requires the fewest function evaluations. If it does, IMPSUB may require fewer.

Table 2 gives the results for one particular matrix when we arbitrarily declare to be a "typical" nonstiff problem. The matrix is of order 3, with eigenvalues $\lambda = 3, 3, 6$; the matrix is defective. We used three different local error tolerances and integrated over $[0, 1]$. The average number of function evaluations for the three starting vectors is given in the table. These can be regarded as typical coefficients of n^2 for the single vector problem or of n^3 for the full matrix exponential; IBM 370 long arithmetic was used.

Although people concerned with the competition between various ODE solvers might be interested in the details of this table, we caution that it is the result of only one experiment. Our main reason for presenting it is to support our contention that the use of any such routine must be regarded as very inefficient. The scaling and squaring method of section 3 and some of the matrix decomposition methods of section 6 require on the order of 10 to 20 n^3 flops and they obtain higher accuracies than those obtained with $200n^3$ or more flops for the ODE solvers.

This excessive cost is due to the fact that the programs are not taking advantage of the linear, constant coefficient nature of the differential equation. They must repeatedly call for the multiplication of various vectors by the matrix A because, as far as they know, the matrix may have changed since the last multiplication.

We now consider the various methods which result from specializing general ODE methods to handle our specific problem.

Method 6. Single step ODE methods. Two of the classical techniques for the solution of differential equations are the fourth order Taylor and Runge–Kutta

methods with fixed step size. For our particular equation they become

$$x_{j+1} = \left(I + hA + \cdots + \frac{h^4}{4!}A^4 \right) x_j = T_4(hA)x_j$$

and

$$x_{j+1} = x_j + \tfrac{1}{6}k_1 + \tfrac{1}{3}k_2 + \tfrac{1}{3}k_3 + \tfrac{1}{6}k_4,$$

where $k_1 = hAx_j$, $k_2 = hA(x_j + \frac{1}{2}k_1)$, $k_3 = hA(x_j + \frac{1}{2}k_2)$, and $k_4 = hA(x_j + k_3)$. A little manipulation reveals that in this case, the two methods would produce identical results were it not for roundoff error. As long as the step size is fixed, the matrix $T_4(hA)$ need be computed just once and then x_{j+1} can be obtained from x_j with just one matrix-vector multiplication. The standard Runge–Kutta method would require 4 such multiplications per step.

Let us consider $x(t)$ for one particular value of t, say $t = 1$. If $h = 1/m$, then

$$x(1) = x(mh) \simeq x_m = [T_4(hA)]^m x_0.$$

Consequently, there is a close connection between this method and Method 3 which involved scaling and squaring [54], [60]. The scaled matrix is hA and its exponential is approximated by $T_4(hA)$. However, even if m is a power of 2, $[T_4(hA)]^m$ is usually not obtained by repeated squaring. The methods have roughly the same roundoff error properties and so there seem to be no important advantages for Runge–Kutta with fixed step size.

Let us now consider the possibility of varying the step size. A simple algorithm might be based on a variable step Taylor method. In such a method, two approximations to x_{j+1} would be computed and their difference used to choose the step size. Specifically, let ε be some prescribed local relative error tolerance and define x_{j+1} and x^*_{j+1} by

$$x_{j+1} = T_5(h_j A)x_j,$$
$$x^*_{j+1} = T_4(h_j A)x_j.$$

One way of determining h_j is to require

$$\|x_{j+1} - x^*_{j+1}\| \simeq \varepsilon \|x_j\|.$$

Notice that we are using a fifth order formula to compute the approximation, and a fourth order formula to control step size.

At first glance, this method appears to be considerably less efficient than one with fixed step size because the matrices $T_4(h_j A)$ and $T_5(h_j A)$ cannot be precomputed. Each step requires $5\,n^2$ flops. However, in those problems which involve large "humps" as described in section 1, a smaller step is needed at the beginning of the computation than at the end. If the step size changes by a factor of more than 5, the variable step method will require less work.

The method does provide some insight into the costs of more sophisticated integrators. Since

$$x_{j+1} - x^*_{j+1} = \frac{h_j^5 A^5}{5!} x_j,$$

we see that the required step size is given approximately by

$$h_j \simeq \left[\frac{5!\varepsilon}{\|A^5\|}\right]^{1/5}.$$

The work required to integrate over some fixed interval is proportional to the inverse of the average step size. So, if we decrease the tolerance ε from, say, 10^{-6} to 10^{-9}, then the work is increased by a factor of $(10^3)^{1/5}$, which is about 4. This is typical of any fifth order error estimate—asking for 3 more figures roughly quadruples the work.

METHOD 7. MULTISTEP ODE SOLVER. As far as we know, the possibility of specializing multistep methods, such as those based on the Adams formulas, to linear, constant coefficient problems has not been explored in detail. Such a method would not be equivalent to scaling and squaring because the approximate solution at a given time is defined in terms of approximate solutions at several previous times. The actual algorithm would depend upon how the starting vectors are obtained, and how the step size and order are determined. It is conceivable that such an algorithm might be effective, particularly for problems which involve a single vector, output at many values of t, large n, and a hump.

The problems associated with roundoff error have not been of as much concern to designers of differential equation solvers as they have been to designers of matrix algebra algorithms since the accuracy requested of ODE solvers is typically less than full machine precision. We do not know what effect rounding errors would have in a problem with a large hump.

5. Polynomial Methods. Let the characteristic polynomial of A be

$$c(z) = \det(zI - A) = z^n - \sum_{k=0}^{n-1} c_k z^k.$$

From the Cayley–Hamilton theorem $c(A) = 0$ and hence

$$A^n = c_0 I + c_1 A + \cdots + c_{n-1} A^{n-1}.$$

It follows that any power of A can be expressed in terms of I, $A, \ldots, A^{n-1}$:

$$A^k = \sum_{j=0}^{n-1} \beta_{kj} A^j.$$

This implies that e^{tA} is a polynomial in A with analytic coefficients in t:

$$\begin{aligned} e^{tA} = \sum_{k=0}^{\infty} \frac{t^k A^k}{k!} &= \sum_{k=0}^{\infty} \frac{t^k}{k!} \left[\sum_{j=0}^{n-1} \beta_{kj} A^j\right] \\ &= \sum_{j=0}^{n-1} \left[\sum_{k=0}^{\infty} \beta_{kj} \frac{t^k}{k!}\right] A^j \\ &\equiv \sum_{j=0}^{n-1} \alpha_j(t) A^j. \end{aligned}$$

The methods of this section involve this kind of exploitation of the characteristic polynomial.

METHOD 8. CAYLEY–HAMILTON. Once the characteristic polynomial is known, the coefficients β_{kj} which define the analytic functions $\alpha_j(t) = \sum \beta_{kj} t^k/k!$ can be generated as follows:

$$\beta_{kj} = \begin{cases} \delta_{kj} & (k < n) \\ c_j & (k = n) \\ c_0\beta_{k-1,n-1} & (k > n, j = 0) \\ c_j\beta_{k-1,n-1} + \beta_{k-1,j-1} & (k > n, j > 0). \end{cases}$$

One difficulty is that these recursive formulas for the β_{kj} are very prone to roundoff error. This can be seen in the 1-by-1 case. If $A = (\alpha)$, then $\beta_{k0} = \alpha^k$ and $\alpha_0(t) = \sum(\alpha t)^k/k!$ is simply the Taylor series for $e^{\alpha t}$. Thus, our criticisms of Method 1 apply. In fact, if $\alpha t = -6$, no partial sum of the series for $e^{\alpha t}$ will have any significant digits when IBM 370 short arithmetic is used.

Another difficulty is the requirement that the characteristic polynomial must be known. If $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A, then $c(z)$ could be computed from $c(z) = \prod_1^n (z - \lambda_i)$. Although the eigenvalues could be stably computed, it is unclear whether the resulting c_j would be acceptable. Other methods for computing $c(z)$ are discussed in Wilkinson [14]. It turns out that methods based upon repeated powers of A and methods based upon formulas for the c_j in terms of various symmetric functions are unstable in the presence of roundoff error and expensive to implement. Techniques based upon similarity transformations break down when A is nearly derogatory. We shall have more to say about these difficulties in connection with Methods 12 and 13.

In Method 8 we attempted to expand e^{tA} in terms of the matrices $I, A, \ldots, A^{n-1}$. If $\{A_0, \ldots, A_{n-1}\}$ is some other set of matrices which span the same subspace, then there exist analytic functions $\beta_j(t)$ such that

$$e^{tA} = \sum_{j=0}^{n-1} \beta_j(t) A_j.$$

The convenience of this formula depends upon how easily the A_j and $\beta_j(t)$ can be generated. If the eigenvalues $\lambda_1, \ldots, \lambda_n$ of A are known, we have the following three methods.

METHOD 9. LAGRANGE INTERPOLATION.

$$e^{tA} = \sum_{j=0}^{n-1} e^{\lambda_j t} \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{(A - \lambda_k I)}{(\lambda_j - \lambda_k)}.$$

METHOD 10. NEWTON INTERPOLATION.

$$e^{tA} = e^{\lambda_1 t} I + \sum_{j=2}^{n} [\lambda_1, \ldots, \lambda_j] \prod_{k=1}^{j-1} (A - \lambda_k I).$$

The divided differences $[\lambda_1, \ldots, \lambda_j]$ depend on t and are defined recursively by

$$[\lambda_1, \lambda_2] = (e^{\lambda_1 t} - e^{\lambda_2 t})/(\lambda_1 - \lambda_2),$$

$$[\lambda_1, \ldots, \lambda_{k+1}] = \frac{[\lambda_1, \ldots, \lambda_k] - [\lambda_2, \ldots, \lambda_{k+1}]}{\lambda_1 - \lambda_{k+1}} \qquad (k \geqq 2).$$

We refer to MacDuffee [9] for a discussion of these formulas in the confluent eigenvalue case.

METHOD 11. VANDERMONDE. There are other methods for computing the matrices

$$A_j = \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{(A - \lambda_k I)}{(\lambda_j - \lambda_k)}$$

which were required in Method 9. One of these involves the Vandermonde matrix

$$V = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \cdots & \lambda_n \\ \vdots & \vdots & & \vdots \\ \lambda_1^{n-1} & \lambda_2^{n-1} & \cdots & \lambda_n^{n-1} \end{bmatrix}.$$

If ν_{jk} is the (j,k) entry of V^{-1}, then

$$A_j = \sum_{k=1}^{n} \nu_{jk} A^{k-1},$$

and

$$e^{tA} = \sum_{j=1}^{n} e^{\lambda_j t} A_j.$$

When A has repeated eigenvalues, the appropriate confluent Vandermonde matrix is involved. Closed expressions for the ν_{jk} are available and Vidysager [92] has proposed their use.

Methods 9, 10, and 11 suffer on several accounts. They are $O(n^4)$ algorithms making them prohibitively expensive except for small n. If the spanning matrices $A_0, \ldots, A_{n-1}$ are saved, then storage is n^3, which is an order of magnitude greater than the amount of storage required by any "nonpolynomial" method. Furthermore, even though the formulas which define Methods 9, 10, and 11 have special form in the confluent case, we do not have a satisfactory situation. The "gray" area of near confluence poses difficult problems which are best discussed in the next section on decomposition techniques.

The next two methods of this section do not require the eigenvalues of A and thus appear to be free of the problems associated with confluence. However, equally formidable difficulties attend these algorithms.

METHOD 12. INVERSE LAPLACE TRANSFORMS. If $\mathcal{L}[e^{tA}]$ is the Laplace transform of the matrix exponential, then

$$\mathcal{L}[e^{tA}] = (sI - A)^{-1}.$$

The entries of this matrix are rational functions of s. In fact,

$$(sI - A)^{-1} = \sum_{k=0}^{n-1} \frac{s^{n-k-1}}{c(s)} A_k,$$

where $c(s) = \det(sI - A) = s^n - \sum_{k=0}^{n-1} c_k s^k$ and for $k = 1, \ldots, n$,

$$c_{n-k} = -\text{trace}(A_{k-1}A)/k, \qquad A_k = A_{k-1}A - c_{n-k}I \qquad (A_0 = I).$$

These recursions were derived by Leverrier and Faddeeva [3] and can be used to evaluate e^{tA}:

$$e^{tA} = \sum_{k=0}^{n-1} \mathcal{L}^{-1}[s^{n-k-1}/c(s)]A_k.$$

The inverse transforms $\mathcal{L}^{-1}[s^{n-k-1}/c(s)]$ can be expressed as a power series in t. Liou [102] suggests evaluating these series using various recursions involving the c_k. We suppress the details of this procedure because of its similarity to Method 8. There are other ways Laplace transforms can be used to evaluate e^{tA} [78], [80], [88], [89], [93]. By and large, these techniques have the same drawbacks as Methods 8–11. They are $O(n^4)$ for general matrices and may be seriously affected by roundoff error.

METHOD 13. COMPANION MATRIX. We now discuss techniques which involve the computation of e^C where C is a companion matrix:

$$C = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ & & & & 1 \\ c_0 & c_1 & c_2 & \cdots & c_{n-1} \end{bmatrix}.$$

Companion matrices have some interesting properties which various authors have tried to exploit:

(i) C is sparse.

(ii) The characteristic polynomial of C is $c(z) = z^n - \sum_{k=0}^{n-1} c_k z^k$.

(iii) If V is the Vandermonde matrix of eigenvalues of C (see Method 11), then $V^{-1}CV$ is in Jordan form. (Confluent Vandermonde matrices are involved in the multiple eigenvalue case.)

(iv) If A is not derogatory, then it is similar to a companion matrix; otherwise it is similar to a direct sum of companion matrices.

Because C is sparse, small powers of C cost considerably less than the usual n^3 flops. Consequently, one could implement Method 3 (scaling and squaring) with a reduced amount of work.

Since the characteristic polynomial of C is known, one can apply Method 8 or various other techniques which involve recursions with the c_k. However, this is not generally advisable in view of the catastrophic cancellation that can occur.

As we mentioned during our discussion of Method 11, the closed expression for V^{-1} is extremely sensitive. Because V^{-1} is so poorly conditioned, exploitation of property (iii) will generally yield a poor estimate of e^A.

If $A = YCY^{-1}$, then from the series definition of the matrix exponential it is easy to verify that

$$e^A = Ye^CY^{-1}.$$

Hence, property (iv) leads us to an algorithm for computing the exponential of a general matrix. Although the reduction of A to companion form is a rational process, the algorithms for accomplishing this are extremely unstable and should be avoided [14].

We mention that if the original differential equation is actually a single nth order equation written as a system of first order equations, then the matrix is already in

companion form. Consequently, the unstable reduction is not necessary. This is the only situation in which companion matrix methods should be considered.

We conclude this section with an interesting aside on computing e^H where $H = (h_{ij})$ is lower Hessenberg ($h_{ij} = 0, j > i + 1$). Notice that companion matrices are lower Hessenberg. Our interest in computing e^H stems from the fact that any real matrix A is orthogonally similar to a lower Hessenberg matrix. Hence, if

$$A = QHQ^T, \qquad Q^TQ = I,$$

then

$$e^A = Qe^HQ^T.$$

Unlike the reduction to companion form, this factorization can be stably computed using the EISPACK routine ORTHES [113].

Now, let f_k denote the kth column of e^H. It is easy to verify that

$$Hf_k = \sum_{i=k-1}^{n} h_{ik}f_i \qquad (k \geqq 2)$$

by equating the kth columns in the matrix identity $He^H = e^HH$. If none of the superdiagonal entries $h_{k-1,k}$ are zero, then once f_n is known, the other f_k follow immediately from

$$f_{k-1} = \frac{1}{h_{k-1,k}}\left[Hf_k - \sum_{i=k}^{n} h_{ik}f_i\right].$$

Similar recursive procedures have been suggested in connection with computing e^C [104]. Since f_n equals $x(1)$ where $x(t)$ solves $Hx = \dot{x}$, $x(0) = (0, \ldots, 0, 1)^T$, it could be found using one of the ODE methods in the previous section.

There are ways to recover in the above algorithm should any of the $h_{k-1,k}$ be zero. However, numerically the problem is when we have a small but nonnegligible $h_{k-1,k}$. In this case rounding errors involving a factor of $1/h_{k-1,k}$ will occur, precluding the possibility of an accurate computation of e^H.

In summary, methods for computing e^A which involve the reduction of A to companion or Hessenberg form are not attractive. However, there are other matrix factorizations which can be more satisfactorily exploited in the course of evaluating e^A, and these will be discussed in the next section.

6. Matrix Decomposition Methods. The methods which are likely to be most efficient for problems involving large matrices and repeated evaluation of e^{tA} are those which are based on factorizations or decompositions of the matrix A. If A happens to be symmetric, then all these methods reduce to a simple very effective algorithm.

All the matrix decompositions are based on similarity transformations of the form

$$A = SBS^{-1}.$$

As we have mentioned, the power series definition of e^{tA} implies

$$e^{tA} = Se^{tB}S^{-1}.$$

The idea is to find an S for which e^{tB} is easy to compute. The difficulty is that S may be close to singular, which means that cond(S) is large.

METHOD 14. EIGENVECTORS. The naive approach is to take S to be the matrix whose columns are eigenvectors of A, that is, $S = V$ where

$$V = [v_1|\cdots|v_n]$$

and

$$Av_j = \lambda_j v_j, \qquad j = 1, \ldots, n.$$

These n equations can be written

$$AV = VD,$$

where $D = \text{diag}(\lambda_1, \ldots, \lambda_n)$. The exponential of D is trivial to compute assuming we have a satisfactory method for computing the exponential of a scalar:

$$e^{tD} = \text{diag}(e^{\lambda_1 t}, \ldots, e^{\lambda_n t}).$$

Since V is nonsingular we have $e^{tA} = Ve^{tD}V^{-1}$.

In terms of the differential equation $\dot{x} = Ax$, the same eigenvector approach takes the following form. The initial condition is a combination of the eigenvectors,

$$x(0) = \sum_{j=1}^{n} \alpha_j v_j,$$

and the solution $x(t)$ is given by

$$x(t) = \sum_{j=0}^{n} \alpha_j e^{\lambda_j t} v_j.$$

Of course, the coefficients α_j are obtained by solving a set of linear equations $V\alpha = x(0)$.

The difficulty with this approach is not confluent eigenvalues per se. For example, the method works very well when A is the identity matrix, which has an eigenvalue of the highest possible multiplicity. It also works well for any other symmetric matrix because the eigenvectors can be chosen orthogonal. If reliable subroutines such as TRED2 and TQL2 in EISPACK [113] are used, then the computed v_j will be orthogonal to the full accuracy of the computer and the resulting algorithm for e^{tA} has all the attributes we desire—except that it is limited to symmetric matrices.

The theoretical difficulty occurs when A does not have a complete set of linearly independent eigenvectors and is thus defective. In this case there is no invertible matrix of eigenvectors V and the algorithm breaks down. An example of a defective matrix is

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

A defective matrix has confluent eigenvalues, but a matrix which has confluent eigenvalues need not be defective.

In practice, difficulties occur when A is "nearly" defective. One way to make this precise is to use the condition number, $\text{cond}(V) = \|V\|\|V^{-1}\|$, of the matrix of eigenvectors. If A is nearly (exactly) defective, then $\text{cond}(V)$ is large (infinite).

Any errors in A, including roundoff errors in its computation and roundoff errors from the eigenvalue computation, may be magnified in the final result by $\text{cond}(V)$. Consequently, when $\text{cond}(V)$ is large, the computed e^{tA} will most likely be inaccurate. For example, if

$$A = \begin{bmatrix} 1+\varepsilon & 1 \\ 0 & 1-\varepsilon \end{bmatrix},$$

then

$$V = \begin{bmatrix} 1 & -1 \\ 0 & 2\varepsilon \end{bmatrix},$$
$$D = \text{diag}(1+\varepsilon, 1-\varepsilon),$$

and

$$\text{cond}(V) = O\left(\frac{1}{\varepsilon}\right).$$

If $\varepsilon = 10^{-5}$ and IBM 370 short floating point arithmetic is used to compute the exponential from the formula $e^A = Ve^DV^{-1}$, we obtain

$$\begin{bmatrix} 2.718307 & 2.750000 \\ 0 & 2.718254 \end{bmatrix}.$$

Since the exact exponential to six decimals is

$$\begin{bmatrix} 2.718309 & 2.718282 \\ 0 & 2.718255 \end{bmatrix},$$

we see that the computed exponential has errors of order 10^5 times the machine precision as conjectured.

One might feel that for this example e^A might be particularly sensitive to perturbations in A. However, when we apply Theorem 3 in section 2 to this example, we find

$$\frac{\|e^{(A+E)} - e^A\|}{\|e^A\|} \leqq 4\|E\|e^{2\|E\|},$$

independent of ε. Certainly, e^A is not overly sensitive to changes in A and so Method 14 must be regarded as unstable.

Before we proceed to the next method it is interesting to note the connection between the use of eigenvectors and Method 9, Lagrange interpolation. When the eigenvalues are distinct the eigenvector approach can be expressed

$$e^{tA} = V\ \text{diag}(e^{\lambda_j t})V^{-1} = \sum_{j=1}^{n} e^{\lambda_j t} v_j y_j^T,$$

where y_j^T is the jth row of V^{-1}. The Lagrange formula is

$$e^{tA} = \sum_{j=1}^{n} e^{\lambda_j t} A_j,$$

where

$$A_j = \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{(A - \lambda_k I)}{(\lambda_j - \lambda_k)}.$$

Because these two expressions hold for all t, the individual terms in the sum must be the same and so

$$A_j = v_j y_j^T.$$

This indicates that the A_j are, in fact, rank one matrices obtained from the eigenvectors. Thus, the $O(n^4)$ work involved in the computation of the A_j is totally unnecessary.

METHOD 15. TRIANGULAR SYSTEMS OF EIGENVECTORS. An improvement in both the efficiency and the reliability of the conventional eigenvector approach can be obtained when the eigenvectors are computed by the QR algorithm [14]. Assume temporarily that although A is not symmetric, all its eigenvalues happen to be real. The idea is to use EISPACK subroutines ORTHES and HQR2 to compute the eigenvalues and eigenvectors [113]. These subroutines produce an orthogonal matrix Q and a triangular matrix T so that

$$Q^T A Q = T.$$

Since $Q^{-1} = Q^T$, this is a similarity transformation and the desired eigenvalues occur on the diagonal of T. HQR2 next attempts to find the eigenvectors of T. This results in a matrix R and a diagonal matrix D, which is simply the diagonal part of T, so that

$$TR = RD.$$

Finally, the eigenvectors of A are obtained by a simple matrix multiplication $V = QR$.

The key observation is that R is upper triangular. In other words, the ORTHES/HQR2 path in EISPACK computes the matrix of eigenvectors by first computing its "QR" factorization. HQR2 can be easily modified to remove the final multiplication of Q and R. The availability of these two matrices has two advantages. First, the time required to find V^{-1} or to solve systems involving V is reduced. However, since this is a small fraction of the total time required, the improvement in efficiency is not very significant. A more important advantage is that $\text{cond}(V) = \text{cond}(R)$ (in the 2-norm) and that the estimation of $\text{cond}(R)$ can be done reliably and efficiently.

The effect of admitting complex eigenvalues is that R is not quite triangular but has 2-by-2 blocks on its diagonal for each complex conjugate pair. Such a matrix is called quasi-triangular and we avoid complex arithmetic with minor inconvenience.

In summary, we suspect the following algorithm to be reliable:

(1) Given A, use ORTHES and a modified HQR2 to find orthogonal Q, diagonal D, and quasi-triangular R so that

$$AQR = QRD.$$

(2) Given x_0, compute y_0 by solving

$$Ry_0 = Q^T x_0.$$

Also estimate $\text{cond}(R)$ and hence the accuracy of y_0.

(3) If cond(R) is too large, indicate that this algorithm cannot solve the problem and exit.
(4) Given t, compute $x(t)$ by

$$x(t) = Ve^{tD}y_0.$$

(If we want to compute the full exponential, then in step 2 we solve $RY = Q^T$ for Y and then use $e^{tA} = Ve^{tD}Y$ in step 4.) It is important to note that the first three steps are independent of t, and that the fourth step, which requires relatively little work, can be repeated for many values of t.

We know there are examples where the exit is taken in step 3 even though the *underlying problem* is not poorly conditioned implying that the algorithm is unstable. Nevertheless, the algorithm is reliable insofar as cond(R) enables us to assess the errors in the computed solution when that solution is found. It would be interesting to code this algorithm and compare it with Ward's scaling and squaring program for Method 3. In addition to comparing timings, the crucial question would be how often the exit in step 3 is taken and how often Ward's program returns an unacceptably large error bound.

METHOD 16. JORDAN CANONICAL FORM. In principle, the problem posed by defective eigensystems can be solved by resorting to the Jordan canonical form (JCF). If

$$A = P[J_1 \oplus \cdots \oplus J_k]P^{-1}$$

is the JCF of A, then

$$e^{tA} = P[e^{tJ_1} \oplus \cdots \oplus e^{tJ_k}]P^{-1}.$$

The exponentials of the Jordan blocks J_i can be given in closed form. For example, if

$$J_i = \begin{bmatrix} \lambda_i & 1 & 0 & 0 \\ 0 & \lambda_i & 1 & 0 \\ 0 & 0 & \lambda_i & 1 \\ 0 & 0 & 0 & \lambda_i \end{bmatrix},$$

then

$$e^{tJ_i} = e^{\lambda_i t} \begin{bmatrix} 1 & t & t^2/2! & t^3/3! \\ 0 & 1 & t & t^2/2! \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The difficulty is that the JCF cannot be computed using floating point arithmetic. A single rounding error may cause some multiple eigenvalue to become distinct or vice versa, altering the entire structure of J and P. A related fact is that there is no a priori bound on cond(P). For further discussion of the difficulties of computing the JCF, see the papers by Golub and Wilkinson [110] and Kågstrom and Ruhe [111].

METHOD 17. SCHUR. The Schur decomposition

$$A = QTQ^T$$

with orthogonal Q and triangular T exists if A has real eigenvalues. If A has complex eigenvalues, then it is necessary to allow 2-by-2 blocks on the diagonal of T or to

make Q and T complex (and replace Q^T with Q^*). The Schur decomposition can be computed reliably and quite efficiently by ORTHES and a short-ended version of HQR2. The required modifications are discussed in the EISPACK guide [113].

Once the Schur decomposition is available,

$$e^{tA} = Qe^{tT}Q^T.$$

The only delicate part is the computation of e^{tT} where T is a triangular or quasi-triangular matrix. Note that the eigenvectors of A are not required.

Computing functions or triangular matrices is the subject of a paper by Parlett [112]. If T is upper triangular with diagonal elements $\lambda_1, \ldots, \lambda_n$, then it is clear that e^{tT} is upper triangular with diagonal elements $e^{\lambda_1 t}, \ldots, e^{\lambda_n t}$. Parlett shows how to compute the off-diagonal elements of e^{tT} recursively from divided differences of the $e^{\lambda_i t}$. The example in section 1 illustrates the 2-by-2 case.

Again, the difficulty is magnification of roundoff error caused by nearly confluent eigenvalues λ_i. As a step towards handling this problem, Parlett describes a generalization of his algorithm applicable to block upper triangular matrices. The diagonal blocks are determined by clusters of nearby eigenvalues. The confluence problems do not disappear, but they are confined to the diagonal blocks where special techniques can be applied.

METHOD 18. BLOCK DIAGONAL. All methods which involve decompositions of the form

$$A = SBS^{-1}$$

involve two conflicting objectives:

1. Make B close to diagonal so that e^{tB} is easy to compute.
2. Make S well conditioned so that errors are not magnified.

The JCF places all the emphasis on the first objective, while the Schur decomposition places most of the emphasis on the second. (We would regard the decomposition with $S = I$ and $B = A$ as placing even more emphasis on the second objective.)

The block diagonal method is a compromise between these two extremes. The idea is to use a nonorthogonal, but well conditioned, S to produce a B which is triangular and block diagonal, as illustrated in Figure 2.

Each block in B involves a cluster of nearly confluent eigenvalues. The number in each cluster (the size of each block) is to be made as small as possible while maintaining some prescribed upper bound for $\text{cond}(S)$, such as $\text{cond}(S) < 100$. The choice of the bound 100 implies roughly that at most 2 significant decimal figures will be lost because of rounding errors when e^{tA} is obtained from e^{tB} via $e^{tA} = Se^{tB}S^{-1}$. A larger bound would mean the loss of more figures while a smaller bound would mean more computer time—both for the factorization itself and for the evaluation of e^{tB}.

In practice, we would expect almost all the blocks to be 1-by-1 or 2-by-2 and the resulting computation of e^{tB} to be very fast. The bound on $\text{cond}(S)$ will mean that it is occasionally necessary to have larger blocks in B, but it will insure against excessive loss of accuracy from confluent eigenvalues.

G. W. Stewart has pointed out that the grouping of the eigenvalues into clusters and the resulting block structure of B is not merely for increased speed. There can be an important improvement in accuracy. Stewart suggests expressing each block B_j in the form

$$B_j = \gamma_j I + E_j,$$

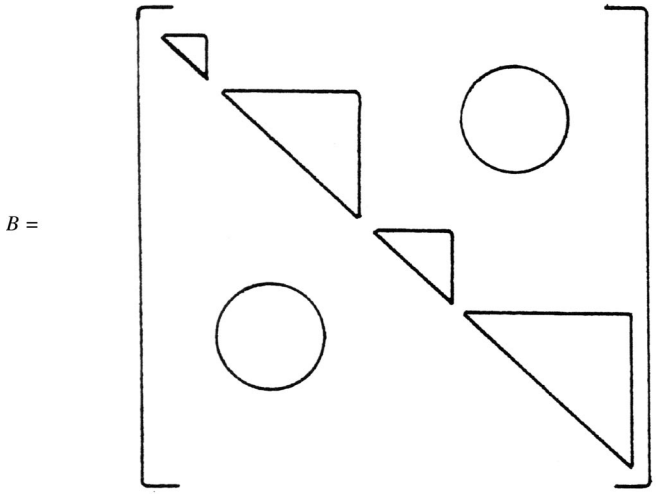

Fig. 2 *Triangular block diagonal form.*

where γ_j is the average value of the eigenvalues in the jth cluster. If the grouping has been done properly, the matrices E_j should then be nearly nilpotent in the sense that E_j^k will rapidly approach zero as k increases. Since E_j is triangular, this will certainly be true if the diagonal part of E_j is small, that is, if all the eigenvalues in the cluster are close together. But it will also be true in another important case. If

$$E_j = \begin{bmatrix} \sqrt{\varepsilon} & 1 \\ 0 & -\sqrt{\varepsilon} \end{bmatrix},$$

where ε is the computer rounding unit, then

$$E_j^2 = \begin{bmatrix} \varepsilon & 0 \\ 0 & \varepsilon \end{bmatrix}$$

can be regarded as negligible. The $\pm\sqrt{\varepsilon}$ perturbations are typical when a double, defective eigenvalue is computed with, say, HQR2.

The fact that E_j is nearly nilpotent means that e^{tB_j} can be found rapidly and accurately from

$$e^{tB_j} = e^{\gamma_j t} e^{tE_j},$$

computing e^{tE_j} by a few terms of the Taylor series.

Several researchers, including Parlett, Ruhe, and Stewart, are currently developing computer programs based on some of these ideas. The most difficult detail is the proper choice of the eigenvalue clustering. It is also important for program efficiency to avoid complex arithmetic as much as possible. When fully developed, these programs will be fairly long and complicated, but they may come close to meeting our other criteria for satisfactory methods.

Most of the computational cost lies in obtaining the basic Schur decomposition. Although this cost varies somewhat from matrix to matrix because of the iterative nature of the QR algorithm, a good average figure is $15n^3$ flops, including the further reduction to block diagonal form. Again we emphasize that the reduction is

independent of t. Once the decomposition is obtained, the calculation of e^{tA} requires about $2n^3$ flops for each t. If we require only $x(t) = e^{tA}x_0$ for various t, the equation $Sy = x_0$ should be solved once at a cost of $n^3/3$ flops, and then each $x(t)$ can be obtained with n^2 flops.

These are rough estimates. There will be differences between programs based on the Schur decomposition and those which work with the block diagonal form, but the timings should be similar because Parlett's algorithm for the exponential is very fast.

7. Splitting Methods. A most aggravating, yet interesting, property of the matrix exponential is that the familiar additive law fails unless we have commutivity:

$$e^{tB}e^{tC} = e^{t(B+C)} \Leftrightarrow BC = CB.$$

Nevertheless, the exponentials of B and C are related to that of $B + C$, for example, by the Trotter product formula [30]:

$$e^{B+C} = \lim_{m\to\infty} (e^{B/m}e^{C/m})^m.$$

Method 19. Splitting. Our colleagues M. Gunzburger and D. Gottleib suggested that the Trotter result be used to approximate e^A by splitting A into $B + C$ and then using the approximation

$$e^A \simeq (e^{B/m}e^{C/m})^m.$$

This approach to computing e^A is of potential interest when the exponentials of B and C can be accurately and efficiently computed. For example, if $B = (A + A^T)/2$ and $C = (A - A^T)/2$, then e^B and e^C can be effectively computed by the methods of section 5. For this choice we show in Appendix B that

$$\|e^A - (e^{B/m}e^{C/m})^m\| \leqq \frac{\|[A^T, A]\|}{4m} e^{\mu(A)}, \tag{7.1}$$

where $\mu(A)$ is the log norm of A as defined in section 2. In the following algorithm, this inequality is used to determine the parameter m.

(a) Set $B = (A + A^T)/2$ and $C = (A - A^T)/2$. Compute the factorization $B = Q\ \mathrm{diag}(\mu_i)Q^T (Q^TQ = I)$ using TRED2 and TQL2 [113]. Variations of these programs can be used to compute the factorization $C = UDU^T$, where $U^TU = I$ and D is the direct sum of zero matrices and real 2-by-2 blocks of the form

$$\begin{bmatrix} 0 & a \\ -a & 0 \end{bmatrix}$$

corresponding to eigenvalues $\pm ia$.

(b) Determine $m = 2^j$ such that the upper bound in (7.1) is less than some prescribed tolerance. Recall that $\mu(A)$ is the most positive eigenvalue of B and that this quantity is known as a result of step (a).

(c) Compute $X = Q\ \mathrm{diag}(e^{\mu_i/m})Q^T$ and $Y = Ue^{D/m}U^T$. In the latter computation, one uses the fact that

$$\exp \begin{bmatrix} 0 & a/m \\ -a/m & 0 \end{bmatrix} = \begin{bmatrix} \cos(a/m) & \sin(a/m) \\ -\sin(a/m) & \cos(a/m) \end{bmatrix}.$$

(d) Compute the approximation, $(XY)^{2^j}$, to e^A by repeated squaring.

If we assume $5n^3$ flops for each of the eigenvalue decompositions in (a), then the overall process outlined above requires about $(13 + j)n^3$ flops. It is difficult to assess the relative efficiency of this splitting method because it depends strongly on the scalars $\|[A^T, A]\|$ and $\mu(A)$, and these quantities have not arisen in connection with any of our previous eighteen methods. On the basis of truncation error bounds, however, it would seem that this technique would be much less efficient than Method 3 (scaling and squaring) unless $\mu(A)$ were negative and $\|[A^T, A]\|$ much less than $\|A\|$.

Accuracy depends on the rounding errors which arise in (d) as a result of the repeated squaring. The remarks about repeated squaring in Method 3 apply also here: there may be severe cancellation, but whether or not this only occurs in sensitive e^A problems is unknown.

For a general splitting $A = B + C$, we can determine m from the inequality

$$\|e^A - (e^{B/m}e^{C/m})^m\| \leqq \frac{\|[B, C]\|}{2m} e^{\|B\|+\|C\|}, \tag{7.2}$$

which we establish in Appendix B.

To illustrate, suppose A has companion form

$$A = \begin{bmatrix} 0 & 1 & & 0 \cdots 0 \\ \vdots & & \ddots & \vdots \\ & & & 1 \\ c_0 & c_1 & \cdots & c_{n-1} \end{bmatrix}.$$

If

$$B = \begin{bmatrix} 0 & I_{n-1} \\ 0 & 0 \end{bmatrix}$$

and $C = e_n c^T$, where $c^T = (c_0, \ldots, c_{n-1})$ and $e_n^T = (0, 0, \ldots, 0, 1)$, then

$$e^{B/m} = \sum_{k=0}^{n-1} \left[\frac{B}{m}\right]^k \frac{1}{k!}$$

and

$$e^{C/m} = I + \frac{e^{c_{n-1}/m} - 1}{c_{n-1}} e_n c^T.$$

Notice that the computation of these scaled exponentials requires only $O(n^2)$ flops. Since $\|B\| = 1$, $\|C\| = \|c\|$, and $\|[B, C]\| \leqq 2\|c\|$, (7.2) becomes

$$\|e^A - (e^{B/m}e^{C/m})^m\| \leqq \frac{e^{1+\|c\|}\|c\|}{m}.$$

The parameter m can be determined from this inequality.

8. Conclusions. A section called "conclusions" must deal with the obvious question: Which method is best? Answering that question is very risky. We don't know enough about the sensitivity of the original problem or about the detailed performance

of careful implementations of various methods to make any firm conclusions. Furthermore, by the time this paper appears in the open literature, any given conclusion might well have to be modified.

We have considered five general classes of methods. What we have called polynomial methods are not really in the competition for "best." Some of them require the characteristic polynomial and so are appropriate only for certain special problems, and others have the same stability difficulties as matrix decomposition methods but are much less efficient. The approaches we have outlined under splitting methods are largely speculative and untried and probably only of interest in special settings. This leaves three classes in the running.

The only generally competitive series method is Method 3, scaling and squaring. Ward's program implementing this method is certainly among the best currently available. The program may fail, but at least it tells you when it does. We don't know yet whether or not such failures usually result from the inherent sensitivity of the problem or from the instability of the algorithm. The method basically computes e^A for a single matrix A. To compute e^{tA} for p arbitrary values of t requires about p times as much work. The amount of work is $O(n^3)$, even for the vector problem $e^{tA}x_0$. The coefficient of n^3 increases as $\|A\|$ increases.

Specializations of ODE methods for the e^A problem have not yet been implemented. The best method would appear to involve a variable order, variable step difference scheme. We suspect it would be stable and reliable but expensive. Its best showing on efficiency would be for the vector problem $e^{tA}x_0$ with many values of t since the amount of work is only $O(n^2)$. It would also work quite well for vector problems involving a large sparse A since no "nonsparse" approximation to the exponential would be explicitly required.

The best programs using matrix decomposition methods are just now being written. They start with the Schur decomposition and include some sort of eigenvalue clustering. There are variants which involve further reduction to a block form. In all cases the initial decomposition costs $O(n^3)$ steps and is independent of t and $\|A\|$. After that, the work involved in using the decomposition to compute $e^{tA}x_0$ for different t and x_0 is only a small multiple of n^2.

Thus, we see perhaps three or four candidates for "best" method. The choice will depend upon the details of implementation and upon the particular problem being solved.

Appendix A. Inverse Error Analysis of Padé Matrix Approximation.

LEMMA 1. *If* $\|H\| < 1$, *then* $\log(I+H)$ *exists and*

$$\|\log(I+H)\| \leqq \frac{\|H\|}{1-\|H\|}.$$

Proof. If $\|H\| < 1$, then $\log(I+H) = \sum_{k=1}^{\infty}(-1)^{k+1}(H^k/k)$ and so

$$\|\log(I+H)\| \leqq \sum_{k=1}^{\infty}\frac{\|H\|^k}{k} \leqq \|H\|\sum_{k=0}^{\infty}\|H\|^k = \frac{\|H\|}{1-\|H\|}. \qquad \square$$

LEMMA 2. *If* $\|A\| \leqq \frac{1}{2}$ *and* $p > 0$, *then* $\|D_{pq}(A)^{-1}\| \leqq (q+p)/p$.

Proof. From the definition of $D_{pq}(A)$ in section 3, $D_{pq}(A) = I + F$, where

$$F = \sum_{j=1}^{q}\frac{(p+q-j)!q!}{(p+q)!(q-j)!}\frac{(-A)^j}{j!}.$$

Using the fact that

$$\frac{(p+q-j)!q!}{(p+q)!(q-j)!} \leqq \left[\frac{q}{p+q}\right]^j,$$

we find

$$\|F\| \leqq \sum_{j=1}^{q} \left[\frac{q}{p+q}\|A\|\right]^j \frac{1}{j!} \leqq \frac{q}{p+q}\|A\|(e-1) \leqq \frac{q}{p+q},$$

and so $\|D_{pq}(A)^{-1}\| = \|(I+F)^{-1}\| \leqq 1/(1-\|F\|) \leqq (q+p)/p.$ ∎

LEMMA 3. *If* $\|A\| \leqq \frac{1}{2}$, $q \leqq p$, *and* $p \geqq 1$, *then* $R_{pq}(A) = e^{A+F}$, *where*

$$\|F\| \leqq 8\|A\|^{p+q+1}\frac{p!q!}{(p+q)!(p+q+1)!}.$$

Proof. From the remainder theorem for Padé approximants [71],

$$R_{pq}(A) = e^A - \frac{(-1)^q}{(p+q)!}A^{p+q+1}D_{pq}(A)^{-1}\int_0^1 e^{(1-u)A}u^p(1-u)^q\,du,$$

and so $e^{-A}R_{pq}(A) = I + H$, where

$$H = \frac{(-1)^{q+1}}{(p+q)!}A^{p+q+1}D_{pq}(A)^{-1}\int_0^1 e^{-uA}u^p(1-u)^q\,du.$$

By taking norms, using Lemma 2, and noting that $(p+q)/pe^{.5} \leqq 4$ we obtain

$$\begin{aligned}\|H\| &\leqq \frac{1}{(p+q)!}\|A\|^{p+q+1}\frac{p+q}{p}\int_0^1 e^{.5}u^p(1-u)^q\,du \\ &\leqq 4\|A\|^{p+q+1}\frac{p!q!}{(p+q)!(p+q+1)!}.\end{aligned}$$

With the assumption $\|A\| \leqq \frac{1}{2}$ it is possible to show that for all admissible p and q, $\|H\| \leqq \frac{1}{2}$ and so from Lemma 1,

$$\|\log(I+H)\| \leqq \frac{\|H\|}{1-\|H\|} \leqq 8\|A\|^{p+q+1}\frac{p!q!}{(p+q)!(p+q+1)!}.$$

Setting $F = \log(I+H)$, we see that $e^{-A}R_{pq}(A) = I + H = e^F$. The lemma now follows because A and F commute implying $R_{pq}(A) = e^A e^F = e^{A+F}$. ∎

LEMMA 4. *If* $\|A\| \leqq \frac{1}{2}$, *then* $R_{pq}(A) = e^{A+F}$, *where*

$$\|F\| \leqq 8\|A\|^{p+q+1}\frac{p!q!}{(p+q)!(p+q+1)!}.$$

Proof. The case $p \geqq q$, $p \geqq 1$ is covered by Lemma 1. If $p+q = 0$, then $F = -A$ and the above inequality holds. Finally, consider the case $q > p$, $q \geqq 1$. From Lemma 3, $R_{qp}(-A) = e^{-A+F}$, where F satisfies the above bound. The lemma now follows because $\|-F\| = \|F\|$ and $R_{pq}(A) = [R_{qp}(-A)]^{-1} = [e^{-A+F}]^{-1} = e^{A-F}$. ∎

THEOREM 4. *If* $\|A\|/2^j \leqq \frac{1}{2}$, *then* $[R_{pq}(A/2^j)]^{2^j} = e^{A+E}$, *where*

$$\frac{\|E\|}{\|A\|} \leqq 8\left(\frac{\|A\|}{2^j}\right)^{p+q} \frac{p!q!}{(p+q)!(p+q+1)!} \leqq \left(\frac{1}{2}\right)^{p+q-3} \frac{p!q!}{(p+q)!(p+q+1)!}.$$

Proof. From Lemma 4, $R_{pq}(A/2^j) = e^{A/2^j+F}$, where

$$\|F\| \leqq 8\left[\frac{\|A\|}{2^j}\right]^{p+q+1} \frac{p!q!}{(p+q)!(p+q+1)!}.$$

The theorem follows by noting that if $E = 2^j F$, then

$$\left[R_{pq}\left(\frac{A}{2^j}\right)\right]^{2^j} = [e^{A/2^j+F}]^{2^j} = e^{A+E}. \qquad \square$$

COROLLARY 1. *If* $\|A\|/2^j \leqq \frac{1}{2}$, *then* $[T_k(A/2^j)]^{2^j} = e^{A+E}$, *where*

$$\frac{\|E\|}{\|A\|} \leqq 8\left(\frac{\|A\|}{2^j}\right)^{k} \cdot \frac{1}{k+1} \leqq \left(\frac{1}{2}\right)^{k-3} \frac{1}{k+1}.$$

COROLLARY 2. *If* $\|A\|/2^j \leqq \frac{1}{2}$, *then* $[R_{qq}(A/2^j)]^{2j} = e^{A+E}$, *where*

$$\frac{\|E\|}{\|A\|} \leqq 8\left(\frac{\|A\|}{2^j}\right)^{2q} \cdot \frac{(q!)^2}{(2q)!(2q+1)!} \leqq \left(\frac{1}{2}\right)^{2q-3} \frac{(q!)^2}{(2q)!(2q+1)!}.$$

Appendix B. Accuracy of Splitting Techniques. In this appendix we derive the inequalities (7.1) and (7.2). We assume throughout that A is an n-by-n matrix and that

$$A = B + C.$$

It is convenient to define the matrices

$$S_m = e^{A/m}$$

and

$$T_m = e^{B/m}e^{C/m},$$

where m is a positive integer. Our goal is to bound $\|S_m^m - T_m^m\|$. To this end we shall have to exploit the following properties of the log norm $\mu(A)$ defined in section 2:

$$\begin{aligned} &\text{(i) } \|e^{tA}\| \leqq e^{\mu(A)t} \qquad (t \geqq 0),\\ &\text{(ii) } \mu(A) \leqq \|A\|,\\ &\text{(iii) } \mu(B+C) \leqq \mu(B) + \|C\|. \end{aligned}$$

These and other results concerning log norms are discussed in references [35], [36], [37], [38], [39], [40], [41], [42].

LEMMA 5. *If* $\Theta \geqq \max\{\mu(A), \mu(B) + \mu(C)\}$, *then*

$$\|S_m^m - T_m^m\| \leqq me^{\Theta(m-1)/m}\|S_m - T_m\|.$$

Proof. Following Reed and Simon [11] we have

$$S_m^m - T_m^m = \sum_{k=0}^{m-1} S_m^k (S_m - T_m) T_m^{m-1-k}.$$

Using log norm property (i) it is easy to show that both $\|S_m\|$ and $\|T_m\|$ are bounded above by $e^{\Theta/m}$ and thus

$$\begin{aligned}\|S_m^m - T_m^m\| &\leqq \sum_{k=0}^{m-1} \|S_m\|^k \|S_m - T_m\| \|T_m\|^{m-1-k} \\ &\leq \|S_m - T_m\| \sum_{k=0}^{m-1} e^{\Theta k/m} e^{\Theta(m-1-k)/m},\end{aligned}$$

from which the lemma immediately follows.

In Lemmas 6 and 7 we shall make use of the notation

$$F(t)|_{t=t_0}^{t=t_1} = F(t_1) - F(t_0),$$

where $F(t)$ is a matrix whose entries are functions of t. ∎

LEMMA 6.

$$T_m - S_m = \int_0^1 e^{tB/m} \left[e^{(1-t)A/m}, \frac{1}{m} C \right] e^{tC/m}\, dt.$$

Proof. We have $T_m - S_m = e^{tB/m} e^{(1-t)A/m} e^{tC/m}|_{t=0}^{t=1}$ and thus

$$T_m - S_m = \int_0^1 \left\{ \frac{d}{dt} [e^{tB/m} e^{(1-t)A/m} e^{tC/m}] \right\} dt.$$

The lemma follows since

$$\frac{d}{dt} [e^{tB/m} e^{(1-t)A/m} e^{tC/m}] = e^{tB/m} \left[e^{(1-t)A/m}, \frac{1}{m} C \right] e^{tC/m}. \qquad ∎$$

LEMMA 7. *If X and Y are matrices, then*

$$\|[e^X, Y]\| \leqq e^{\mu(X)} \|[X, Y]\|.$$

Proof. We have $[e^X, Y] = e^{tX} Y e^{(1-t)X}|_{t=0}^{t=1}$ and thus

$$[e^X, Y] = \int_0^1 \left\{ \frac{d}{dt} [e^{tX} Y e^{(1-t)X}] \right\} dt.$$

Since $d/dt[e^{tX} Y e^{(1-t)X}] = e^{tX}[X, Y] e^{(1-t)X}$ we get

$$\begin{aligned}\|[e^X, Y]\| &\leqq \int_0^1 \|e^{tX}\| \|[X, Y]\| \|e^{(1-t)X}\|\, dt \\ &\leqq \|[X, Y]\| \int_0^1 e^{\mu(X)t} e^{\mu(X)(1-t)}\, dt,\end{aligned}$$

from which the lemma immediately follows. ∎

THEOREM 5. *If* $\Theta \geqq \max\{\mu(A), \mu(B) + \mu(C)\}$, *then*

$$\|S_m^m - T_m^m\| \leqq \frac{1}{2m} e^{\Theta} \|[B, C]\|.$$

Proof. If $0 \leqq t \leqq 1$, then an application of Lemma 7 with $X \equiv (1-t)A/m$ and $Y \equiv C/m$ yields

$$\begin{aligned} \|[e^{(1-t)A/m}, C/m]\| &\leqq e^{\mu(A)(1-t)/m} \|[(1-t)A/m, C/m]\| \\ &\leq e^{\Theta(1-t)/m} \frac{(1-t)}{m^2} \|[B, C]\|. \end{aligned}$$

By coupling this inequality with Lemma 6 we can bound $\|T_m - S_m\|$:

$$\begin{aligned} \|T_m - S_m\| &\leqq \int_0^1 \|e^{tB/m}\| \|[e^{(1-t)A/m}, C/m]\| \|e^{tC/m}\| \, dt \\ &\leqq \int_0^1 e^{\mu(B)t/m} e^{\Theta(1-t)/m} \frac{(1-t)}{m^2} \|[B, C]\| e^{\mu(C)t/m} \, dt \\ &\leqq \frac{1}{2} e^{\Theta/m} \frac{\|[B, C]\|}{m^2}. \end{aligned}$$

The theorem follows by combining this result with Lemma 5. ∎

COROLLARY 3. *If* $B = (A + A^*)/2$ *and* $C = (A - A^*)/2$, *then*

$$\|S_m^m - T_m^m\| \leqq \frac{1}{4m} e^{\mu(A)} \|[A^*, A]\|.$$

Proof. Since $\mu(A) = \mu(B)$ and $\mu(C) = 0$, we can set $\Theta = \mu(A)$. The corollary is established by noting that $[B, C] = \frac{1}{2}[A^*, A]$. ∎

COROLLARY 4.

$$\|S_m^m - T_m^m\| \leqq \frac{1}{2m} e^{\mu(B) + \|C\|} \|[B, C]\| \leqq \frac{1}{2m} e^{\|B\| + \|C\|} \|[B, C]\|.$$

Proof. $\max\{\mu(A), \mu(B) + \mu(C)\} \leqq \mu(B) + \|C\| \leqq \|B\| + \|C\|$. ∎

Acknowledgments. We have greatly profited from the comments and suggestions of so many people that it is impossible to mention them all. However, we are particularly obliged to B. N. Parlett and G. W. Stewart for their very perceptive remarks and to G. H. Golub for encouraging us to write this paper. We are obliged to Professor Paul Federbush of the University of Michigan for helping us with the analysis in Appendix B. Finally, we would like to thank the referees for their numerous and most helpful comments.

REFERENCES

Background.

[1] R. BELLMAN, *Introduction to Matrix Analysis*, McGraw-Hill, New York, 1969.

[2] C. DAVIS, *Explicit functional calculus*, J. Linear Algebra Appl., 6 (1973), pp. 193–199.

[3] V. N. FADDEEVA, *Computational Methods of Linear Algebra*, Dover, New York, 1959.

[4] G. E. FORSYTHE, M. A. MALCOLM, AND C. B. MOLER, *Computer Methods for Mathematical Computations*, Prentice–Hall, Englewood Cliffs, NJ, 1977.

[5] J. S. FRAME, *Matrix functions and applications, Part* II: *Functions of matrices*, IEEE Spectrum, 1 (4) (1964), pp. 102–108.

[6] J. S. Frame, *Matrix functions and applications, Part* IV: *Matrix functions and constituent matrices*, IEEE Spectrum, 1 (6) (1964), pp. 123–131.
[7] J. S. Frame, *Matrix functions and applications, Part* V: *Similarity reductions by rational or orthogonal matrices*, IEEE Spectrum, 1 (7) (1964), pp. 103–116.
[8] F. R. Gantmacher, *The Theory of Matrices*, Vols. I and II, Chelsea, New York, 1959.
[9] C. C. MacDuffee, *The Theory of Matrices*, Chelsea, New York, 1956.
[10] L. Mirsky, *An Introduction to Linear Algebra*, Oxford University Press, London, 1955.
[11] M. Reed and B. Simon, *Functional Analysis*, Academic Press, New York, 1972.
[12] R. F. Rinehart, *The equivalence of definitions of a matrix function*, Amer. Math. Monthly, 62 (1955), pp. 395–414.
[13] P. C. Rosenbloom, *Bounds on functions of matrices*, Amer. Math. Monthly, 74 (1967), pp. 920–926.
[14] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

Properties and representations.

[15] T. M. Apostol, *Some explicit formulas for the matrix exponential*, Amer. Math. Monthly, 76 (1969), pp. 284–292.
[16] R. W. Atherton and A. E. De Gance, *On the evaluation of the derivative of the matrix exponential function*, IEEE Trans. Automat. Control, AC-20 (1975), pp. 707–708.
[17] A. Bradshaw, *The eigenstructure of sample data systems with confluent eigenvalues*, Internat. J. Systems Sci., 5 (1975), pp. 607–613.
[18] R. Bellman, *Perturbation Techniques in Mathematics, Physics, and Engineering*, Holt, Rinehart and Winston, New York, 1964.
[19] J. C. Cavendish, *On the norm of a matrix exponential*, SIAM Rev., 17 (1975), pp. 174–175.
[20] C. G. Cullen, *Remarks on computing* e^{At}, IEEE Trans. Automat. Control, AC-16 (1971), pp. 94–95.
[21] F. Fer, *Resolution de l'equation matricelle* $dU/dt = pU$ *par produit infini d'exponentielles*, Acad. Roy. Belg. Bull. Cl. Sci., 44 (1958), pp. 819–829.
[22] E. P. Fulmer, *Computation of the matrix exponential*, Amer. Math. Monthly, 82 (1975), pp. 156–159.
[23] B. Kågstrom, *Bounds and perturbation bounds for the matrix exponential*, BIT, 17 (1977), pp. 39–57.
[24] T. Kato, *Perturbation Theory for Linear Operators*, Chap. 9, Springer-Verlag, New York, 1966.
[25] R. B. Kirchner, *An explicit formula for* e^{At}, Amer. Math. Monthly, 74 (1967), pp. 1200–1204.
[26] H. O. Kreiss, *Über Matrizen die beschränkte Halbgruppen erzeuge*, Math. Scand., 7 (1959), pp. 71–80.
[27] D. L. Powers, *On the eigenstructure of the matrix exponential*, Internat. J. Systems Sci., 7 (1976), pp. 723–725.
[28] E. J. Putzer, *Avoiding the Jordan canonical form in the discussion of linear systems with constant coefficients*, Amer. Math. Monthly, 73 (1966), pp. 2–7.
[29] N. M. Rice, *More Explicit Formulas for the Exponential Matrix*, Queen's Mathematical Reprints 1970–21, Queen's University, Kingston, ON, Canada, 1970.
[30] H. F. Trotter, *Product of semigroups of operators*, Proc. Amer. Math. Soc., 10 (1959), pp. 545–551.
[31] C. F. Van Loan, *A Study of the Matrix Exponential*, Numerical Analysis Report 7, Department of Mathematics, University of Manchester, Manchester, UK, 1975.
[32] C. F. Van Loan, *The sensitivity of the matrix exponential*, SIAM J. Numer. Anal., 14 (1977), pp. 971–981.
[33] G. H. Weiss and A. A. Maradudin, *The Baker–Hausdorff formula and a problem in crystal physics*, J. Math. Phys., 3 (1962), pp. 771–777.
[34] A. D. Ziebur, *On determining the structure of A by analyzing* e^{At}, SIAM Rev., 12 (1970), pp. 98–102.

Log norms and stability.

[35] W. A. Coppel, *Stability and Asymptotic Behavior of Differential Equations*, D. C. Heath, Boston, 1965.
[36] G. Dahlquist, *Stability and Error Bounds in the Numerical Integration of Ordinary Differential Equations*, Transactions of the Royal Institute of Technology 130, Stockholm, Sweden, 1959.
[37] C. A. Desoer and H. Haneda, *The measure of a matrix as a tool to analyze computer algorithms for circuit analysis*, IEEE Trans. Circuit Theory, CT-19 (1972), pp. 480–486.

[38] E. Deutsch, *On matrix norms and logarithmic norms*, Numer. Math., 24 (1975), pp. 49–51.
[39] C. V. Pao, *Logarithmic derivatives of a square matrix*, J. Linear Algebra Appl., 6 (1973), pp. 159–164.
[40] C. V. Pao, *A further remark on the logarithmic derivatives of a square matrix*, J. Linear Algebra Appl., 7 (1973), pp. 275–278.
[41] T. Ström, *Minimization of norms and logarithmic norms by diagonal similarities*, Computing, 10 (1972), pp. 1–9.
[42] T. Ström, *On logarithmic norms*, SIAM J. Numer. Anal., 12 (1975), pp. 741–753.

Survey articles.

[43] M. Healey, *Study of methods of computing transition matrices*, Proc. IEEE, 120 (1973), pp. 905–912.
[44] C. B. Moler, *Difficulties in computing the exponential of a matrix*, Proceedings of the Second USA–Japan Computer Conference, A.F.I.P.S., Montvale, NJ, 1975, pp. 79–82.

Truncated Taylor series.

[45] L. Y. Bahar and A. K. Sinha, *Matrix exponential approach to dynamic response*, Comput. & Structures, 5 (1975), pp. 159–165.
[46] T. A. Bickart, *Matrix exponential: Approximation by truncated power series*, Proc. IEEE, 56 (1968), pp. 372–373.
[47] G. J. Bierman, *Power series evaluation of transition and covariance matrices*, IEEE Trans. Automat. Control, AC-17 (1972), pp. 228–231.
[48] D. A. Calahan, *Numerical solution of linear systems with widely separated time constants*, Proc. IEEE, 55 (1967), pp. 2016–2017.
[49] K. C. Daly, *Evaluating the matrix exponential*, Electron. Lett., 8 (1972), p. 390.
[50] W. Everling, *On the evaluation of e^{At} by power series*, Proc. IEEE, 55 (1967), p. 413.
[51] D. A. Gall, *The solution of linear constant coefficient ordinary differential equations with APL*, Comput. Methods Mechanics and Engrg., 1 (1972), pp. 189–196.
[52] M. L. Liou, *A novel method of evaluating transient response*, Proc. IEEE, 54 (1966), pp. 20–23.
[53] J. B. Mankin and J. C. Hung, *On Roundoff Errors in Computation of Transition Matrices*, Reprints of the Joint Automatic Control Conference, University of Colorado, Boulder, CO, 1969, pp. 60–64.
[54] E. J. Mastascusa, *A relation between Liou's method and fourth order Runge–Kutta method for evaluation of transient response*, Proc. IEEE, 57 (1969), pp. 803–804.
[55] J. B. Plant, *On the computation of transient matrices for time invariant systems*, Proc. IEEE, 56 (1968), pp. 1397–1398.
[56] M. M. Shah, *On the Evaluation of e^{At}*, Cambridge Report CUED/B-Control TR8, Cambridge University, Cambridge, UK, 1971.
[57] M. M. Shah, *Analysis of Roundoff and Truncation Errors in the Computation of Transition Matrices*, Cambridge Report CUED/B-Control TR12, Cambridge University, Cambridge, UK, 1971.
[58] C. J. Standish, *Truncated Taylor series approximation to the state transition matrix of a continuous parameter Markov chain*, Linear Algebra Appl., 12 (1975), pp. 179–183.
[59] D. E. Whitney, *Propagated error bounds for numerical solution of transient response*, Proc. IEEE, 54 (1966), pp. 1084–1085.
[60] D. E. Whitney, *More similarities between Runge–Kutta and matrix exponential methods for evaluating transient response*, Proc. IEEE, 57 (1969), pp. 2053–2054.

Rational approximation.

[61] J. L. Blue and H. K. Gummel, *Rational approximations to the matrix exponential for systems of stiff differential equations*, J. Comput. Phys., 5 (1970), pp. 70–83.
[62] W. J. Cody, G. Meinardus, and R. S. Varga, *Chebyshev rational approximation to $exp(-x)$ in $[0, +\infty]$ and applications to heat conduction problems*, J. Approx. Theory, 2 (1969), pp. 50–65.
[63] W. Fair and Y. Luke, *Padé approximations to the operator exponential*, Numer. Math., 14 (1970), pp. 379–382.
[64] S. P. Norsett, *C-polynomials for rational approximation to the exponential function*, Numer. Math., 25 (1975), pp. 39–56.
[65] E. B. Saff, *On the degree of best rational approximation to the exponential function*, J. Approx. Theory, 9 (1973), pp. 97–101.
[66] E. B. Saff and R. S. Varga, *On the zeros and poles of Padé approximants to* $\exp(z)$, Numer. Math., 25 (1975), pp. 1–14.

[67] R. E. Scraton, *Comment on rational approximants to the matrix exponential*, Electron. Lett., 7 (1971), pp. 260–261.
[68] J. L. Siemieniuch, *Properties of certain rational approximations to* e^{-z}, BIT, 16 (1976), pp. 172–191.
[69] G. Siemieniuch and I. Gladwell, *On Time Discretizations for Linear Time Dependent Partial Differential Equations*, Numerical Analysis Report 5, Department of Mathematics, University of Manchester, Manchester, UK, 1974.
[70] D. M. Trujillo, *The direct numerical integration of linear matrix differential equations using Padé approximations*, Internat. J. Numer. Methods Engrg., 9 (1975), pp. 259–270.
[71] R. S. Varga, *On higher order stable implicit methods for solving parabolic partial differential equations*, J. Math. Phys., 40 (1961), pp. 220–231.
[72] R. C. Ward, *Numerical computation of the matrix exponential with accuracy estimate*, SIAM J. Numer. Anal., 14 (1977), pp. 600–610.
[73] A. Wragg and C. Davies, *Evaluation of the matrix exponential*, Electron. Lett., 9 (1973), pp. 525–526.
[74] A. Wragg and C. Davies, *Computation of the exponential of a matrix* I: *Theoretical considerations*, J. Inst. Math. Appl., 11 (1973), pp. 369–375.
[75] A. Wragg and C. Davies, *Computation of the exponential of a matrix* II: *Practical considerations*, J. Inst. Math. Appl., 15 (1975), pp. 273–278.
[76] V. Zakian, *Rational approximants to the matrix exponential*, Electron. Lett., 6 (1970), pp. 814–815.
[77] V. Zakian and R. E. Scraton, *Comments on rational approximations to the matrix exponential*, Electron. Lett., 7 (1971), pp. 260–262.

Polynomial methods.

[78] G. J. Bierman, *Finite series solutions for the transition matrix and covariance of a time-invariant system*, IEEE Trans. Automat. Control, AC-16 (1971), pp. 173–175.
[79] J. A. Boehm and J. A. Thurman, *An algorithm for generating constituent matrices*, IEEE Trans. Circuit Theory, CT-18 (1971), pp. 178–179.
[80] C. F. Chen and R. R. Parker, *Generalization of Heaviside's expansion technique to transition matrix evaluation*, IEEE Trans. Educ., E-9 (1966), pp. 209–212.
[81] W. C. Davidon, *Exponential Function of a 2-by-2 Matrix*, Hewlett-Packard HP65 Library Program.
[82] S. Deards, *On the evaluation of* $exp(tA)$, Matrix Tensor Quart., 23 (1973), pp. 141–142.
[83] S. Ganapathy and R. S. Rao, *Transient response evaluation from the state transition matrix*, Proc. IEEE, 57 (1969), pp. 347–349.
[84] I. C. Goknar, *On the evaluation of constituent matrices*, Internat. J. Systems Sci., 5 (1974), pp. 213–218.
[85] I. I. Kolodner, *On* $exp(tA)$ *with A satisfying a polynomial*, J. Math. Anal. Appl., 52 (1975), pp. 514–524.
[86] Y. L. Kuo and M. L. Liou, *Comments on "A novel method of evaluating* e^{At} *in closed form,"* IEEE Trans. Automat. Control, AC-16 (1971), p. 521.
[87] E. J. Mastascusa, *A method of calculating* e^{At} *based on the Cayley–Hamilton theorem*, Proc. IEEE, 57 (1969), pp. 1328–1329.
[88] K. R. Rao and N. Ahmed, *Heaviside expansion of transition matrices*, Proc. IEEE, 56 (1968), pp. 884–886.
[89] K. R. Rao and N. Ahmed, *Evaluation of transition matrices*, IEEE Trans. Automat. Control, AC-14 (1969), pp. 779–780.
[90] B. Roy, A. K. Mandal, D. Roy Choudhury, and A. K. Choudhury, *On the evaluation of the state transition matrix*, Proc. IEEE, 57 (1969), pp. 234–235.
[91] M. N. S. Swamy, *On a formula for evaluating* e^{At} *when the eigenvalues are not necessarily distinct*, Matrix Tensor Quart., 23 (1972), pp. 67–72.
[92] M. Vidysager, *A novel method of evaluating* e^{At} *in closed form*, IEEE Trans. Automat. Control, AC-15 (1970), pp. 600–601.
[93] V. Zakian, *Solution of homogeneous ordinary linear differential equations by numerical inversion of Laplace transforms*, Electron. Lett., 7 (1971), pp. 546–548.

Companion matrix methods.

[94] A. K. Choudhury, D. R. Choudhury, B. Roy, and A. K. Mandal, *On the evaluation of* e^{At}, Proc. IEEE, 56 (1968), pp. 1110–1111.
[95] L. Falcidieno and A. Luvinson, *A Numerical Approach to Computing the Companion Matrix Exponential*, CSELT technical report 4, 1975, pp. 69–71.

[96] C. J. Harris, *Evaluation of matrix polynomials in the state companion matrix of linear time invariant systems*, Internat. J. Systems Sci., 4 (1973), pp. 301–307.
[97] D. W. Kammler, *Numerical Evaluation of exp(At) When A is a Companion Matrix*, manuscript, University of Southern Illinois, Carbondale, IL, 1976.
[98] I. Kaufman, *Evaluation of an analytical function of a companion matrix with distinct eigenvalues*, Proc. IEEE, 57 (1969), pp. 1180–1181.
[99] I. Kaufman, *A note on the "Evaluation of an analytical function of a companion matrix with distinct eigenvalues*," Proc. IEEE, 57 (1969), pp. 2083–2084.
[100] I. Kaufman and P. H. Roe, *On systems described by a companion matrix*, IEEE Trans. Automat. Control, AC-15 (1970), pp. 692–693.
[101] I. Kaufman, H. Mann, and J. Vlach, *A fast procedure for the analysis of linear time invariant networks*, IEEE Trans. Circuit Theory, CT-18 (1971), pp. 739–741.
[102] M. L. Liou, *Evaluation of the transition matrix*, Proc. IEEE, 55 (1967), pp. 228–229.
[103] A. K. Mandal, et al., *Numerical computation method for the evaluation of the transition matrix*, Proc. IEEE, 116 (1969), pp. 500–502.
[104] W. E. Thomson, *Evaluation of transient response*, Proc. IEEE, 54 (1966), p. 1584.

Ordinary differential equations.

[105] B. L. Ehle and J. D. Lawson, *Generalized Runge–Kutta processes for stiff initial value problems*, J. Inst. Math. Appl., 16 (1975), pp. 11–21.
[106] C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice–Hall, Englewood Cliffs, NJ, 1971.
[107] L. F. Shampine and M. K. Gordon, *Computer Solution of Ordinary Differential Equations—The Initial Value Problem*, W. H. Freeman, San Francisco, 1975.
[108] L. F. Shampine and H. A. Watts, *Practical Solution of Ordinary Differential Equations by Runge–Kutta Methods*, Sandia Lab Report SAND 76-0585, Albuquerque, NM, 1976.
[109] J. Starner, *Numerical Solution of Implicit Differential-Algebraic Equations*, Ph.D. Thesis, University of New Mexico, Albuquerque, NM, 1976.

Matrix decomposition methods.

[110] G. H. Golub and J. H. Wilkinson, *Ill-conditioned eigensystems and the computation of the Jordan canonical form*, SIAM Rev., 18 (1976), pp. 578–619.
[111] B. Kågstrom and A. Ruhe, *An algorithm for numerical computation of the Jordan normal form of a complex matrix*, ACM Trans. Math. Software, 6 (1980), pp. 398–419.
[112] B. N. Parlett, *A recurrence among the elements of functions of triangular matrices*, Linear Algebra Appl., 14 (1976), pp. 117–121.
[113] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines: EISPACK Guide*, 2nd ed., Lecture Notes in Comput. Sci. 6, Springer-Verlag, New York, 1976.

Integrals involving e^{At}.

[114] E. S. Armstrong and A. K. Caglayan, *An Algorithm for the Weighting Matrices in the Sampled-Data Optimal Linear Regulator Problem*, NASA technical note, TN D-8372, 1976.
[115] J. Johnson and C. L. Phillips, *An algorithm for the computation of the integral of the state transition matrix*, IEEE Trans. Automat. Control, AC-16 (1971), pp. 204–205.
[116] C. Kallstrom, *Computing exp(A) and $\int exp(As)\,ds$*, Report 7309, Division of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1973.
[117] A. H. Levis, *Some computational aspects of the matrix exponential*, IEEE Trans. Automat. Control, AC-14 (1969), pp. 410–411.
[118] C. F. Van Loan, *Computing integrals involving the matrix exponential*, IEEE Trans. Automat. Control, AC-23 (1978), pp. 395–404.

Selected applications.

[119] F. H. Branin, *Computer methods of network analysis*, Proc. IEEE, 55 (1967), pp. 1787–1801.
[120] R. Brockett, *Finite Dimensional Linear Systems*, John Wiley, New York, 1970.
[121] K. F. Hansen, B. V. Koen, and W. W. Little, *Stable numerical solutions of the reactor kinetics equations*, Nuclear Sci. Engrg., 22 (1965), pp. 51–59.
[122] J. A. W. da Nobrega, *A new solution of the point kinetics equations*, Nuclear Sci. Engrg., 46 (1971), pp. 366–375.

9. Update, Twenty-Five Years Later. Since the publication of "Nineteen Ways" twenty-five years ago, the field of matrix computations has matured in several important directions:

- There is a greater appreciation for nonnormality and related issues of conditioning.
- Krylov-type methods for large sparse problems have been developed that are very effective.
- Many structure-exploiting variants of the basic algorithms have been developed which work well in specific application areas.
- High-performance methods have been developed for computers that have a hierarchical memory and/or multiple processing units.

All these developments have affected the matrix exponential "business," some more than others. This brief account of the past twenty-five years is not the distillation of an exhaustive literature search. Instead, we have chosen to highlight just a few research threads, especially those that complement and expand upon the key ideas in "Nineteen Ways." We trust that our very incomplete bibliography provides adequate pointers for the interested reader.

A good place to begin the discussion is with the scaling and squaring method that is based upon Padé approximation (Method 3). It continues to attract a lot of attention relative to the other methods presented in "Nineteen Ways." Bochev and Markov [124] discuss its execution using interval arithmetic, while Arioli, Codenatti, and Fassino [123] offer analyses for several important structured examples.

While on the topic of Padé approximation and scaling and squaring, it is interesting to note that analogous algorithmic techniques have been developed for the matrix logarithm problem. See Cheng et al. [129] and Higham [138]. Intelligent "e^A thinking" has affected how researchers approach other matrix function problems that arise in application areas such as control engineering and signal processing. Early studies of the condition of the matrix exponential problem prompted others to investigate similar issues for general matrix functions, e.g., Kenny and Laub [143], [144] and Mathias [147]. The sensitivity of the map $A \rightarrow e^A$ requires the analysis of the underlying Fréchet derivative. See Najfeld and Havel [149], Mathias [148], and and Dieci and Papini [131].

Methods 5 through 7 are based on using numerical ODE solvers to approximate the matrix exponential. Conversely, the matrix exponential can be used to solve differential equations. This has been an increasingly popular approach recently, particularly for stiff problems. For example, Hochbruck, Lubich, and Selhofer [140] show that by using Krylov space methods for the matrix exponentiation (as discussed the next section), iterations for the matrix exponential may converge faster than those for the system of linear equations that would arise with a traditional implicit ODE method. See also the paper by Edwards et al. [134].

Several authors have developed new approaches to the exponentiation of the Schur canonical form, thereby adding new dimensions to Methods 17 and 18. Dieci and Papini [130] work with Padé approximation on a block triangular structure while Kenny and Laub [145] exploit some practical connections to the Fréchet derivative.

The splitting method (Method 19) was a purely speculative inclusion in "Nineteen Ways." We had no experience in using it but thought that it was interesting to include a technique that worked around "the great matrix exponential tragedy," namely, the fact that e^{A+B} does not generally equal $e^A e^B$. It turns out that there is

heightened interest in the exponential splitting idea in various numerical partial differential equation settings. See Celledoni and Iserles [127], [128], Sheng [153], Jahnke and Lubich [142], and Zanna and Munthe-Kaas [161].

As mentioned in the fourth bullet item above, changes in hardware have prompted a major reevaluation of many algorithms in the matrix computation field. For example, data reuse issues have prompted the design of methods that are rich in matrix multiplication. Thus, matrix exponentiation methods (such as scaling and squaring) that are rich in this operation have an even greater appeal than they did twenty-five years ago. A few remarks about level-2 and level-3 BLAS as they apply to the matrix log problem can be found in Higham [138]. Partial fraction approximants to the exponential have also attracted attention in parallel computing settings. See Calvetti, Gallopoulos, and Reichel [126].

10. Method 20: Krylov Space Methods. In many applications one does not need the full matrix e^A but only the product $e^A v$ for some given vector v. This is true, for example, when solving the initial value problem

$$\dot{x} = Ax, \quad x(0) = x_0$$

with solution $x(t) = e^{At}v$. Often A is large and sparse, in particular if this ODE arises from the spatial discretization of a partial differential equation. Since e^A will typically be dense even if A is sparse, we would like to avoid computing this matrix.

One of the most significant changes in numerical linear algebra in the past twenty-five years is the rise of iterative methods for sparse matrix problems, in which only matrix-vector products are needed. A powerful class of methods that are applicable to many problems are the Krylov space methods, in which approximations to the solution are obtained from the Krylov spaces spanned by the vectors $\{v,\ Av,\ A^2v, \ldots,\ A^mv\}$ for some m that is typically small compared to the dimension of A. The Lanczos method for solving symmetric eigenvalue problems is of this form, and for nonsymmetric matrices the Arnoldi iteration can be used. In this method the eigenvalues of a large matrix are approximated by the eigenvalues of a Hessenberg matrix of dimension m. After m steps in Arnoldi with starting vector v (assumed to have unit length) we have the *partial* Hessenberg reduction

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T,$$

where V_m has m orthonormal columns, H is m-by-m and upper Hessenberg, e_m is the last column of I_m, and v_{m+1} is a unit vector that satisfies $V_m^T v_{m+1} = 0$. Returning to the problem of matrix exponentiation, it turns out that

$$e^A v \approx V_m e^{H_m} e_1,$$

where e_1 is the first column of I_m.

Very good approximations are often obtained with relatively small m, and computable error bounds exist for the approximation. Thus, the large sparse e^A problem is replaced with a small dense e^{H_m} problem.

For some interesting analysis and applications, see Knizhnerman [146], Druskin and Knizhnerman [132], Druskin, Greenbaum, and Knizhnerman [133], Gallopoulos and Saad [136], Hochbruck and Lubich [139], Hochbruck, Lubich, and Selhofer [140] Nour-Omid [151], Saad [152], and Stewart and Leyk [157].

If we were to revise thoroughly "Nineteen Ways," we would have to revise the title because Krylov space methods constitute a twentieth approach!

11. Matrix Exponential in MATLAB. The first version of MATLAB was being developed at the same time "Nineteen Ways" was being written in the late 1970s. Even though the original MATLAB had only 80 functions, one of them was the matrix exponential. The first MathWorks MATLAB, released in 1984, included both `exp(A)`, the element-by-element array exponential, and `expm(A)`, the matrix exponential.

MATLAB was initially intended for the academic numerical linear algebra community, but researchers and practitioners in the control design community soon found it useful. The matrix exponential is an important computational tool in control theory, so availability of `expm(A)` in early versions of MATLAB quite possibly contributed to the system's technical and commercial success.

The `expm` function is used by MATLAB in its Control Toolbox, System Identification Toolbox, Neural Net Toolbox, Mu-Analysis and Synthesis Toolbox, and Model Predictive Control toolbox, as well as in Simulink. It is also used in MATLAB itself for a Newton-step correction in computing `logm`.

The `expm` function in MATLAB is built-in, so the source code is not distributed. However, the `help` entry says

```
EXPM(A) is the matrix exponential of A.  EXPM is computed using
a scaling and squaring algorithm with a Pade approximation.
```

In other words, MATLAB uses Methods 3 and 2 from "Nineteen Ways."

The MATLAB `demos` directory contains three M-files, `expm1, expm2` and `expm3`, that implement three different methods for computing the matrix exponential. The first of these, `expm1`, is essentially an M-file description of the built-in algorithm. The function begins by determining the appropriate scaling.

```
function E = expm1(A)
[f,e] = log2(norm(A,'inf'));
s = max(0,e+1);
A = A/2^s;
```

For a floating point number x, the statement `[f,e] = log2(x)` finds the fraction f, with $1/2 \le |f| < 1$, and integer exponent e so that $x = f \cdot 2^e$. This leads to a scaling parameter s so that the scaled matrix has $\|A/2^s\|_\infty < 1/2$. The next section of code in `expm1.m` finds the numerator E and denominator D of the $(6,6)$ Padé approximation.

```
X = A;
c = 1/2;
E = eye(size(A)) + c*A;
D = eye(size(A)) - c*A;
q = 6; p = 1;
for k = 2:q
   c = c * (q-k+1) / (k*(2*q-k+1));
   X = A*X;
   cX = c*X;
   E = E + cX;
   if p, D = D + cX;
   else, D = D - cX; end
   p = ~p;
end
```

The computation of $D^{-1}E$ is done with the MATLAB backslash operator.

```
E = D\E;
```

The computation is completed by squaring the approximant s times.

```
for k=1:s, E = E*E; end
```

The second MATLAB `demo` function, `expm2`, is included only for pedagogical purposes. It implements Method 1, Taylor series. This function is accurate, but not very efficient, for matrices with $\|A\| < 1$. The accuracy deteriorates and the execution time increases as $\|A\|$ increases.

```
function E = expm2(A)
E = zeros(size(A));
F = eye(size(A));
k = 1;
while norm(E+F-E,1) > 0
   E = E + F;
   F = A*F/k;
   k = k+1;
end
```

The third MATLAB `demo` function, `expm3`, is also primarily of pedagogical interest. It implements Method 14, eigenvectors. This function is accurate and efficient for symmetric, orthogonal, and other normal matrices. The accuracy deteriorates as `cond(V)`, the condition number of the matrix of eigenvectors, increases. The function fails completely, and without warning, when A is defective.

```
function E = expm3(A)
[V,D] = eig(A);
E = V * diag(exp(diag(D))) / V;
```

12. Nonnormality. We now know that the behavior of the matrix exponential is closely related to various matrix properties that express its nonnormality. These include quantities such as the condition numbers of eigenvalues and eigenvectors, the field of values, the polynomial numerical hulls of various degrees, and the pseudospectra.

A recent paper by Anne Greenbaum [137] discusses generalizations of the field of values that are relevant to the matrix exponential, although the details have not yet been worked out. Background is provided by [141] and [150].

The study of pseudospectra has been led by L. N. Trefethen [158], [159]. The *Pseudospectra Gateway* Web page [135] includes a bibliography with, at the time of this writing, nearly 200 entries.

Let $\Lambda(A)$ denote the set of eigenvalues of A. The ϵ-pseudospectrum of a matrix A, denoted by $\Lambda_\epsilon(A)$, is a set in the complex plane that depends upon a scalar parameter ϵ and that converges to $\Lambda(A)$ as $\epsilon \to 0$. There are three equivalent definitions. One involves the resolvent:

$$\Lambda_\epsilon(A) = \{z : \|(zI - A)^{-1}\| \geqq \epsilon^{-1}\}.$$

A second definition involves perturbations of the matrix:

$$\Lambda_\epsilon(A) = \{z : z \in \Lambda(A+E), \|E\| \leqq \epsilon\}.$$

A third definition involves pseudoeigenvectors:

$$\Lambda_\epsilon(A) = \{z : \|(A - zI)v\| \leqq \epsilon, \|v\| = 1\}.$$

If A is normal, then $\Lambda_\epsilon(A)$ is simply the set of disks of radius ϵ around the eigenvalues. But if A is not normal, then $\Lambda_\epsilon(A)$ can be a much larger and more interesting set. The size and structure of this set determines the behavior of e^{tA} and of many of the methods for computing it.

13. EigTool and Expokit. EigTool is a MATLAB toolkit and graphical interface developed by Tom Wright [160] at Oxford University and available via the Pseudospecta Gateway. Once EigTool has been installed, the MATLAB command

```
eigtool(A)
```

plots the boundary of $\Lambda_\epsilon(A)$ for several values of ϵ. A graphical interface provides a number of related quantities and options, including graphs of $\|A^k\|$ and $\|e^{tA}\|$ as functions of k and t, together with lower bounds for these quantities based on the pseudospectra.

EigTool uses a close cousin of Method 3, scaling and squaring, to compute the graph of $\|e^{tA}\|$. A step size h must be chosen. No attempt is made to automatically determine a "good" value. The default value is $h = 0.1$. The matrix $E_1 = e^{hA}$ is computed using the MATLAB function `expm`. Then matrices E_n, which approximate e^{nhA}, are computed by repeated multiplication

$$E_n = E_1 \cdot E_{n-1}, \qquad n = 2, \ldots.$$

The only errors in this approach are the truncation error in the initial Padé approximation and the roundoff error in the repeated matrix multiplication. The magnitude of the error at each step is on the order floating point accuracy, relative to the quantities involved in that step. Consequently, for stable matrices with a large hump, the relative accuracy deteriorates for large t.

The most extensive software for computing the matrix exponential that we are aware of is Expokit, developed by Roger Sidje [154], [155]. Both Fortran and MATLAB versions are available. In addition to computing the matrix-valued function e^{tA} for small, dense matrices A, Expokit has functions for computing the vector-valued function $e^{tA}x_0$ for both small, dense matrices and large, sparse matrices. There are also functions for computing the solution to inhomogeneous, constant coefficient linear ODEs

$$\dot{x} = Ax + u, \quad x(0) = x_0$$

for both dense and sparse A. Transient states of Markov chains are given special attention in the package.

The methods for dense matrices use Chebyshev approximations instead of Padé approximations. The methods for sparse matrices use Krylov sequences and require only operator-vector products, Av, for various vectors v.

14. Transient Example. Figure 3 illustrates the hump phenomenon with the graph of $\|e^{tA}\|$ for a matrix adapted from the transient demo example in EigTool.

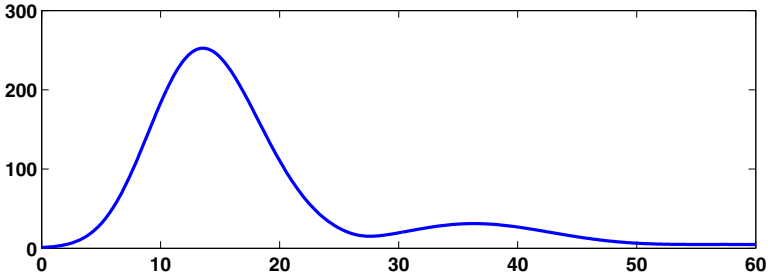

Fig. 3 $\|e^{tA}\|$, *the hump for the transient example.*

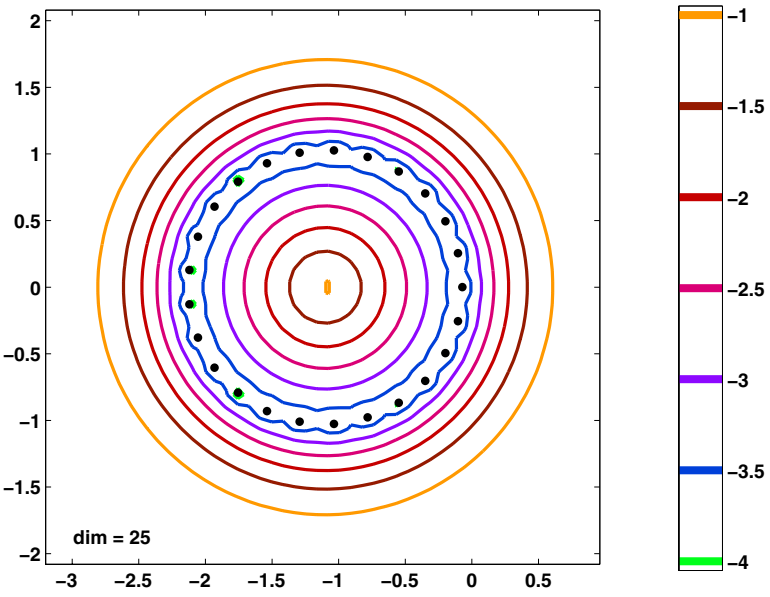

Fig. 4 ϵ*-pseudospectra for the transient example.*

The hump is not very large; the graph reaches a height of only slightly over 250 before it decays to zero.

The matrix has order $n = 25$. It is generated in MATLAB by adding the diagonal matrix formed from the nth roots of unity to the circulant matrix with ones on the superdiagonal and in the lower left-hand corner. The spectrum is shifted to make the matrix stable; the right-most eigenvalue is -0.0719.

```
n = 25;
C = diag(ones(n-1,1),1); C(n,1) = 1;
D = diag(exp(2*pi*i*(0:n-1)/n));
I = eye(n);
A = C + D - 1.1*I;
```

Figure 4 is the plot from EigTool of the pseudospectra for this example. The matrix is not normal, but the condition number of the matrix of eigenvectors is only

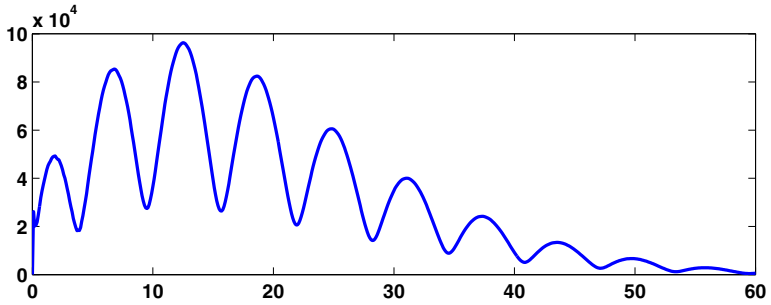

Fig. 5 $\|e^{tA}\|$, *the hump for the stabilized Boeing* 767.

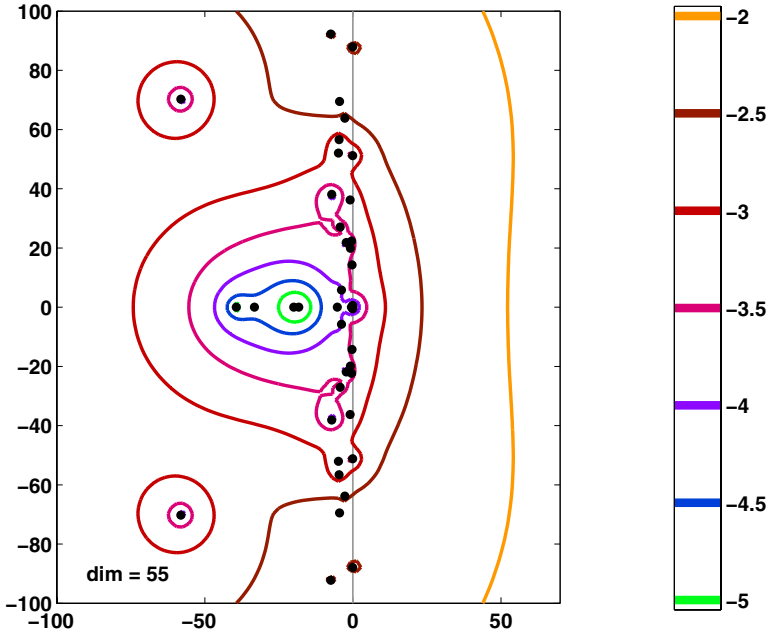

Fig. 6 ϵ-*pseudospectra for the stabilized Boeing* 767.

about $2 \cdot 10^3$. The Frobenius norm of the off-diagonal portion of the Schur form, which is the Henrici measure of departure from normality, is only 4.86. These figures are consistent with the height of the hump and the spread of the pseudospectra contours.

15. Boeing 767 Example. Figure 5 plots $\|e^{tA}\|$ and Figure 6 shows the pseudospectra for an example from control theory. Burke, Lewis, and Overton [125] describe nonsmooth, nonconvex optimization methods that move the spectrum of a matrix A into the left half plane by adding a low-order stabilizer, $A + BKC$. They start with a matrix of order $n = 55$ that models a Boeing 767 aircraft at a flutter condition. The matrix has eigenvalues with positive real part, corresponding to an unstable system. The control matrix K is 2-by-2, so there are only four control parameters. The optimization task is to choose these parameters in such a way that all

the eigenvalues are moved into the left half plane. Their final matrix is stable, but only barely.

The height of the hump is almost 10^5, so initial transients are greatly magnified before they eventually die out. The eigenvalue closest to the real axis has $\mathrm{Re}(\lambda) = -0.0788$. This should be compared with $\max |\lambda| = 10^3$ and $\|A\| = 1.69 \cdot 10^7$. The condition number of the matrix of eigenvectors is $9.5 \cdot 10^6$. These values are all consistent with the pseudospectra plot that shows small perturbations easily move the eigenvalues into the right half plane and destabilize the system.

The B767 example also provides some timing results. On a 700 MHz Pentium III laptop, MATLAB requires 0.024 seconds to compute `expm(A)`. The demo function `expm1(A)` requires only slightly more time. The norm of this matrix is large, $1.69 \cdot 10^7$, which is greater than 2^{24}, so the scaling and squaring parameter is $s = 26$. Consequently, almost 70% of the time is spent in the repeated squaring. The remaining 30% is spent in the Padé approximation, including 16% in matrix multiplications, `A*X`, and 11% in the "matrix division," `D\E`. By way of comparison, the computation of eigenvalues and eigenvectors with `[V,D] = eig(A)` on this machine requires 0.020 seconds, so for this example, computation of the exponential of a matrix takes about the same time as computation of its eigenvalues and eigenvectors. However, as `t` increases, the time required to compute `expm(t*A)` also increases, while the time required to compute `eig(t*A)` does not change very much.

16. Conclusions, Again. Scaling and squaring has emerged as the least dubious of the original nineteen ways, but we still do not understand the method completely. The roundoff errors introduced during the repeated squaring have the same general effect as perturbations in the original matrix. However, the method is probably not backwards stable in the strict sense. The analysis might be something like that for inverting a matrix. Each column of a computed inverse is a column of the exact inverse of a perturbation of the original matrix, but the perturbation is different for each column. Perhaps someone can prove a similar result for scaling and squaring.

Scaling and squaring is an efficient method for computing e^A, but not so efficient for computing $e^{tA}v$ for a given vector v and many values of t. That's like computing A^{-1}, when what is really needed is the solution to $Ax = b$ for many different b's. Recent research work, particularly for sparse matrices, has focused on the $e^{tA}v$ problem, but the software is not so widely available.

New methods continue to be developed for matrices with particular properties. The ground rules for efficient computation change as machines change. Like any good saga, the story of the matrix exponential is not finished. Another update might be warranted in 2028.

Acknowledgments. Thanks to the editors of *SIAM Review* for republishing "Nineteen Ways" and, especially, to Randy LeVeque for his valuable contributions.

REFERENCES

[123] M. ARIOLI, B. CODENOTTI, AND C. FASSINO, *The Padé method for computing the matrix exponential*, Linear Algebra Appl., 240 (1996), pp. 111–130.

[124] P. BOCHEV AND S. MARKOV, *A self-validating numerical method for the matrix exponential*, Computing, 43 (1989), pp. 59–72.

[125] J. V. BURKE, A. S. LEWIS, AND M. L. OVERTON, *A nonsmooth, nonconvex optimization approach to robust stabilization by static output feedback and low-order controllers*, in 4th IFAC Symposium on Robust Control Design, 2003, submitted.

[126] D. Calvetti, E. Gallopoulos, and L. Reichel, *Incomplete partial fractions for parallel evaluation of rational matrix functions*, J. Comput. Appl. Math., 59 (1995), pp. 349–380.
[127] E. Celledoni and A. Iserles, *Approximating the exponential from a Lie algebra to a Lie group*, Math. Comp., 69 (2000), pp. 1457–1480.
[128] E. Celledoni and A. Iserles, *Methods for the approximation of the matrix exponential in a Lie-algebraic setting*, IMA J. Numer. Anal., 21 (2001), pp. 463–488.
[129] S. H. Cheng, N. J. Higham, C. S. Kenney, and A. J. Laub, *Approximating the logarithm of a matrix to specified accuracy*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1112–1125.
[130] L. Dieci and A. Papini, *Padé approximation for the exponential of a block triangular matrix*, Linear Algebra Appl., 308 (2000), pp. 183–202.
[131] L. Dieci and A. Papini, *Conditioning of the exponential of a block triangular matrix*, Numer. Algorithms., 28 (2001), pp. 137–150.
[132] V. L. Druskin and L. A. Knizhnerman, *Krylov space approximations of eigenpairs and matrix functions in exact and computer arithmetic*, Numer. Linear Algebra Appl., 2 (1995), pp. 205–217.
[133] V. Druskin, A. Greenbaum, and L. Knizhnerman, *Using nonorthogonal Lanczos vectors in the computation of matrix functions*, SIAM J. Sci. Comput., 19 (1998), pp. 38–54.
[134] W. S. Edwards, L. S. Tuckerman, R. A. Friesner, and D. C. Sorensen, *Krylov methods for the incompressible Navier Stokes equations*, J. Comput. Phys., 110 (1994), pp. 82–102.
[135] M. Embree and L. N. Trefethen, *Pseudospectra Gateway*, http://web.comlab.ox.ac.uk/pseudospectra.
[136] E. Gallopoulos and Y. Saad, *Efficient solution of parabolic equations by Krylov approximation methods*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1236–1264.
[137] A. Greenbaum, *Generalizations of the field of values useful in the study of polynomial functions of a matrix*, Linear Algebra Appl., 347 (2002), pp. 233–249.
[138] N. J. Higham, *Evaluating Padé approximants of the matrix logarithm*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1126–1135.
[139] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.
[140] M. Hochbruck, C. Lubich, and H. Selhofer, *Exponential integrators for large systems of differential equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1552–1574.
[141] R. Horn and C. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1991.
[142] T. Jahnke and C. Lubich, *Error bounds for exponential operator splittings*, BIT, 40 (2000), pp. 735–744.
[143] C. Kenney and A. J. Laub, *Condition estimates for matrix functions*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 191–209.
[144] C. S. Kenney and A. J. Laub, *Small-sample statistical condition estimates for general matrix functions*, SIAM J. Sci. Comput., 15 (1994), pp. 36–61.
[145] C. S. Kenney and A. J. Laub, *A Schur–Fréchet algorithm for computing the logarithm and exponential of a matrix*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 640–663.
[146] L. A. Knizhnerman, *Calculation of functions of unsymmetric matrices using Arnoldi's method*, Comput. Math. and Math. Phys., 31 (1991), pp. 1–9.
[147] R. Mathias, *Condition estimation for matrix functions via the Schur decomposition*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 565–578.
[148] R. Mathias, *Evaluating the Fréchet derivative of the matrix exponential*, Numer. Math., 63 (1992), pp. 213–226.
[149] I. Najfeld and T. F. Havel, *Derivatives of the matrix exponential and their computation*, Adv. Appl. Math., 16 (1995), pp. 321–375.
[150] O. Nevanlinna, *Convergence of Iterations for Linear Equations*, Birkhäuser-Verlag, Basel, 1993.
[151] B. Nour-Omid, *Applications of the Lanczos algorithm*, Comput. Phys. Comm., 53 (1989), pp. 157–168.
[152] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.
[153] Q. Sheng, *Global error estimates for exponential splitting*, IMA J. Numer. Anal., 14 (1993), pp. 27–56.
[154] R. B. Sidje, *Expokit: Software package for computing matrix exponentials*, ACM Trans. Math. Software, 24 (1998), pp. 130–156.
[155] R. B. Sidje, *Expokit software*, http://www.maths.uq.edu.au/expokit.
[156] R. B. Sidje and W. J. Stewart, *A survey of methods for computing large sparse matrix exponentials arising in Markov chains*, Comput. Statist. Data Anal., 29 (1999), pp. 345–368.

[157] D. E. Stewart and T. S. Leyk, *Error estimates for Krylov subspace approximations of matrix exponentials*, J. Comput. Appl. Math., 72 (1996), pp. 359–369.
[158] L. N. Trefethen, *Pseudospectra of linear operators*, SIAM Rev., 39 (1997), pp. 383–406.
[159] L. N. Trefethen, *Computation of pseudospectra*, Acta Numer. 8, Cambridge University Press, Cambridge, UK, 1999, pp. 247–295.
[160] T. G. Wright, *EigTool software package*, http://web.comlab.ox.ac.uk/projects/pseudospectra/eigtool/.
[161] A. Zanna and H. Z. Munthe-Kaas, *Generalized polar decompositions for the approximation of the matrix exponential*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 840–862.