NINETEEN WAYS TO COMPUTE THE EXPONENTIAL OF A MATRIX

C.B. Moler[+] and C.F. Van Loan[++]

TR 76-283

July 1976

[+] Department of Mathematics
University of New Mexico
Albuquerque, New Mexico
87106

[++] Department of Computer Science
Cornell University
Ithaca, New York
14853

[++] Supported in part by NSF grant MCS76-08686.

Errata

p.17 Nonsingularity of
large enough or if
negative real axis.

p.39 $\mathcal{L}^{-1}$

p.59 ...corresponding to

$D_{pq}(A)$ is assured if p and q are
the eigenvalues of A are on the

eigenvalues $\pm$ i a .

NINETEEN WAYS TO COMPUTE THE EXPONENTIAL OF A MATRIX

CLEVE MOLER
University of New Mexico

CHARLES VAN LOAN †
Cornell University

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.

---

† Partially supported by NSF Grant MCS76-08686 .

1. INTRODUCTION

Mathematical models of many physical, biological, and economic processes often 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

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

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

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.

Reliability means that the method does not produce completely erroneous results, or at least that some indication is given if the method cannot perform satisfactorily on a particular matrix.

Stability means that the method does not introduce any more sensitivity to perturbation than is inherent in the underlying problem. A method can be stable and still not produce accurate results if those results are not well determined by the data for the problem. Accuracy refers primarily to the error introduced by truncating infinite series or terminating iterations. Often, using more time will increase accuracy provided the method is stable.

Efficiency is measured by the amount of computer time required to solve a particular problem. There are several problems to distinguish. For example, computing e^A once is different from computing e^{tA} for several values of t. Methods which make use of some decomposition of A which is 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 $O(n^3)$ time.

In estimating the time required by matrix computations it is traditional to estimate the time required by the multiplications and then increase it by 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 actually involves a single, n-th 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 problem is based in part on the following dilemma. Attempts to exploit the special properties of the differential equation lead naturally to consideration of the eigenvalues λ_i and eigenvectors v_i of A and to a representation of $x(t)$ by

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

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 effected by roundoff error in problems where the matrix tA has large elements.

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

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

The exponential of this matrix is

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

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

$$e^{tA} = \begin{bmatrix} e^{\lambda t} & at\,e^{\lambda t} \\ 0 & e^{\lambda t} \end{bmatrix}$$

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 a, 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.

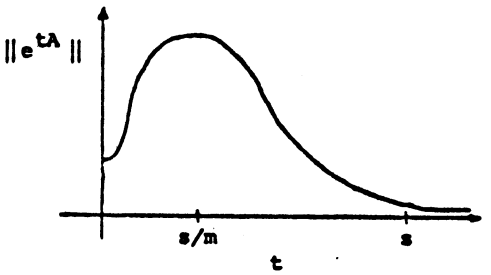

Figure 1. The "Hump"

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 beyond it for then

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

Unfortunately, the roundoff errors in the m-th power of a ma-

trix, 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 normal matrices. (See below.) The example matrix is normal if and only if $\alpha = 0$. When $\alpha = 0$, the difficulties with multiple eigenvalues and the hump both disappear.

It is convenient to review some conventions and definitions at this time. All matrices are denoted by capital letters (A,B,C,etc.). A will always be n-by-n and the dimensions of other matrices will be clear from context. 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 will have an eminent role to play:

A	symmetric	$\leftrightarrow$	$A^T = A$
A	Hermitian	$\leftrightarrow$	$A^* = A$
A	normal	$\leftrightarrow$	$A^*A = A\,A^*$
Q	orthogonal	$\leftrightarrow$	$Q^TQ = I$
Q	unitary	$\leftrightarrow$	$Q^*Q = I$
T	triangular	$\leftrightarrow$	$t_{ij} = 0$, $i > j$
D	diagonal	$\leftrightarrow$	$d_{ij} = 0$, $i \neq j$

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

$$\|x\| = \left[\sum_{i=1}^{n} |x_i|^2\right]^{1/2} , \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\|A\| = \max_{\|x\|=1} \|Ax\|$$

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

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

$$\mathrm{cond}(A) = \|A\| \, \|A^{-1}\|$$

Should A be singular, we adopt the convention that it has infinite condition.

The commutator of two matrices B and C is given by $[B,C]$ where

$$[B,C] = BC - CB$$

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

$$Q^* A Q = T$$

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,

$$J = J_1 \oplus \ldots \oplus J_k$$

of Jordan blocks

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

The λ_i are eigenvalues of A. If any of the m_i are greater than 1, A is said to be <u>defective</u>. This means that A does not have a full set of n linearly independent eigenvectors. A is <u>derogatory</u> 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 thus 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 [B15].

Theorem 1

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

$$\phi(t) \leq t \, \|E\| \, e^{[\mu(A) - \alpha(A) + \|E\|]t} \qquad (t \geq 0)$$

The scalar $\mu(A)$ is the "log norm" of A (associated with the 2-norm) and has many interesting properties [C1-C6]. In particular, $\mu(A) \geq \alpha(A)$.

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

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

where

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

Theorem 3

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

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

where

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

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

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

This shows that the perturbation bounds on $\phi(t)$ for normal matrices are as small as can be expected. This leads 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 this might be the case when A has a poorly conditioned eigensystem as measured by cond(P). This is related to having 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 [B15] . One can show that there exists a perturbation E such that

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

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

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

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

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 play a direct role.

Method 1. Taylor Series

The definition

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

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

Such an algorithm is known to be unsatisfactory even in the 1-by-1 case 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 360 using "short" arithmetic, which corresponds to a relative accuracy of $16^{-5} \approx 0.95 \cdot 10^{-6}$. We input

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

and obtained the output

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

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,

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

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 cal-

culation 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 would have used 360 "long" arithmetic (which does not require significantly more time), 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 [E7] is frequently cited. If δ is some prescribed error tolerance, Liou suggests choosing K large enough so that

$$\| T_K(A) - e^A \| < \frac{\|A\|^{K+1}}{(K+1)!} \quad \frac{1}{1 - \|A\|/(K+2)} \leq \delta$$

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 [E4] has sharpened the truncation error bound implemented by Liou, and Bickhart [E1] 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 in the left half plane. Zakian [F12] and Wragg and Davies [F11] consider 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

$$\text{cond}[D_{qq}(A)] = \text{cond}(e^{-A/2}) \geq e^{(a_1 - a_n)/2}$$

where $a_1 \geq \ldots \geq a_n$ 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 super-diagonal 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 $D_{qq}(A)$.

We briefly mention that 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 Pade 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:

$$e^{A} = \left(e^{A/m} \right)^{m}$$

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 forming 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 $\|A\|/m \leq 1$. 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 [F8], Kammler [E6], Kallstrom [L2], Scraton [F5], and Shah [E11,E12].

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 1 we show that if $\|A\| \leq 2^{j-1}$ then

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

where

$$\|E\| \leq \frac{8\|A\|^{2q+1}}{4^{qj}} \cdot \frac{(q!)^2}{(2q)!\,(2q+1)!}$$

This "inverse error analysis" result 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

$$\|E\| \leq \varepsilon \|A\|$$

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

TABLE 1.

OPTIMUM SCALING AND SQUARING PARAMETERS WITH PADÉ AND TAYLOR 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)

(q,j)
(k,j)

To read the array, 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 as the Taylor approximant 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 1 that $AE = EA$, we have

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

$$\leq \|E\| e^{\|E\|} \leq c \|A\| e^{c\|A\|}$$

A similar bound can be derived for the Taylor approximants.

The analysis and the resulting table that we have presented do <u>not</u> 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. This occurs when $\|R^2\|$ is much less than $\|R\|^2$ since the roundoff errors are small when compared to $\|R\|^2$ but not necessarily small when compared to $\|R^2\|$. Such cancellation can only happen

when cond(R) is large because $R^{-1} R^2 = R$ implies

$$\text{cond}(R) \geq \frac{\|R\|^2}{\|R^2\|}$$

The particular 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 only occurs in problems which involve a large hump . We regard it as an open question to carefully analyze the roundoff error of the repeated squaring of matrices of the form $e^{A/m}$ and to relate the analysis to a realistic assessment of the sensitivity of e^A .

In his implementation of the scaling and squaring method, Ward [F8] is aware of the possibility of cancellation. He computes an à posteriori bound on the size of the error, including the effects of both truncation and roundoff. This is certainly preferable to no error estimate at all, but it is still not completely satisfactory. A large error estimate could be the result of any of three different 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 other 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 $\max_{0\le x<\infty} | c_{qq}(x) - e^{-x} |$. For various values of q, Cody, Meinardus, and Varga [F2] 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 nonhermitian (non-normal) A, it is hard to determine how cisely $c_{qq}(A)$ 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

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

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 Pade 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 kinds of discrepancies arise from the fact that A is not normal. The example illustrates that non-normality exerts a subtle influence upon the methods of this section even though the eigensystem, per se, is not explicitly involved in any of the algorithm

4. ORDINARY DIFFERENTIAL EQUATION METHODS

Since e^{tA} and $e^{tA}x_0$ are solutions to ordinary differential equations, 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 have the advantage of being easy to use and require very little additional programming or other thought. The primary disadvantage is a relatively high cost in computer time. There is also some danger of roundoff error difficulties, although this is probably not too severe.

The o.d.e. programs are designed to solve a single system of equations

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

and obtain the solution at many values of t. By setting $f(x,t) = Ax$ the k-th column of e^{tA} can be obtained by setting x_0 to the k-th 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 O.D.E. Solver

Most computer center libraries contain programs for solving initial value problems in ordinary differential equations. 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 o.d.e. solver. This is obviously an expensive luxury since the o.d.e. 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 o.d.e. solvers to compute the exponentials of about a dozen matrices and have measured the amount of work required. The programs are:

(1) RKF45. Written by Shampine and Watts[J3], 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.

(2) DE/STEP. Written by Shampine and Gordon[J2], this program uses variable order, variable step Adams predictor-corrector formulas. Two function evaluations are required per step.

(3) IMPSUB. Written by Starner[J4], this program is a modification of Gear's DIFSUB [J1] and is based on implicit backward differentiation formulas intended for stiff differential equations. Starner's modifications add the ability to solve

"infinitely 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 such function evaluations required 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 it is fixed order. 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 give the indication, IMPSUB may require fewer.

The following table gives the results for one particular matrix which we arbitrarily declare to be a "typical" nonstiff problem. The matrix is of order 3, with a double eigenvalue equal to 3 and a simple eigenvalue equal to 6. The matrix is defective. We used three different local error tolerances and

integrated over the interval [0,1] . The average number of function evaluations for the three different 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 problem. IBM/360 long arithmetic was used.

TABLE 2.

Local Error Tolerance Versus Subroutine

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

Although people concerned with the competition between various o.d.e. 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 lend general support to 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 to obtain higher accuracies than those obtained with 200 n^3 or more flops for the o.d.e. 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 o.d.e. methods to handle our specific problem.

Method 6. Single Step O.D.E. 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

$$x_{j+1} = (I + hA + \ldots + \frac{h^4}{4!} A^4)\, x_j = T_4(Ah)\, x_j$$

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

$$\begin{aligned} k_1 &= hAx_j \\ k_2 &= hA(x_j + \tfrac{1}{2} k_1) \\ k_3 &= hA(x_j + \tfrac{1}{2} k_2) \\ k_4 &= hA(x_j + k_3) \end{aligned}$$

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 $h = 1/m$ where m is a positive integer, then

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

Consequently, there is a close connection between this method and Method 3 which involved scaling and squaring [E9,E15]. The scaled matrix is hA and the exponential of the scaled matrix 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 seems to be no important advantages for fixed step size Runge-Kutta.

Let us now consider the possibility of variable 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

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

Notice that we are using a 5-th order formula to actually compute the approximation, but are using a 4-th 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 $T_4(h_jA)$ and $T_5(h_jA)$ cannot be precomputed. Each step requires $5n^2$ flops. However, in those problems which involve large "humps" as described in Section 1 , a smaller step size is needed at the beginning of the computation than at the end. If the step size ε 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

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

we see that the required step size is given approximately by

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

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 5-th order error estimate -- asking for 3 more figures roughly quadruples the work.

Method 7. Multistep O.D.E. 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 such an algorithm might be among the more effective methods, 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. In particular, we do not know what effect rounding errors would have in a problem with a large hump for detailed studies are lacking.

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

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

Using this result it is possible to show 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 :

$$e^{tA} = \sum_{k=0}^{\infty} \frac{t^k A^k}{k!} = \sum_{k=0}^{\infty} \frac{t^k}{k!} \left[\sum_{j=0}^{n-1} \beta_{kj} A^j \right]$$

$$= \sum_{j=0}^{n-1} \left[\sum_{k=0}^{\infty} \beta_{kj} \frac{t^k}{k!} \right] A^j = \sum_{j=0}^{n-1} \alpha_j(t) A^j$$

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 $\alpha_j(t) = \Sigma \beta_{kj} t^k/k!$ 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 of this method 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 $s_0(t) = \Sigma\,(\alpha t)^k/k!$ 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 360 short arithmetic is used.

Another problem with this method is the requirement that the characteristic polynomial be known. If $\lambda_1,\ldots,\lambda_n$ are the eigenvalues of A, then $c(z)$ could be computed from the formula $c(z) = \prod_1^n (z - \lambda_i)$. 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[A13]. 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

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

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

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

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

Generalizations of these formulas exist for the case when the eigenvalues are exactly confluent and we refer the reader to MacDuffee [A8] for a discussion of them.

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 of eigenvalues

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

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

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

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 v_{jk} are available and Vidysager[G15] has proposed their use.

Methods 9,10, and 11 suffer on several accounts. They are $O(n^4)$ algorithms making then 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 "non-polynomial" method. Another weakness of these algorithms is their behavior when the eigenvalues are nearly confluent. Even though the formulas which define methods 9,10, and 11 have special form in the confluent case , we do not have a pleasing numerical situation. The "gray" area of near confluence poses difficult problems which are best discussed in the next section on decomposition techniques.

The last 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,

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

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 [A3] 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 [H7] 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} [G1,G3,G11,G12,G16].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 effected by roundoff error.

Method 13. Companion Matrix

We now discuss techniques which involve the computation of e^C where C is a companion matrix:

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

Companion matrices have some interesting properties which various authors have tried to exploit [H1-H9] :

(i) C is sparse.

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

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

(iv) If a general matrix A is not derogatory, then it is similar to a companion matrix. If A is derogatory, then 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 when these formulas are used.

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

$$e^A = Y e^C Y^{-1}$$

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 [A13].

We mention that if the original differential equation is actually a single n-th 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

$$A = Q H Q^T \qquad Q^T Q = I$$

then

$$e^A = Q e^H Q^T$$

Unlike the reduction to companion form, this factorization can be stably computed using the EISPACK routine ORTHES [K4].

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

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

by equating the k-th columns in the matrix identity $He^H = e^H H$.

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 [H9] . 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 o.d.e. 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 non-negligible $h_{k-1,k}$. 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 various factorizations or decompositions of the matrix A. Unfortunately, these are also the methods which suffer the most from the difficulties which attend confluent eigen values. However, if A happens to be symmetric, then all these methods reduce to the same method and that method is very effective.

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

$$A = S B S^{-1}$$

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

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

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 $\text{cond}(S)$ 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

$$V = [\, v_1 \mid \cdots \mid v_n \,]$$

and

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

These n equations can be written

$$AV = VD$$

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

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

Consequently, if V is nonsingular,

$$e^{tA} = V e^{tD} V^{-1}$$

In terms of the differential equation $\dot{x} = Ax$, the same eigenvector approach takes the following form. The initial condition is expressed as a linear 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 $Va = x(0)$.

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 very well for any other symmetric matrix because the eigenvectors can then be taken to be orthoganal regardless of eigenvalue multiplicity. If reliable subroutines such as TRED2 and TQL2 in EISPACK [K4] are used to compute the eigenvalues and eigenvectors, the computed v_j will be orthogonal to the full accuracy of the computer. 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 $\dot{A}$ 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 whole 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 cond(V). Consequently, when 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

$$V \quad \begin{bmatrix} 1 & -1 \\ 0 & 2\varepsilon \end{bmatrix},$$

$$D = \text{diag}(1+\varepsilon , 1-\varepsilon) ,$$

and

$$\text{cond}(V) = O(\tfrac{1}{\varepsilon})$$

If $\varepsilon = 10^{-5}$ and IBM/360 short floating point arithmetic is used to compute the exponential from the formula $e^A = V e^D V^{-1}$, 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

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

regardless of what ε is. 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 Lagrange interpolation. When the eigenvalues are distinct the eigenvector approach can be expressed

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

where y_j^T is the j-th 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 [A13]. 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 [K4]. 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

$$T R = R D$$

Finally, the eigenvectors of A are obtained by a simple matrix

multiplication

$$V = Q R$$

The key observation is that R happens to be triangular. In other words, the ORTHES/HQR2 path in EISPACK computes the matrix of eigenvectors by first computing its "Q-R" 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 of equations involving V is reduced. However, since this is a fairly small fraction of the total time required, the improvement in overall efficiency is not very significant. A more important advantage is that

$$\text{cond}(V) = \text{cond}(R)$$

(assuming use of the 2-norm) and that the estimation of cond(R) can be done fairly reliably and efficiently.

The effect of dropping the assumption that A has real eigenvalues is that R is not quite triangular, but has 2-by-2 blocks on its diagonal for each pair of complex eigenvalues. Such a matrix is called "quasi-triangular". The minor inconvenience that loss of proper triangularity implies allows us to avoid complex arithmetic.

In summary, we suspect the following algorithm might be fairly reliable:

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

$$A Q R = Q R D$$

2. Given x_0, compute y_0 by solving

$$R y_0 = Q^T x_0$$

Also estimate 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

$$x(t) = V e^{tD} y_0$$

(If we want to compute the full exponential, then in Step 2 we solve $RY = Q^T$ for Y and then use $e^{tA} = V e^{tD} Y$ 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 reasonably assess the errors in the computed solution when that solution is found. It would be interesting to code this algorithm and com-

pare it with Ward's scaling and squaring program. (See 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

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

is the JCF of A, then

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

The exponentials of the Jordan blocks J_i can be given in close 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 problem with this technique is that the JCF cannot be computed using floating point arithmetic. A single rounding error may cause some multiple eigenvalue to become distinct or vica versa and as a result, the entire structure of J and P may be altered. Another way of saying essentially the same thing is that there is no à priori bound on cond(P). An appreciation of the computational difficulties which attend the JCF may be obtained by reading the papers by Golub and Wilkinson [K1] and Kagstrom and Ruhe [K2] .

Method 17. Schur.

The Schur decomposition

$$A = Q T Q^T$$

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 shortended version of HQR2. The required modifications are discussed in the EISPACK guide [K4] .

Once the Schur decomposition is available,

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

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 of triangular matrices is the subject of a recent paper by Parlett [K3] . 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

$$A = S B S^{-1}$$

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 Jordan Canonical Form places all the emphasis on the first objective, while the Schur decomposition places most of the em-

phasis 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.

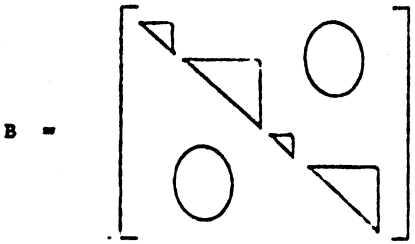

Figure 2. Triangular Block Diagonal Form

Each of the blocks in B involves a cluster of nearly confluent eigenvalues. The number in each cluster, and hence, the size of each block is to be made as small as possible while maintaining some prescribed upper bound for cond(S), such as

$$\text{cond}(S) \leq 100$$

The choice of 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 $e^{tA} = S\, e^{tB}\, S^{-1}$. A larger bound would mean more figures would be lost. A smaller bound would mean that more computer time would be required, both for the factoriz-

ation 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 very fast. The bound on cond(S) will mean that it is occasiona necessary to have larger blocks in B , but it will insure aga excessive loss of accuracy from confluent eigenvalues.

G.W. Stewart has pointed out that the grouping of the ei envalues into clusters and the resulting block structure of B is not merely for increased speed. There can be an important a curacy benefit as well. Stewart suggests expressing each block B_j in the form

$$B_j = \gamma_j I + E_j$$

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

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

where ϵ is the computer rounding unit, then

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

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 computed rapidly and accurately by using

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

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

Several researchers, including Parlett and Stewart, are currently developing computer programs based on some of these ideas. There are many details to be worked out. The most difficult is how to choose the proper 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 well come close to meeting all our other criteria for satisfactory methods.

Once the confluent eigenvalue difficulties are dealt with in a stable way, concern focuses on the efficiency of the 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 about $10n^3$ flops. The clustering of the eigenvalues and the further reduction to block diagonal requires a few more n^3 flops. Again we emphasize that the reduction is independent of t . Once the decomposition is obtained, the cal-

culation of the matrix e^{tA} requires about $2n^3$ flops for each t. If all that is required are the vectors $x(t) = e^{tA} x_0$ for various t, the equation $Sy_0 = x_0$ should be solved only once, at a cost of $n^3/3$ flops, and then each $x(t)$ can be obtained with only n^2 flops.

Of course, these are only rough work estimates. There will be differences between programs which work with 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 the matrix exponential is that the familiar additive law does not carry over from the scalar case unless we have commutivity:

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

Nevertheless, there are ways of relating the exponentials of B and C to that of $B + C$. One of these is the Trotter product formula [B13] :

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

Method 19. Splitting

The Trotter result suggests another method for approximating e^A . First, find a convenient split of the form $A = B+C$ and second, use the approximation

$$e^A \approx (e^{B/m}\, e^{C/m})^m$$

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

$$B = (A + A^T)/2$$

$$C = (A - A^T)/2$$

then the exponentials of the symmetric matrix B and the skew-

symmetric matrix C can be effectively computed by the methods of Section 5. For the above choice of B and C we show in Appendix 2 that

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

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 $B = Q \operatorname{diag}(\mu_i) Q^T$ $(Q^TQ=I)$ using TRED2 and TQL2 [K4]. Variations of these programs can be used to compute the factorization $C = U D U^T$ 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

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

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

(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 access the relative efficiency of this splitting method because the amount of work 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 $\mu(A)$ was negative and/or $\|[A^T,A]\|$ was much less than $\|A\|$.

Turning to the question of accuracy, our main concern are the rounding errors which arise in (d) as a result of the repeated squaring. The remarks we made about repeated squaring in connection with Method 3 are applicable here, that is, there may be severe cancellation during the powering but whether or not this only occurs in sensitive e^A problems is unknown.

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

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

which we establish in Appendix 2.

To illustrate, suppose A has companion form

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

If $B = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}$ $\begin{matrix} \\ \}n-1 \end{matrix}$ $\begin{matrix} \\ n-1 \end{matrix}$ 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 require only $O(n^2)$ flops. Since $\|B\| = 1$, $\|C\| = \|c\|$, and $\|[B,C]\| \leq 2\|c\|$, (7.2) becomes

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

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 in front of the n^3 increases as $\|A\|$ increases.

What is probably the best o.d.e. method has not been implemented. It would be a specialization of a variable order, variable step o.d.e. solver to the linear, constant coefficient problem. We suspect it would be very stable and reliable although somewhat expensive in computer time. 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 variations which involve further reduction to some block form. In all cases the initial decomposition is $O(n^3)$ 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 there are perhaps three or four candidates for "best" method. The choice among them will depend upon the detail of implementation and upon the particular problem being solved.

APPENDIX 1 . INVERSE ERROR ANALYSIS OF PADE MATRIX APPROXIMATION

Lemma 1.

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

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

Proof.

If $\|H\| < 1$ then $\log(I + H) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{H^k}{k}$

and so

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

Lemma 2.

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

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

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

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

Lemma 3.

If $\|A\| \leq \frac{1}{2}$, $q \leq p$, and $p \geq 1$, then $R_{pq}(A) = e^{A+F}$ where

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

Proof.

From the remainder theorem for Pade approximants [F7] ,

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

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 $\frac{p+q}{p} e^{.5} \leq 4$ we obtain

$$\|H\| \leq \frac{1}{(p+q)!} \|A\|^{p+q+1} \frac{p+q}{p} \int_0^1 e^{.5} u^p (1-u)^q \, du$$

$$\leq 4\|A\|^{p+q+1} \frac{p!\,q!}{(p+q)!\,(p+q+1)!}$$

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

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

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 $\|A\| \leq \frac{1}{2}$ then $R_{pq}(A) = e^{A+F}$ where

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

Proof.

The case $p \geq q$, $p > 1$ is covered by Lemma 1 . If $p+q = 0$, then $F = -A$ and the above inequality holds. Finally, consider the case $q > p$, $q \geq 1$. 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

If $\frac{\|A\|}{2^j} \leq \frac{1}{2}$, then $[R_{pq}(\frac{A}{2^j})]^{2^j} = e^{A+E}$

where

$$\|E\| \leq 8 \frac{\|A\|^{p+q+1}}{2^{j(p+q)}} \frac{p!\, q!}{(p+q)!\,(p+q+1)!}$$

Proof.

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

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

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

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

Corollary 1.

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

$$\|E\| \leq 8 \frac{\|A\|^{k+1}}{2^{jk}} \frac{1}{k+1}$$

Corollary 2.

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

where

$$\|E\| \leq 8 \frac{\|A\|^{2q+1}}{4^j} \frac{(q!)^2}{(2q)!(2q+1)!}$$

APPENDIX TWO. ACCURACY OF THE TROTTER APPROXIMATION.

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 :

$$\text{(i)} \quad \|e^{tA}\| \leq e^{\mu(A)t} \qquad (t \geq 0)$$

$$\text{(ii)} \quad \mu(A) \leq \|A\|$$

$$\text{(iii)} \quad \mu(B + C) \leq \mu(B) + \|C\|$$

These and other results concerning log norms are discussed in references [C1]-[C6] .

Lemma 1

If $\theta \geq \max\{\, \mu(A) \,,\, \mu(B) + \mu(C) \,\}$ then

$$\|S_m^m - T_m^m\| \leq m\, e^{\theta(m-1)/m}\, \|S_m - T_m\|$$

Proof.

Following Reed and Simon[A10] 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

$$\|S_m^m - T_m^m\| \leq \sum_{k=0}^{m-1} \|S_m\|^k \|S_m - T_m\| \|T_m\|^{m-1-k}$$

$$\leq \|S_m - T_m\| \sum_{k=0}^{m-1} e^{\theta k/m} e^{\theta(m-1-k)/m}$$

from which the lemma immediately follows.

In Lemmas 2 and 3 we shall make use of the notation

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

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

Lemma 2

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

Proof.

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

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

The lemma follows since

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

Lemma 3

If X and Y are matrices then

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

Proof

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

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

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

$$\| [e^X, Y] \| \leq \int_0^1 \| e^{tX} \| \, \| [X,Y] \| \, \| e^{(1-t)X} \| \, dt$$

$$\leq \| [X,Y] \| \int_0^1 e^{\mu(X)t} \, e^{\mu(X)(1-t)} \, dt$$

from which the lemma immediately follows.

Theorem

If $\theta \geq \max\{ \mu(A) , \mu(B) + \mu(C) \}$, then

$$\| S_m^m - T_m^m \| \leq \frac{1}{2m} e^{\theta} \| [B,C] \|$$

Proof

If $0 \leq t \leq 1$ then an application of Lemma 3 with $X \equiv (1 - t)A/m$ and $Y \equiv C/m$ yields

$$\| [e^{(1-t)A/m}, C/m] \| \leq e^{\mu(A)(1-t)/m} \| [(1-t)A/m, C/m] \|$$

$$\leq e^{\theta(1-t)/m} \frac{(1-t)}{m^2} \| [B,C] \|$$

By coupling this inequality with Lemma 2 we can bound $\| T_m - S_m \|$:

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

$$\leq \int_0^1 e^{\mu(B)t/m} e^{\theta(1-t)/m} \frac{(1-t)}{m^2} \| [B,C] \| e^{\mu(C)t/m} dt$$

$$\leq \frac{1}{2} e^{\theta/m} \frac{\| [B,C] \|}{m^2}$$

The theorem follows by combining this result with Lemma 1.

Corollary 1.

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

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

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 2.

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

Proof.

$$\max\{\mu(A), \mu(B)+\mu(C)\} \leq \mu(B) + \| C \| \leq \| B \| + \| C \|$$

(Note: We are obliged to Professor Paul Federbush of the University of Michigan for helping us with the analysis in this appendix)

Acknowledgements

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.

BIBLIOGRAPHY

ATTENDING LINEAR ALGEBRA AND ANALYSIS

[A1] R.Bellman, Introduction to Matrix Analysis, McGraw-Hill, New York, 1969.

[A2] C.Davis, Explicit Functional Calculus, J.Linear Algebra and It's Applications, 6(1973), pp.193-199.

[A3] V.N.Faddeeva, Computational Methods of Linear Algebra, Dover Publications, New York, 1959.

[A4] J.S.Frame, Matrix Functions and Applications Part II : Functions of Matrices, IEEE Spectrum, 1(April, 1964), pp.102-108.

[A5] J.S.Frame, Matrix Functions and Applications Part IV: Matrix Functions and Constituent Matrices, 1(June, 1964), pp.123-131.

[A6] J.S.Frame, Matrix Functions and Applications Part V: Similarity Reductions by Rational or Orthogonal Matrices, IEEE Spectrum, 1(July, 1964), pp.103-109.

[A7] F.R.Gantmacher, The Theory of Matrices Volumes I and II, Chelsea Publishing Co., New York, 1959.

[A8] C.C.MacDuffee, The Theory of Matrices, Chelsea Publishing Co., New York, 1956.

[A9] L.Mirsky, An Introduction to Linear Algebra, Oxford University Press, London, 1955.

[A10] M.Reed and B.Simon, Functional Analysis, Academic Press, New York, 1972.

[A11] R.F.Rinehart, The Equivalence of Definitions of a Matrix Function, Amer.Math.Monthly, 62(1955), pp.395-414.

[A12] P.C.Rosenbloom, Bounds on Functions of Matrices, Amer.Math. Monthly, 74(1967), pp.920-926.

[A13] J.H.Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, Oxford, 1965.

PROPERTIES AND REPRESENTATIONS

[B1] T.M.Apostol, Some Explicit Formulas for the Matrix Exponential, Amer.Math.Monthly 76(1969), pp.284-292.

[B2] A.Bradshaw, The Eigenstructure of Sample Data Systems with Confluent Eigenvalues, Int.J.Systems Science, 5(1974), pp.607-613.

[B3] R.Bellman, Perturbation Techniques in Mathematics, Physics, and Engineering, Holt,Rinehart, and Winston, New York, 1964.

[B4] J.C.Cavendish (et al), On the Norm of a Matrix Exponential, SIAM Review, 17(1975), pp.174-175.

[B5] C.G.Cullen, Remarks on Computing e^{At}, IEEE Trans.Auto.Cont., AC-16(1971), pp.94-95.

[B6] F.Fer, Resolution de l'Equation Matricelle $\frac{dU}{dt} = pU$ par Produit Infini D'exponentielles, Acad.Roy.Belg.Cl. Sci., 44(1958), pp.819-829.

[B7] E.P.Fulmer, Computation of the Matrix Exponential, Amer. Math.Monthly, 82(1975), pp.156-159.

[B8] B.Kagstrom, Bounds and Perturbation Bounds for the Matrix Exponential, Report 55.76, Department of Information Processing, University of Umea, Umea, Sweden, 1976.

[B9] T.Kato, Perturbation Theory for Linear Operators (Chap.9), Springer-Verlag, New York, 1966.

[B10] R.B.Kirchner, An Explicit Formula for e^{At}, Amer.Math.Monthly, 74(1967), pp.1200-1204.

[B11] 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.

[B12] N.M.Rice, More Explicit Formulas for the Exponential Matrix, Queen's Mathematical Reprints No.1970-21, Queen's University, Kingston, Ontario, 1970.

[B13] H.F.Trotter, Product of Semigroups of Operators, Proc.AMS, 10(1959), pp.545-551.

[B14] C.F.Van Loan, A Study of the Matrix Exponential, Numerical Analysis Report No.7, Department of Mathematics, University of Manchester, Manchester, England, 1975.

[B15] C.F.Van Loan, The Sensitivity of the Matrix Exponential, submitted for publication.

[B16] G.H.Weiss and A.A.Maradudin, The Baker-Hausdorff Formula and a Problem in Crystal Physics, J.Math.and Physics, 3(1962), pp.771-777.

[B17] A.D.Ziebur, On Determining the Structure of A by Analyzing e^{At} , SIAM Review, 12(1970), pp.98-102.

LOG NORMS AND STABILITY

[C1] W.A.Coppel, Stability and Asymptotic Behavior of Differential Equations, D.C.Heath, Boston, 1965.

[C2] G.Dahlquist, Stability and Error Bounds in the Numerical Integration of Ordinary Differential Equations, Transactions of the Royal Institute of Technology, No. 130, Stockholm, Sweden, 1959.

[C3] C.V.Pao, Logarithmic Derivatives of a Square Matrix, J.Linear Algebra and Its Applications, 6(1973), pp.159-164.

[C4] C.V.Pao, A Further Remark on the Logarithmic Derivatives of a Square Matrix, J.Linear Algebra and Its Applications 7(1973), pp.275-278.

[C5] T.Strom, Minimization of Norms and Logarithmic Norms by Diagonal Similarities, Computing, 10(1972), pp.1-9.

[C6] T.Strom, On Logarithmic Derivatives, SIAM J.Num.Anal., 12(1975), pp.741-753.

SURVEY ARTICLES

[D1] M.Healey, Study of Methods of Computing Transition Matrices, Proc.IEE, 120(1973), pp.905-912.

[D2] C.B.Moler, Difficulties in Computing the Exponential of a Matrix, Proceedings of the Second USA-Japan Computer Conference, 1975 , pp.79-82.

TRUNCATED TAYLOR SERIES

[E1] T.A.Bickart, Matrix Exponential: Approximation by Truncated Power Series, Proc.IEEE, 56(1968), pp.872-873.

[E2] G.J.Bierman, Power Series Evaluation of Transition and Covariance Matrices, IEEE Trans.Auto.Cont., AC-17(1972), pp. 228-231.

[E3] K.C.Daly, Evaluating the Matrix Exponential, Elect.Letters, 8(1972), p.390.

[E4] W.Everling, On the Evaluation of e^{At} by Power Series, Proc. IEEE, 55(1967), p.413.

[E5] D.A.Gall, The Solution of Linear Constant Coefficient Ordinary Differential Equations with APL, Computer Methods in Mechanics and Engineering, 1(1972), pp.189-196.

[E6] D.W.Kammler, An Algorithm for Numerically Solving a First Order System of Linear Differential Equations with Constant Coefficients, unpublished manuscript, 1973.

[E7] M.L.Liou, A Novel Method of Evaluating Transient Response, Proc.IEEE, 54(1966), pp.20-23.

[E8] J.B.Mankin and J.C.Hung, On Roundoff Errors in Computation of Transition Matrices, Reprints of the Joint Automatic Control Conference, pp.60-64, Univ.of Colorado, Boulder, Colorado, 1969.

[E9] 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.

[E10] J.B.Plant, On the Computation of Transition Matrices for Time Invariant Systems, Proc.IEEE, 56(1968), pp.1397-1398.

[E11] M.M.Shah, On the Evaluation of e^{At}, Cambridge Report CUED/B-Control TR8, Cambridge, England, 1971.

[E12] M.M.Shah, Analysis of Roundoff and Truncation Errors in the Computation of Transition Matrices, Cambridge Report CUED/B-Control TR12, Cambridge, England, 1971.

[E13] C.J.Standish, Truncated Taylor Series Approximation to the State Transition Matrix of a Continuous Parameter Markov Chain, J.Linear Algebra and Its Applications, 12(1975), pp.179-183.

[E14] D.E.Whitney, Propogated Error Bounds for Numerical Solution of Transient Response, Proc.IEEE, 54(1966), pp. 1084-1085.

[E15] 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

[F1] J.L.Blue and H.K.Gummel, Rational Approximations to the Matrix Exponential for Systems of Stiff Differential Equations, J.Computational Physics, 5(1970), pp.70-83.

[F2] 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.Appx.Theory, 2(1969), pp.50-65.

[F3] W.Fair and Y.Luke, Pade Approximations to the Operator Exponential, Numer.Math., 14(1970), pp.379-382.

[F4] E.B.Saff and R.S.Varga, On the Zeros and Poles of Pade Approximants to exp(z), Numer.Math., 25(1975), pp.1-14.

[F5] R.E.Scraton, Comment on Rational Approximants to the Matrix Exponential, Electronics Letters, 7(1971), pp.260-261.

[F6] G.Siemieniuch and I.Gladwell, On Time Discretizations for Linear Time Dependent Partial Differential Equations, Numerical Analysis Report No.5, Department of Mathematics, University of Manchester, Manchester, England, 1974.

[F7] R.S.Varga, On Higher Order Stable Implicit Methods for Solving Parabolic Partial Differential Equations, J.Math. Phys., 40(1961), pp.220-231.

[F8] R.C.Ward, Numerical Computation of the Matrix Exponential with Accuracy Estimate, Union Carbide Corp. Nuclear Division Technical Report UCCND CSD 24 , Knoxville Tennessee, 1975.

[F9] A.Wragg and C.Davies, Evaluation of the Matrix Exponential, Electronics Letters, 9(1973), pp.525-526.

[F10] A.Wragg and C.Davies, Computation of the Exponential of a Matrix I: Theoretical Considerations, 11(1973), pp. 369-375.

[F10] A.Wragg and C.Davies, Computation of the Exponential of a Matrix I: Theoretical Considerations, J.Inst.Math. Applic., 11(1973), pp.369-375.

[F11] A.Wragg and C.Davies, Computation of the Exponential of a Matrix II: Practical Considerations, J.Inst.Math. Applic., 15(1975), pp.273-278.

[F12] V.Zakian, Rational Approximants to the Matrix Exponential, Electronics Letters, 6(1970), pp.814-815.

POLYNOMIAL METHODS

[G1] G.J.Bierman, Finite Series Solutions for the Transition Matrix and Covariance of a Time-Invariant System, IEEE Trans.Auto.Cont., AC-16(1971), pp.173-175.

[G2] J.A.Boehm and J.A.Thurman, An Algorithm for Generating Constituent Matrices, IEEE Tran.Circuit Theory, CT-18 (1971), pp.178-179.

[G3] 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.

[G4] W.C.Davidon, Exponential Function of a 2-by-2 Matrix, Hewlett-Packard HP65 Library Program.

[G5] S.Deards, On the Evaluation of Exp(tA), Matrix and Tensor Quarterly, 23(1973), pp.141-142.

[G6] S.Ganapathy and R.S.Rao, Transient Response Evaluation from the State Transition Matrix, Proc.IEEE, 57(1969), pp.347-349.

[G7] I.C.Goknar, On the Evaluation of Constituent Matrices, Int. J.Sys.Sci. 5(1974), pp.213-218.

[G8] I.I.Kolodner, On Exp(tA) with A Satisfying a Polynomial, J.Math.Anal.and Appl., 52(1975), pp.514-524.

[G9] Y.L.Kuo and M.L.Liou, Comments on a "Novel Method of Evaluating e^{At} in Closed Form", IEEE Trans.Auto.Cont., AC-16(1971), p.521.

[G10] E.J.Mastascusa, A Method of Calculating e^{At} Based on the Cayley-Hamilton Theorem, Proc.IEEE, 57(1969), pp. 1328-1329.

[G11] K.R.Rao and N.Ahmed, Heaviside Expansion of Transition Matrices, Proc.IEEE, 56(1968), pp.884-886.

[G12] K.R.Rao and N.Ahmed, Evaluation of Transition Matrices, IEEE Trans.Auto.Contr., AC-14(1969), pp.779-780.

[G13] B.Roy (et al), On the Evaluation of the State Transition Matrix, Proc.IEEE, 57(1969), pp.234-235.

[G14] M.N.S.Swamy, On a Formula for Evaluating e^{At} When the Eigenvalues are not Necessarily Distinct, Matrix and Tensor Quarterly, 23(1972), pp.67-72.

[G15] M.Vidysager, A Novel Method of Evaluating e^{At} in Closed Form, IEEE Trans.Auto.Contr., AC-15(1970), pp.600-601.

[G16] V.Zakian, Solution of Homogeneous Ordinary Linear Differential Equations by Numerical Inversion of Laplace Transforms, Electronics Letters, 7(1971), pp.546-54

COMPANION MATRIX METHODS

[H1] A.K.Choudhury (et al), On the Evaluation of e^{At}, Proc. IEEE, 56(1968), pp.1110-1111.

[H2] C.J.Harris, Evaluation of Matrix Polynomials in the State Companion Matrix of Linear Time Invariant Systems Int.J.Sys.Sci., 4(1973), pp.301-307.

[H3] I.Kaufman, Evaluation of an Analytical Function of a Companion Matrix with Distinct Eigenvalues, Proc.IEEE, 57(1969), pp.1180-1181.

[H4] I.Kaufman, A Note on the "Evaluation of an Analytical Function of a Companion Matrix with Distinct Eigenvalue Proc.IEEE, 57(1969), pp.2083-2084.

[H5] I.Kaufman and P.H.Roe, On Systems Described by a Companion Matrix, IEEE Trans.Auto.Contr., AC-15(1970), pp.692-693.

[H6] I.Kaufman, H.Mann, and J.Vlach, A Fast Procedure for the Analysis of Linear Time Invariant Networks, IEEE Trans.Cir.Th., CT-18(1971), pp.739-741.

[H7] M.L.Liou, Evaluation of the Transition Matrix, Proc.IEEE, 55(1967), pp.228-229.

[H8] A.K.Mandal (et al), Numerical Computation Method for the Evaluation of the Transition Matrix, Proc.IEE, 116(1969), pp.500-502.

[H9] W.E.Thomson, Evaluation of Transient Response, Proc.IEEE, 54(1966), p.1584.

ORDINARY DIFFERENTIAL EQUATIONS

[J1] C.W.Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice Hall, Englewood Cliffs, New Jersey, 1971.

[J2] L.F.Shampine and M.K.Gordon, Computer Solution of Ordinary Differential Equations - the Initial Value Problem, W.H.Freeman and Co., San Francisco, 1975.

[J3] L.F.Shampine and H.A.Watts, Report on RKF45 , Sandia Laboratories, in preparation.

[J4] J.Starner, Numerical Solution of Implicit Differential-Algebraic Equations, Ph.D. Thesis, University of New Mexico, 1976.

MATRIX DECOMPOSITION METHODS

[K1] G.H.Golub and J.H.Wilkinson, Ill-Conditioned Eigensystems and the Computation of the Jordan Canonical Form, Stanford Computer Science Report CS75-478, 1975.

[K2] B.Kagstrom and A.Ruhe , An Algorithm for Numerical Computation of the Jordan Normal Form of a Complex Matrix, Report UMINF 51.74, Department of Information Processing, University of Umea, Umea, Sweden, 1974.

[K3] B.N.Parlett, Computation of Functions of Triangular Matrices, Memo No. ERL-M481, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, 1974.

[K4] B.T.Smith (et al), Matrix Eigensystem Routines: EISPAK Guide, Second Edition, Springer-Verlag, New York, 1976.

INTEGRALS INVOLVING e^{tA}

[L1] J.C.Johnson and C.L.Phillips, An Algorithm for the Computation of the Integral of the State Transition Matrix, IEEE Trans.Auto.Cont., AC-16(1971), pp.204-205.

[L2] C.Kallstrom, Computing Exp(A) and $\int \mathrm{Exp}(As)ds$, Report 7309, Division of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1973.

[L3] A.H.Levis, Some Computational Aspects of the Matrix Exponential, IEEE Trans.Auto.Cont., AC-14(1969), pp.410-411.

SELECTED APPLICATIONS

[M1] F.H.Branin, Computer Methods of Network Analysis, Proc. IEEE, 55(1967), pp.1787-1801.

[M2] R.Brockett, Finite Dimensional Linear Systems, Wiley, New York, 1970.

[M3] K.F.Hansen, B.V.Koen, and W.W.Little, Stable Numerical Solutions of the Reactor Kinetics Equations, Nuclear Science and Engineering, 22(1965), pp.51-59.

[M4] J.A.W.da Nobrega, A New Solution of the Point Kinetics Equations, Nuclear Science and Engineering, 46(1971), pp.366-375.